BLACK-BOX OPTIMIZATION REVISITED: IMPROVING ALGORITHM SELECTION WIZARDS THROUGH MASSIVE BENCHMARKING

Anonymous authors
Paper under double-blind review

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 different optimization algorithms. Among other issues, this practice promotes overfitting and poor-performing user guidelines. To address this shortcoming, we propose in this work a benchmark suite which covers a broad range of black-box optimization problems, ranging from academic benchmarks to real-world optimization problems, from discrete over numerical to mixed-integer problems, from small to very large-scale problems, from noisy over dynamic to static problems, etc. We demonstrate the advantages of such a broad collection by deriving from it Automated Black Box Optimizer (ABBO), a general-purpose algorithm selection wizard. Using three different types of algorithm selection techniques, ABBO achieves competitive performance on all benchmark suites. It significantly outperforms previous state of the art on some of them, including the MuJoCo collection, YABBOB, and LSGO. A single algorithm therefore performed best on these three important benchmarks, without any task-specific parametrization. The benchmark collection, the wizard, its low-level solvers, as well as all experimental data are fully reproducible and open source. They are made available as a fork of Nevergrad, termed OptimSuite.

1 INTRODUCTION: STATE OF THE ART

Many real-world optimization challenges are black-box problems; i.e., instead of having an explicit problem formulation, they can only be accessed through the evaluation of solution candidates, often requiring simulations or even physical experiments. Black-box optimization methods are particularly widespread in machine learning (Salimans et al., 2016; Wang et al., 2020), to the point that it is 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 (Rice, 1976; Smith-Miles, 2009; Kotthoff, 2014; Bischl et al., 2016; Kerschke & Trautmann, 2018; Kerschke et al., 2018). But whereas automated algorithm selection has become standard in SAT solving (Xu et al., 2008) and AI planning (Vallati et al., 2015), a manual selection and configuration of the algorithms 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. Addressing this flaw in black-box optimization, we present a unified benchmark collection which covers a previously unseen breadth of problem instances. We use this collection to develop a high-performing algorithm selection and configuration wizard, ABBO. ABBO uses high-level problem characteristics to select one or several algorithms, which are run for the allocated budget of function evaluations. Originally derived from a subset of the available benchmark collection, in particular YABBOB, the excellent performance of ABBO generalizes across all settings of our broad benchmark suite. Implemented as a fork of Nevergrad (Rapin & Teytaud, 2018), the benchmark collection, the wizard, and all performance data are

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 in (Anonymous, 2020).

Case	Choice
Discrete decision variables only	
Noisy optimization with categorical variables	Genetic algorithm mixed with bandits (Heidrich-Meisner & Igel, 2009; Liu et al., 2020).
alphabets of size < 10, sequential evaluations	$(1+1)$-Evolutionary Alg. with linearly decreasing stepsize
alphabets of size < 10, parallel case	Adaptive $(1+1)$-Evolutionary Alg. (Doerr et al., 2019).
Other discrete cases with finite alphabets	Convert to the continuous case using SoftMax as in (Liu et al., 2020) and apply CMandAS2 (Rapin et al., 2019)
Presence of infinite discrete domains	FastGA (Doerr et al., 2017)
Numerical decision variables only, evaluations are subject to noise	
$d > 100$	progressive optimization as in (Berthier, 2016).
$d \leq 30$	TBPSA (Hellwig & Beyer, 2016)
$b > 100$	sequential quadratic programming
Other cases	TBPSA (Hellwig & Beyer, 2016)
Numerical decision variables only, high degree of parallelism	
Parallelism $> b/2$ or $b < d$	MetaTuneRecentering (Meunier et al., 2020)
Parallelism $> b/5$, $d < 5$, and $b < 100$	DiagonalCMA (Ros & Hansen, 2008)
Parallelism $> b/5$, $d < 5$, and $b < 500$	Chaining of DiagonalCMA (100 asks), then CMA+meta-model (Auger et al., 2005)
Parallelism $> b/5$, other cases	NaiveTBPSA as in (Cauwet & Teytaud, 2020)
Numerical decision variables only, sequential evaluations	
$b > 6000$ and $d > 7$	Chaining of CMA and Powell, half budget each.
$b < 30d$	$(1+1)$-Evol. Strategy w/ 1/5-th rule (Rechenberg, 1973b)
$d < 5$ and $b < 30d$	CMA-ES + meta-model (Auger et al., 2005)
$b < 30d$	Cobyla (Powell, 1994)
For all other cases and all details, please refer to the source code	

open source. The algorithms are automatically rerun at certain time intervals and all data is exported to the public dashboard (Rapin & Teytaud, 2020). For ICLR reviewers, all code is available, thanks to github-anonymizer, at (Anonymous, 2020).

