Versatile Black-Box Optimization

Jialin Liu, Antoine Moreau, Mike Preuss, Jeremy Rapin, Baptiste Roziere,

Southern University of Science and Technology
Shenzhen, China
liujl@sustech.edu.cn

Universitié Clermont Auvergne,
CNRS, SIGMA Clermont, Institut Pascal
Clermont-Ferrand, France
antoine.moreau@univ-bpclermont.fr

Facebook AI Research &
Paris-Dauphine University
Paris, France
jrapin@fb.com, broz@fb.com

Fabien Teytaud
Univ. Littoral Côte d’Opale
Calais, France
teytaud@univ-littoral.fr

Olivier Teytaud
Facebook AI Research
Paris, France
oteytaud@fb.com

ABSTRACT
Choosing automatically the right algorithm using problem descriptors is a classical component of combinatorial optimization. It is also a good tool for making evolutionary algorithms fast, robust and versatile. We present Shiwa, an algorithm good at both discrete and continuous, noisy and noise-free, sequential and parallel, black-box optimization. Our algorithm is experimentally compared to competitors on YABBOB, a BBOB comparable testbed, and on some variants of it, and then validated on several real world testbeds.

CCS CONCEPTS
• Theory of computation → Optimization with randomized search heuristics.

KEYWORDS
Black-box optimization, portfolio algorithm, gradient-free algorithms, open source platform

ACM Reference Format:
Jialin Liu, Antoine Moreau, Mike Preuss, Jeremy Rapin, Baptiste Roziere, Fabien Teytaud, and Olivier Teytaud. 2020. Versatile Black-Box Optimization. In Genetic and Evolutionary Computation Conference (GECCO ’20), July 8–12, 2020, Cancún, Mexico. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3377930.3389838

1 INTRODUCTION: ALGORITHM SELECTION
Selecting automatically the right algorithm is critical for success in combinatorial optimization: our goal is to investigate how this can be applied in derivative-free optimization. Algorithm selection [8, 29, 30, 32, 48, 49] can be made a priori or dynamically. In the dynamic case, the idea is to take into account preliminary numerical results, i.e. starting with several optimizers concurrently and, afterwards, focusing on the best only [12, 29, 34, 35, 37, 41]. Algorithm selection among a portfolio of methods routinely wins SAT competitions. Passive algorithm selection [4] is the special case in which the algorithm selection is made a priori, from high level characteristics which are typically known in advance, such as the following problem descriptors: dimension; computational budget (measured in terms of number of fitness evaluations); degree of parallelism in the optimization; nature of variables, whether they are discrete or not, ordered or not, whether the domain is metrizable or not; presence of constraints; noise in the objective function; and multiobjective nature of the problem. Besides the performance improvement on a given family of optimization problems, algorithm selection also aims at designing a versatile algorithm, i.e. an algorithm which works for arbitrary domains and goals. We use benchmark experiments in an extendable open source platform [46], which contains a collection of benchmark problems and state of the-art gradient-free algorithms, for designing a vast algorithm selection process - and test it on real world objective functions. Up to our knowledge, there is no direct competitor we could compare to because algorithm selection is usually only performed on a subset of the problem types we tackle. However, we do outperform the native algorithm selection methods of Nevergrad (CMandAS and CMandAS2 [45]) on YABBOB (Fig. 2), though not by a wide margin as on testbeds far from their usual context (Fig. 3).

2 DESIGN OF SHIWA
We propose Shiwa for algorithm selection. It uses a vast collection of algorithm selection rules for optimization. Most of these rules are passive, but some are active.

2.1 Components
Our basic tools are methods from [46]:

• Continuous domains: evolution strategies (ES) [5] including the Covariance Matrix Adaptation ES [23] (abbreviated as “CMA” in this paper), estimation of distribution algorithms (EDA) [9, 38, 40], Bayesian optimization [27], particle swarm optimization (PSO) [28], differential evolution (DE) [36, 50],

1We consider that a domain is metrizable if it is equipped with a meaningful, non-binary, metric. In the present paper, any domain containing a discrete categorical variable is considered as non-metrizable. This is not the mathematical notion of metrizability.
sequential quadratic programming [1], Cobyla [43] and Powell [42].

- Discrete optimization: the classical (1 + 1)-evolutionary algorithm, Fast-GA [19] and uniform mixing of mutation rates [17]. We also include some recombination operators [26].
- Noisy optimization: bandits [10], algorithms with repeated sampling [18] and population control [25].

