Compiling a Benchmarking Test-Suite for Combinatorial Black-Box Optimization: A Position Paper

Ofer M. Shir
Tel-Hai College and Migal Institute
Upper Galilee, Israel
ofersh@telhai.ac.il

Carola Doerr
Sorbonne University, CNRS, LIP6
Paris, France
Carola.Doerr@mpi-inf.mpg.de

Thomas Bäck
LIACS, Leiden University
Leiden, The Netherlands
t.h.w.baeck@liacs.leidenuniv.nl

ABSTRACT

This contribution focuses on the challenge of formulating a set of benchmark problems and/or a test-suite for Combinatorial Optimization problems when treated as black-box global optimization problems. We discuss the involved dilemmas and possible obstacles of such a compilation. To this end, we formulate a list of design questions that need to be answered as a first step in this compilation process. We articulate our perspective on these questions by proposing a rough classification of relevant problem classes, answering the posed questions, and suggesting a preliminary set of problems. While this position paper addresses the Evolutionary Computation community, it intends to offer an open-minded Computer Science perspective – by considering the broad definition of Combinatorial Optimization and by accounting for equivalent threads within Operations Research and Mathematical Programming communities. At the same time, this work capitalizes on prior art in algorithms' benchmarking, including the authors' own experience with the continuous BBOB benchmark problem set, as well as a number of discrete black-box optimization challenges frequently encountered in practice.

CCS CONCEPTS

• **Computing methodologies** → **Randomized search**; *Discrete space search*;

KEYWORDS

discrete black-box optimization, combinatorial optimization, benchmarking, algorithm evaluation, algorithm comparison

ACM Reference Format:
Ofer M. Shir, Carola Doerr, and Thomas Bäck. 2018. Compiling a Benchmarking Test-Suite for Combinatorial Black-Box Optimization: A Position Paper. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205651.3208251

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

ACM ISBN 978-1-4503-5764-7/18/07…$15.00
https://doi.org/10.1145/3205651.3208251

1 INTRODUCTION

A subclass of global optimization problems is denoted as Combinatorial Optimization (CO) — referring to problems formulated either by integral decision variables (possibly mixed-integer, where continuous variables are incorporated as well), or by discrete structures, e.g., graphs [40]. Formally, a CO problem

$$\mathcal{P} := \left(\mathcal{S},\, f : \mathcal{S} \to \mathbb{R}^{+}\right),$$

is defined by a finite set $\mathcal{S}$ with an objective function f assigning a non-negative value to any of its elements $s \in \mathcal{S}$. An optimization process is defined as the search over $\mathcal{S}$ with the explicit goal of locating an element s^* with the *minimal* f-value.

CO constitutes a broad, well-studied field that has been addressed by multiple scientific sub-communities of Computer Science (CS) and Applied Mathematics for at least six decades. It has been approached by means of formal algorithms and Mathematical Programming (MP) [13] (often branded as Operations Research (OR), yet strongly rooted at Theoretical CS [38]), and has simultaneously been treated by a wide range of dedicated heuristics (frequently under the label of Soft Computing [33, 36, 43]). The domain of Constraints Satisfaction Problems (CSPs) [25] covers combinatorial problems which do not formulate an objective function and are predominantly concerned with satisfying the set of imposed constraints. Such problems are associated with models for which *obtaining feasible solutions already sets a hard challenge*, or otherwise to models for which the objective function is of secondary importance. CSPs have practical implications in Functional Verification, and are commonly addressed in OR by so-called Constraints Programming (CP) – which is treated by an independent branch of dedicated techniques for constraints satisfaction. In the "formal algorithms" end, OR of either MP or CP is practically carried out by two sub-communities, employing different classes of algorithms. In the current paper, however, unless specified otherwise, **CO-problems are referred to as an umbrella term**, encompassing all types of discrete optimization problems, as well as CSPs.

CO problems arise almost everywhere in theoretical and practical optimization, while a large volume of so-called *solvers* is constantly under development. An important subclass of CO problem-solvers is *black-box optimization heuristics*, which operate in a trial-and-error fashion, by evaluating candidate solutions and using the function values to evolve the strategy upon which the next *search points* are drawn. This approach is in sharp contrast to classical *white-box optimizers*, which construct solutions *bottom-up*, by exploiting the explicit problem-structure and the available instance-data.[1] Problem-specific white-box approaches are often superior to

[1] In practice, mixed forms of these approaches exist, often subsumed as *gray-box optimizers*; see, e.g., [45].

general-purpose black-box approaches – when comparisons over specific problems are available (cf. discussion below). At the same time, the practical relevance of black-box optimizers is rooted in the fact that an explicit problem structure is not a prerequisite for their operation – i.e., they do not require that the optimization problem is formulated via an explicit map $f : \mathcal{S} \rightarrow \mathbb{R}^{+}$.

The current focus of the targeted benchmarking suite is set on algorithms adhering to the black-box approach. Within this class, Evolutionary Algorithms (EAs) [7], and more broadly randomized iterative black-box optimization heuristics, commonly referred to as Randomized Search Heuristics (RSHs) [6], constitute popular black-box solvers. They have been concerned with CO-problems since their early development because of their natural fit to address them. Importantly, since they employ a black-box approach by default, and since CSPs may be transformed into optimization problems, **the aforementioned distinction between MP to CP has no equivalence in the RSHs/EAs communities**. In other words, RSHs treat all CO problems, including CSPs, indistinguishably. At the same time, loosely speaking, there are no common grounds for performance comparisons between RSHs to MP/CP solvers when targeting similar CO problems. Traditionally, this has been explained by the dramatic differences in the models' scales when comparing the two approaches: MP/CP (representing white-box approaches) are known to handle millions of decision variables and constraints, whereas RSHs (representing black-box approaches) operate well over thousands of variables. On this note, however, it should be kept in mind that MP-solvers occasionally "*hit a wall*" on CO problems, and are sometimes outperformed by so-called Hybrid Metaheuristics [9]. Such a scenario is mentioned herein in order to shed light on this borderline between the two scientific communities. As an example, we present an MP formulation of the Multidimensional Knapsack Problem (MKP), utilizing n binary decision variables x_i for items' selection (relying on instance-specific data: m knapsacks' capacities c_k, the profits of the n items, p_i, and the resources consumptions $r_{i,k}$ of items per knapsacks):

