Handling Real-World Problems within the COCO Platform

Tea Tušar
Jožef Stefan Institute
Ljubljana, Slovenia
tea.tusar@ijs.si

Vanessa Volz
modl.ai
Kopenhagen, Denmark
vanessa@modl.ai

Dimo Brockhoff
Inria, CMAP, Ecole Polytechnique
IP Paris, France
name.surname@inria.fr

Nikolaus Hansen
Inria, CMAP, Ecole Polytechnique
IP Paris, France
name.surname@inria.fr

ABSTRACT

Until recently, the problems employed for benchmarking optimization algorithms within the Comparing Continuous Optimizers (COCO) platform needed to have continuous variables and known optimal values. In addition, they had to be implemented within the platform (in the C language). These restrictions made COCO difficult to use for benchmarking algorithms on real-world problems. This paper describes the adaptations to the COCO platform that facilitate its use on real-world and other problems with integer or mixed-integer variables and unknown optimal values. Evaluation of solutions can now be done with external programs that are interfaced with COCO through socket communication.

Keywords

Real-world problems, algorithm benchmarking, the COCO platform

1. INTRODUCTION

Although Evolutionary Computation (EC) methods are often applied to real-world problems, they are almost exclusively benchmarked on artificial ones [7]. This is especially problematic in the field of Evolutionary Multi-Objective Optimization (EMO) where the most popular test problem suites like DTLZ [2] and WFG [5] have some unintended characteristics that stem from their construction and are not likely to be present in the real world. Consequently, we cannot expect algorithms that perform well on such test problems to also work well on real-world problems, which defies an important aspect of algorithm benchmarking [8].

To amend this issue, new test problems from the real world are being proposed. For example, the Mazda problem is a highly constrained problem with a large number of integer variables and two objectives [6]. It requires setting the thickness of several car parts so that their total weight is minimized and the number of parts with common thickness is maximized. The main challenge of this problem stems from its large search space dimension and the difficulty of finding feasible solutions due to the many constraints. Another example is the suite of three diverse design optimization problems that require Computational Fluid Dynamics (CFD) simulations for evaluating solutions [1]. The problems have a different number of objectives (two are single- and one is bi-objective) and can have varying search space dimension. Since the CFD simulations are time-consuming, the biggest challenge is to find good solutions to the problems in reasonable time.

Using such problems for algorithm benchmarking is nontrivial since nothing is provided but the problems themselves. A researcher who wants to use these problems in a benchmarking study still needs to take care of the performance assessment methodology as well as run additional algorithms on the same problems to acquire data for comparisons.

An alternative is to propose real-world problems within a framework that takes care of the cumbersome aspects of algorithm benchmarking. The Comparing Continuous Optimizers (COCO) platform[1] was designed exactly for facilitating this task [3]. It incorporates several suites of test problems, takes care of all the performance assessment and makes it easy to include data from previous experiments in the comparisons. The selection of its problem suites was recently extended to include mixed-integer problems [9] as well as real-world problems based on games [12]. This required some adaptations of the platform that are also expected to simplify future inclusions of real-world problem suites.

This paper presents the modifications that were needed for COCO to support problems with integer variables, problems with unknown optimal values and external evaluation of solutions. They were not previously explained in [9] and [12].

After a brief presentation of the COCO platform and its latest suites in Section 2, we explain the adaptations needed to support real-world problems within COCO in Section 3. The paper ends with concluding remarks in Section 4.

2. THE COCO PLATFORM

2.1 Overview

The aim of the COCO platform [3] is to simplify the benchmarking of numerical optimization algorithms and make the data from those experiments available to the scientific community. The platform consists of two main parts (see Figure 1). The first, called *COCO experiments*, is implemented in C. It is used for running an algorithm on the chosen test problem suite and recording its performance. The algorithm

