Black-Box Optimization Revisited: Improving Algorithm Selection Wizards through Massive Benchmarking

Laurent Meunier, Herilalaina Rakotoarison, Pak Kan Wong, Baptiste Roziere,
Jérémy Rapin, Olivier Teytaud, Antoine Moreau, Carola Doerr

***Abstract*—Existing studies in black-box optimization suffer from low generalizability, caused by a typically selective choice of problem instances used for training and testing of different optimization algorithms. Among other issues, this practice promotes overfitting and poor-performing user guidelines. We address this shortcoming by introducing in this work a general-purpose algorithm selection wizard that was designed and tested on a previously unseen breadth of black-box optimization problems, ranging from academic benchmarks to real-world applications, from discrete over numerical to mixed-integer problems, from small to very large-scale problems, from noisy over dynamic to static problems, etc. Not only did we use the already very extensive benchmark environment available in Nevergrad, but we also extended it significantly by adding a number of additional benchmark suites, including Pyomo, Photonics, LSGO, and MuJoCo. Our wizard achieves competitive performance on all benchmark suites. It significantly outperforms previous state-of-the-art algorithms on some of the suites, including YABBOB and LSGO. Its excellent performance is obtained without any task-specific parametrization.**

The algorithm selection wizard, all of its base solvers, as well as the benchmark suites are available for reproducible research in the open-source Nevergrad platform.

***Index Terms*—Black-Box Optimization, Benchmarking**

I. Introduction: State of the Art

Many real-world optimization challenges are effectively black-box problems; i.e., the main source of information when solving them is the evaluation of solution candidates. These evaluations often require simulations or even physical experiments. Black-box optimization methods are thus widely applied in practice, with a particularly growing impact in machine learning [1], [2], to the point that they are considered a key research area of artificial intelligence. Black-box optimization algorithms are typically easy to implement and easy to adjust to different problem types. To achieve peak performance, however, proper algorithm selection and configuration are key, since black-box optimization algorithms have complementary strengths and weaknesses [3], [4]. But while automated algorithm selection has become standard in SAT solving [5] and AI planning [6], a manual selection and configuration of the algorithms—often entirely based on users previous experience and not necessarily on broader performance data—is still predominant in the broader black-box optimization context. To reduce the bias inherent to such manual choices, and to support the automation of algorithm selection and configuration, sound comparisons of the different black-box optimization approaches are needed. Existing benchmarking suites, however, are rather selective in the problems they cover. This leads to specialized algorithm frameworks whose performance suffer from poor generalizability. We address this flaw in black-box optimization by presenting a high-performing algorithm selection wizard, ABBO (Automated Black-Box Optimization). ABBO uses very basic information about the problem and the available computational resources to select one or several algorithms, which are run for the allocated budget of function evaluations. The wizard was developed within the Nevergrad platform [7], which we have significantly extended for this work to obtain an even broader set of benchmark problems.

In summary, our key contributions are as follows: **(1) Algorithm Selection Wizard ABBO:** Our algorithm selection technique, ABBO can be seen as an extension of the Shiwa wizard presented in [8]. The wizard takes as input information on the *problem* (the dimension of the search domain, the type and range of each variable, and their order), the *presence of noise* in the evaluation (but not its intensity), and the *computational resources* that are available to solve the problem (the budget and the degree of parallelism, i.e., number of solution candidates that can be evaluated simultaneously). Based on this input, the wizard outputs one or several algorithms that it suggests to execute on the given problem. ABBO uses three types of selection techniques: *passive algorithm selection* (choosing an algorithm as a function of a priori available features [9], [8]), *active algorithm selection* (a bet-and-run strategy which runs several algorithms for some time and stops all but the strongest after a predefined number of evaluations [10], [11], [12], [13], [14], [15], [4]), and *chaining* (running several algorithms in turn, in an a priori defined order [16]).

Our wizard selects from and combines a very large number of base algorithms, among them algorithms suggested in [17], [18], [19], [20], [21], [8], [22], [23], [24], [25], [26]. We compare the performance of ABBO to all these algorithms, as well as to its predecessor Shiwa, and to all other algorithms

Manuscript received August xx, 2021; revised xx yy, zz.
This work was supported by the Paris Ile-de-France Region.
Laurent Meunier, Baptiste Roziere, Jérémy Rapin, and Olivier Teytaud are with Facebook AI Research, Paris, France. Herilalaina Rakotoarison is with TAU, LRI, INRIA, Université Paris-Saclay, Orsay, France. Pak Kan Wong is with The Chinese University of Hong Kong. Antoine Moreau is with the Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, Clermont-Ferrand, France. Carola Doerr is a CNRS researcher at LIP6, Sorbonne Université, Paris, France.

available in Nevergrad.

(2) Benchmark Collection: By integrating a number of additional benchmark suites into the Nevergrad platform, we obtain a huge benchmark collection that covers a previously unseen breadth of black-box optimization problems, ranging from academic benchmarks to real-world applications, from discrete over numerical to mixed-integer problems, from small to very large-scale problems, from noisy over dynamic to static problems, etc.

Structure of the Paper: We motivate in Sec. II why we have chosen to develop ABBO within the Nevergrad benchmarking environment and how we have extended it. Sec. III summarizes ABBO and discusses differences to previous versions. Experimental results can be found in Sec. IV. Sec. V concludes our paper with an outlook to future work.

Availability of Data and Code: Our algorithm selection wizard ABBO, its base solvers, and the benchmark collection have all been merged to the **main Nevergrad master**, where they are available for reproducible, open-source research. Nevergrad periodically reruns all algorithms and makes all data available on the **public dashboard** [27]. Note that ABBO is called **NGOpt8** within the Nevergrad environment, to allow for better version control in the latter: NGOpt is the name of the latest version of the optimization wizard, the current version (July 2021) is NGOpt14. Since Nevergrad is developing at fast pace, we have saved a **frozen version of its code base**. Together with the **performance data** used for this paper and the **videos** illustrating the performance of the policies learned for MuJoCo, this code is available on zenodo [73].

II. Sound Black-Box Optimization Benchmarking

We discuss in this section which features we consider desirable for sound benchmarking, and how different suites address these. This discussion motivates our decision to design and to evaluate ABBO within the Nevergrad environment [7]. Tab. I summarizes how some of the common benchmarking environments address the properties discussed below.

A. Desirable Properties for Sound Benchmarking

Generalization: The most obvious issue in terms of generalization is the statistical one: we need sufficiently many experiments for conducting valid statistical tests and for evaluating the robustness of algorithms' performance. However, this is probably not the main issue. A biased benchmark, excluding large parts of the real world needs, leads to biased conclusions, no matter how many experiments we perform. Inspired by [32] in the case of image classification, and similar to the spirit of cross-validation for supervised learning, we use a much broader collection of benchmark problems for evaluating algorithms in an unbiased manner. Another subtle issue in terms of generalization is the case of instance-based choices of (hyper-)parameters: an experimenter modifying the algorithm or its parameters specifically for each instance can easily achieve considerable performance improvements. In this paper, we consider that only the following problem properties are known in advance (and can hence be used for algorithm selection and configuration): the dimension of the domain, the type and range of each variable, their order, the presence of noise (but not its intensity), the budget, and the degree of parallelism (i.e., number of solution candidates that can be evaluated simultaneously). To mitigate the common risk of over-tuning, we evaluate algorithms on a broad range of problems, from academic benchmark problems to real-world applications. Each algorithm runs on all benchmarks without any change or task-specific tuning.

Large scale: Since practical problems can reach very large dimensions, we consider it desirable to include benchmark suites that comprise such problems. A "+" in Table I indicates that the collection provides benchmark problems in dimension $\geq 1\,000$.

Translations: The search point zero frequently plays a special role in optimization. For example, complexity penalization often "pushes" towards zero. Also, large values in a neural network lead to saturation [33]: then, we get a plateau and cannot learn from the samples. In artificial experiments, several classical test functions have their optimum at $(0, \ldots, 0)$. To avoid misleading conclusions, it is now a standard procedure, advocated in particular in [28], to randomly translate the objective functions. Concretely, we consider that there is translation when optima are randomly translated by a $\mathcal{N}(0, \sigma^2)$ shift. This property is mainly interesting for artificially created benchmarks, but is unfortunately not always applied.

