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

GECCO '18: Genetic and Evolutionary Computation Conference
Kyoto, Japan, July 15–19, 2018

Session: Black Box Discrete Optimization Benchmarking
July 16, 11:00-12:40, Room 3 (2F)

Aleš Zamuda, Miguel Nicolau, Christine Zarges

University of Maribor

University College Dublin

Prifysgol Aberystwyth University

Motivation

- Taxonomical identification survey of classes
  - in **discrete optimization** challenges
  - that can be found in the literature.
- Black-Box Discrete Optimization Benchmarking (BB-DOB).
- Including a proposed **pipeline perspective** for benchmarking,
  - inspired by previous computational optimization competitions.

- Main topic: **why certain classes** together with their **properties** should be included in the perspective,
  - like deception and separability or toy problem label.

- Moreover, guidelines are discussed on:
  - how to **select significant instances** within these classes,
  - the design of experiments **setup**,
  - performance **measures**, and
  - presentation methods and **formats**.

Other Existing Benchmarks

Inspired by previous computational optimization competitions in **continuous settings** that used test functions for optimization application domains:

- single-objective: CEC 2005, 2013, 2014, 2015
  - constrained: CEC 2006, CEC 2007, CEC 2010
  - multi-modal: CEC 2010, SWEVO 2016
  - black-box (target value): BBOB 2009, COCO 2016
  - noisy optimization: BBOB 2009
  - large-scale: CEC 2008, CEC 2010
  - dynamic: CEC 2009, CEC 2014
  - real-world: CEC 2011
  - computationally expensive: CEC 2013, CEC 2015
  - learning-based: CEC 2015
- multi-objective: CEC 2002, CEC 2007, CEC 2009, CEC 2014
- bi-objective: CEC 2008
- many objective: CEC 2018

Tuning/ranking/hyperheuristics use. $\rightarrow$ DEs as usual winner algorithms.

Discrete Optimization Functions Classes: Perspectives

- Including grey-box knowledge: black $\rightarrow$ white (box).
- More is known about the problem, the better the algorithm.
- **Representation** (known knowledge) and **budget cost** (knowledge from **new**/**online** fitness calls):

1. Modality:
   - unimodal, bimodal, multimodal – over GA fixed genotypes.
2. Programming representations:
   - fixed vs. dynamic – using GP trees.
3. Real-world challenges modeling:
   - for tailored problem representation.
4. Budget planning:
   - for new problems.

Perspective 1: Fixed Genotype Functions – Modality

- Modality:
  - **Pseudo-Boolean**: $f : \{0,1\}^n \to \mathbb{R}$
  - **unimodal**: there is a unique local optimum (e.g. OneMax)
    - a search point x^* is a local optimum if:
      for all x with $H(x^*, x) = 1$
      (i.e., the direct Hamming neighbors of x^*), $f(x^*) \geq f(x)$
  - **weakly unimodal**: all its local optima have the same fitness
  - **multimodal**: otherwise (not (weakly)unimodal) – e.g. Trap
    - **bimodal**: have two local optima (e.g. TwoMax)
    - generalization of TwoMax to **arbitrary no. local optima**

Perspective 1: Fixed Genotype Functions – More Properties

- Other properties for Boolean functions:
  - **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,
  - **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).

Perspective 2: Sample Symbolic Regression Problems with GP

- Genetic Programming (GP) has seen a recent effort towards standardization of benchmarks, particularly in the application area of **Symbolic Regression** and **Classification**.
- 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 (15 functions), Pagie (1 function), Korns (15 functions), and Vladislavleva (8 functions).

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} : $f(x_1, x_2, x_3, x_4, x_5) = 2 - 2.1\cos(9.8x_1)\sin(1.3x_5)$

Perspective 2: Dynamic Genotype Functions, GP – Guidelines

- Guidelines on improving benchmarking GP by Nicolau et al.:
  - 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.
- Mostly GP researchers resort to **UCI datasets** for real-world

Perspective 2: Dynamic Genotype Functions, GP – Example Classes

- Scalable Genetic Programming (function classes): e.g. through **gene-pool optimal mixing** and **input-space entropy-based building-block learning**
  - To learn and exploit **model structures** in black-box optimization for possibly atomic representations of partial solutions,
  - e.g. in dimension (number of input variables), but also through definition of function/terminal set (with scalable number of inputs)
  - artificial problems **Order** (GP version of OneMax) and **Trap** (with scalable problem size as the maximum binary tree height); applied: **Boolean Circuits design** (Comparator, Even Parity, Majority, and Multiplexer).
- GP at other areas: most popular in recent years: **gaming** and **automatic program synthesis** with a range of competitions and workshops
  e.g. Evolving levels for Super Mario Bros, Virtual Creatures Contest (https://virtualcreatures.github.io/vc2018/), and The General Video Game AI Competition (http://www.gvgai.net/).
- 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.