$$\begin{aligned} &\textbf{[MKP]} \text{ maximize } \sum_{i=1}^{n} p_i \cdot x_i \\ &\text{subject to:} \\ &\sum_{i=1}^{n} r_{i,k} x_i \geq c_k \;\; \forall k \in 1 \ldots m \\ &x_i \in \{0,1\} \;\; \forall i \in 1 \ldots n \end{aligned} \tag{1}$$

This problem may seem straightforward for an MP-solver, but is in fact highly-challenging and hard to be accurately treated by modern commercial solvers without hybridization (i.e., introduction of black-box techniques) [9].

Finally, in order to target an effective suite compilation, we restrict by choice the optimization scope to **single-objective, noise-free, and static problems**. We suggest to dedicate separate threads for targeting the benchmarking of multi-objective, noisy, and/or dynamic problems.

1.1 The Role of Benchmarking

Benchmarking aims at supporting practitioners in choosing the "best" algorithmic technique and its fittest instantiation for the problem at hand through a systematic empirical investigation and comparison amongst competing techniques in light of the following more detailed questions:

- How does the algorithm perform on different **classes of problems** (e.g., network/flow optimization or scheduling, to mention a couple) and how does its performance compare to that of other approaches?
- Which **problem features** possess the strongest impact on the accuracy and/or the convergence speed, and how this dependency may be quantified? Examples for relevant features are the *modality* of a problem (proportion of local and global optima), its *separability* (grade of inter-dependencies between the decision variables), the degree of *constraints*, and its *monotonicity*.
- How does the performance **scale with increasing problem complexity** (i.e., dimensionality, cardinality of categories per a decision variable, etc.)?
- How **sensitive** is a given algorithm with respect to small changes in the problem instance or the algorithmic components?

For theoreticians, benchmarking can be an essential tool for the enhancements of mathematically-derived ideas into techniques being broadly applicable in practical optimization. In addition, empirical performance comparisons constitute an important source for formulating new research questions.

Designing sound benchmarking is a **very demanding task** that requires a very good knowledge of the research literature, existing techniques, and applications. In addition, systematic benchmarking typically necessitates original research, to design test problems whose features are controllable and scalable – elements that usually do not characterize real-world optimization tasks. Finally, significant implementation efforts are needed to realize an easily-accessible, well-designed, and well-documented testbed. All this is not feasible unless a deliberate approach takes place.

1.2 Existing Work and State-of-the-Art

Benchmarking RSHs on CO-problems is an open issue for debate. While white-box algorithms routinely compete within the OR community on an established and increasingly growing set of problems' libraries (see, e.g., the Mixed-Integer Programming (MIP) library [1] or the CSP library [2]), black-box algorithms do not enjoy equivalent settings. There exist isolated examples of benchmarking environments for black-box CO, but they are typically designed by practitioners and focus on specific sets of challenging problems. As a consequence, benchmarking an algorithm on such problems may become irrelevant to theoreticians, since the problems' complexity would render analytical attempts infeasible. We therefore believe that an effort to accommodate a balanced compilation of problems is much needed, to serve the large spectrum of black-box algorithms' researchers.

Our aim is thus to design a benchmarking environment that constitutes an open-minded CS framework, facilitating a broad range of challenges frequently encountered in practice. We argue that it should include relevant benchmarks of the OR community – while at the same time subsume problems that help understand

algorithms' behavior on archetypal "*toy problems*", whose underlying landscapes are well understood, and mathematical analyses may thus turn fruitful. If successful, this compilation effort is expected to establish an effective benchmarking tool, useful both for practitioners and theoreticians, with potential impact beyond the scope of standard black-box optimization (that is, with outcome potentially relevant to the OR community).

The remainder of this paper is organized as follows. In Section 2 we outline the existing challenges and the apparent requirements for addressing them. Section 3 includes an overview of relevant problem classes with their representative problem-instances – on which we raise certain design questions that should be debated prior to a compilation process. We note our answers to the prescribed questions in Section 4, and then devise a preliminary set of test-problems for the suite in Section 5. Finally, we summarize this position paper in Section 6.

2 CHALLENGES AND REQUIREMENTS

We are concerned with the design of a benchmark environment for discrete black-box optimization, and more concretely as previously mentioned, with the black-box single-objective optimization of noise-free and static CO problems. Our task is therefore to identify a set $\{(\mathcal{S}_i, F_i : \mathcal{S}_i \to \mathbb{R}) \mid i \in [N]\}$ of CO problems for which we want to compare algorithms' performances.

The black-box setting requires that the algorithms undergoing benchmarking hold basic information on the search space $\mathcal{S}_i$ per a given problem. They iteratively propose candidate solutions

$$x^{(1)}, \ldots, x^{(s)} \in \mathcal{S}_i,$$

for which the objective function values,

$$F_i(x^{(1)}), \ldots, F_i(x^{(s)}) \in \mathbb{R},$$

are evaluated without revealing any other information about the problem $(\mathcal{S}_i, F_i)$.