Symmetrizations / Rotations: Some optimization methods may perform well on separable objective functions but degrade significantly when optimizing non-separable functions. If the dimension of a separable objective function is d, these methods can reduce the objective function into d one-dimensional optimization processes [34]. Therefore, [28], [35] proposed that objective functions should be rotated to generate more difficult non-separable objective functions. In [36], the importance of dummy variables, which are not invariant under rotation, was pointed out. Several references in the genetic algorithms literature, including [37], argue that rotations may not always be the right approach, in particular when the order of the variables carries a meaning. Nevergrad uses rotations, but separates the rotated and non-rotated cases in its evaluation, allowing users to focus on the setting of their choice. Assuming an optimum at 0 up to a translation step, we consider *rotation* as the replacement of the function $x \mapsto f(x)$ by $x \mapsto f(M(x))$ for a randomly selected rotation matrix M. We speak of *symmetrization* when $x \mapsto f(x)$ is replaced by $x \mapsto f(S(x))$, where S is a randomly chosen diagonal matrix whose entries are either 1 or -1.

One-line reproducibility: Where reproducibility requires significant coding, it is unlikely to be of great use outside of a very small set of specialists. One-line reproducibility is given when the effort to reproduce an entire experiment does not require more than the execution of a single line. This is possible in Nevergrad, as an example *<python -m nevergrad.benchmark yabbob –plot >* will reproduce YABBOB results on 30 cores.

Periodically automated dashboard: Some platforms do not collect the algorithms, which severely limits their reproducibility, as their implementations may not be available

TABLE I
PROPERTIES OF SELECTED BENCHMARK COLLECTIONS (DETAILS IN THE MAIN TEXT). "+" MEANS THAT THE FEATURE IS PRESENT, "-" THAT THE FEATURE IS MISSING, AND "NA" MEANS THAT IT IS NOT APPLICABLE.

Testbed	BBOB [28]	MuJoCo [29]	LSGO [30]	BBComp [31]	Nevergrad [7]
Large scale	-	NA	+	-	+
Translations	+	NA	+	+	+
Symmetrizations / rotations	+	NA	+	-	+
One-line reproducibility	-	-	-	-	+
Periodically automated dashboard	NA	NA	NA	NA	+
Complex or real-world	-	+	-	+	+
Available open source and no licensing issues	+	-	+	+	+
Ask/tell/recommend framework	-	NA	+	+	+
Human excluded / client-server	-	-	-	+	-

for public comparison. An automated and periodically rerun dashboard mitigates this risk. It is also convenient because new problems can be added "on the go" without causing problems, as all algorithms will be executed on all these new problem instances.

Complex or real-world: Benchmarks that contain real-world optimization problems, or at least complex simulators are desirable to evaluate our methods in realistic environments. MuJoCo is an example of a complex simulator.

Open sourced / no license: Another important aspect of benchmarking environments is whether or not algorithms, problems, and data are available under an open source agreement. BBOB does not collect algorithms, MuJoCo requires a license, BBComp is no longer maintained. As part of our work we integrated MuJoCo into Nevergrad, making it available to a broad public, since users can upload their algorithms in Nevergrad and they will be run on all benchmarks, including MuJoCo.

Ask/tell/recommend framework: Formalizing the concept of numerical optimization is typically made through the formalism of oracles or parallel oracles [40]. A recent trend is the adoption of the ask-and-tell format developed in [38]. The bandit literature pointed out that we should distinguish *ask*, *tell*, and *recommend*: the way we choose a point for gathering more information ("ask") is not necessarily close to the way we choose an approximation of the optimum ("recommend"), see [39], [41], [42] for detailed discussions. The difference is particularly important in noisy optimization, where an algorithm that just happens to do one lucky evaluation should not be able to get credit unless it would actively recommend that solution. A closely related issue is that a run with budget T is not necessarily close to the truncation of a run in budget $10T$.

Human excluded / client-server: Whether or not the problem instances are truly black-box. In the proper black-box setting, algorithms can only suggest points and observe function values, but neither the algorithm nor its designer have access to any other information about the problem apart from the number of variables, their type, ranges, and order. It is impossible to repeat experiments for tuning hyperparameters without "paying" the budget of the tuning. Nevergrad does not reproduce the extreme black-box nature of BBComp [31], where the objective function is evaluated on a server and the algorithms really only perform function evaluations over the internet without having access to any other source of information about the problem at hand. Still, by integrating a wide range of benchmarks in a single open-source framework, which, in addition, is periodically re-run, we nevertheless conclude that Nevergrad provides the right environment for the development and the evaluation of ABBO.

B. Benchmarking Suites (Now) Available in Nevergrad

As a result of our work, Nevergrad now includes PBT (a small scale version of Population-Based Training [44]), Pyomo [45], Photonics (problems related to optical properties and nanometric materials), YABBOB and variants, LSGO [30], MLDA [46], PowerSystems, FastGames, 007, Rocket, SimpleTSP, Realworld [7], [8], MuJoCo [29], and others, including a (currently small) benchmark of hyperparameters of Scikit-Learn [47], and Keras-tuning. In this list, underlined means that the benchmark is either new (i.e., created by us), or, in the case of PowerSystems and SimpleTSP, significantly modified compared to previous works, or, in the case of Pyomo, LSGO, and MuJoCo, included for the first time inside Nevergrad. For MuJoCo, we believe that interfacing with Nevergrad is particularly useful, to ensure fair comparisons, which rely very much on the precise setup of MuJoCo. Some more details about the suites will be given in Sec. IV when we discuss results for selected benchmark collections.

III. THE ABBO ALGORITHM SELECTION WIZARD

Base Solvers: Black-box optimization problems are often tackled using evolutionary computation. Evolution strategies [56], [57], [55] have been particularly dominant in the continuous case, in experimental comparisons based on the Black-Box Optimization Benchmark BBOB [28] or variants thereof. Parallelization advantages [1] are particularly appreciated in the machine learning context. Differential Evolution [19] is a key component of most winning algorithms in competitions based on variants of Large Scale Global Optimization (LSGO [30]). LSGO is more based on correctly identifying a partial decomposition and scaling to $\geq 1\,000$ variables, whereas BBOB focuses (mostly, except [58]) on ≤ 40 variables. Mathematical programming techniques [20], [21], [59], [23] are rarely used in the evolutionary computation world, but they have won competitions [23] and significantly improved evolution strategies through memetic methods [60]. Methods focused on MuJoCo [61], [62] have rarely been tested on other benchmarks such as BBOB or LSGO. Reproducibility in MuJoCo is problematic as its results can depend on

Algorithm 1 High-level overview of ABBO. Selection rules are followed in this order, first match applied. d = dimension, budget b = number of evaluations. Details of ABBO and the configuration of its base solvers are available in the Nevergrad platform [7], where ABBO is listed as NGOpt8.

Case	Choice
A: Discrete decision variables only, noise-free case	
Noisy optimization with categorical variables	Genetic algorithm mixed with bandits [48], [8].
alphabets of size < 5, sequential evaluations	$(1+1)$-Evolutionary Alg. with linearly decreasing stepsize
alphabets of size < 5, parallel case	Adaptive $(1+1)$-Evolutionary Alg. [25].
Other discrete cases with finite alphabets	Convert to the continuous case using SoftMax as in [8] and apply CMandAS2 [49]
Presence of infinite discrete domains	FastGA [24]
B: Numerical decision variables only, evaluations are subject to noise	
$d > 100$	progressive optimization as in [50].
$d \leq 30$	TBPSA [22]
$b > 100$	sequential quadratic programming
Other cases	TBPSA [22]
C: Numerical decision variables only, high degree of parallelism, noise-free	
Parallelism $> b/2$ or $b < d$	MetaTuneRecentering [51]
Parallelism $> b/5$, $d < 5$, and $b < 100$	DiagonalCMA-ES [52]
Parallelism $> b/5$, $d < 5$, and $b < 500$	Chaining of DiagonalCMA-ES (100 asks), then CMA-ES+meta-model [53]
Parallelism $> b/5$, other cases	NaiveTBPSA as in [54]
D: Numerical decision variables only, sequential evaluations, noise-free	
$b > 6\,000$ and $d > 7$	Chaining of CMA-ES and Powell, half budget each.
$b < 30d$ and $d > 30$	$(1+1)$-Evol. Strategy w/ 1/5-th rule [55]
$d < 5$ and $b < 30d$	CMA-ES + meta-model [53]
$b < 30d$	Cobyla [21]
E: other cases. Noisy discrete cases: progressive methods. Other continuous noise-free cases than C and E: apply DE or CMA depending on the dimension (see code and [8]).	

very small details in the implementation. Closer to machine learning, efficient global optimization [63] is widely used, although it suffers from the curse of dimensionality more than other methods [64]. The LAMTCS algorithm presented in [2] applies black-box optimization on MuJoCo, i.e., for the control of various realistic robots [29].