In summary, our contributions are as follows. **(1) OptimSuite Benchmark Collection:** OptimSuite combines several contributions that recently led to improved reliability and generalizability of black-box optimization benchmarking, among them LSGO (Li et al., 2013), YABBOB (Hansen et al., 2009a; Liu et al., 2020; Anonymous, 2020), Pyomo (Hart et al., 2017; Anonymous, 2020), MLDA (Gallagher & Saleem, 2018), and MuJoCo (Todorov et al., 2012; Mania et al., 2018). **(2) Algorithm Selection Wizard ABBO:** Our algorithm selection technique, ABBO, can be seen as an extension of the Shiwa wizard presented in (Liu et al., 2020). It uses three types of selection techniques: *passive algorithm selection* (choosing an algorithm as a function of a priori available features (Baskiotis & Sebag, 2004; Liu et al., 2020)), *active algorithm selection* (a bet-and-run strategy which runs several algorithms for some time and stops all but the strongest (Mersmann et al., 2011; Pitzer & Affenzeller, 2012; Fischetti & Monaci, 2014; Malan & Engelbrecht, 2013; Muñoz Acosta et al., 2015; Cauwet et al., 2016; Kerschke et al., 2018)), and *chaining* (running several algorithms in turn, in an a priori defined order (Molina et al., 2009)). Our wizard combines, among others, algorithms suggested in (Virtanen et al., 2019; Hansen & Ostermeier, 2003; Storn & Price, 1997; Powell, 1964; 1994; Liu et al., 2020; Hellwig & Beyer, 2016; Artelys, 2015; Doerr et al., 2017; 2019; Dang & Lehre, 2016). Another core contribution of our work is a sound comparison of our wizard to Shiwa, and to the long list of algorithms available in Nevergrad.

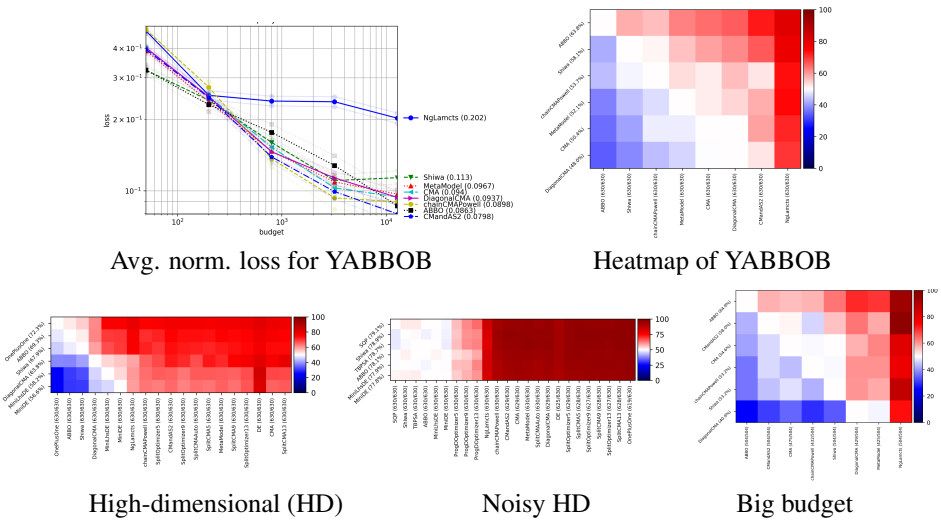
2 Sound Black-Box Optimization Benchmarking

We summarize desirable features and common shortcomings of black-box optimization benchmarks and discuss how OptimSuite addresses these.

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. This, however, is probably not the main issue. A biased benchmark, excluding large parts of the industrial needs, leads to biased conclusions, no matter how many experiments we perform. Inspired by (Recht et al., 2018) 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

Testbed	BBOB	MuJoCo	LSGO	Nevergrad	BBComp	OptimSuite
Large scale	-		+	+		+
Translations	+		+	+	+	+
Symmetrizations / rotations	+		+	+		+
One-line reproducibility	-		-	+		+
Periodic automated dashboard				+		+
Complex or real-world	-	+	-	+		+
Multimodal	+	+	+	+	+	+
Open sourced / no license		-				+
Ask/tell/recommend correct	-		+	+	+	+
Far-optimum	+			+		+
Human excluded / client-server					+	

Table 1: Properties of selected benchmark collections. "+" means that the feature is present, "-" that the feature is missing, and an empty case that it is not applicable. Far-optimum refers to problems with optimum far from the center or on the side of the domain; such benchmarks test the ability of optimization algorithms to answer promptly to a bad initialization (Chotard et al., 2012). "Translations" applies only to artificial benchmarks. "Multimodal" (resp. "symmetrization", "real-world") do not imply that *all* test functions are multimodal (resp. symmetrized, real-world). "Open sourced" refers to open access to most algorithms involved in the published comparison; here, "-" refers to license issues for the benchmark itself.



Avg. norm. loss for YABBOB

Heatmap of YABBOB

High-dimensional (HD)

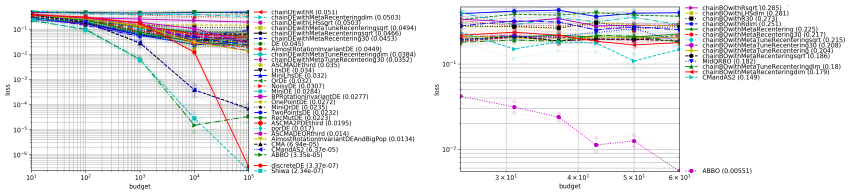
Noisy HD

Big budget

Figure 1: YABBOB and some extensions. Other variants include parallel, differences of budgets and combinations of those variants, with excellent results for ABBO (see Nevergrad's dashboard `https://dl.fbaipublicfiles.com/nevergrad/allxps/list.html`, publicly visible with anonymized entries, for the part of our code which is already merged there).

modifying the algorithm or its parameters specifically for each instance can easily improve results by a vast margin. 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, 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.