[1] `https://github.com/numbbo/coco`

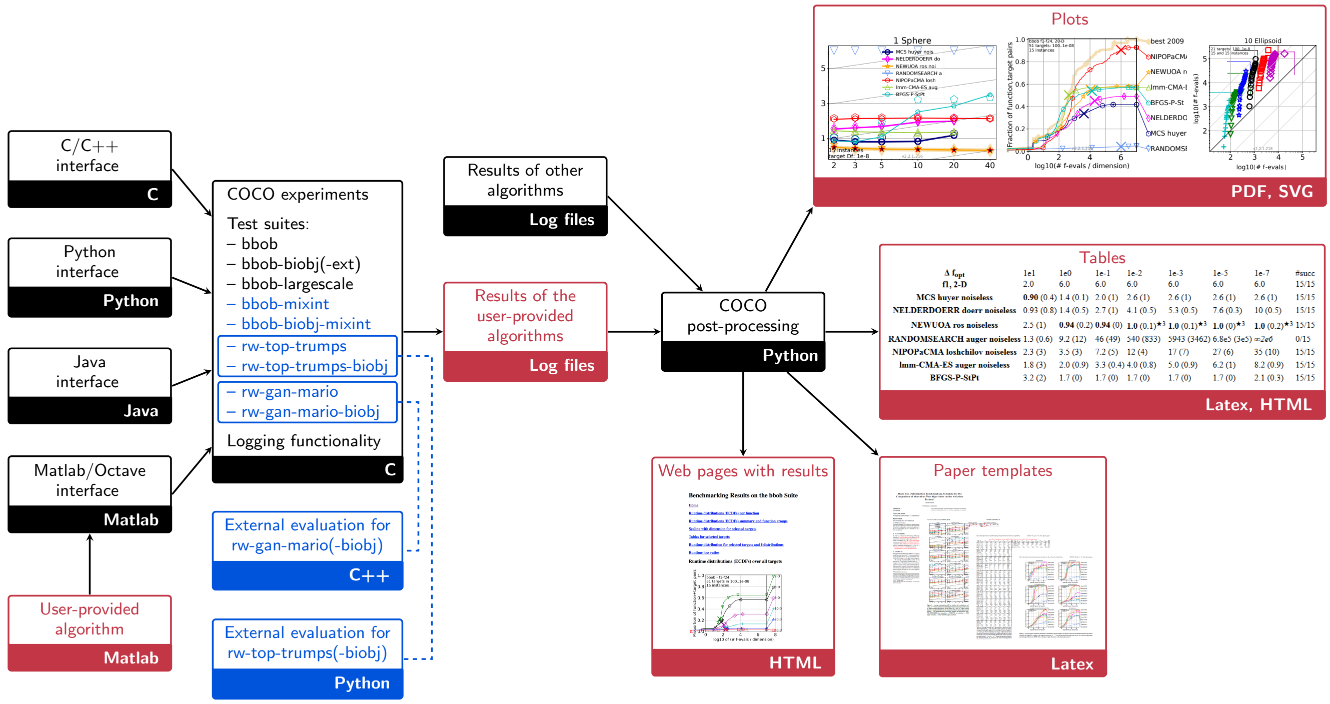

Figure 1: The COCO platform scheme. Its main components are presented in black, while the user-provided algorithm and its results are shown in red. Blue color is used to denote the recent additions.

can be connected to the platform using one of the available interfaces in C/C++, Python, Java and Matlab/Octave. During the algorithm run, its results are logged into files whenever one of the performance targets is achieved.

The second part, called *COCO post-processing*, is implemented in Python. From the log files created by the experiments, it produces plots and tables with information on the performance of the algorithm as well as HTML pages to facilitate browsing through them and paper templates with the most relevant results already included. With COCO post-processing it is very easy to add the performance of other algorithms to the comparisons. Currently, results of more than 200 experiments are available. Most were collected on the `bbob` suite of 24 continuous single-objective problems without constraints or noise [4].

Until of late, all the problem suites in COCO were based on the `bbob` problems. For example, the `bbob-largescale` suite contains large-scale instantiations of the `bbob` problems [11], while the `bbob-biobj` and `bbob-biobj-ext` suites consist of bi-objective problems constructed by using the `bbob` functions as their separate objectives [10].

2.2 Recent Problem Suites

We have recently proposed a total of six new problem suites that are more real-world-like than those previously included in COCO [9, 12]. Table 1 shows summary information for some of their properties. All can be initialized with various search space dimensions and provide multiple instances that represent small perturbations of the problems. In all these suites the bi-objective problems were created by using two single-objective functions as the two objectives.

Table 1: Basic properties of the six recently proposed problem suites.

Suite name	`bbob-mixint`	`bbob-biobj-mixint`
# objectives	1	2
Dimensions	5, 10, 20, 40, 80, 160	5, 10, 20, 40, 80, 160
# functions	24	92
# instances	15	15

Suite name	`rw-top-trumps`	`rw-top-trumps-biobj`
# objectives	1	2
Dimensions	88, 128, 168, 208	88, 128, 168, 208
# functions	5	3
# instances	15	15

Suite name	`rw-gan-mario`	`rw-gan-mario-biobj`
# objectives	1	2
Dimensions	10, 20, 30, 40	10, 20, 30, 40
# functions	28	10
# instances	7	7

The `bbob-mixint` and `bbob-biobj-mixint` suites contain single- and bi-objective mixed-integer problems, respectively. They were constructed by discretizing the first 80% of the variables of the corresponding `bbob` and `bbob-biobj` problems. Because of this, the problem dimensions were set to be larger than those of the `bbob` problems, while the functions and instances remained the same.