Algorithm Selection Wizards: As mentioned, ABBO combines the various base algorithms available in Nevergrad in three different ways (see Sec. I). Its high-level structure is summarized in Algorithm 1 for selected optimization scenarios. We cannot replicate the full set of case distinctions here. All details are accessible via the implementation of ABBO in Nevergrad. Written in Python, this implementation is comparatively easy to navigate even for users with limited programming experience.

The most relevant predecessor of ABBO is the Shiwa algorithm presented in [8]. Shiwa was also developed within Nevergrad and was shown to outperform each of the base algorithms when averaged over diverse benchmark problems. Also our ABBO entirely relies on the base algorithms as available in Nevergrad; that is, we did not modify the configuration of any method. We have, however, added a number of different algorithms for the development of ABBO, almost exclusively taken from the research literature (see below for details). We therefore acknowledge that the efficiency of ABBO heavily relies on the quality of these base components, which is based on cumulative effort of numerous research teams.

From a high-level perspective, ABBO extends Shiwa by the following features:

(1) Better use of chaining [16] and more intensive use of mathematical programming techniques for the last part of the optimization run, i.e., the local convergence, thanks to Meta-Models (simple quadratic forms trained on the best points, used in the parallel case) and more time spent in Powell's method [20] (in the sequential case). This explains the improvement visible in Sec. IV-A.

(2) Better performance in discrete optimization, achieved, in particular, by adaptive mutation rates (i.e., step size distributions).

(3) Better segmentation of the different cases of continuous optimization.

More concretely, and for the parts of the ABBO that are detailed in Algorithm 1, the main differences between ABBO and Shiwa are as follows. (A) We have added several evolutionary algorithms with variable mutation rates, 36 for the parallel cases and one for the sequential case (using a linearly decreasing mutation rate). We also introduced active algorithm selection ("bet-and-run") with CMandAS2, which—depending on the budget b—races three copies of CMA-ES or two copies of CMA-ES and a (1+1) ES for $b/10$ steps. (B) We use progressive methods (i.e., progressively adding variables in the optimization run, starting at a small set and then growing to the entire set of variables, as in [50]) for high-dimensional cases, and we use Sequential Quadratic Programming (SQP) [23] when the budget is sufficient for training a quadratic model. (C) We make use of the space filling design MetaTuneRecentering proposed in [51]: we use it in the highly parallel case, but also in the sequential setting if the budget is smaller than the dimension. (D) We also use meta-models for some small dimensional cases. Overall, meta-models are helping our algorithms in the continuous setting except for sequential or high-dimensional cases.

IV. Experimental Results

When presenting results on a single benchmark function, we present the average objective function value for different budget values. When a collection comprises multiple benchmark problems, we present the aggregated experimental results with two distinct types of plots:

(1) Loss: normalized average (over all runs) objective value for each budget, averaged over all problems. The normalized objective value is the average objective value linearly rescaled to $[0, 1]$: then we normalize over different problems.

(2) Heatmaps, showing for each pair (x, y) of optimization algorithms the frequency at which Algorithm x outperforms Algorithm y. Algorithms are ranked by average winning frequency. For instance, in Figure 1, ABBO wins in 63.8% of the cases against other algorithms, whereas CMA wins 50.4%.

High-Level Overview: Tab. II summarizes the rank of ABBO on some of the benchmark suites. The rank is based on the winning rate in Nevergrad's dashboard [27]. Each of the suites listed in Tab. II comprises several problems and different settings with respect to budget, objective function, possibly dimension, and noise level. We separate benchmarks that were used for designing ABBO from those used for its validation, and those only used for testing.

The "*" symbol marks suites that were used for designing ABBO's predecessor Shiwa. Some of our modifications also improve the performance of Shiwa compared to the version published in [8]; for example, our chaining implies that the $(k+1)$-st code starts from the best point obtained by the k-th algorithm, which significantly improves in particular the chaining CMA-ES+Powell or CMA-ES+SQP. Experiments with "†" in the ranking of Shiwa correspond to this improved version of Shiwa.

Since the submission of this paper, several variants of bandit-based algorithms have been added for high-dimensional noisy optimization. They outperform ABBO, hence its poor rank for these cases.

A. Suites Used for Designing and Validating ABBO

YABBOB (Yet Another Black-Box Optimization Benchmark [49]), is an adaptation of BBOB [28], with extensions such as parallelism and noise management. It contains many variants, including noise, parallelism, high-dimension (prior to [58] BBOB was limited to dimension < 50). Results are available in Figures 1 and 4. The high-dimensional suite is inspired by [30], the noisy one is related to the noisy counterpart of BBOB but implements the difference between ask and recommend as discussed in Sec. II. The parallel one generalizes YABBOB to settings in which several evaluations can be executed in parallel. Results on PARAMULTIMODAL are presented in Fig. 6 (left). In addition, ABBO was run on ILLCONDI & ILLCONDIPARA (ill conditioned functions), HDMULTIMODAL (a multimodal case focusing on high-dimension), NOISY & RANKNOISY (two noisy continuous testbeds), YAWIDEBBOB (a broad range of functions including discrete cases and cases with constraints).

AllDEs and Hdbo are benchmark collections specifically designed to compare DE variants (AllDEs) and high-dimensional Bayesian Optimization (Hdbo), respectively [7]. These benchmark functions are similar to the ones used in YABBOB. Many variants of DE (resp. BO) are considered. Results are presented in Fig. 5. They show that the performance of ABBO, relatively to DE or BO, is consistent over a wide range of parametrizations of DE or BO, at least in their most classical variants, which are all available in Nevergrad for empirical comparisons.

Realworld: A test of ABBO is performed on the Realworld optimization benchmark suite proposed in [7]. This suite includes testbeds from MLDA [46] and from [8]. Results for this suite, presented in Fig. 6, confirm that ABBO performs well also on benchmarks that were not explicitly used for its design. However, this benchmark was used for designing Shiwa, which was the starting point for the design of ABBO. A rigorous cross-validation, on benchmarks totally independent from the design of Shiwa, is provided in the next sections.

B. New Benchmark Suites Used Only for Evaluating ABBO

Pyomo is a modeling language in Python for optimization problems [45]. It has been adopted for formulating large models for complex and real-world systems, including energy systems and network resource systems. We implemented an interface to Pyomo for Nevergrad. Experimental results are summarized in Fig. 2. They show that ABBO also performs decently in discrete settings and in constrained cases.

Additional new artificial and real-world functions: LSGO [30] combines various functions into an aggregated testbed including composite highly multimodal functions. Correctly decomposing the problem is essential. Various implementations of LSGO exist; in particular, the Octave and C++ versions do not match exactly for F3/F6/F10. We match the C++ version, which is the one used in [30]. For F7, there is a difference between the code and the paper and we match the code rather than the paper. Following [30], our implementation comprises functions with subcomponents (i.e., groups of decision variables) having non-uniform sizes and non-uniform, even conflicting, contributions to the objective function. We also present experimental results on **SequentialFastgames** from the Nevergrad benchmarks, and three newly introduced benchmarks, namely **Rocket**, **Simple TSP** (a set of traveling salesman problems, where a vector $x \in \mathbb{R}^d$ is converted into a permutation σ by letting $\sigma(i)$ be the index of the i-th largest element in x (ties broken at random)), and **power systems** (unit commitment problems [65]). Experimental results are presented in Figs. 2, 7, and 8, respectively. They show that ABBO performs well on new benchmarks, never used for its design nor for that of the low-level heuristics used inside ABBO.

MuJoCo: Some articles [67], [2] studied the MuJoCo testbeds [29] in the black-box setting. MuJoCo tasks correspond to control problems. Defined in [2], [61], the objective is to learn a linear mapping from states to actions. It turned out that the scaling of the variables is critical [61]: following the recommendation from [51] to sample close to zero in the high dimensional setting, we chose to initialize all the variables of the problem with a variance decaying with the dimension, for all methods run in Fig. 3. We remark that ABBO and Shiwa perform well, even when compared to gradient-based methods, while having the advantage of being applicable to settings in which gradients are not available. In comparison to gradient-based methods, black-box methods do not require computation of the gradient, and hence, save computational time.