Use the ask, tell, and recommend pattern. Formalizing the concept of numerical optimization is typically made through the formalism of oracles or parallel oracles (Rogers, 1987). A recent trend is the adoption of the ask-and-tell format (Collette et al., 2010). The bandit literature pointed out that we should distinguish *ask*, *tell*, and *recommend*: the way we choose a point for gathering more information is not necessarily close to the way we choose an approximation of the optimum (Bubeck et al., 2011; Coulom, 2012b; Decock & Teytaud, 2013). We adopt the following framework: given an objective function f and an *optimizer*, for $i \in \{1, \dots, T\}$, do $x \leftarrow optimizer.ask$ and



AllDEs ($d \in \{5, 20, 100\}$.)
Cigar, Hm, Ellipsoid, Sphere.

HDBO ($d \in \{20, 2000\}$)
Cigar, Hm, Ellipsoid, Sphere.

Figure 2: ABBO vs specific families of optimization algorithms (DE, and BO in the high-dimensional case). Not all run algorithms are mentioned, for short. Bayesian optimization, often exploring boundaries first, is outperformed in high dimension (Wang et al., 2020).

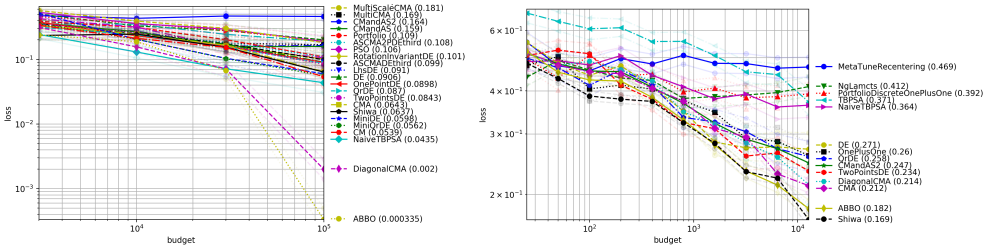


Figure 3: Left: experiments for the parallel multimodal setting PARAMULTIMODAL. Budget up to 100000, parallelism 1000, Ackley+Rosenbrock+DeceptiveMultimodal+Griewank+Lunacek+Hm. Right: Realworld benchmark from Nevergrad: games, Sammon mappings, clustering, small traveling salesman instance, small power systems.

$optimizer.tell(x, f(x))$. Then, evaluate the performance with $f(optimizer.recommend)$. The requirement of a recommend method distinct from the ask is critical in noisy optimization. A debate pointed out some shortcomings in the the noisy counterpart of BBOB (Hansen et al., 2009b) which was assuming that ask = recommend: (Beyer, 2012a;b; Coulom, 2012a) have shown that in the noisy case, this difference was particularly critical, and a framework should allow algorithms to "recommend" differently than they "ask". A related issue is that a run with budget T is not necessarily close to the truncation of a run in budget $10T$.

Translation-invariance. Zero frequently plays a special role in optimization. For example, complexity penalizations often "push" towards zero. In control, numbers far from zero are often more likely to lead to bang-bang solutions (and therefore extract zero signal, leading to a needle-in-the-haystack optimization situation), in particular with neural networks. In one-shot optimization, (Cauwet et al., 2019; Meunier et al., 2020) have shown how much focusing to the center is a good idea in particular in high-dimension. Our experiments in control confirm that the scale of the optimization search is critical, and explains results observed in some optimization papers (Section 4.2). In artificial experiments, several classical test functions have their optimum in zero: it is now a standard procedure, advocated in particular in (Hansen et al., 2009a), to randomly translate the objective functions. This is unfortunately not always applied.

Rotation and symmetrization. Some optimization methods may perform well on separable objective functions but degrade significantly in 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 (Salomon, 1996). Therefore, Hansen et al. (2009a; 2011) have insisted that objective functions should be rotated to generate more difficult non-separable objective functions. However, Bousquet et al. (2017) pointed out the importance of dummy variables, which are not invariant per rotation; and (Holland, 1975) and more generally the genetic algorithms literature insist that rotation does not always makes sense – we lose some properties of a real-world objective function, and in some real-world scenarios rotating would e.g. mix temperature, distance and electric intensity. Permutating the order of variables is also risky, as their order is sometimes critical - k-point crossovers a la Holland (Holland, 1975) typically assume some order of variables, which would be broken. Also, users sometimes rank variables with the most important first – and