The problems from the single- and bi-objective suites `rw-top-trumps` and `rw-top-trumps-biobj` are based on the Top Trumps card game. The goal (optimization problem) is to construct a deck for the game with desirable properties (objectives). The number of dimensions corresponds to the number of cards (22, 32, 42, 52) multiplied by the number of categories on a card (4), and the all-integer variable values are the values of the categories on the cards. Out of the five different single objectives that measure a quality indicator of the deck, two can be computed directly and three require simulations of gameplay. The three bi-objective functions are constructed from the five single-objective ones in such a way that the two objectives are (at least partially) conflicting.

Lastly, the `rw-gan-mario` and `rw-gan-mario-biobj` suites contain single- and bi-objective problems of constructing levels for the well-known Super Mario Bros. platformer game to optimize the chosen objectives. The levels are computed by a Generative Adversarial Network (GAN), i.e., the solutions correspond to continuous latent vectors [13]. The dimension of the search space therefore matches the dimension of the latent vectors and can be set almost arbitrary. Out of the 28 single objectives, ten can be computed directly and the rest require simulations of gameplay. Again, the bi-objective functions were constructed by looking at the conflicts between objectives.

3. SUPPORTING PROPERTIES OF REAL-WORLD PROBLEMS

COCO was initially designed to work with the `bbob` problems that are continuous, have known optima and use the C code within COCO experiments to evaluate solutions. Here we explain in more detail the changes brought by the shift to real-world problems, which do not share these properties (see Figure 1).

3.1 Integer Variables

The Top Trumps and mixed-integer suites required supporting problems where either all or just some of the variables are integer. This entailed adding an additional parameter, which gives the number of integer variables to the internal problem class in COCO experiments as well as to the interfaces to all supported languages. Without any loss of generality we set that all the integer variables come before any continuous ones, which means that this single addition is enough to support problems with (some) integer variables (the parameter is naturally set to zero for continuous problems). The integer variables are internally still represented as real values with double precision. It is then up to the evaluation function to make sure they are correctly interpreted as integers.

In addition, the COCO loggers can be configured to output these variables as integers, which can save considerable space in case of a large number of integer variables (see the Top Trumps suites). This is done through the observer's `log_discrete_as_int` parameter, which is set to false by default.

3.2 Unknown Optimal Values

In COCO, an evaluation is logged whenever it surpasses a target value. When an algorithm is run on problems with known optimal values, the target values are defined as differences to the optimal function value (in the single-objective case) or to the optimal value of a multi-objective performance indicator (in the multi-objective case). In the usual benchmarking setting in COCO, the targets are chosen equidistantly in logarithmic scale. Therefore, it is very important that the optimal value is known (or is at least very well approximated). If the estimate of the optimal value is (much) lower than the true optimal value, the smallest target values will never be reached. If, on the other hand, it is (much) higher, the algorithm will be able to reach all targets while still being arbitrarily far away from the optimal value.

The discretization of the `bbob` and `bbob-biobj` problems that produced mixed-integer problems was performed in such a way that the optimal values remained equal and are therefore known (see [9] for more details). This means that similarly to their corresponding continuous predecessors, the optima for the `bbob-mixint` problems are known, while for the `bbob-biobj-mixint` problems, the ideal and nadir points are known, but not the Pareto sets and fronts (in this case not even for the double sphere function). In contrast, most Top Trumps and Mario GAN problems have unknown optimal values already in their single-objective formulation, which is to be expected in the majority of real-world problems. Consequently, neither the Pareto sets and fronts nor the ideal and nadir points are known for the bi-objective game-based problems.

While the issue of unknown optimal indicator values for the `bbob-biobj` problems is amended by providing an estimate of indicator values using all nondominated solutions from several runs of a number of algorithms, this approach is not feasible for real-world problems.

In order to support real-world problems with unknown optimal values, we are using an infinite number of equally spaced absolute target values aligned at zero with a step of 10^{-5}. In this way, the logger records an evaluation any time the algorithm finds a function (or performance indicator) value that improves the best found one by at least 10^{-5}. Such a strategy makes sure that the convergence to the optimal value can be detected (up to the precision of 10^{-5}) regardless of its absolute value.