We use the same experimental setup as [2] (linear policy, offline whitening of states). We get better results than LA-MCTS, in a setting without using any expensive surrogate model (Tab. III). Our runs with CMA-ES and Shiwa are

TABLE II
RANK OF ABBO, SHIWA, AND CMA-ES ON SELECTED BENCHMARK SUITES. SEE MAIN TEXT FOR AN EXPLANATION OF SYMBOLS.

Benchmark	**Use for ABBO**	**# of configs**	**ranking**			**ABBO best competitor**
			ABBO	Shiwa	CMA-ES	
HDBO	Designing	24	2/21	$1^{\dagger}$	2	Shiwa
PARAMULTIMODAL	Designing	112	**1/27**	$3^{\dagger}$	6	DiagonalCMA-ES [52]
Realworld	Designing	486	**1/6**	$2^{\dagger}$	3	Shiwa
Illcondi	Designing	12	**1/24**	$3^{\dagger}$	8	Cobyla
Illcondipara	Designing	12	5/28	$7^{\dagger}$	3	DiagonalCMA-ES
YABBOB	Designing*	630	**1/8**	2	5	Shiwa
YAPARABBOB	Designing*	630	**1/6**	4	5	MetaModel
YAHDBBOB	Designing*	378	2/19	3	18	$(1+1)$-ES
YANOISYBBOB	Designing*	630	2/11	6	10	SQP
YAHDNOISYBBOB	Designing*	630	4/24	2	13	SQP
YASMALLBBOB	Designing*	378	**1/8**	2	7	Shiwa
HdMultimodal	Validation	42	**1/14**	$2^{\dagger}$	4	Shiwa
Noisy	Validation	96	16/28	$19^{\dagger}$	NA	RecombiningOptimisticNoisyDiscrete$(1+1)$
RankNoisy	Validation	72	4/25	NA	19	ProgD13
AllDEs	Validation	60	**1/28**	$2^{\dagger}$	3	Shiwa
Pyomo	Evaluating	104	**1/19**	$3^{\dagger}$	10	Shiwa
Rocket	Evaluating	13	5/18	$4^{\dagger}$	3	DiagonalCMA-ES [52]
SimpleTSP	Evaluating	52	3/15	$2^{\dagger}$	7	PortfolioDiscrete$(1+1)$
Seq. Fastgames	Evaluating	20	3/28	$4^{\dagger}$	23	OptimisticDiscrete$(1+1)$
LSGO	Evaluating	45	**1/6**	$4^{\dagger}$	6	MiniLHSDE
Powersystems	Evaluating	48	10/26	NA	25	$(1+1)$-ES

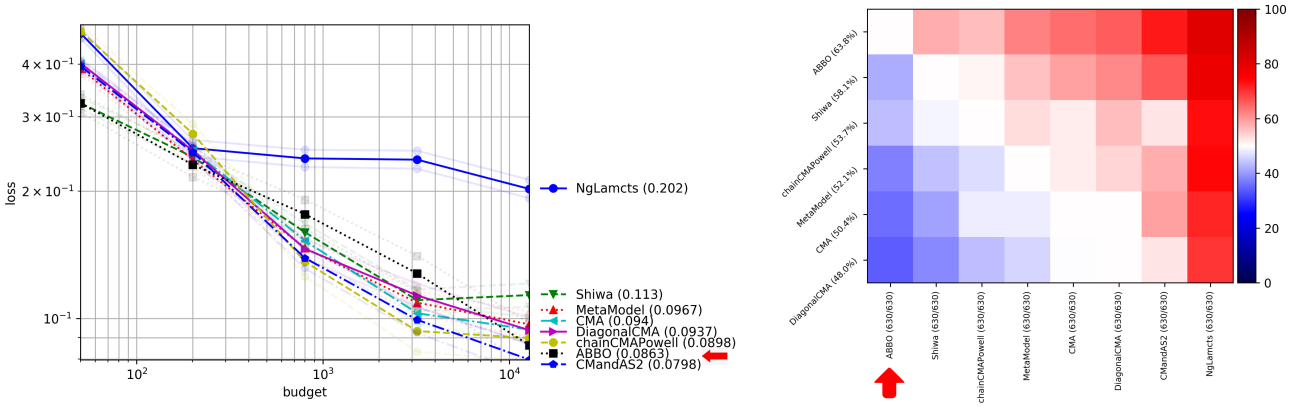

Fig. 1. Average normalized loss and heatmap for YABBOB. Additional plots for High-dimensional (HD), NoisyHD, and Large budgets are available in Fig. 4. Other variants include parallel, differences of budgets, and combinations of those variants, with excellent results for ABBO.

better than those in [2]. We acknowledge that LMRS [67] outperforms our method on all MuJoCo tasks, using a deep network as a surrogate model: however, we point out that a part of their code is not open sourced, making the experiments not reproducible. In addition, when rerunning their repository without the closed source part, it solved Half-Cheetah within budget 56k, which is larger than ours. For Humanoid, the target was reached at 768k, which is again larger than our budget. The results from ABBO are comparable to, and are usually better than (for the 3 hardest problems) the results from LA-MCTS, while ABBO is entirely reproducible. In addition, it runs the same method for all benchmarks and it is not optimized for each task specifically as in [67], [2]. In contrast to ABBO, LA-MCTS [2] uses different underlying regression methods and sampling methods depending on the MuJoCo task, and it is not run on other benchmarks except for some of the HDMULTIMODAL ones. On the latter, ABBO performances are significantly better for Ackley and Rosenbrock in dimension 100 (average results around 100 and 10^{-8} after 10k iterations for Rosenbrock and Ackley respectively for ABBO, vs. 500 and 0.5 in [2]). From the curves in [2] and those presented here in this paper, we expect LA-MCTS to perform well with an adapted choice of parametrization and with a low budget, for tasks related to MuJoCo, whereas ABBO is adapted for wide ranges of tasks and budgets.

As mentioned at the end of the introduction, videos illustrating the performance of the learnt policies are available at [73].

V. CONCLUSIONS

We have introduced in this paper ABBO, an improved algorithm selection wizard that significantly improves upon its predecessor Shiwa [8]. For the development and the evaluation of ABBO we have considerably extended the Nevergrad platform by adding several real-world and academic benchmark suites. All our work is available in the master branch of

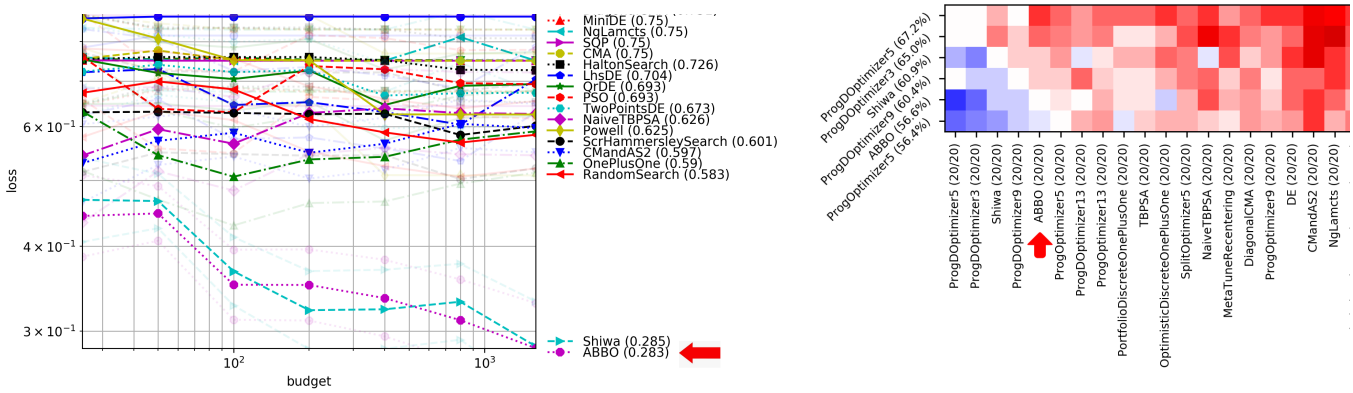

Fig. 2. Additional problems: Pyomo (left figure, covering Knapsack, P-median and others) and SequentialFastgames (on the right, presented as heatmaps due to the high noise. Subsumes GuessWho, War, Batawaf, Flip). Rockets, SimpleTSP, PowerSystems, and LSGO plots are available in Figs. 7, and 8. Pyomo and SimpleTSP include discrete variables. Pyomo includes constraints. Rocket, PowerSystems, SequentialFastGames are based on open source simulators.