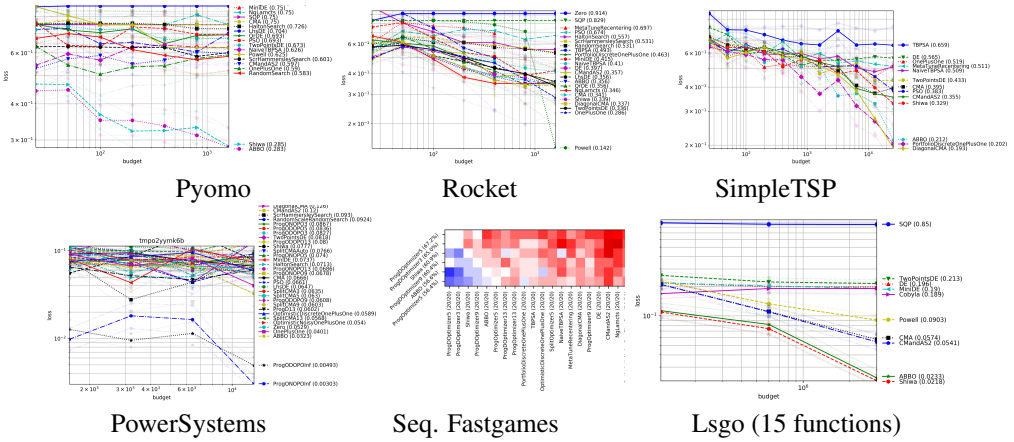


Figure 4: Additional Pyomo problems and real-world problems: Pyomo (covering Knapsack, P-median and others). Rocket (26 continuous variables, budget up to 1600, sequential or parallelism 30), SimpleTSP (10 to 1000 decision variables), PowerSystems (1806 to 9646 neural decision variables), SequentialFastgames (presented as heatmaps due to the high noise: GuessWho, War, Batawaf, Flip). We also add the well known LSGO (mix of partially separable, overlapping, shifted cases as in (Li et al., 2013)). Pyomo and SimpleTSP include discrete variables. Pyomo includes constraints. Rocket, PowerSystems, SequentialFastGames are based on open source simulators and are already merged from OptimSuite to Nevergrad.

some optimization methods do take care of this (Cauwet et al., 2019). We do include rotations, but include both cases, rotated or not. For composite functions which use various objective functions on various subsets of variables, we consider the case with rotations – without excluding the non-rotated case. A possibly better form of symmetrization, which makes sense for replicating an objective function without exact identity, consists in symmetrizing some variables: for example, if the ith variable has range $[a, b]$, we can replace x_i by $b + a - x_i$. Applying this on various subsets of variables leads to 2^d symmetries of an objective function, if the dimension is d. This variation can reduce the bias toward symmetric search operations (Li et al., 2013).

Benchmarking in OptimSuite. We summarize in Table 1 some existing benchmark collections and their (desirable) properties. We inherited various advantages from Nevergrad, namely the automatic rerun of experiments and reproducibility in one-line. Our fork includes PBT (a small scale version of Population-Based Training Jaderberg et al. (2017)), Pyomo Hart et al. (2017), Photonics (problems related to optical properties and nanometric materials), YABBOB and variants, LSGO (Li et al., 2013), MLDA(Gallagher & Saleem, 2018), Rocket, PowerSystems, FastGames, 007, Rocket, SimpleTSP, Realworld (Liu et al., 2020), and others including a (currently small) benchmark of hyperparameters of Scikit-Learn (Pedregosa et al., 2011), all of those being visible for review at the above-mentioned anonymized URL. We note that, at present, we do not reproduce the extreme black-box nature of Loshchilov & Glasmachers (2017).

3 A New Algorithm Selection Wizard: ABBO

Black-box optimization is sometimes dominated by evolutionary computation. Evolution strategies (Beyer & Schwefel, 2002; Beyer, 2001; Rechenberg, 1973a) have been particularly dominant in the continuous case, in experimental comparisons based on the Black-Box Optimization Benchmark BBOB (Hansen et al., 2009a) or variants thereof. Parallelization advantages (Salimans et al., 2016) are particularly appreciated in the machine learning context. However, differential evolution (Storn & Price, 1997) is a key component of all winning algorithms in competitions based on variants of Large Scale Global Optimization (LSGO (Li et al., 2013)), suggesting a significant difference between these benchmarks. In particular, LSGO is more based on correctly identifying a partial decomposition and scaling to ≥ 1000 variables, whereas BBOB focuses (mostly, except (Varelas et al., 2018)) on ≤ 40 variables. Mathematical programming techniques (Powell, 1964; 1994; Nelder & Mead, 1965; Artelys, 2015) are rarely used in the evolutionary computation world,

MuJoCo Problem	Target	SOTA w/o grad	SOTA with grad (in ioa)	ABBO result	LA-MCTS avg reward	ABBO avg reward
Swimmer-v2	325	132 ioa	427 (Ars V2-t)	around 450 iter	358	**383**
Hopper-v2	3120	2897 ioa	1973 (Ars V2-t)	around 3000 iter	**3292**	1787
HalfCheetah-v2	3430	3877 ioa	1707 (Ars V2-t)	around 4000 iter	3227	**4105**
Walker2d-v2*	4390	BR: 3314 (not reached)	14250 (TRPO-nn)	BR: **4398, budget** < 64000 **(Target reached!)**	2769	**2949**
Ant-v2*	3580	BR: 2791 (not reached)	20800 (Ars V2-t)	BR: **5325, budget** < 32000 **(Target reached!)**	2511	**3532**
Humanoid-v2*	6000	BR: 3384 (not reached)	130000 (NG-lin)	BR (budget 500000): **4870**	2511	**4620**

Table 2: State-of-the-art results from Wang et al. (2020) and references therein, compared to results from ABBO. Two last cols = average reward for the maximum budget tested in Wang et al. (2020), namely 1k, 4k, 4k, 40k, 30k, 40k respectively. "ioa" = iterations on average for reaching the target loss. "iter" = iterations for target reached for median run. "*" refer to problem for which the target was never reached in previous (black-box) papers: 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 loss instead of computing the ioa, so we cannot compare exactly; see Fig. 5 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 (Wang et al., 2020), which acknowledges the poor results of LA-MCTS for high-dimensional Ackley and Rosenbrock.