2.1 Previously Proposed Guidelines for Black-Box Benchmarking

Over two decades ago, Whitley et al. [46] criticized the commonly tested artificial landscapes in the EC community at that time, and offered general guidelines for constructing test problems. Those problems were mostly continuous parameter optimization in their nature, but we consider the statements valid for the current scope. We summarize those guidelines in light of the current context, i.e., an environment for discrete black-box optimization benchmark:

(A) *"Test suites should contain problems that are resistant to hill-climbers"*. Hill-climbing strategies, including greedy local search, are typically faster than EAs, when they are successful. Hence, it is justified to test EAs on landscapes which cannot be easily hill-climbed.
(B) *"Test suites should contain problems that are non-linear, non-separable, and non-symmetric"*.
(C) *"Test suites should contain scalable functions"*. The dimensionality of the search space is an important issue, and thus should be tested accordingly.
(D) *"Test suites should contain problems with scalable evaluation cost"*. The cost of some evaluation functions grows as a function of the search space dimensionality. This typically characterizes real-world problems, and should be considered.
(E) *"Test problems should have a canonical form"*. This demand is relevant to encoding-based algorithms, such as GAs.

We hereby adopt those recommendations, and intend to pose a list of additional design questions in Section 3 that need to be addressed prior to the benchmark design process takes place.

2.2 The Continuous Counterpart: BBOB

The continuous Black-box Optimization Benchmarking suite BBOB for real-valued search [29] constitutes an established testing framework for evaluating performance of continuous optimizers. The noise-free suite encompasses 24 functions classified as follows:

(1) Separable functions
(2) Functions with low or moderate conditioning
(3) Functions with high conditioning and unimodal
(4) Multi-modal functions with adequate global structure
(5) Multi-modal functions with weak global structure

We do not formally define the aforementioned terms nor do we explain the rationale in this classification, due to scope and space limitations, and refer the interested reader to the documentation [30]. These classes, nevertheless, may inspire us when searching for typical equivalent features to be covered by the discrete black-box optimization benchmark. Another conclusion we draw from this classification is the option of aggregating performance over a class of functions, a question that we will return to in Section 3.4.

3 NINE FUNDAMENTAL DESIGN QUESTIONS

This section aims at identifying the desirable properties of CO benchmarking problems. In this identification process we pose relevant questions and examine several specific CO-problems that feature some of these properties.

Next, we begin by presenting the *archetypical* Traveling Salesman Problem (TSP), which would illustrate multiple design issues to be discussed. The TSP is posed as finding a Hamilton circuit of minimal total cost. Explicitly, given a directed graph G, with a vertex set $V = \{1, \ldots, |V|\}$ and an edge set $E = \{\langle i, j\rangle\}$, each edge is associated with cost information $c_{ij} \in \mathbb{R}^+$. Two TSP formulations are provided herein:

$$\begin{aligned}
&\textbf{[TSP-ILP]} \text{ minimize } \sum_{\langle i,j\rangle \in E} c_{ij} \cdot x_{ij} \\
&\text{subject to:} \\
&\sum_{j \in V} x_{ij} = 1 \ \forall i \in V \\
&\sum_{i \in V} x_{ij} = 1 \ \forall j \in V \\
&u_i - u_j + 1 \leq (|V| - 1)\left(1 - x_{ij}\right) \ \forall i, j \in \{1, \ldots, |V|\} \\
&|V| \geq u_i \geq 2 \ \forall i \in \{2, 3, \ldots, |V|\} \\
&x_{ij} \in \{0, 1\} \ \forall i, j \in V
\end{aligned} \tag{2}$$

This ILP, known as the Miller-Tucker-Zemlin formulation [40], utilizes n^2 binary assignment variables x_{ij} in addition to n integers u_i that play the role of inner-circles eliminators (replacing, otherwise,

an exponential number of subtour elimination constraints [40]).

$$\boxed{\begin{array}{l}\textbf{[TSP-perm]}\ \text{minimize}\ \sum_{i=0}^{n-1} c_{\pi(i),\pi((i+1)_{\mathrm{mod}n})} \\ \text{subject to:} \\ \pi \in P_\pi^{(n)}\end{array}} \tag{3}$$

In the latter, $P_\pi^{(n)}$ denotes the set of permutations of length n. This permutation formulation does not qualify as an MP, since it violates the canonical form. Nevertheless, it is often useful for RSHs which utilize dedicated permutation variation operators. These two alternative formulations would raise the problem representation question in what follows.

3.1 Problem Representation

An important design question in CO is the formulation of the original problem as a function $f : \mathcal{S} \to \mathbb{R}$. It is well known that care should be given to this step, since the representation of candidate solutions can have a decisive influence on the complexity of the problem, as well as on the performance of the algorithms [21]. For some CO problems, a canonical problem representation exists, but for a considerable number of problems this does not hold. At the same time, since black-box optimization algorithms are often tailored to a specific search-space at hand, the problem representations must be well-defined *in advance*.

Among the most common representations are the n-dimensional cube of alphabet size r, $\{0,...,r-1\}^n$, and the set $P_\pi^{(n)}$ of permutations over the set $[n] := \{1,\ldots,n\}$ – which were both illustrated within the TSP formulations above.

> **[Q1]** Should a problem representation be dictated per each benchmarking problem?

3.2 Instance-Based Problems

Importantly, each TSP-instance constitutes an independent CO problem, whose properties are essentially dictated by the graph G and the associated cost information c_{ij}. Therefore, TSP problems are not scalable.

Other instance-based CO-problems are addressed by RSHs as a common practice, among which Kauffman's NK-landscapes [5, 34] and the Quadratic Assignment Problem [11] are worth noting in this context.

> **[Q2]** Should instance-based problems be incorporated within the test-suite?