TABLE III

RESULTS ON MUJOCO FOR A LINEAR POLICY IN THE BLACK-BOX SETTING FROM [2] AND REFERENCES THEREIN. WE COMPARE VARIOUS PUBLISHED RESULTS TO RESULTS FROM ABBO. TWO LAST COLUMNS = AVERAGE REWARD FOR THE MAXIMUM BUDGET TESTED IN [2], NAMELY 1K, 4K, 4K, 40K, 30K, 40K, RESPECTIVELY. "IOA" = ITERATIONS ON AVERAGE FOR REACHING THE TARGET. "ITER" = ITERATIONS FOR TARGET REACHED FOR MEDIAN RUN. "*" REFERS TO PROBLEMS FOR WHICH THE TARGET WAS NOT REACHED BY [2]: THEN BR MEANS "BEST RESULT IN 10 RUNS". ABBO REACHES THE TARGET FOR HUMANOID AND ANT WHEREAS PREVIOUS (BLACK-BOX) PAPERS DID NOT; WE GET NEARLY THE SAME IOA FOR HOPPER AND HALFCHEETAH (NEVERGRAD COMPUTED THE EXPECTED VALUE INSTEAD OF COMPUTING THE IOA, SO WE CANNOT COMPARE EXACTLY; SEE FIG. 3 FOR CURVES). ABBO IS SLOWER THAN LA-MCTS ON SWIMMER. NOTE THAT WE KEEP THE SAME METHOD FOR ALL BENCHMARKS WHEREAS LA-MCTS MODIFIED THE ALGORITHM FOR 3 ROWS. ON HDMULTIMODAL, ABBO PERFORMS BETTER THAN LA-MCTS, AS DETAILED IN THE TEXT, AND AS CONFIRMED IN [2], WHICH ACKNOWLEDGES THE POOR RESULTS OF LA-MCTS FOR HIGH-DIMENSIONAL ACKLEY AND ROSENBROCK.

Task	Target	LA-MCTS results	ABBO result	LA-MCTS avg reward	ABBO avg reward
Swimmer-v2	325	132 ioa	around 450 iter	358	**365**
Hopper-v2	3120	2897 ioa	around 3 000 iter	**3292**	1787
HalfCheetah-v2	3430	3877 ioa	around 4 000 iter	3227	**4730**
Walker2d-v2*	4390	BR: 3314 (not reached)	BR: **4398, budget** $<$ 64 000 **(reached!)**	2769	**2949**
Ant-v2*	3580	BR: 2791 (not reached)	BR: **5325, budget** $<$ 32 000 **(reached!)**	2511	**3532**
Humanoid-v2*	6 000	BR: 3384 (not reached)	BR (budget 5 00000): **4870**	2511	**4620**

Nevergrad, where it is available for reproducible, open-source research. ABBO is listed as NGOpt8 in Nevergrad.

Despite its simplicity, ABBO shows very promising performance across the whole benchmark suite, often outperforming the previous state-of-the-art, problem-specific solvers. Highlights of our performance comparison include: (a) by solving 5 of the 6 MuJoCo cases without any task-specific hyperparameter tuning, ABBO outperforms LA-MCTS, which was specialized for each single task, (b) ABBO outperforms Shiwa on YABBOB and its variants, which is the benchmark suite that was used to design Shiwa in the first place, (c) ABBO is also among the best methods on LSGO and almost all other benchmarks.

Future work: Nevergrad implements most of the desirable features outlined in Sec. II, with one notable exception, the true black-box setting, which other benchmark environments have implemented through a client-server interaction [31]. A possible combination between our platform and such a challenge, using the dashboard to publish the results, could be useful, to offer a meaningful way for cross-validation. Further improving ABBO is on the roadmap. In particular, we are experimenting with the automation of the still hand-crafted selection rules. Note, though, that it is important to us to maintain a high level of interpretability, which we consider key for a wide acceptance of the wizard. Another avenue for future work is a proper configuration of the low-level heuristics subsumed by ABBO. At present, some of them are merely textbook implementations, and significant room for improvement can therefore be expected. Newer variants of CMA-ES [68], [69], [70], of LMRS [67], recent Bayesian optimization libraries (e.g. [71]), as well as per-instance algorithm configuration such as [72] are not unlikely to result in important improvements for various benchmarks. We also plan on extending the benchmark collection available through Nevergrad further, both via interfacing existing benchmark collections/problems and by designing new benchmark problems ourselves.

REFERENCES

[1] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[2] L. Wang, R. Fonseca, and Y. Tian, "Learning search space partition for black-box optimization using Monte Carlo Tree Search," in *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[3] J. R. Rice, "The Algorithm Selection Problem," *Advances in Computers*, vol. 15, pp. 65-118, 1976.

[4] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann, "Automated Algorithm Selection: Survey and Perspectives," *Evolutionary Computation*, pp. 1-47, 2018.

[5] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "SATzilla: Portfolio-based algorithm selection for SAT," *J. Artif. Int. Res.*, vol. 32, no. 1, pp. 565–606, 2008.