but they sometimes won competitions (Artelys, 2015) and significantly improved evolution strategies through memetic methods (Radcliffe & Surry, 1994). The wide literature in algorithm selection (Rice, 1976; Smith-Miles, 2009; Kotthoff, 2014; Bischl et al., 2016; Kerschke & Trautmann, 2018; Kerschke et al., 2018) was applied to continuous black-box optimization and in a public platform in (Liu et al., 2020): their optimization algorithm combines many optimization methods and outperforms each of them when averaged over diverse test functions. Closer to machine learning, efficient global optimization (Jones et al., 1998) is widely used, though it suffers from the curse of dimensionality more than other methods (Snoek et al., 2012); (Wang et al., 2020) presented a state-of-the-art algorithm in black-box optimization on MuJoCo, i.e. control of various realistic robots (Todorov et al., 2012). We propose ABBO, which extends (Liu et al., 2020) by the following features: (1) Better use of chaining (Molina et al., 2009) 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 (in the parallel case) and more time spent in Powell's method (in the sequential case). This explains the improvement visible in Section 4.1. (2) Better performance in discrete optimization, using additional codes recently introduced in Nevergrad, in particular adaptive step-sizes. (3) Better segmentation of the different cases of continuous optimization. The obtained algorithm selection wizard, ABBO, is presented in Alg. 1.

4 Experimental Results

When presenting results on a single benchmark function, we present the usual average loss for different budget values. When a collection comprises multiple benchmark problems, we present our aggregated experimental results with two distinct types of plots: (1) Average normalized loss for each budget, averaged over all problems. The normalized loss is the loss linearly rescaled to $[0, 1]$. (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. Benchmarks in Sections 4.1-4.2 are included in our experiments.

4.1 Benchmarks in OptimSuite Used for Designing ABBO

YABBOB (Yet Another Black-Box Optimization Benchmark (Rapin et al., 2019)), is an adaptation of BBOB (Hansen et al., 2009a), with extensions such as parallelism and noise management. It contains many variants, including noise, parallelism, high-dimension (BBOB was limited to dimension < 50). Several extensions, for the high-dimensional, the parallel or the big budget case, have been developed: we present results in Fig. 1. The high-dimensional one is inspired by (Li et al., 2013), the noisy one is related to the noisy counterpart of BBOB but correctly implements the difference

between ask and recommend as discussed in Section 2, the parallel one generalizes YABBOB to settings in which several evaluations can be executed in parallel. Results on PARAMULTIMODAL are presented in Fig. 3 (left). In addition, ABBO was run on ILLCONDI & ILLCONDIPARA (ill conditionned 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 (Anonymous, 2020). These benchmark functions are similar to the ones used in YABBOB. Many variants of DE (resp. BO) are considered. Results are presented in Fig. 2. 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. All these variants are publicly visible in Nevergrad and/or in our anonymized branch.

Realworld: A test of ABBO is performed on the Realworld optimization benchmark suite proposed in (Rapin & Teytaud, 2018). This suite includes testbeds from MLDA (Gallagher & Saleem, 2018) and from (Liu et al., 2020). Results for this suite, presented in Fig. 3, 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 basis of our ABBO. A rigorous cross-validation, on benchmarks totally independent from the design of Shiwa, is provided in next sections.

4.2 New Benchmarks in OptimSuite Used Only for Evaluating ABBO

Pyomo is a modeling language (Hart et al., 2017) in Python for optimization problems. It is popular and has been adopted in formulating large models for complex and real-world systems, including energy systems and networked resource systems. We implemented an interface to Pyomo for Nevergrad and enriched our benchmark problems (Anonymous, 2020), which include discrete variables and constraints. Experimental results are shown in Fig. 4. They show that ABBO also performs decently in discrete settings and in constrained cases.

Additional new artificial and real-world functions: LSGO (large scale global optimization) combines various functions into an aggregated difficult testbed including composite highly multimodal functions. Correctly decomposing the problem is essential. Various implementations of LSGO exist; in particular we believe that some of them do not match exactly. Our implementation follows (Li et al., 2013) , which introduces functions with subcomponents (i.e., groups of decision variables) having non-uniform sizes and non-uniform, even conflicting, contributions to the objective function. Furthermore, we present here experimental results on SequentialFastgames from the Nevergrad benchmarks, and three newly introduced benchmarks, namely Rocket, SimpleTsp (a set of traveling salesman problems), power systems (unit commitment problems (Padhy, 2004)). Experimental results are presented in Fig. 4. 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: (Todorov et al., 2012; Wang et al., 2020; Mania et al., 2018) proposed the MuJoCo testbeds in the black-box setting. MuJoCo tasks correspond to control problems. Defined in (Wang et al., 2020; Mania et al., 2018), the objective is to learn a linear mapping from states to actions. It turned out that the scaling is critical (Mania et al., 2018): for reasons mentioned in Section 2, solutions are close to 0. We chose to scale all algorithms at the power of 0.1 the closest to $1.2/d$, for all methods run in Fig. 5. Refer to the comparison shown in Table 2 and Fig. 5: we see that ABBO and Shiwa perform well, including comparatively to gradient-based methods in some cases, while having the ability to work when the gradient is not available. Black-box methods, when the gradient is available, have, in addition, the advantage of saving up the computational cost of the gradient. The algorithm performing best in (Wang et al., 2020) is LA-MCTS. Results from ABBO are comparable to, and usually better than (for the 3 hardest problems), results from LA-MCTS, while ABBO is entirely reproducible and the same method is run for all benchmarks and was not optimized for the task. In contrast to ABBO, (Wang et al., 2020) uses different underlying regression methods and sampling methods depending on the Mujoco task, and is not run on other benchmarks except a part of HDMULTIMODAL. Regarding HDMULTIMODAL, ABBO performances are significantly better for Ackley and Rosenbrock in dimension 100 (expected results around 100 and 10^{-8} after 10k iterations for Rosenbrock and Ackley respectively for ABBO, vs 0.5 and 500 in (Wang et al.,

