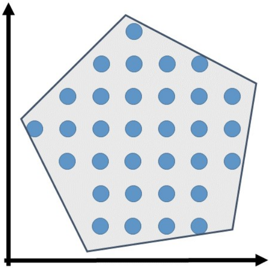
Compiling a Benchmarking Test-Suite for Combinatorial Black-Box Optimization: A Position Paper

Ofer M. **Shir** (Tel-Hai College & Migal Institute, ISRAEL)
Carola **Doerr** (CNRS & Sorbonne University, FRANCE)
Thomas **Bäck** (LIACS, Leiden University, The NETHERLANDS)

The Genetic and Evolutionary Computation Conference,
GECCO-2018: BB-DOB Workshop
July 2018, Kyoto, JAPAN

Benchmarking Discrete Optimization: Fundamentals



Introduction

- Combinatorial Optimization (CO),

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

  is defined by a finite set $\mathcal{S}$ with an objective function f assigning a non-negative value to any of its elements $s \in \mathcal{S}$.
  [**Seminal work**: "Combinatorial Optimization" by Papadimitriou & Steiglitz]
- Problem-solving in practice: Mathematical Programming (MP) / Operations Research, *versus* Randomized Search Heuristics (RSHs) / Soft Computing
- There are no common grounds for performance comparisons between RSHs to MP solvers when targeting similar CO problems.
- **Benchmarking RSHs on CO-problems is an open issue**.

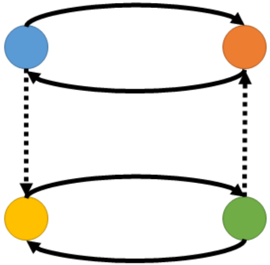
The Role of Benchmarking

- How does the algorithm perform on different **classes of problems** 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? E.g., *modality* of a problem, its *separability*, 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?

Starting Point

Current focus: formulating a set of benchmark problems and/or a test-suite for CO problems when treated as black-boxes by RSHs.

- Previously proposed guidelines for black-box benchmarking (Whitley et al., 1996):
  - (A) *"Test suites should contain problems that are resistant to hill-climbers"*.
  - (B) *"Test suites should contain problems that are non-linear, non-separable, and non-symmetric"*.
  - (C) *"Test suites should contain scalable functions"*.
  - (D) *"Test suites should contain problems with scalable evaluation cost"*.
  - (E) *"Test problems should have a canonical form"*.
- **BBOB** is an established testing framework for evaluating performance of continuous optimizers. The noise-free suite encompasses 24 functions.



$$9 \times Q \rightsquigarrow A$$

The Traveling Salesman Problem

The *archetypical* Traveling Salesman Problem (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 has cost information $c_{ij} \in \mathbb{R}^+$.

Black-box formulation: permutations

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

ILP Formulation [Miller-Tucker-Zemlin]

TSP as an ILP utilizes n^2 binary decision variables $\mathtt{x}_{ij}$:

$$\boxed{\begin{array}{l} \textbf{[TSP-ILP]} \quad \text{minimize} \displaystyle\sum_{\langle i,j\rangle \in E} c_{ij} \cdot \mathtt{x}_{ij} \\ \text{subject to:} \\ \qquad \displaystyle\sum_{j \in V} \mathtt{x}_{ij} = 1 \quad \forall i \in V \\ \qquad \displaystyle\sum_{i \in V} \mathtt{x}_{ij} = 1 \quad \forall j \in V \\ \qquad \mathtt{x}_{ij} \in \{0,1\} \quad \forall i,j \in V \\ \qquad \mathtt{u}_i - \mathtt{u}_j + 1 \leq (|V|-1)(1-\mathtt{x}_{ij}) \quad \forall i,j \in 1 \ldots |V| \\ \qquad |V| \geq \mathtt{u}_i \geq 2 \quad \forall i \in \{2,3,\ldots,|V|\} \end{array}} \tag{2}$$

where n integers $\mathtt{u}_i$ are needed as decision variables to prevent inner-circles.

Q1: Problem Representation

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

Q1: Problem Representation

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

[A1]

- A certain problem formulation **should be set fixed** — TSP-ILP and TSP-perm are two different search-problems!
- We suggest to restrict the benchmark suite to functions $f : \mathcal{S} \to \mathbb{R}^+$ ($\mathcal{S}$ being a finite set of integers — also the most common representation in the EC literature)

Q2: Instance-Based Problems

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

Q2: Instance-Based Problems

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

[A2]

- The hardness of an instance-based CO problem can differ substantially between two different instances
- Problem instances require specific descriptions and are therefore, in general, **not arbitrarily scalable with respect to their dimensionality**
- We suggest that **preference should be given to instance-free problems**; instance-based problems be included only to the extent needed to understand performance behavior that cannot be otherwise observed over instance-free problems

Q3: Invariant Problem Formulation

[Q3] Should the benchmarking framework cover the invariance aspect, and implicitly **favor algorithms that are invariant**? If so, which invariances should be respected?

Q3: Invariant Problem Formulation

[Q3] Should the benchmarking framework cover the invariance aspect, and implicitly **favor algorithms that are invariant**? If so, which invariances should be respected?

[A3]

- Every benchmark suite might focus on a too narrow representative set of problems – with the risk of *overfitting*
- Therefore, the suite **should account for problem invariances**.
- We believe that conforming to certain, natural invariances reduces the risk of such overfitting. Our full paper elaborates on such meaningful invariances.

Q4-Q6: Performance Evaluation

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

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

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

Performance Evaluation Answered

[**A4**] We advocate the use of **function evaluations as the main performance measure** — as in COCO.

[**A5**] Yes, we support **performance aggregation** (though not over problem dimensions, since it should be used for algorithm selection).