3.3 Invariant Problem Formulation

An important feature of the BBOB suite is a set of invariances that the problems are required to adhere. That is, BBOB considers as a *problem* not a single pair $(S, f : \mathbb{R}^n \to \mathbb{R}^+)$ of search space and objective function, but rather a whole collection of problems with identical, but rotated, translated, and scaled *fitness landscapes*. More formally, for a set Φ of automorphisms $\phi : \mathbb{R}^n \to \mathbb{R}^n$ and a set $\mathcal{T}$ of automorphisms $\tau : \mathbb{R} \to \mathbb{R}$, the problem $(S, f : \mathbb{R}^n \to \mathbb{R}^+)$ is identified with the set $\{(S, \tau \circ f \circ \phi : \mathbb{R}^n \to \mathbb{R}^+) \mid \phi \in \Phi, \tau \in \mathcal{T}\}$.

> **[Q3]** Should the benchmarking framework cover the invariance aspect, and implicitly favor algorithms that are invariant with respect to problem representation, and if so, which invariances should be respected?

3.4 Performance Evaluation

In black-box optimization two competing types of performance indicators exist: elapsed CPU time and the number of function evaluations (typically coined *query complexity* in the broader CS-literature). While CPU time is arguably the measure that predominantly matters in practical applications, it has the substantial drawback that it is hardware and software specific, and thus hard to generalize. Function evaluations, in contrast, have the advantage of being a *universal* performance measure, independent of the implementation and the hardware that it is executed on.

> **[Q4]** Which primary performance evaluation measure should be adopted?

Similarly as BBOB defines the five subclasses presented in Section 2.2, it might make sense to categorize the benchmark problems, and to compute some aggregated performance measures for these problem clusters as a standard output of the testbed.

> **[Q5]** Should performance aggregation be conducted?

Most benchmarking environments allow for a comparison of algorithms' performances, but do not necessarily facilitate an in-depth analysis of the optimization process beyond the evolution of quality indicators such as the best function value identified so far. From an analytical perspective, however, additional information, such as the evolution of dynamic parameter values, diversity measures, etc. might be desirable.

> **[Q6]** Should the test-suite also facilitate *algorithm profiling* in the sense of algorithmic analysis beyond pure performance evaluation?

3.5 Simple Benchmark Problems Admitting Runtime Analysis

Amongst a number of highly challenging black-box optimization problems, BBOB also encompasses a few rather simple problems like the Sphere function $(\mathbb{R}^n, F_1(x) := \sum_{i=1}^n x_i^2)$ as well as a few additional *unconstrained convex* problems.

The mathematical property of convexity in continuous domains naturally does not hold in CO. Yet, in terms of problem hardness, we consider the equivalent to convex problems [10] as *simple problems admitting runtime analysis*. A well-known representative of this class is the "OneMax" problem [3], also known as the "Counting Ones" problem. OneMax is considered to be one of the simplest non-trivial CO problem. We consider its formulation as a minimization problem, which is named the "Hamming Distance" (HD) problem: Let a binary string of length n, $x := (x_1,\ldots,x_n)$ with $x_i \in \{0,1\}$, form the decision variables. The objective function to be minimized is simply the summation over those variables:

$$\boxed{\begin{array}{l}\textbf{[HD]}\ \text{minimize}\ \sum_{i=1}^{n} x_i \\ \text{subject to:} \\ x_i \in \{0,1\}\ \ \forall i \in \{1,\ldots,n\}\end{array}} \tag{4}$$

The HD problem enjoys a number of available, quite precise, runtime results per a broad class of different RSHs, including the (1+1) EA [31], population-based EAs [26], GAs using recombination [14], and algorithms with dynamic parameter choices [15, 19]. It is also one of the few problems for which the complexity is very well understood [20, 24].

> **[Q7]** Should the test-suite encompass simple CO problems admitting runtime analysis?

3.6 Facing Operations Research

A large volume of commonly-addressed combinatorial optimization problems may be formulated as integer problems that are linear both in the objective function and the imposed constraints. This formulation yields a so-called Integer or Mixed-Integer Linear Program ([M]ILP), that consequently may efficiently be treated by adecquate solvers [8]. Progressive applied reseach throughout several decades in OR has obtained a large variety of forceful MILP solvers – either commercial (e.g., IBM's ILOG CPLEX [32]) or non-commercial. Such solvers typically tackle MILP problems very effectively, featuring a smaller order of performance magnitude in terms of CPU time when compared to RSHs.[2] Clearly, drawing such a comparison is problematic, due to the fact that MILP-solvers enjoy a white-box perspective, while RSHs are subject to either gray- or black-box perspectives. But even so, a legitimate question could be raised within this context on benchmarking:

> **[Q8]** Should RSHs' performance be evaluated on problems that are known to be effectively treated by MP-solvers in practice?

3.7 MP versus CP

Since MP and CP are clearly distinguished within the OR community, we pose a general question on the distinction between optimization to CSPs in the black-box perspective:

> **[Q9]** Should RSHs' performance be indistinguishably evaluated on CSPs as well? That is, should a distinction between standard optimization to CSPs be avoided in the black-box perspective?

4 PROPOSED POSITION AND PROBLEMS

We offer our perspective by answering the preceding questions:

(1) A certain problem formulation dictates the test-case, and should be set fixed (i.e., TSP-ILP and TSP-perm are essentially two different search-problems). We suggest to restrict the benchmark suite to functions $f : \{0, \ldots, r-1\}^n \to \mathbb{R}^+$. This problem representation allows for a large flexibility and subsumes a large fraction of classical CO problems. It is also the arguably most common representation used in the EC literature for CO problems.