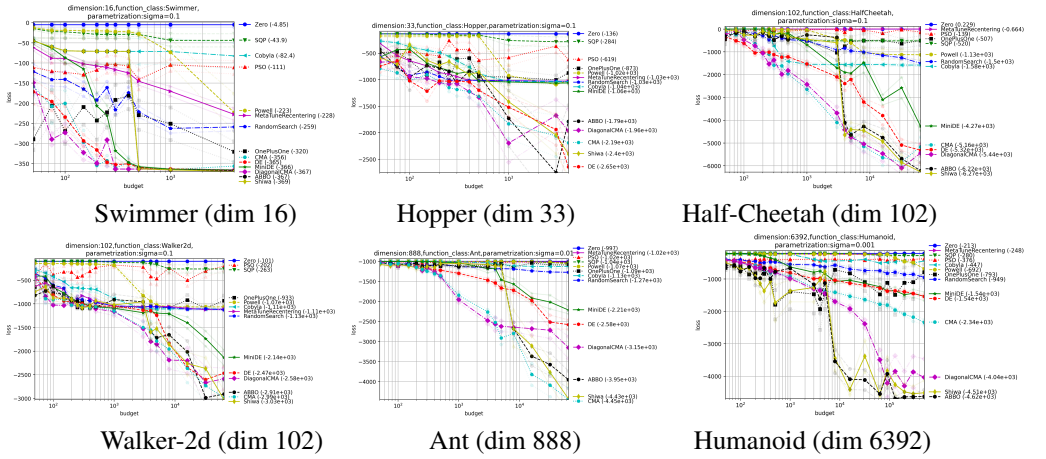


Figure 5: Results on the MuJoCo testbeds. Dashed lines show the standard deviation. Compared to the state of the art in (Wang et al., 2020), with an algorithm adapted manually for the different tasks, we get overall better results for Humanoid, Ant, Walker, worse for Swimmer (could match if we had modified our code for the 3 easier tasks as done in (Wang et al., 2020)), similar for Hopper and Cheetah: we reach the target for 5 of the 6 problems, whereas previous black-box algorithms solved 3 tasks at best.

2020)). From the curves in (Wang et al., 2020) and in the present work, 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 a wide range of tasks and a wide range of budgets.

5 Conclusions

This paper proposes OptimSuite, a very broad benchmark suite composed of real-world and of artificial benchmark problems. OptimSuite is implemented as a fork of Nevergrad (Rapin & Teytaud, 2018), from which it inherits a strong reproducibility: our (Python) code is open source (Anonymous, 2020), tests are rerun periodically, and up-to-date results are available in the public dashboard (Rapin & Teytaud, 2020). A whole experiment can be done as a one-liner. OptimSuite fixes issues common in existing benchmarking environments. The suite subsumes MuJoCo, Pyomo, LSGO, YABBOB, MLDA, and several new real-world problems. We also propose ABBO, an improved algorithm selection wizard. Despite its simplicity, ABBO shows very promising performance across the whole benchmark suite, often outperforming the previous state-of-the-art, problem-specific solvers: (a) ABBO outperforms LA-MCTS on MuJoCo, despite LA-MCTS being strongly specialized on that particular benchmark, solving 5 out of 6 benchmarks including 2 which were never previously solved by black-box methods (b) ABBO outperforms Shiwa on YABBOB and its variants, which were the benchmark used for creating Shiwa in the first place (c) ABBO outperforms all methods from Nevergrad in LSGO except one specific variant of DE (d) ABBO is also the best-performing algorithm or among the best on almost all other benchmarks.

Further work. OptimSuite subsumes most of the desirable features outlined in Section 2, with one notable exception, the true black-box setting, which other benchmark environments have implemented through a client-server interaction (Loshchilov & Glasmachers, 2017). 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; as of today, some of them are merely textbook implementations, some room for improvement can therefore be expected. We also plan on extending OptimSuite further, both through interfacing existing benchmark collections/problems, and by designing new benchmark problems.

REFERENCES

Anonymous. Anonymized github repository with all source code of our work. https://anonymous.4open.science/r/5118cc2c-e2b0-475f-ba8d-3a2971a93acd/, 2020.

SME Artelys, 2015. URL https://www.artelys.com/news/159/16/KNITRO-wins-the-GECCO-2015-Black-Box-Optimization-Competition.

Anne Auger, Marc Schoenauer, and Olivier Teytaud. Local and global order 3/2 convergence of a surrogate evolutionnary algorithm. In *Gecco*, pp. 8, 2005.