Perspective 3: Tailored Problem Representation

Additionally to those functions already mentioned, real-world challenges for optimization algorithms include e.g.:

- **knapsack problems** (e.g. automatic summarization),
- **routing** (Traveling Salesperson Problem (TSP), Chinese Postman (CP), path planning)
- **scheduling** (including job shop and flow shop)
- **bioinformatics** (including sequencing, alignment, and protein folding),
- **cryptography**, and
- **computer vision**.

Perspective 4: Budget Planning During Optimizer Design

- In order to introduce solutions to industry that require process improvement after very latent evaluation of the process,
  - 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.

Perspective 4: New Algorithm Design

- Application of an algorithm to a domain could be **reflective**:
  - influence the quality of the produced results due to a limited fitness evaluation budget allowed for setting up an optimizer:

  A measure of **performance inclination to any function class** should 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.
    - Example: 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 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.

Properties and Usage

- Use the **4 identified aspects** as **BB-DOB challenges**:
  - through **function class types** representative instances.

- 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.

Significant Instances Methodology: Openness Requirement

- Methodology requirement: when preparing a benchmark, **formulation** as well as **data** should be accessible.

- Example: namely, some e.g. **domain specific benchmarks** might not be fully disclosed, which would not allow testing of other (new) algorithms.
  - Explained: 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.

$\rightarrow$ Take into account **existing and recognized** benchmark sets when preparing a new discrete optimization benchmark,

- like the **MIPLIB 2010** benchmark set[1].

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

Significant Instances Methodology: Connectivity Requirement

New benchmark set should connect to existing optimizer providers

- Gurobi `http://www.gurobi.com/downloads/download-center`,
- CPLEX `https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer`,
- XPRESS `https://www.solver.com/xpress-solver-engine`,
- Mosek `https://www.mosek.com/`,
- SCIP `http://scip.zib.de/`,
- CBC `https://projects.coin-or.org/Cbc`,
- GLPK `https://www.gnu.org/software/glpk/`,
- LP_Solve `http://lpsolve.sourceforge.net/`, and
- MATLAB `https://www.mathworks.com/products/optimization.html`.

Significant Instances Methodology: General Guidelines

As a black-box benchmarking suite, the perspectives of the **COCO platform** could be followed.

The functions referenced previously in this talk capture:

- combinatorial optimization problems **difficulties 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.

Significant Instances Methodology: Selecting Classes

- We suggest:
  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, which 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 as the benchmark as a whole.
- Select classes: at the workshop and until PPSN.

Experiments Setup: Outline

1. Optimize the instance functions listed in the previous part,
2. 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.
3. The number of independent runs for an optimizer on a specific test instance should be set based on the test instance **complexity**
4. The runtime of optimizers should be measured **proportional** to the time it takes to execute the fitness functions.

Performance Measures: Requirements

Benchmark: practically **relevant** & theoretically **accessible**

- Vital practicality measure: algorithms' mutual **advantages** and **disadvantages** insight
- Accessibility: public website w/ results submission

Measured performance:

- runtime,
- fixed budget yield,
- fixed precision yield.

Performance Measures: Analysis

Comparing algorithms:

- Friedman **ranking** & post-hoc procedures for significance (fixed budget)
- observing **order of magnitude** improvements (limited budget for new problems)
  - fast **incremental rating** systems or
  - single value **performance mark**.

Deep statistics for evolution and **behavior tracking** of:

- values of population and memory within the algorithm,
- traits visualization, e.g. with graph plots for:
  - fitness convergence,
  - control parameters,
  - optimizer population memory,
  - inter-connectedness of evolved population members through generations (i.e. **communication complexity** analysis).

Results Tables: Content

The tables 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.

→ 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.

Results Tables: Compatibility

Compatibility for:

- generation of data output,
- procedures for post-processing,
- ranking, and
- presentation formats for the results.

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.

Conclusion

- Surveyed previous successes in benchmarking of optimization algorithms (CEC, GECCO).
- Listed toy problems and also specific domain benchmarks,
  - GP, knapsack, routing, scheduling, bioinformatics, and CV.
- **Taxonomical identification** survey:
  - **classes** in discrete optimization,
  - perspective on Black-Box Discrete Optimization Benchmarking (BB-DOB).
- **Benchmarking pipeline for BB-DOB**, providing:
  - properties, usage, instances of listed classes,
  - experimental setup, performance measures, and
  - formats for result representation.
- Towards challenges for more general discrete optimization:
  - able to **tackle new unknown problems** as a black box and
  - **algorithms** automating performance over these problems.

Future Work

- When the WG3 of **ImAppNIO wiki**[2] will list sufficient benchmarking suggestions, it is expected that a benchmark **code package** (WG4) 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.

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

Acknowledgement. This article is based upon work from COST Action CA15140 'Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (**ImAppNIO**)' 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**.

Thank you for your attention.

Questions?