(2) We suggest that (i) preference be given to problems that are not instance-based, and that (ii) instance-based problems be included only to the extent needed to understand performance behavior that cannot be otherwise observed over instance-free CO problems.
In light of existing benchmarking suites, this claim may appear surprising at first. We base our position on the following argumentation: Firstly, it is well known that the complexity of an instance-based CO problem can differ substantially between two different instances. Depending on the particular instance, the problem can be relatively easy to solve (i.e., polynomial or better time complexity), or very difficult (i.e., exponential time complexity). For a fair comparison, all algorithms would therefore need to be tested on the same set of instances, which carries the risk of *overfitting*. Secondly, problem instances require specific descriptions and are therefore, in general, not arbitrarily scalable with respect to their dimensionality.[3] Finally, we wish to avoid future scenarios in which the competition of black-box approaches over an instance-based CO problem splits into an independent research activity – as evidently occurred for the TSP and SAT sub-communities. We argue that such specialized empirical investigations of particular benchmark problems should not overcast the goal of a broad-scope RSHs' performance comparison.
Finally, should instance-based problems be included, we suggest to give priority to problems that are scalable, in the sense that the fitness landscapes adhere similar patterns. We also recommend to consider each selected problem-instance as a separate problem.

(3) Yes, the suite should account for problem invariances. As no selection of CO problems can encompass *all* the relevant features of real-world optimization challenges, every benchmark suite bears the risk of focusing the research community on a too small, not too representative set of problems. As a consequence, *overfitting* may occur. We believe that conforming to certain, natural invariances reduces the risk of such overfitting. This belief is seemingly backed up by the BBOB development as witnessed by its participants/users. We present in Section 5.1 a number of invariances that we consider meaningful in the context of minimizing functions of the type $f : \{0, \ldots, r-1\}^n \to \mathbb{R}^+$.

(4) In line with the arguments presented in [27], we advocate the use of function evaluations as the main performance measure. The benchmarking system, however, should also record the elapsed CPU times so that they can be accounted for if needed. If scenarios of evident large differences in CPU times occur, they should be reported in the respective documentation. When run on the same machine, CPU times can be used as a secondary performance indicator.

(5) Yes, we support performance aggregation. We do not support, however, aggregation over problem dimension, since, as already argued in [28], problem dimension should be used for algorithm selection. Note also that even in the black-box setting the algorithm needs to know the search space that

[2] This comparative statement is noted with a reservation, since some MILP solvers actually employ evolutionary operators in their heuristic components, such as CPLEX's *polish* subroutine [42].

[3] Arguably, a large cluster of problem instances could be viewed altogether as a scalable problem, e.g., Random-3-SAT [12]; this idea however is not put forward herein.

it is asked to operate upon. For the same reason, we also object to aggregation over the alphabet size r.

(6) Yes, the benchmark suite should allow for algorithm profiling. A mere algorithm comparison/competition tool bears the risk of being less appealing to the EC community, as one of the main research ambitions is to understand the working principles behind RSHs, with the ultimate goal of enabling a more effective algorithm design. However, the additional data to be tracked should not result in a significant slowdown of the benchmarking activities. We therefore suggest to implement it as an optional feature. Standard observables may include population sizes, mutation/crossover rates and probabilities, selective pressure, and the diversity of populations and/or objective values.

(7) Yes, selected analyzable functions, such as HD, should be incorporated into the test-suite, also to promote intensified discussions between theory-driven to practice-oriented scholars. To foster such an exchange, it is important to carefully argue which aspect of black-box optimization each of these selected problems covers.

(8) Yes, problems that are (easily) solvable by MILP-solvers could be incorporated into the test-suite, as long as they are not instance-based. Notably, a preference should be given to more challenging problems.

(9) Yes, RSHs' performance should be indistinguishably evaluated on CSPs as well, since in the black-box perspective they are merely CO-problems.

5 EXAMPLE BENCHMARK PROBLEMS

We collect in this section a few benchmark problems that adhere to the answers given in the previous section. Although we classify them according to their origin, we recall that this classification might be less relevant in the black-box setting. Put differently, the suggested performance aggregation does not necessarily need to follow this historic development.

Before we state the example problems, we first discuss the invariances that the problems should obey, in a similar fashion as the 24 BBOB benchmark functions respect invariance with respect to rotations, scaling, and translation.

5.1 Problem Invariances

We have argued that we restrict our attention to problems of the type $(\{0,\dots,r-1\}^n, f : \{0,\dots,r-1\}^n \to \mathbb{R}^+)$. Already for $r = 2$ this class includes an important number of classical optimization problems. We start our discussion with a set of invariances that we consider most relevant in the context of such *pseudo-Boolean* optimization problems. To this end, we recall that the hyper-cube is equipped with a natural distance measure, the Hamming distance H, which assigns to each pair of length-n bit strings the number of bits in which they differ. Composing a pseudo-Boolean function with an automorphism of the hyper-cube that respects this distance measure (so-called *Hamming automorphisms*) results in a function of the same, but rotated fitness landscape. We suggest to include this important set of automorphisms in the set of invariances that the problems should adhere. More concretely, we require to treat as invariances of the same problem $f : \{0,1\}^n \to \mathbb{R}^+$ all functions that can be expressed as

$$f_{z,\sigma} : \{0,1\}^n \to \mathbb{R}, x \mapsto f(\sigma(x \oplus z)) \tag{5}$$

where the string $z \in \{0,1\}^n$ determines the shift and the permutation $\sigma \in P_\pi^{(n)}$ the rotation. Note here that we abuse notation and the define as $\sigma(x)$ the re-arranged string $\sigma(x) := (x_{\sigma(1)} \dots x_{\sigma(n)})$.

It has been argued in [35] that many important RSH respect problem invariance with respect to Hamming automorphisms. In black-box optimization, the invariance under Hamming automorphisms is intensively studied in the so-called *unbiased black-box complexity model,* cf. [23] for a survey.