[**A6**] Yes, the benchmark suite **should allow for algorithm profiling**.

Admitting Runtime Analysis

- BBOB also encompasses a few rather simple problems like the Sphere function ($\mathbb{R}^n, F_1(x) := \sum_{i=1}^{n} x_i^2$) and other *unconstrained convex* problems.
- Convexity is irrelevant here, but the equivalent in problem hardness could be *simple problems admitting runtime analysis*.
- A well-known representative of this class is the "OneMax" problem

$$\boxed{\begin{array}{l} \textbf{[HD]} \;\; \text{minimize} \displaystyle\sum_{i=1}^{n} x_i \\ \text{subject to:} \\ \qquad x_i \in \{0,1\} \;\; \forall i \in \{1,\ldots,n\} \end{array}} \tag{3}$$

Q7: Problems Admitting Runtime Analysis

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

Q7: Problems Admitting Runtime Analysis

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

[A7]
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.

Q8: Facing Operations Research

Many CO problems may be formulated as Integer Linear Programs and treated by Mathematical Programming (MP) solvers in **extreme efficiency**.
Performance differences may be significant when compared to RSH.

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

Q8: Facing Operations Research

Many CO problems may be formulated as Integer Linear Programs and treated by Mathematical Programming (MP) solvers in **extreme efficiency**.
Performance differences may be significant when compared to RSH.

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

[A8]
Yes, problems that are (easily) solvable by MP-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.

Constraints Satisfaction Problems

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

[NQP-CSP] satisfy:

$$\begin{aligned}
&\sum_{i,j} x_{ij} = n \\
&\sum_{i,j|j-i:=k} x_{ij} \leq 1 \quad k \in \{-n+2, -n+3, \ldots, n-3, n-2\} \\
&\sum_{i,j|i+j:=\ell} x_{ij} \leq 1 \quad \ell \in \{2, 3, \ldots, 2n-3, 2n-2\} \\
&x_{ij} \in \{0,1\} \;\; \forall i,j \in \{1, \ldots, n\}
\end{aligned} \tag{4}$$

n^2 binary decision variables x_{ij} are associated with the chessboard's coordinates, having an origin $(1,1)$ at the top-left corner.

Q9: Distinguishing CSP?

The OR community distinguishes between standard optimization problems to Constraints Satisfaction Problems: Constraints Programming has forked into an independent subcommunity.

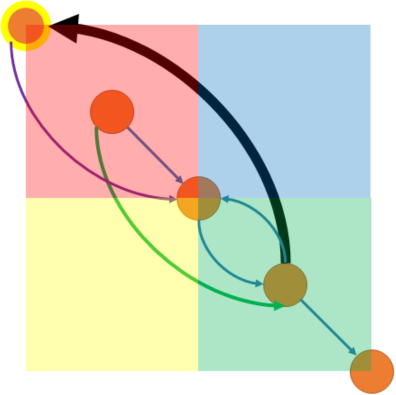
[Q9] Should RSHs' **performance be indistinguishably evaluated on CSPs as well**? That is, should a distinction between standard optimization to CSPs be avoided in the black-box perspective?

Q9: Distinguishing CSP?

The OR community distinguishes between standard optimization problems to Constraints Satisfaction Problems: Constraints Programming has forked into an independent subcommunity.

[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?

[A9]
Yes, RSHs' performance **should be indistinguishably evaluated on CSPs as well**, since in the black-box perspective they are merely CO-problems.



Discussion

Nonlinear Hard Problems

This class of problems is meant to capture challenging CO problems that do not subscribe to MP/OR.

The Low-Autocorrelation Binary Sequence (LABS) problem is a hard CO problem with practical applications in electrical engineering. Given a sequence of length n, $S := (s_1, \ldots, s_n)$ with $s_i = \pm 1$,

$$\begin{array}{l} \textbf{[LABS]} \ \text{maximize} \ \dfrac{n^2}{2E(S)} \\ \text{subject to:} \\ \qquad E(S) := \displaystyle\sum_{k=1}^{n-1} \left(\sum_{i=1}^{n-k} s_i \cdot s_{i+k} \right)^2 \\ \qquad s_i \in \{-1, +1\} \ \ \forall i \in \{1 \ldots n\} \end{array} \tag{5}$$

The Human Factor

The **human factor plays a crucial role in such processes**.

- Formulation of a test-suite may involve three types of scholars: theoreticians, algorithms' designers, and practitioners.
  (i) theoreticians naturally favor analyzable functions
  (ii) algorithms' engineers may prefer families of functions that are successfully treated by their designs
  (iii) practitioners may have the best insights into which functions most accurately represent real-world problems (thus having their biased preferences)

- A proper balance should be made amongst those three parties to effectively compile a test-suite meaningful to a broad audience.

Communities and Resources

- INFORMS: The Institute for Operations Research and the Management Sciences; `https://www.informs.org/`
- COIN-OR: Computational Infrastructure for Operations Research – a project that aims to "create for mathematical software what the open literature is for mathematical theory"; `https://www.coin-or.org/`
- MATHEURISTICS: model-based metaheuristics, exploiting MP in a metaheuristic framework; `http://mh2018.sciencesconf.org/`

Benchmarking and Competitions

- MIPLIB: the Mixed Integer Programming LIBrary

  http://miplib.zib.de/

- CSPLib: a problem library for constraints

  http://csplib.org/

- SAT-LIB: the Satisfiability Library - Benchmark Problems

  http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

- TSP-LIB: the Traveling Salesman Problem sample instances

  http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

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

Acknowledgements go to Pietro Oliveto for motivating the effort.

dōmo arigatō