Nicolas Baskiotis and Michèle Sebag. 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*, 2004.

Vincent Berthier. Progressive differential evolution on clustering real world problems. In *Revised Selected Papers of the 12th International Conference on Artificial Evolution - Volume 9554*, pp. 71–82. Springer-Verlag New York, Inc., 2016.

Hans-Georg Beyer. *The Theory of Evolution Strategies*. Natural Computing Series. Springer, Heideberg, 2001.

Hans-Georg Beyer. http://lists.lri.fr/pipermail/bbob-discuss/2012-April/000270.html, 2012a.

Hans-Georg Beyer. http://lists.lri.fr/pipermail/bbob-discuss/2012-April/000258.html, 2012b.

Hans-Georg Beyer and Hans-Paul Schwefel. Evolution Strategies - A Comprehensive Introduction. *Natural Computing*, 1(1):3–52, May 2002.

Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Thomas Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. ASlib: A Benchmark Library for Algorithm Selection. *Artificial Intelligence (AIJ)*, 237: 41 – 58, 2016.

Olivier Bousquet, Sylvain Gelly, Kurach Karol, Olivier Teytaud, and Damien Vincent. Critical hyper-parameters: No random, no cry. *CoRR*, abs/1706.03200, 2017.

Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in finitely-armed and continuous-armed bandits. *Theor. Comput. Sci.*, 412(19):1832–1852, 2011.

Marie-Liesse Cauwet and Olivier Teytaud. Population control meets Doob's martingale theorems: The noise-free multimodal case. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, GECCO '20, pp. 321–322. Association for Computing Machinery, 2020.

Marie-Liesse Cauwet, Jialin Liu, Baptiste Rozière, and Olivier Teytaud. Algorithm portfolios for noisy optimization. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):143–172, 2016.

Marie-Liesse Cauwet, Camille Couprie, Julien Dehos, Pauline Luc, Jérémy Rapin, Morgane Riviere, Fabien Teytaud, and Olivier Teytaud. Fully parallel hyperparameter search: Reshaped space-filling. *arXiv preprint arXiv:1910.08406. To appear in Proc. of ICML 2020*, 2019.

Alexandre Adrien Chotard, Anne Auger, and Nikolaus Hansen. Cumulative Step-size Adaptation on Linear Functions: Technical Report. Research report, Inria Saclay, June 2012.

Yann Collette, Nikolaus Hansen, Gilles Pujol, Daniel Salazar, and Rodolphe Le Riche. On object-oriented programming of optimizers - examples in scilab. 01 2010.

Remi Coulom. http://lists.lri.fr/pipermail/bbob-discuss/2012-April/000252.html, 2012a.

Rémi Coulom. Clop: Confident local optimization for noisyblack-box parameter tuning. In *Advances in Computer Games*, pp. 146–157. Springer Berlin Heidelberg, 2012b.

Duc-Cuong Dang and Per Kristian Lehre. Self-adaptation of mutation rates in non-elitist populations. In *Parallel Problem Solving from Nature - PPSN XIV - 14th International Conference*, pp. 803–813, 2016.

Jérémie Decock and Olivier Teytaud. Noisy optimization complexity under locality assumption. In *Proceedings of the Twelfth Workshop on Foundations of Genetic Algorithms XII*, FOGA XII '13, pp. 183–190. ACM, 2013.

Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. Fast genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pp. 777–784. ACM, 2017.

Benjamin Doerr, Carola Doerr, and Johannes Lengler. Self-adjusting mutation rates with provably optimal success rules. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, pp. 1479–1487. Association for Computing Machinery, 2019.

Matteo Fischetti and Michele Monaci. Exploiting erraticism in search. *Operations Research*, 62(1): 114–122, 2014.

Marcus Gallagher and Sobia Saleem. Exploratory landscape analysis of the mlda problem set. In *PPSN'18 workshop*, 2018.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 11(1), 2003.

Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report RR-6828, INRIA, France, 2009a.

Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. *Technical Report PPE, 2009/21*, 01 2009b.

Nikolaus Hansen, Raymond Ros, Nikolas Mauny, Marc Schoenauer, and Anne Auger. Impacts of Invariance in Search: When CMA-ES and PSO Face Ill-Conditioned and Non-Separable Problems. *Applied Soft Computing*, 11:5755–5769, 2011.

William E Hart, Carl D Laird, Jean-Paul Watson, David L Woodruff, Gabriel A Hackebeil, Bethany L Nicholson, and John D Siirola. *Pyomo-optimization modeling in python*, volume 67. Springer, 2017.

Verena Heidrich-Meisner and Christian Igel. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 401–408. ACM, 2009.

Michael Hellwig and Hans-Georg Beyer. Evolution under strong noise: A self-adaptive evolution strategy can reach the lower performance bound - the pcCMSA-ES. In *Parallel Problem Solving from Nature – PPSN XIV*, pp. 26–36. Springer International Publishing, 2016.

John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks, 2017.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, Dec 1998.

Pascal Kerschke and Heike Trautmann. Automated Algorithm Selection on Continuous Black-Box Problems By Combining Exploratory Landscape Analysis and Machine Learning. *Evolutionary Computation*, pp. 1 – 28, 2018.

Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann. Automated Algorithm Selection: Survey and Perspectives. *Evolutionary Computation*, pp. 1 – 47, 2018.