After the experiments, the targets of interest need to be chosen for the post-processing part. This requires some preliminary analysis of the results. Once the targets have been chosen, they can remain the same for future experiments or change in order to account for better solutions found in time. This does not affect the ability to add previously computed results to the comparison as the post-processing is always run anew.

3.3 External Evaluations

While artificial problem suites can be implemented in C with some moderate effort, this is much harder to do for real-world problems (especially those that are not originally available in C). To address this issue, we added the possibility to evaluate solutions using an external evaluator that is not provided by COCO.

This is achieved by the means of *socket communication*, where the external evaluator acts as a server waiting to be queried and COCO as the client that continuously queries the server with proposed solutions. In such a case, the 'shell' of the suite that provides the general information about its problems still needs to be implemented in COCO, however, this is rather straightforward and has been automated with a script.

Evaluation of solutions using socket communication works as follows. COCO (the client) sends to the external evaluator (the server) a solution together with the information needed to identify the problem, that is, the function and instance identifier and the number of dimensions. If needed, other parameters can also be passed at the same time. When the external evaluator receives the query, it evaluates the given solution with the right problem and returns the objective and constraint values as a response to the query.

This is a quite flexible and efficient way to communicate with an external evaluator. It is much faster than writing to and reading from files. It is also very flexible—the external evaluator can really be external (not even run on the same computer as COCO), which might be important for some real-world problems that cannot be disclosed.

4. CONCLUSIONS

By adding to COCO the support for problems with integer variables, unknown optimal values and external evaluation of solutions, we have opened its use for benchmarking optimization algorithms on real-world problems. We hope that the mixed-integer and game-based problem suites described in this paper are just the start and other real-world problems, such as the Mazda problem and the CFD problems mentioned in the Introduction will follow soon.

The code with the functionality described in the paper can be found at `https://github.com/ttusar/coco/tree/gbea`.

5. ACKNOWLEDGMENTS

The first author acknowledges the financial support from the Slovenian Research Agency (project No. Z2-8177).

6. REFERENCES

[1] S. J. Daniels, A. A. M. Rahat, R. M. Everson, G. R. Tabor, and J. E. Fieldsend. A suite of computationally expensive shape optimisation problems using computational fluid dynamics. In *Proceedings of the 15th International Conference on Parallel Problem Solving from Nature, PPSN XV*, pages 296–307. Springer, Cham, Switzerland, 2018.

[2] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multiobjective optimization. In A. Abraham, L. Jain, and R. Goldberg, editors, *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, pages 105–145. Springer London, 2005.

[3] N. Hansen, A. Auger, O. Mersmann, T. Tušar, and D. Brockhoff. COCO: A platform for comparing continuous optimizers in a black-box setting. *ArXiv e-prints*, arXiv:1603.08785, 2016.

[4] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009.

[5] S. Huband, P. Hingston, L. Barone, and R. L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

[6] T. Kohira, H. Kemmotsu, A. Oyama, and T. Tatsukawa. Proposal of benchmark problem based on real-world car structure design optimization. In *Companion Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 183–184. ACM, 2018.

[7] Z. Michalewicz. Quo vadis, evolutionary computation?: On a growing gap between theory and practice. In *Proceedings of the 2012 World Congress Conference on Advances in Computational Intelligence*, WCCI'12, pages 98–121. Springer-Verlag, 2012.

[8] T. Tušar. On using real-world problems for benchmarking multiobjective optimization algorithms. In *Proceedings of the International Conference on High-Performance Optimization in Industry, HPOI 2018, 21st International Multiconference Information Society, IS 2018*, volume D, pages 7–10. Jožef Stefan Institute, 2018.

[9] T. Tušar, D. Brockhoff, and N. Hansen. Mixed-integer benchmark problems for single- and bi-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, pages 718–726. ACM, 2019.

[10] T. Tušar, D. Brockhoff, N. Hansen, and A. Auger. COCO: The bi-objective black-box optimization benchmarking (bbob-biobj) test suite. *ArXiv e-prints*, arXiv:1604.00359, 2016.

[11] K. Varelas, A. Auger, D. Brockhoff, N. Hansen, O. A. ElHara, Y. Semet, R. Kassab, and F. Barbaresco. A comparative study of large-scale variants of cma-es. In *Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN XV)*, volume 11101 of *Lecture Notes in Computer Science*, pages 3–15. Springer, 2018.

[12] V. Volz, B. Naujoks, P. Kerschke, and T. Tušar. Single- and multi-objective game-benchmark for evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, pages 647–655. ACM, 2019.

[13] V. Volz, J. Schrum, J. Liu, S. M. Lucas, A. Smith, and S. Risi. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 221–228. ACM Press, New York, 2018.