In addition to the invariance with respect to Hamming automorphisms, we suggest to also conform invariance with respect to *translation*

$$\{f + r : \{0,1\}^n \to \mathbb{R}, x \mapsto f(x) + r \mid r \in \mathbb{R}\} \tag{6}$$

and with respect to *scaling*

$$\{cf : \{0,1\}^n \to \mathbb{R}, x \mapsto c \cdot f(x) \mid c \in \mathbb{R}\} \tag{7}$$

For CO problems represented as search over the n-dimensional cube $\{0,\dots,r-1\}^n$ of alphabet size $r \geq 2$, no natural equivalent to Hamming distance exists. In contrast, there are a number of different distance measures that are meaningful in different contexts, even when restricting to a single optimization problem; cf. [17] for the example of three different multi-valued OneMax problems. The three distance measures discussed there are the classic L_1 distance $d_1(x,z) := \sum_{i=1}^n |x_i - z_i|$, the ring distance $d_r(x,z) := \min\{x_i - (z_i - r), |x_i - z_i|, (z_i + r) - x_i\}$, and the indicator distance $d_i(x,z) := n - |\{j \in [n] \mid x_j = z_j\}|$. Other distance measures exist. We suggest a problem-specific discussion of the metric that is used to define the automorphisms that the problem should adhere. Regardless of the chosen set of search-space invariances, we also suggest to respect translation- and scaling-invariance.

5.2 Analyzable Functions

We introduce some of the classic benchmark functions that allow for a rigorous runtime analysis. Following our recommendation with respect to Q7, each one of them highlights some important problem features that are meaningful to showcase. We formulate these problems as pseudo-Boolean functions $f : \{0,1\}^n \to \mathbb{R}$, and note that multiple ways exist to generalize these to multi-valued analogs. As indicated in the previous subsection, such a generalization requires an intensified discussion of the underlying metric—a question that we would like to put aside for the current paper.

OneMax (HD). As mentioned in Section 3.5, the OneMax function is *the* archetypal problem for a simple hill-climbing exercise. Despite being the best-studied problem in the theory of discrete black-box optimization,[4] a number of questions concerning the performance of classical EAs remain unsolved, among them the innocently looking problem of determining the optimal adaptive mutation rate for the (1+1) EA [20] and the usefulness of crossover [15].

BinaryValue. While greedy local search algorithms behave identically on all strictly monotonous functions, performance gaps between different linear functions $f : \{0,1\}^n \to \mathbb{R}, x \mapsto \sum_{i=1}^n w_i x_i$

[4] Published results date back to Erdős and Rényi [24]; cf. [18] for a discussion on this and related works from discrete mathematics in the context of black-box optimization.

can be observed already for the $(1+\lambda)$ EA with standard bit mutation [22]. A particular linear function that has received much attention in the runtime analysis community is BinaryValue BV : $\{0,1\}^n \to \mathbb{R}, x \mapsto \sum_{i=1}^{n} 2^{n-i} x_i$. In contrast to the OneMax function, where the correlation between the function values and the distance to the optimal solution is perfect, the exponential decay of the bit weights causes a much different fitness landscape for BinaryValue. Note, for example, that the BV-value of $(1,0,\ldots,0)$ is strictly larger than that of $(0,1,\ldots,1)$, despite being much closer to the optimal solution $(0,\ldots,0)$.

Intermediate Linear Functions. OneMax with its constant bit weights and BinaryValue with the exponentially decaying weights can be seen as extreme examples for linear functions. Intermediate linear functions, e.g., $f : \{0,1\}^n \to \mathbb{R}, x \mapsto \sum_i i x_i$ may be worthwile to benchmark, in order to detect how sensitive an algorithm is with respect to the bit weights.

LeadingOnes. Another very well studied function in the theory of pseudo-Boolean black-box optimization is LeadingOnes, a function for counting the number of initial ones in a bit string. This problem is an interesting object in theoretical investigations, because it is a non-separable problem with an easy to understand structure. Optimal algorithms for this problem are surprisingly complex [4]; they achieve a running time of $\Theta(n \log\log n)$, whereas most RSHs have running times quadratic in n, and problem-tailored binary-search-based solvers exhibit $O(n \log n)$ running times.

Jump Functions. For a non-negative integer ℓ, the function jump_ℓ is derived from the OneMax function, denoted here as OM, by "blanking out" any information within the strict ℓ-neighborhood of the optimum and its bitwise complement (i.e., by setting a function value of 0). That is, for all $x \in \{0,1\}^n$ we have

$$\text{jump}_\ell(x) := \begin{cases} n, & \text{if } \text{OM}(x) = n, \\ \text{OM}(x), & \text{if } \ell \le \text{OM}(x) \le n - \ell, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Multiple similar variants of this function exist, and they differ most notably in the size of the search space "between" local and global optima, and in the values assigned to these points, cf. [16, Section 2] for a discussion. What is common to any of these functions is that they all expose the algorithms' ability to excavate the gap between the local optima and the global optimum. Variants of this function with consecutive or shifted objective function gaps may be worthwhile to study. Note, however, that the former introduces a *multimodal* problem, which should be treated as a separate case.

RoyalRoad Functions. One way to introduce "consecutive" plateaus is exhibited in RoyalRoad functions. These functions were originally designed to showcase a situation in which the use of recombination operators can be beneficial over purely mutation-based search. For a given *block size* k, the function partitions a string x into n/k blocks of size k and counts the number of blocks in which all bits are set correctly. Thus, formally, $\text{RR}_k(x) := |\{i \in [1..\lceil n/k \rceil] \mid \forall j \in [k] : x_{(i-1)k+j} = 1\}|$.

The discussion above shows that already these simple problems admitting runtime analysis define a large set of parametrized problems. Care has to be taken when deciding which linear functions, which jump functions, which royal road functions, and which multimodal variants thereof to include in the benchmark suite.