### 2.2 Combinations of algorithms

Various tools exist for combining existing optimization algorithms. Terminology varies depending on authors. We adopt the following definitions: **Chaining**: running an algorithm, then another, and so on, initializing each algorithm using the results obtained by previous ones. In the present paper, we refer to chaining when an algorithm is run for a part of the computational budget, and another algorithm is run for the rest. For example, a memetic algorithm [44] running an evolution strategy at the beginning and another algorithm is run for the rest. For example, a memetic algorithm [44] running an evolution strategy at the beginning and Powell’s method [42] afterwards is a form of chaining. **Passive algorithm selection** [4]: the decision is made at the beginning of the optimization run, before any evaluation is performed. **Active algorithm selection**: preliminary results of several optimization algorithms can be used to select one of them. **Splitting**: the variables are partitioned into $k$ groups of variables $G_1, \ldots, G_k$. Then an optimizer $O_i$ optimizes variables in $G_i$; there are $k$ optimizers running concurrently. All optimizers see the same fitness values, and, at each iteration, the candidate is obtained by concatenating the candidates proposed by each optimizer. Our optimization algorithm uses all these combinations, except splitting - though there are problems for which splitting looks like a good solution and should be used. Most of the improvement is due to passive algorithm selection for moderate computational budgets, whereas asymptotically active algorithm selection becomes critical (in particular for multimodal cases) and chaining (combining an evolution strategy for early stages and fast mathematical programming techniques at the end) is an important tool for large computational budget.

### 2.3 Preliminaries

#### 2.3.1 Ask and tell and recommend

In the present document, we use a “ask and tell” presentation of algorithms, convenient for presenting combinations of methods. Given an optimizer $o$, $o.ask$ returns a candidate to be evaluated next. $o.tell(x, v)$ informs $o$ that the value at $x$ is $v$. $o.numask$ is the number of times $o.ask$ has been used. $o.archive$ is the list of visited points, with, for each of them, the list of evaluations (note that a same candidate can have been evaluated more than once, typically for noise management). $o.recommend$ provides an approximation of the optimum; this is typically the final step of an optimization run.

#### 2.3.2 Softmax transformations

[46] uses a softmax transformation for converting a discrete optimization problem into a noisy continuous optimization problem; for example, a variable with 3 possible values $a$, $b$ and $c$, becomes a triplet of continuous variables $v_a$, $v_b$ and $v_c$. The discrete variable has value $v_a$ with probability $\exp(v_a)/\left(\exp(v_a) + \exp(v_b) + \exp(v_c)\right)$. Preliminary experiments show that this simple transformation can perform well for discrete variables with more than 2 values.

#### 2.3.3 Optimism, pessimism, progressive widening

Following [14, 24, 39, 51], we use combinations of bandits, progressive growing of the search space and evolutionary computation as already developed in Nevergrad. Such combinations are labelled “Optimistic” in Fig. 1. We refer to [46] for full details.

### 2.4 Algorithm design

#### 2.4.1 Hypotheses

The design of our algorithm Shiwa relies on the following hypotheses: (i) The standard $(1 + 1)$-ES in continuous domains is reliable for low computational budget / high dimension. (ii) DE is a well known baseline algorithm. (iii) CMA performs well in moderate dimension [23], in particular for rotated ill-conditioned problems. (iv) Population control [25] is designed for continuous parameter optimization in high-dimension with a noisy optimization function. (v) In other noisy optimization settings one can use an “optimistic” combination of bandit algorithms and evolutionary computation [24, 31] as detailed above. (vi) Population control can be used for increasing the population size in case of stagnation, even without noise (see the “NaiveTBPSA” code in [46]). (vii) The MetaRecentering [11] algorithm wins benchmarks in many

---

**Figure 1**: The Shiwa algorithm using the components listed in Section 2.1 based on the hypotheses presented in Section 2.4.1. The problem dimension is denoted by $d$. 

---

**Diagram**: The Shiwa algorithm using the components listed in Section 2.1 based on the hypotheses presented in Section 2.4.1. The problem dimension is denoted by $d$. 

---

**Algorithm Design**

- **Hypotheses**
  - The standard $(1 + 1)$-ES in continuous domains is reliable for low computational budget / high dimension.
  - DE is a well known baseline algorithm.
  - CMA performs well in moderate dimension.
  - Population control is designed for continuous parameter optimization in high-dimension with a noisy optimization function.
  - In other noisy optimization settings one can use an “optimistic” combination of bandit algorithms and evolutionary computation.
  - Population control can be used for increasing the population size in case of stagnation, even without noise.
  - The MetaRecentering algorithm wins benchmarks in many.
one-shot optimization problems in [46]; it is a combination of Hammersley sampling [21], scrambling [2], and automatic rescaling [11].
(vii) Continuous optimization algorithms combined with softmax values of visible there.