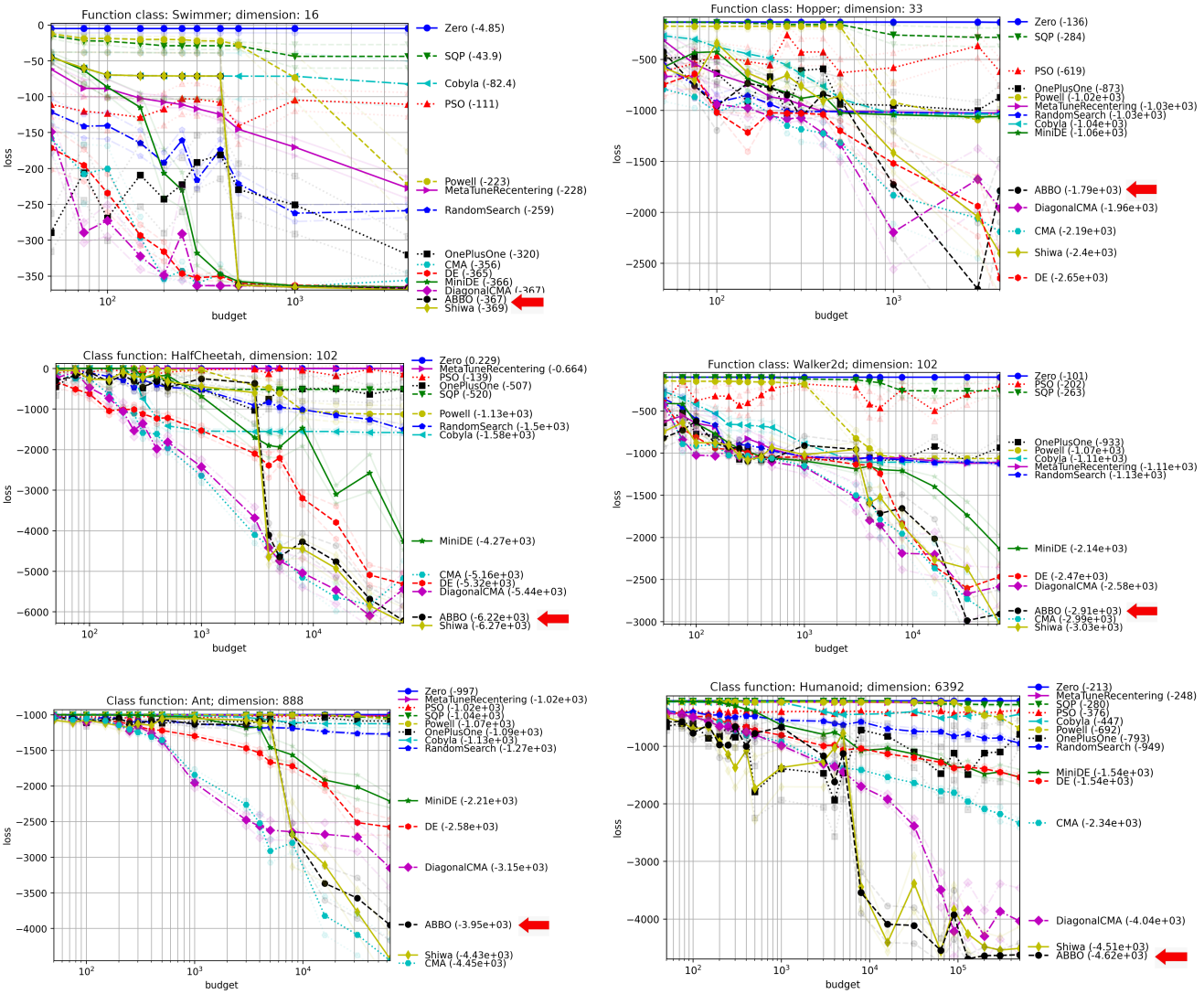

Fig. 3. Results on the MuJoCo testbeds. Dashed lines show the standard deviation. Compared to the state of the art in [2], with an algorithm adapted manually for the different tasks, we get overall better results for Humanoid, Ant, and Walker. We get worse results for Swimmer (could match if we had modified our code for the three easier tasks as done in [2]), similar for Hopper and Cheetah: we reach the target for 5 of the 6 problems (see main text). Runs of Shiwa correspond to the improvement of Shiwa due to chaining, as explained in Sec. III.

[6] M. Vallati, F. Hutter, L. Chrpa, and T. L. McCluskey, "On the effective configuration of planning domain models," in *Proc. of International Joint Conference on Artificial Intelligence (IJCAI'15)*, 2015.

[7] J. Rapin and O. Teytaud, "Nevergrad - A gradient-free optimization platform," https://GitHub.com/FacebookResearch/Nevergrad, 2018.

[8] J. Liu, A. Moreau, M. Preuss, J. Rapin, B. Roziere, F. Teytaud, and O. Teytaud, "Versatile black-box optimization," in *Proc. of Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2020, pp. 620–628.

[9] N. Baskiotis and M. Sebag, "C4.5 competence map: a phase transition-inspired approach," in *Proc. of International Conference on Machine Learning (ICML*, 2004.

[10] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph, "Exploratory Landscape Analysis," in *Proc. of Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2011, pp. 829-836.

[11] E. Pitzer and M. Affenzeller, "A Comprehensive Survey on Fitness Landscape Analysis," in *Recent Advances in Intelligent Engineering Systems*. Springer, 2012, pp. 161-191.

[12] M. Fischetti and M. Monaci, "Exploiting erraticism in search," *Operations Research*, vol. 62, no. 1, pp. 114–122, 2014.

[13] K. M. Malan and A. P. Engelbrecht, "A Survey of Techniques for Characterising Fitness Landscapes and Some Possible Ways Forward," *Information Sciences (JIS)*, vol. 241, pp. 148-163, 2013.

[14] M. A. Muñoz Acosta, Y. Sun, M. Kirley, and S. K. Halgamuge, "Algorithm Selection for Black-Box Continuous Optimization Problems: A Survey on Methods and Challenges," *Information Sciences (JIS)*, vol. 317, pp. 224-245, 2015.

[15] M.-L. Cauwet, J. Liu, B. Rozière, and O. Teytaud, "Algorithm portfolios for noisy optimization," *Annals of Mathematics and Artificial Intelligence*, vol. 76, no. 1-2, pp. 143–172, 2016.

[16] D. Molina, M. Lozano, and F. Herrera, "Memetic algorithm with local search chaining for continuous optimization problems: A scalability test," in *2009 Ninth International Conference on Intelligent Systems Design and Applications*, 2009, pp. 1068–1073.

[17] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser and et al., "SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python," *arXiv*, p. arXiv:1907.10121, 2019.

[18] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 11, no. 1, 2003.

[19] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

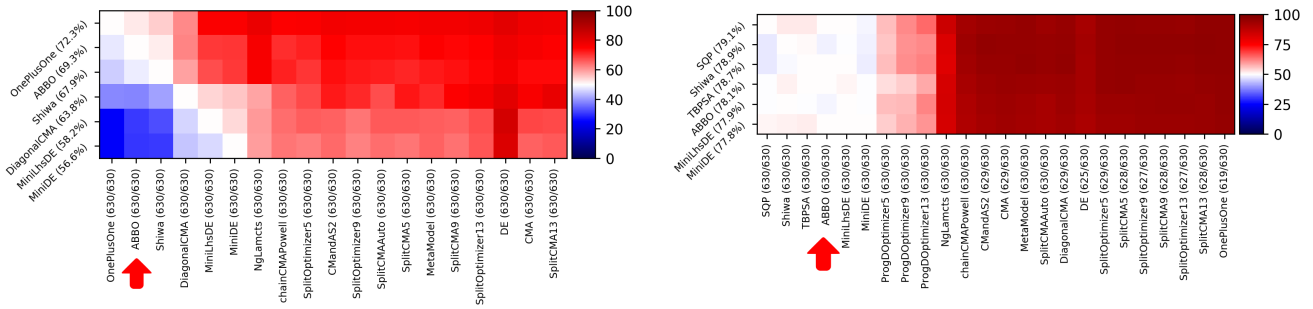

Fig. 4. YAHDBBOB (dimension ≥ 50) and YANOISYHDBBOB (noisy + dimension ≥ 50) heatmaps.

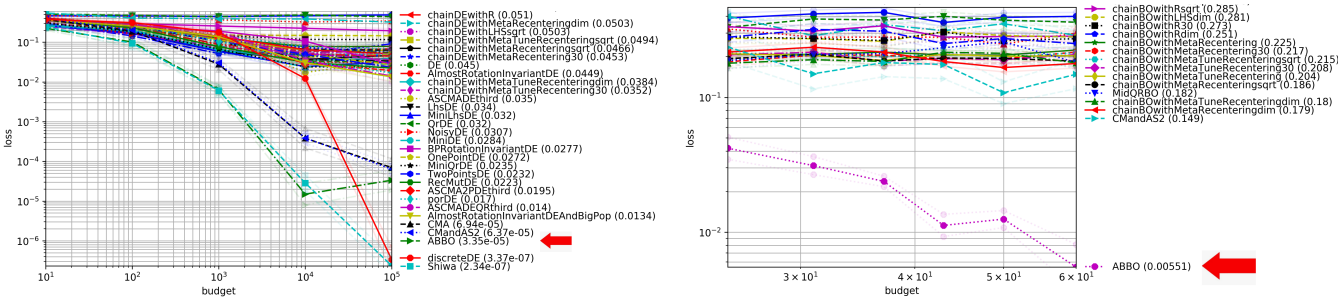

Fig. 5. ABBO vs specific families of optimization algorithms (DE on the left in dimension 5, 20 and 100; and BO in dimension 20 on the right) on Cigar, Hm, Ellipsoid, Sphere functions. Not all run algorithms are mentioned, for short. Bayesian optimization (Nevergrad uses [66]), often exploring boundaries first, is outperformed in high dimension [2].

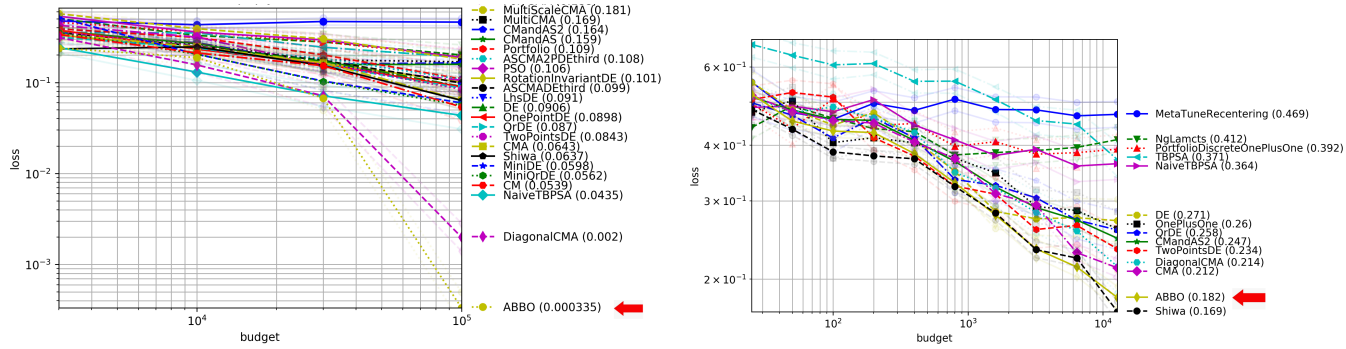

Fig. 6. Left: experiments for the parallel multimodal setting PARAMULTIMODAL. Budget up to 100 000, parallelism 1 000, Ackley+Rosenbrock+DeceptiveMultimodal+Griewank+Lunacek+Hm. Right: Realworld benchmark from Nevergrad: games, Sammon mappings, clustering, small traveling salesman instance, small power systems.