5.3 CSP

The n-queens problem (NQP) is defined as the task to place n queens on an $n \times n$ chessboard in such a way that they cannot *capture* each other. NQP formally constitutes a CSP:

$$\begin{array}{l} \textbf{[NQP-CSP]} \text{ satisfy:} \\ \displaystyle\sum_{i,j} x_{ij} = n \\ \displaystyle\sum_{i,j \mid j-i:=k} x_{ij} \le 1 \; k \in \{-n+2, -n+3, \ldots, n-3, n-2\} \\ \displaystyle\sum_{i,j \mid i+j:=\ell} x_{ij} \le 1 \; \ell \in \{2,3,\ldots,2n-3,2n-2\} \\ x_{ij} \in \{0,1\} \;\; \forall i,j \in \{1,\ldots,n\} \end{array} \tag{9}$$

This formulation utilizes n^2 binary decision variables x_{ij}, which are associated with the chessboard's coordinates, having an origin $(1,1)$ at the top-left corner. Setting a binary to 1 implies a single queen assignment in that cell. This CSP formulation devises the demand to place n queens as the first constraint, followed by two sets of constraints eliminating queens' *mutual threats* at the increasing-diagonal (using the dummy indexing k) and decreasing-diagonal (using the dummy indexing ℓ). As the majority of CSPs, it can be reformulated as a canonical optimization problem (transforming into an ILP by, e.g., maximizing the summation over $\sum_{i,j} x_{ij}$ while dropping the first constraint). Also, it should be noted that a *permutation formulation* also exists for this problem, and is sometimes attractive for RSHs, as mentioned for the **TSP-perm** formulation. Due to chessboard symmetries, NQP possesses multiplicity of optimal solutions. Its attractiveness, however, lies in its hardness.

5.4 Non-Linear Hard Problems

This class of problems is meant to capture challenging CO problems that do not fall under the umbrella of OR. They naturally fit to be treated by RSHs.

Obtaining binary sequences possessing a high merit factor, also known as the Low-Autocorrelation Binary Sequence (LABS) problem, constitutes a grand combinatorial challenge with practical applications in radar engineering and measurements [41, 44] as well as several open questions concerning its mathematical nature. Given a sequence of length n, $S := (s_1,\ldots,s_n)$ with $s_i = \pm 1$, the merit factor is proportional to the reciprocal of the sequence's autocorrelations. The LABS optimization problem is defined as searching over the sequence space to yield the maximal merit factor:

$$\begin{array}{l} \textbf{[LABS]} \text{ maximize } \dfrac{n^2}{2E(S)} \\ \text{subject to:} \\ E(S) := \displaystyle\sum_{k=1}^{n-1} \left(\sum_{i=1}^{n-k} s_i \cdot s_{i+k} \right)^2 \\ s_i \in \{-1,+1\} \;\; \forall i \in \{1 \ldots n\} \end{array} \tag{10}$$

While its current formulation does not adhere to the standard form presented earlier, $\{0,\ldots,r-1\}^n \to \mathbb{R}^+$, it could be reformulated to follow it. This hard, non-linear problem has been studied over several decades (see, e.g., [37, 39]), where the only way to obtain exact solutions remains exhaustive search.

6 DISCUSSION AND SUMMARY

This position paper proposed an explanatory overview of the existing dilemmas and difficulties involving the formulation of a CO benchmarking environment for black-box algorithms. It aimed at offering a broad CS-perspective by accounting for the OR community and its contribution on the one hand, and by examining existing benchmarks for black-box algorithms in numerical optimization (BBOB) on the other hand. By posing 9 critical questions while investigating related issues, it is our belief that we set the ground for a constructive process of compiling such an environment.

In our opinion, the **human factor plays a crucial role in such processes**. Generally speaking, formulation of a test-suite may involve three types of scholars: theoreticians, algorithms' designers, and practitioners. While theoreticians naturally favor analyzable functions, algorithms' engineers may possess preferences toward families of functions that are successfully treated by their designs, and practitioners may have the best insights into which functions most accurately represent real-world problems, and their preferences would then be biased accordingly. These human tendencies should be acknowledged and accounted for. Importantly, we state that a proper balance should be made amongst those three parties in order to accomplish the ultimate goal of compiling an effective test-suite that is appealing and meaningful to a broad audience, and offers novel insights into the working principles of discrete black-box optimization techniques.

ACKNOWLEDGEMENTS

This publication is based upon work from COST Action CA15140, "*Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice*".