3 EXPERIMENTS

We compare Shiwa to algorithms from Nevergrad on a large and diverse set of problems. The results of these experiments, which have all been produced using Nevergrad, are summarized in Figs. 2, 3, 4, 5, 6 and 7. The plots are made as follows.

A limited number of rows is presented (top for the best). Each row corresponds to an optimizer. Optimizers are ranked by their scores. The score of an optimizer $o$ in a testbed $T$ is the frequency, averaged over all problems $p \in T$ and over all optimizers $o'$ included in the experiments, of $o$ outperforming $o'$ on $p \in T$. The problems differ in the objective function, parameterization, computational budget $T$, dimension $d$, presence of random rotation or not, and the degree of parallelization. "o outperforming o' on a function $f$" means that $f(o.recommend) < f(o'.recommend)$, with a truth value of $\frac{1}{2}$ in case of tie. Similarly to [22], the comparison is made among implementations, not among abstract algorithms; the detailed implementations are freely available and open sourced in [46] or in modules imported there. The columns correspond to the same optimizers, but there are more optimizers included. Still, not all optimizers are shown. Between parentheses, we can see how many cases were run out of how many instances exist; e.g. 244/252 means that there are 252 instances, but only 244 were successfully run. A failure can be due to a bug, or, in most cases, to the timeout - some optimizers become computationally expensive in high-dimensional problems and fail to complete.

Detailed setup and reproducibility. The detailed setup of our benchmarks is available online\(^2\). We did not change any setting. Our plots are those produced automatically and periodically by Nevergrad. In the artificial testbeds, Nevergrad considers cases with different numbers of critical variables (i.e. variables which have an impact on the fitness functions) and of useless variables (possibly zero). There are also rotated cases and non-rotated cases. The detailed setup is at the URL above. We provide a high-level presentation of benchmarks in the present document.

Statistical significance. Each experiment contains several settings. The total number of repetitions varies, but there are always at least 200 random repetitions (cumulated over the different settings) for each result displayed in our plots, and at least 5 repetitions per setting. Importantly, Nevergrad periodically reruns all benchmarks with all algorithms: we can see independent reruns of all results presented in the present paper at http://dl.fbaipublicfiles.com/nevergrad/allxps/list.html, plus additional experiments (including a so-called YAWIDEBBOB which extends YABBOB) and results are essentially the same.

3.1 YABBOB: Yet Another Black-Box Optimization Benchmark

YABBOB is a benchmark of black-box optimization problems. It roughly approximates BBOB [22], without exactly sticking to it. Consequently, CMA performs the best on it. YABBOB has the following advantages. It is part of a maintained and easy to use platform. Optimization algorithms are natively included in an open sourced platform. Everyone can rerun everything. There is a noisy optimization counterpart, without the issues known in BBOB [6, 7, 15, 16].
There are several variants, parallel or not, noisy or not, high-dimensional or not, large computational budget or not. The same platform includes a large range of real world testbeds, switching form artificial to real world is just one parameter to change in the command line.

3.2 Training: artificial testbeds

We designed Shiwa by handcrafting an algorithm as shown in Fig. 1, in which the constants have been tuned on YABBOB (Fig. 2) by trial-and-error. Then, variants of YABBOB, namely YANOISYBBOB, YAHDBBOB, YAPARABBOB, YABIGBBOB, corresponding to the counterpart one with noise, the high-dimensional one, the parallel one, the big computational budget, respectively. Shiwa is good in all categories, often performing the best, and always competing decently with the best in that category.
Figure 4: Results on specific homogeneous families of problems (to be continued in Fig. 5). Shiwa performs well overall.

Table 1: YABBOB testbed and variants. The objective functions are Hm, Rastrigin, Griewank, Rosenbrock, Ackley, Lunacek, DeceptiveMultimodal, BucheRastrigin, Multipeak, Sphere, DoubleLinearSlope, StepDoubleLinearSlope, Cigar, AltCigar, Ellipsoid, AltEllipsoid, StepEllipsoid, Discus, BentCigar, DeceptiveIllcond, DeceptiveMultimodal, DeceptivePath. Full details in [46].