[20] M. J. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, vol. 7, no. 2, pp. 155-162, 1964.

[21] ——, *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*. Springer, 1994, pp. 51-67.

[22] M. Hellwig and H.-G. Beyer, "Evolution under strong noise: A self-adaptive evolution strategy can reach the lower performance bound - the pcCMSA-ES," in *Proc. of Parallel Problem Solving from Nature (PPSN)*. Springer, 2016, pp. 26–36.

[23] S. Artelys, "Artelys sqp wins the bbcomp competition," 2015. [Online]. Available: https://www.ini.rub.de/PEOPLE/glasmtbl/projects/bbcomp/index.html

[24] B. Doerr, H. P. Le, R. Makhmara, and T. D. Nguyen, "Fast genetic algorithms," in *Proc. of Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2017, pp. 777–784.

[25] B. Doerr, C. Doerr, and J. Lengler, "Self-adjusting mutation rates with provably optimal success rules," in *Proc. of Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2019, p. 1479–1487.

[26] D. Dang and P. K. Lehre, "Self-adaptation of mutation rates in non-elitist populations," in *Proc. of Parallel Problem Solving from Nature (PPSN)*, 2016, pp. 803–813.

[27] J. Rapin and O. Teytaud, "Dashboard of results for Nevergrad platform," https://dl.fbaipublicfiles.com/nevergrad/allxps/list.html, 2020.

[28] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking 2009: Experimental setup," INRIA, France, Tech. Rep. RR-6828, 2009.

[29] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Proc. of International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2012, pp. 5026–5033.

[30] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Qin, "Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization," Available: https://www.researchgate.net/publication/261562928_Benchmark_Functions_for_the_CEC%272013_Special_Session_and_Competition_on_Large-Scale_Global_Optimization, 2013.

[31] I. Loshchilov and T. Glasmachers, "Black box optimization competition," 2017. [Online]. Available: https://www.ini.rub.de/

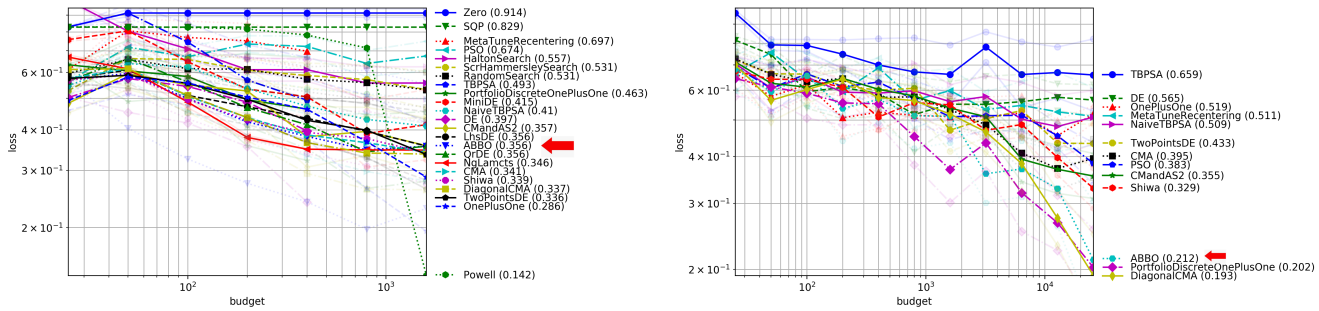

Fig. 7. Additional problems (1): on left, Rocket (26 continuous variables, budget up to 1600, sequential or parallelism 30) and on right, SimpleTSP (10 to 1 000 decision variables).

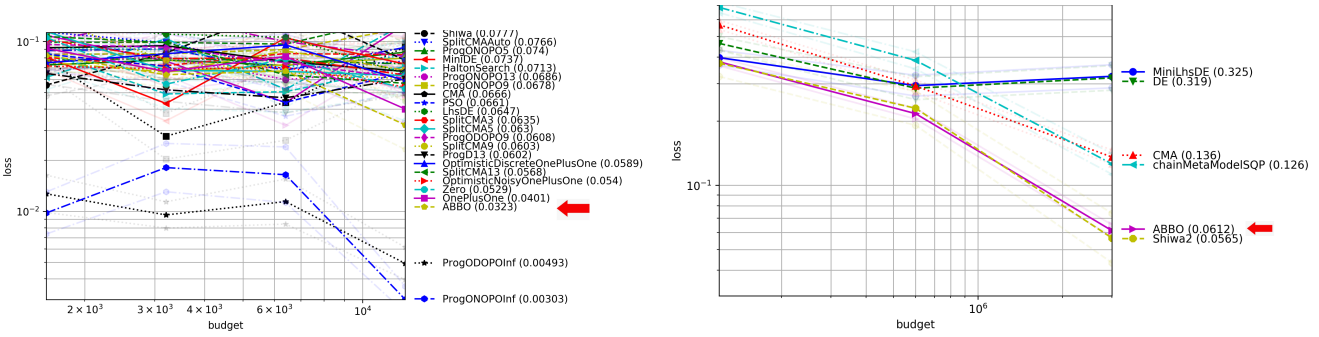

Fig. 8. Additional problems (2): on left, PowerSystems (1806 to 9646 neural decision variables) and on right, LSGO (mix of partially separable, overlapping, shifted cases as in [30]).

PEOPLE/glasmtbl/projects/bbcomp/index.html

[32] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do CIFAR-10 classifiers generalize to CIFAR-10 ?" *arXiv*, p. arXiv:1806.00451, 2018.

[33] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. of Artificial Intelligence and Statistics (AISTATS'10)*. PMLR 9:249-256, 2010.

[34] R. Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms," *BioSystems*, vol. 39, no. 3, pp. 263–278, 1996.

[35] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, and A. Auger, "Impacts of Invariance in Search: When CMA-ES and PSO Face Ill-Conditioned and Non-Separable Problems," *Applied Soft Computing*, vol. 11, pp. 5755–5769, 2011.

[36] O. Bousquet, S. Gelly, K. Karol, O. Teytaud, and D. Vincent, "Critical hyper-parameters: No random, no cry," *arXiv*, p. arXiv:1706.03200, 2017.

[37] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[38] Y. Collette, N. Hansen, G. Pujol, D. Salazar, and R. Le Riche, "On object-oriented programming of optimizers - examples in scilab," in *Multidisciplinary Design Optimization in Computational Mechanics*. Wiley, pp. 499-538, 2010.

[39] S. Bubeck, R. Munos, and G. Stoltz, "Pure exploration in finitely-armed and continuous-armed bandits," *Theor. Comput. Sci.*, vol. 412, no. 19, pp. 1832–1852, 2011.

[40] H. Rogers, *Theory of Recursive Functions and Effective Computability*. MIT Press, 1987.

[41] R. Coulom, "Clop: Confident local optimization for noisyblack-box parameter tuning," in *Advances in Computer Games*. Springer, 2012, pp. 146–157.

[42] J. Decock and O. Teytaud, "Noisy optimization complexity under locality assumption," in *Proc. of Foundations of Genetic Algorithms (FOGA)*. ACM, 2013, pp. 183–190.

[43] A. A. Chotard, A. Auger, and N. Hansen, "Cumulative Step-size Adaptation on Linear Functions: Technical Report," Inria, 2012.

[44] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu, "Population based training of neural networks." *arXiv*, p. arXiv:1711.09846, 2017.

[45] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola, *Pyomo-optimization modeling in python*. Springer, 2017.

[46] J. Rapin, M. Gallagher, P. Kerschke, M. Preuss, O. Teytaud, "Exploring the MLDA benchmark on the nevergrad platform," in *Proc. of Genetic and Evolutionary Computation Conference (GECCO, Companion material)*. ACM, 2018, pp 1888–1896.