REFERENCES

[1] MIPLIB: the Mixed Integer Programming LIBrary. http://miplib.zib.de/, 2017.
[2] CSPLib: A problem library for constraints. http://csplib.org/, 2018.
[3] D. H. Ackley. *A Connectionist Machine for Genetic Hillclimbing*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.
[4] P. Afshani, M. Agrawal, B. Doerr, C. Doerr, K. G. Larsen, and K. Mehlhorn. The query complexity of finding a hidden permutation. In *Space-Efficient Data Structures, Streams, and Algorithms - Papers in Honor of J. Ian Munro on the Occasion of His 66th Birthday*, volume 8066 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 2013.
[5] H. Aguirre and K. Tanaka. *Performance and Scalability of Genetic Algorithms on NK-Landscapes*, pages 37–52. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
[6] A. Auger and B. Doerr. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2011.
[7] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, NY, USA, 1996.
[8] E. R. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. MIP: Theory and practice — closing the gap. In M. J. D. Powell and S. Scholtes, editors, *System Modelling and Optimization*, pages 19–49, Boston, MA, 2000. Springer US.
[9] C. Blum and G. R. Raidl. *Hybrid Metaheuristics: Powerful Tools for Optimization*. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer International Publishing, 2016.
[10] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, 2004.
[11] R. E. Burkard. *Quadratic Assignment Problems*, pages 2741–2814. Springer New York, New York, NY, 2013.
[12] C. Coarfa, D. D. Demopoulos, A. San Miguel Aguirre, D. Subramanian, and M. Y. Vardi. Random 3-SAT: The plot thickens. *Constraints*, 8(3):243–261, July 2003.
[13] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley and Sons, New York, NY, USA, 2011.
[14] D. Corus and P. S. Oliveto. Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms. *CoRR*, abs/1708.01571, 2017.
[15] B. Doerr and C. Doerr. Optimal static and self-adjusting parameter choices for the $(1 + (\lambda, \lambda))$ genetic algorithm. *Algorithmica*, 80:1658–1709, 2018.
[16] B. Doerr, C. Doerr, and T. Kötzing. Unbiased black-box complexities of jump functions. *Evolutionary Computation*, 23:641–670, 2015.
[17] B. Doerr, C. Doerr, and T. Kötzing. Static and self-adjusting mutation strengths for multi-valued decision variables. *Algorithmica*, 80:1732–1768, 2018.
[18] B. Doerr, C. Doerr, R. Spöhel, and H. Thomas. Playing Mastermind with many colors. *Journal of the ACM*, 63:42:1–42:23, 2016.
[19] B. Doerr, C. Doerr, and J. Yang. k-bit mutation with self-adjusting k outperforms standard bit mutation. In *Proc. of Parallel Problem Solving from Nature (PPSN'16)*, volume 9921 of *Lecture Notes in Computer Science*, pages 824–834. Springer, 2016.
[20] B. Doerr, C. Doerr, and J. Yang. Optimal parameter choices via precise black-box analysis. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'16)*, pages 1123–1130. ACM, 2016.
[21] B. Doerr and D. Johannsen. Edge-based representation beats vertex-based representation in shortest path problems. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'10)*, pages 758–766. ACM, 2010.
[22] B. Doerr and M. Künnemann. Optimizing linear functions with the $(1+\lambda)$ evolutionary algorithm—different asymptotic runtimes for different instances. *Theoretical Computer Science*, 561:3–23, 2015.
[23] C. Doerr. Complexity theory for discrete black-box optimization heuristics. *CoRR*, abs/1801.02037, 2018.
[24] P. Erdős and A. Rényi. On two problems of information theory. *Magyar Tudományos Akadémia Matematikai Kutató Intézet Közleményei*, 8:229–243, 1963.
[25] K. Ghédira and B. Dubuisson, editors. *Constraint Satisfaction Problems*. John Wiley and Sons, 2013.
[26] C. Gießen and C. Witt. The interplay of population size and mutation probability in the $(1 + \lambda)$ EA on OneMax. *Algorithmica*, 78(2):587–609, 2017.
[27] N. Hansen, A. Auger, D. Brockhoff, D. Tusar, and T. Tusar. COCO: performance assessment. *CoRR*, abs/1605.03560, 2016.
[28] N. Hansen, A. Auger, O. Mersmann, T. Tusar, and D. Brockhoff. COCO: A platform for comparing continuous optimizers in a black-box setting. *CoRR*, abs/1603.08785, 2016.
[29] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '10, pages 1689–1696, New York, NY, USA, 2010. ACM.
[30] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. *Inria Research Report RR-6829*, 2009.
[31] H.-K. Hwang, A. Panholzer, N. Rolin, T.-H. Tsai, and W.-M. Chen. Probabilistic analysis of the (1 + 1)-evolutionary algorithm. *Evolutionary Computation*, pages 1–47, 2018. To appear.
[32] IBM ILOG. The CPLEX Optimizer. www.ibm.com/software/, 2018.
[33] J. Kacprzyk and W. Pedrycz, editors. *Springer Handbook of Computational Intelligence*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.
[34] S. A. Kauffman and E. D. Weinberger. The NK model of rugged fitness landscapes and its application to maturation of the immune response. *Journal of Theoretical Biology*, 141(2):211 – 245, 1989.
[35] P. K. Lehre and C. Witt. Black-box search by unbiased variation. *Algorithmica*, 64:623–642, 2012.
[36] R. Martí, P. Pardalos, and M. G. Resende, editors. *Handbook of Heuristics*. Springer International Publishing, 2018.
[37] B. Militzer, M. Zamparelli, and D. Beule. Evolutionary search for low autocorrelated binary sequences. *IEEE Transactions on Evolutionary Computation*, 2(1):34–39, Apr 1998.
[38] C. Moore and S. Mertens. *The Nature of Computation*. Oxford University Press, 2011.
[39] T. Packebusch and S. Mertens. Low autocorrelation binary sequences. *Journal of Physics A: Mathematical and Theoretical*, 49(16):165001, 2016.
[40] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Books on Computer Science. Dover Publications, 1998.
[41] I. A. Pasha, P. S. Moharir, and N. S. Rao. Bi-alphabetic pulse compression radar signal design. *Sadhana*, 25(5):481–488, Oct 2000.
[42] E. Rothberg. An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions. *INFORMS Journal on Computing*, 19(4):534–541, 2007.
[43] G. Rozenberg, T. Bäck, and J. N. Kok, editors. *Handbook of Natural Computing: Theory, Experiments, and Applications*. Springer-Verlag, Berlin-Heidelberg, Germany, 2012.
[44] I. I. Shapiro, G. H. Pettengill, M. E. Ash, M. L. Stone, W. B. Smith, R. P. Ingalls, and R. A. Brockelman. Fourth test of general relativity: Preliminary results. *Phys. Rev. Lett.*, 20:1265–1269, May 1968.
[45] D. Whitley. Next generation genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 922–941, New York, NY, USA, 2017. ACM.
[46] D. Whitley, K. E. Mathias, S. B. Rana, and J. Dzubera. Evaluating Evolutionary Algorithms. *Artificial Intelligence*, 85(1-2):245–276, 1996.