| Name          | Dimensions | Budget            | Parallelism | Translated | Rotated | Noisy |
|---------------|------------|-------------------|-------------|------------|---------|-------|
| YABBOB        | 2, 10, 50  | 50, 200, 800, 3200, 12800 | 1 (sequential) | +N(0, Id) | {yes, no}  | No    |
| YABIGBOB      | 2, 10, 50  | 40000, 80000      | 1 (sequential) | +N(0, Id) | {yes, no}  | No    |
| YAHDBBOB      | 100, 1000, 3000 | 50, 200, 800, 3200, 12800 | 1 (sequential) | +N(0, Id) | {yes, no}  | No    |
| YANOISYBBOB   | 2, 10, 50  | 50, 200, 800, 3200, 12800 | 1 (sequential) | +N(0, Id) | {yes, no}  | Yes   |

Table 2: Other benchmarks used in this paper, besides YABBOB and its variants. Full details in [46].

| Category     | Name          | Description |
|--------------|---------------|-------------|
| Real world   | ARCoating     | Anti-reflective coating optimization. |
|              | Photonics     | Optical properties: Bragg, Morpho and Chirped. |
|              | FastGames     | Tuning of agents playing the game of war, Batawaf, GuessWho, BigGuessWho Flip. |
|              | MLDA          | Machine learning and data analysis testbed [20] |
|              | Realworld     | Includes many of the above and others, e.g. traveling salesman problems. |
| Artificial   | Multiobjective| 2 or 3 objective functions among Sphere, Ellipsoid, Hm and Cigar in dimension 6 or 7, sequential or 100 workers, budget from 100 to 5900. |
|              | Manyobjective | Similar with 6 objective functions. |
|              | Deceptive     | Similar with 6 objective functions. |
|              | Multimodal    | Hm, Rastrigin, Griewank, Rosenbrock, Ackley, Lunacek, DeceptiveMultimodal in dimension 3 to 25, sequential, budget from 3000 to 100000 |
|              | Paramultimodal| Similar with 1000 workers. |
|              | Illcond       | Cigar, Ellipsoid, dimension 50, budget 100 to 10000, both rotated and not rotated. |
|              | SPSA          | Sphere, Cigar, various translations, strong noise. |
3.3 Illustrating: homogeneous families of benchmarks.

We here consider tests which are not directly from YABBOB variants. Some functions are common to some of YABBOB; this section, as opposed to the next one, is not intended to be a completely independent test: these tests are either related to YABBOB or to real world tests from the next section. This section illustrates the behavior of Shiwa on various families of functions, for analysis purposes. Results are presented in Fig. 4, 5 and 6 and 7. \( T \) and \( d \) in the captions refer to the computational budget and dimension, respectively. We keep the name of these experiments as in [46]. The context of Fig. 7 (“illcondi” problem in Nevergrad) is as follows: smooth ill-conditioned problems, namely Cigar and Ellipsoid, both rotated and unrotated, in dimension 50 with moderate computational budget (from 100 to 10000). Unsurprisingly, for these problems for which ruggedness is null and the challenge is to tackle ill-conditioning within a moderate computational budget, SQP and Cobyla outperform all methods based on random exploration: Shiwa (and its variant Urchin) can compete, without knowing anything about the objective functions, because, based on the low computational budget it switches to a chaining of CMA and Powell or to Cobyla (depending on the budget, see Fig. 1), and not to classical evolutionary methods alone.
3.4 Testing: real world testbeds
We test our method on the “realworld” collection of problems in [46]. This contains Traveling Salesman Problems (TSPs), Power Systems Management, Photonics, ARCoating (design of anti-reflective coatings in optics [13]), Machine Learning and Data Applications (MLDA [20]). Results are presented in Fig. 8, and problems in this figure are all independent of YABBOB variants which were used for fine-tuning Shiwa. Photonics and ARCoating problems are one of our contributions. The Photonics problems are truly evolutionary in the sense that the best solutions correspond to optical structures which occur in nature [3] and have thus been produced by evolution. These testbeds are characterized by a very large number of local minima because these structures present potentially a lot of resonances. Finally, these problems are particularly modular, the different parts of the structures playing different roles without being truly independent.

4 CONCLUSIONS
We designed Shiwa (Section 2.4.2), a versatile optimization algorithm tackling optimization problems on arbitrary domains (continuous or discrete or mixed), noisy or not, parallel or oneshot or sequential, and outperforming many methods thanks to a combination of a wide range of optimization algorithms. The remarks in
We met classes of functions for which we failed to become better with dimension ranging from 60 to 80 and computational without degrading more classical settings. The extension to noisy well. Shiwa can also take into account the presence of noise (with-ill-conditioned cases: we get good results for any of these cases, happy with the results in parallel, one-shot, high-dimensional or our code is integrated in [46] and publicly available.