[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[48] V. Heidrich-Meisner and C. Igel, "Hoeffding and bernstein races for selecting policies in evolutionary direct policy search," in *Proc. of International Conference on Machine Learning (ICML)*. ACM, 2009, pp. 401–408.

[49] J. Rapin, P. Dorval, J. Pondard, N. Vasilache, M. Cauwet, C. Couprie, and O. Teytaud, "Openly revisiting derivative-free optimization," in *Proc. of Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2019, pp. 267–268.

[50] V. Berthier, "Progressive differential evolution on clustering real world problems," in *Proc. of Conference on Artificial Evolution*. Springer, 2016, pp. 71–82.

[51] L. Meunier, C. Doerr, J. Rapin, and O. Teytaud, "Variance reduction for better sampling in continuous domains," in *Proc. of Parallel Problem Solving from Nature (PPSN)*. Springer, 2020, pp. 154–168.

[52] R. Ros and N. Hansen, "A simple modification in CMA-ES achieving linear time and space complexity," in *Proc. of Parallel Problem Solving from Nature (PPSN)*. Springer, 2008, pp. 296–305.

[53] A. Auger, M. Schoenauer, and O. Teytaud, "Local and global order 3/2 convergence of a surrogate evolutionary algorithm," in *Proc. of Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2005, pp. 857–864.

[54] M.-L. Cauwet and O. Teytaud, "Population control meets Doob's martingale theorems: The noise-free multimodal case," in *Proc. of Genetic and*

Evolutionary Computation Conference (GECCO, Companion material). ACM, 2020, p. 321–322.
[55] I. Rechenberg, *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Holzboog Verlag, 1973.
[56] H.-G. Beyer and H.-P. Schwefel, "Evolution Strategies - A Comprehensive Introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.
[57] H.-G. Beyer, *The Theory of Evolution Strategies*. Springer, 2001.
[58] K. Varelas, O. A. ElHara, D. Brockhoff, N. Hansen, D. M. Nguyen, T. Tusar, and A. Auger, "Benchmarking large-scale continuous optimizers: The BBOB-largescale testbed, a COCO software guide and beyond," *Appl. Soft Comput.*, vol. 97, no. Part, p. 106737, 2020.
[59] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, pp. 308–313, 1965.
[60] N. J. Radcliffe and P. D. Surry, "Formal memetic algorithms," in *Evolutionary Computing: AISB Workshop*. Springer, LNCS 865, 1994, pp. 1–16.
[61] H. Mania, A. Guy, and B. Recht, "Simple random search provides a competitive approach to reinforcement learning," *arXiv*, p. arXiv:1803.07055, 2018.
[62] O. Sener and V. Koltun, "Learning to guide random search," in *Proc. of International Conference on Learning Representations (ICLR)*. Available: https://openreview.net/forum?id=B1gHokBKwS
[63] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.
[64] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2012, pp. 2951–2959.
[65] N. P. Padhy, "Unit commitment-a bibliographical survey," *IEEE Transactions on Power Systems*, vol. 19, no. 2, pp. 1196–1205, 2004.
[66] F. Nogueira, "Bayesian Optimization: Open source constrained global optimization tool for Python," 2014–. [Online]. Available: https://github.com/fmfn/BayesianOptimization
[67] O. Sener and V. Koltun, "Learning to guide random search," in *International Conference on Learning Representations (ICLR)*, 2020. [Online]. Available: https://openreview.net/forum?id=B1gHokBKwS
[68] I. Loshchilov, "A computationally efficient limited memory CMA-ES for large scale optimization," in *Proc. of Genetic and Evolutionary Computation (GECCO)*. ACM, 2014, pp. 397–404.
[69] Y. Akimoto and N. Hansen, "Projection-based restricted covariance matrix adaptation for high dimension," in *Proc. of Genetic and Evolutionary Computation (GECCO)*. ACM, 2016, pp. 197–204.
[70] I. Loshchilov, T. Glasmachers, and H.-G. Beyer, "Large scale black-box optimization by limited-memory matrix adaptation," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 353–358, 2018.
[71] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local Bayesian optimization," in *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 5496–5507.
[72] N. Belkhir, J. Dréo, P. Savéant, and M. Schoenauer, "Per instance algorithm configuration of cma-es with limited budget," in *Proc. of Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2017, p. 681–688.
[73] L. Meunier, H. Rakotoarison, P. K. Wong, B. Roziere, J. Rapin, O. Teytaud, A. Moreau and C. Doerr "Black-Box Optimization Revisited: Improving Algorithm Selection Wizards through Massive Benchmarking," in *Zenodo*. Available: https://doi.org/10.5281/zenodo.4744977

Laurent Meunier is a PhD student at Facebook AI Research and Miles Team in LAMSADE at Université Paris Dauphine. He is co-supervised by Olivier Teytaud (Facebook) and Jamal Atif (Dauphine). Prior to his PhD studies, he was a student of the French École Polytechnique. He works on theoretical aspects of black-box optimization, and robustness of machine learning algorithms to adversarial examples.

Herilalaina Rakotoarison received a Master degree in Computer Science from Paris-Saclay University in 2017. He is currently pursuing a PhD in Machine Learning at INRIA Saclay under the TAU (Tackling the Underspecified) team. His research is about leveraging algorithm selection and hyper-parameter optimization strategies to automate the design of machine learning pipelines. He is also involved with the AI4Health ITU Focus Group on benchmarking AI-based malaria diagnosis systems.

Pak Kan Wong received the B.Eng. degree in Information Engineering and the Ph.D. degree in Computer Science and Engineering from the Chinese University of Hong Kong, Hong Kong. He is currently a machine learning software engineer at GoodNotes, which develops one of the best note-taking apps for digital handwritten notes on iPad and iPhone. His research is mainly focused on grammar-based genetic programming and deep neural network, as well as their applications. Besides, he has also contributed to Nevergrad, a platform for derivative-free optimization.

Baptiste Roziere is a PhD student at Facebook AI Research and Université Paris Dauphine in Paris. He worked mostly on latent space optimization for generative adversarial networks and on machine learning for programming languages.

Jérémy Rapin is a research engineer at Facebook AI Research in Paris. He is a maintainer and core developer of Nevergrad, a derivative-free optimization platform. Before that, he completed a PhD in signal processing at Paris-Saclay University in 2014, and he applied deep learning in a startup to help physicians interpret electrocardiograms.

Olivier Teytaud is a research scientist at Facebook AI Research in Paris. He has been working in power systems, arithmetics, games, control, optimization, computer vision. He currently contributes to Nevergrad (https://github.com/facebookresearch/nevergrad), a platform for derivative-free optimization, and Polygames, which recently won the first games against a top level player in the game of Hex and in the game of Havannah. He currently combines evolutionary optimization and deep learning, including applications in computer vision, and maintains NGOpt, a powerful optimization wizard. He claims that NGOpt is the most versatile black-box optimization algorithm with a strong performance on a wide range of benchmarks.

Antoine Moreau is an associate professor at Université Clermont-Auvergne, currently an Academy CAP 20-25 chair holder (16-IDEX-0001 CAP 20-25). He has been working in the field of nanophotonics since 2003, studying numerically how light can be manipulated using either dielectric or metallic nanostructures. He enjoys doing so very much. He has applied optimization and artificial intelligence to various problems in physics, including molecular clusters, fluid mechanics, photovoltaics or biophotonics. He has recently shown that optimization is able to retrieve naturally occurring photonic structures, contributing to Nevergrad in this context.

Carola Doerr, formerly Winzen, is since 2013 a permanent CNRS researcher at Sorbonne Université in Paris, France. Carola's main research activities are in the mathematical analysis of randomized algorithms, with a strong focus on evolutionary algorithms and other sampling-based optimization heuristics. She is also interested in all aspects of benchmarking these algorithms.

She is associate editor of ACM Transactions on Evolutionary Learning and Optimization, editorial board member of the Evolutionary Computation journal, advisory board member of the Springer Natural Computing Book Series. She was program chair of PPSN 2020, FOGA 2019, and the theory tracks of GECCO 2015 and 2017. Carola is/was guest editor of a special issue in TEVC and two special issues in Algorithmica. She is a founding and coordinating member of the Benchmarking Network, an initiative created to consolidate and to stimulate activities on benchmarking sampling-based optimization heuristics.