Lars Kotthoff. Algorithm Selection for Combinatorial Search Problems: A Survey. *AI Magazine*, 35(3):48 – 60, 2014.

Xiaodong Li, Ke Tang, Mohammmad Nabi Omidvar, Zhenyu Yang, and Kai Qin. Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization. 01 2013.

Jialin Liu, Antoine Moreau, Mike Preuss, Jeremy Rapin, Baptiste Roziere, Fabien Teytaud, and Olivier Teytaud. Versatile black-box optimization. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, pp. 620–628, 2020.

Ilya Loshchilov and T. Glasmachers. Black box optimization competition, 2017. URL `https://bbcomp.ini.rub.de/`.

Katherine Mary Malan and Andries Petrus Engelbrecht. A Survey of Techniques for Characterising Fitness Landscapes and Some Possible Ways Forward. *Information Sciences (JIS)*, 241:148 – 163, 2013.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning, 2018.

Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory Landscape Analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 829 – 836. ACM, 2011. ISBN 978-1-4503-0557-0.

Laurent Meunier, Carola Doerr, Jérémy Rapin, and Olivier Teytaud. Variance reduction for better sampling in continuous domains. In *Parallel Problem Solving from Nature - PPSN XVI - 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part I*, volume 12269 of *Lecture Notes in Computer Science*, pp. 154–168. Springer, 2020.

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*, pp. 1068–1073, 2009.

Mario Andrés Muñoz Acosta, Yuan Sun, Michael Kirley, and Saman K. Halgamuge. Algorithm Selection for Black-Box Continuous Optimization Problems: A Survey on Methods and Challenges. *Information Sciences (JIS)*, 317:224 – 245, October 2015.

John A. Nelder and Roger Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

Narayana Prasad Padhy. Unit commitment-a bibliographical survey. *IEEE Transactions on Power Systems*, 19(2):1196–1205, 2004.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

Erik Pitzer and Michael Affenzeller. A Comprehensive Survey on Fitness Landscape Analysis. In János Fodor, Ryszard Klempous, and Carmen Paz Suárez Araujo (eds.), *Recent Advances in Intelligent Engineering Systems*, Studies in Computational Intelligence, pp. 161 – 191. Springer, 2012.

Michael J.D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964.

Michael J.D. Powell. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, pp. 51–67. Springer Netherlands, 1994. ISBN 978-94-015-8330-5.

Nicholas J. Radcliffe and Patrick D. Surry. Formal memetic algorithms. In T.C. Fogarty (ed.), *Evolutionary Computing: AISB Workshop*, pp. 1–16. Springer Verlag LNCS 865, 1994.

Jeremy Rapin and Olivier Teytaud. Nevergrad - A gradient-free optimization platform. `https://GitHub.com/FacebookResearch/Nevergrad`, 2018.

Jeremy Rapin and Olivier Teytaud. Dashboard of results for Nevergrad platform. `https://dl.fbaipublicfiles.com/nevergrad/allxps/list.html`, 2020.

Jérémy Rapin, Pauline Dorval, Jules Pondard, Nicolas Vasilache, Marie-Liesse Cauwet, Camille Couprie, and Olivier Teytaud. Openly revisiting derivative-free optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019, Prague, Czech Republic, July 13-17, 2019*, pp. 267–268. ACM, 2019.

Ingo Rechenberg. *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Holzboog Verlag, 1973a.

Ingo Rechenberg. *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Holzboog Verlag, 1973b.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do CIFAR-10 classifiers generalize to CIFAR-10 ?, 2018.

John Rischard Rice. The Algorithm Selection Problem. *Advances in Computers*, 15:65 – 118, 1976.

Hartley Rogers. *Theory of Recursive Functions and Effective Computability*. MIT Press, 1987. ISBN 0262680521.

Raymond Ros and Nikolaus Hansen. A simple modification in CMA-ES achieving linear time and space complexity. In *Parallel Problem Solving from Nature – PPSN X*, pp. 296–305. Springer Berlin Heidelberg, 2008.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016.

Ralf Salomon. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39(3):263–278, 1996.

Kate Amanda Smith-Miles. Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection. *ACM Computing Surveys (CSUR)*, 41:1 – 25, January 2009.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2951–2959, 2012.

Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359, December 1997.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. pp. 5026–5033, 10 2012. ISBN 978-1-4673-1737-5.

Mauro Vallati, Frank Hutter, Lukás Chrpa, and Thomas Leo McCluskey. On the effective configuration of planning domain models. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'15)*, 2015.

Konstantinos Varelas, Anne Auger, Dimo Brockhoff, Nikolaus Hansen, Ouassim Ait ElHara, Yann Semet, Rami Kassab, and Frédéric Barbaresco. A comparative study of large-scale variants of CMA-ES. In *Proc. of Parallel Problem Solving from Nature (PPSN XV)*, volume 11101 of *LNCS*, pp. 3–15. Springer, 2018.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv e-prints*, art. arXiv:1907.10121, Jul 2019.

Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning search space partition for black-box optimization using Monte Carlo Tree Search. *arXiv:2007.00708. To appear in Proc. of Neurips 2020*, 2020.

Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: Portfolio-based algorithm selection for SAT. *J. Artif. Int. Res.*, 32(1):565–606, June 2008.