Limitations. design of Shiwa, Shiwa performed well. Reproducibility matters: all ing a wide range of problems which have never been used in the outperforms most one-shot optimization methods. In Fig. 8, cover-photonics problems, and on ill-conditioned quadratic problems, and on YABBOB, is excellent on real world benchmarks, on the new methods. As an example of versatility, Shiwa outperforms CMA for extracting the best of each method and outperforming existing type of variables, computational budget, dimension, parallelism, our knowledge of the noise intensity, rarely available in real life), of problems, for which no homogeneous method alone can perform well. Shiwa can also take into account the presence of noise (without knowledge of the noise intensity, rarely available in real life), type of variables, computational budget, dimension, parallelism, for extracting the best of each method and outperforming existing methods. As an example of versatility, Shiwa outperforms CMA on YABBOB, is excellent on real world benchmarks, on the new photonics problems, and on ill-conditioned quadratic problems, and outperforms most one-shot optimization methods. In Fig. 8, covering a wide range of problems which have never been used in the design of Shiwa, Shiwa performed well. Reproducibility matters: all our code is integrated in [46] and publicly available. Limitations. We met classes of functions for which we failed to become better on the first, without degrading on the second. Typically, we are happy with the results in parallel, one-shot, high-dimensional or ill-conditioned cases: we get good results for any of these cases, without degrading more classical settings. The extension to noisy cases or discrete cases was also straightforward and without draw-back. But for hard multimodal settings vs ill-conditioned unimodal functions, we did our best to choose a realistic compromise but could not have something optimal for both. The no-free lunch [52] actually tells us that we can not be universally optimal. Additional research in the active selection part might help.

Further work. 1. We consider taking into account, as a problem feature, the computational cost of the objective function, the presence of constraints, and, in particular for being more versatile regarding multimodal vs monomodal problems and rugged vs smooth problems, using more active (i.e. online) portfolios, or fitness analysis. 2. A challenge is to recursively analyze the domain for combining different algorithms on different groups of variables: this is in progress. 3. Splitting algorithms are present in [46]. We feel that they are quite good in high dimension and could be part of Shiwa. 4. Particle Swarm Optimization is excellent in some difficult cases: we try to leverage it by integrating it into Shiwa. 5. We did not present our experiments in discrete settings: they can however be retrieved in the section entitled “wide-discrete” of http://dl.baipublicfiles.com/nevergrad/alltests/list.html. We see there the good performance of tools using softmax, including Shiwa. We did not include this in the present paper because, in the context of limited space, there was not enough diversity in the experiments. A thorough analysis of this method, rather new and beyond the scope of the present work dedicated to algorithm selection, is left as further work. 6. Shiwa was mainly designed by scientific knowledge, and a bit by trial-and-error on YABBOB. We guess much better results could be obtained by a more systematic analysis, e.g. numerical optimization on a wider counterpart of YABBOB. 7. Integrating e.g. [33] for high-dimensional cases is also a possibility.

ACKNOWLEDGMENTS

Author J. Liu was supported by the National Key R&D Program of China (Grant No. 2017YFC0804003), the National Natural Science Foundation of China (Grant No. 61906083), the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017TQ07X386), the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. JCYJ2019080912403553), the Shenzhen Science and Technology Program (Grant No. KQTD201611251435531) and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).

REFERENCES

[1] Artelys. 2019. Artelys Knitro, winner of the 2019 BBComp edition. https://www.artelys.com/news/solvers-news/knitro-bbccomp-winner/
[2] Emanouil I Atanassov. 2004. On the discrepancy of the Halton sequences. Math. Balkanica (NS) 18, 1-2 (2004), 15–32.
[3] Mamadou Aliou Barry, Vincent Berthier, Bodo D. Wilts, Marie-Claire Cambourieux, Rémi Pollès, Olivier Teytaud, Emmanuel Centeno, Nicolas Biais, and Antoine Moreau. 2018. Evolutionary algorithms converge towards evolved biological photonic structures. arXiv:1808.04689 [physics.optics]
[4] Nicolas Baskiotis and Michèle Sebag. 2004. C4.5 competence map: a phase transition-inspired approach. In Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004.
[5] Hans-Georg Beyer. 2001. The Theory of Evolution Strategies. Springer, Heidelberg.
[6] Hans-Georg Beyer. 2012. http://lists.irit.fr/pipermail/bbob-discuss/2012-April/00270.html.
[7] Hans-Georg Beyer. 2012. http://lists.irit.fr/pipermail/bbob-discuss/2012-April/00258.html.
