mathematics

MDPI

Article

On the Performance Evaluation of Anytime Algorithms in Noisy Optimization

Ramona Boeh and Thomas Hanne *

Institute for Information Systems, University of Applied Sciences and Arts Northwestern Switzerland, 4600 Olten, Switzerland; ramona.boeh@fhnw.ch
* Correspondence: thomas.hanne@fhnw.ch

Abstract

A standardized, elaborate and well-argued performance metric for benchmarking anytime algorithms on noisy optimization problems is missing in the current benchmarking frameworks. This research aims at developing such a metric as an artifact by following the Design Science Research methodology. Existing noise handling practices and benchmarking guidelines are reviewed. Benchmarking experiments, including a test suite containing 13 artificial test functions, were conducted with two state-of-the-art algorithms to assess the performance metric. Based on the experimental results and the existing literature, this study concludes that, rather than relying on a single quantitative value, algorithm performance should be assessed through a comprehensive performance report comprising the numerical loss value, the number of function evaluations, wall-clock runtime, and a performance profile illustrating the trajectory of loss values over time. This approach aims to maximize the reliability, expressiveness, and informational value of performance assessments.

Keywords: anytime algorithms; noisy optimization; evolutionary algorithms; benchmarking; evolutionary computation; black-box optimization

MSC: 68W50; 90C59; 90C26

1. Introduction

Many real-world optimization problems are characterized by high complexity, uncertainty, and limited computational resources. To address these challenges, evolutionary computation (EC) emerged in the mid-twentieth century, inspired by the Darwinian principles of selection, recombination, and mutation [1]. Since then, evolutionary algorithms (EAs) and related metaheuristics have become widely applied to challenging optimization problems across engineering and scientific domains. However, the No Free Lunch Theorem (NFLT) demonstrated that no optimization algorithm performs best across all possible problem classes, highlighting the importance of benchmarking for identifying suitable algorithms for specific applications [2,3].

Benchmarking plays an essential role in optimization research enabling systematic comparisons of algorithm performance. Yet, designing meaningful benchmarking studies remains challenging due to methodological requirements, the need for standardized procedures, and the selection of appropriate performance measures [4]. Although several benchmarking frameworks have been proposed to improve comparability and reproducibility, continuous refinement is required as optimization methods and application domains evolve [5].

Check for updates

Academic Editor: Zhongming Wu

Received: 18 September 2026
Revised: 20 October 2026
Accepted: 27 October 2026
Published: 29 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

A particular challenge arises in noisy optimization problems, where objective-function evaluations are affected by uncertainty such as measurement errors, stochastic simulations, or environmental variability [6]. Such problems are common in practical engineering applications including planning, scheduling, and pathfinding. EAs are often considered suitable for these settings because their population-based search and iterative improvement processes provide a certain degree of robustness against noise. Furthermore, they can be viewed as Anytime Algorithms (AAs), meaning that they can produce progressively better solutions as additional computational resources become available [7,8].

Despite considerable advances in both benchmarking and noisy optimization, a widely accepted performance metric that jointly captures solution quality and computational effort for AAs in single-objective noisy optimization is still lacking [3]. Existing benchmarking studies employ a variety of performance indicators and stopping criteria, limiting comparability across studies and reducing the interpretability of experimental results [9,10]. While performance profiles provide valuable visual insights into algorithm behavior, they do not offer a standardized quantitative measure of overall performance [11–13].

The development of such a metric is further complicated by three challenges: First, benchmarking studies often differ substantially in their experimental design and reporting standards, making general conclusions difficult [5,9,10]. Second, different interpretations of anytime behavior and stopping criteria hinder the definition of a universally applicable performance measure [8,11,12]. Third, noisy optimization research follows two distinct paradigms: one focuses on algorithmic mechanisms such as resampling and population control to mitigate noise, whereas the other emphasizes stopping criteria and confidence in near-optimal solutions. Reconciling these perspectives is essential for establishing a generally applicable benchmarking metric.

This study addresses this research gap by investigating the development of an overall performance metric for benchmarking AAs in single-objective noisy optimization. Section 2 provides theoretical background on noisy optimization, AAs, and benchmarking. Section 3 describes the experimental methodology and benchmarking setup. Section 4 presents and discusses the experimental results used to assess the proposed performance metric. The findings contribute to ongoing efforts toward more reliable, interpretable, and standardized benchmarking practices for noisy optimization. Section 5 concludes the paper and highlights avenues for future research.

2. Literature Review

2.1. Noisy Optimization Problems

The literature distinguishes four major sources of noise in optimization problems: fitness function noise, input variable perturbations, fitness approximation, and dynamic optimization noise. Fitness function noise is commonly addressed through resampling and averaging combined with population control for symmetric noise [14], whereas input perturbations are handled through robust optimization approaches that perform well despite slight input changes [15]. When objective-function evaluations are expensive, meta models are used to approximate fitness values using simulation-generated data [6,16]. In dynamic optimization, the objective function itself changes over time, and algorithms must continuously track these changes. While the true fitness value can only be estimated in noisy conditions, solving approaches vary depending on the noise type [17].

2.2. AAs for Noisy Optimization Problems

Reference [18] defined the following key properties of AAs: measurable result quality, larger improvement steps at the beginning, and minimal overhead when suspending or

resuming. While these may hold for noisy optimization, noise reduces the correlation between solution quality and computation time, blurring the assessment of progress [7].

Performance profiles are the standard tool for analyzing AAs displaying solution quality over runtime [19]. However, it cannot indicate a stopping criterion for the runtime of the algorithm. Instead, utility functions are commonly used to express solution approximation. The five stopping methods are:

1. Fixed runtime budget: Assumes monotonic quality improvement and works if the quality is estimable [7].
2. Marginal value: Terminates when the expected utility gain becomes negative but assumes monotonic improvement, which is not guaranteed in noisy optimization [20].
3. Monitoring policy: Balances runtime allocation and monitoring costs based on observed features and expected utility, suitable under uncertainty [21].
4. Fixed-target value: Relies on known optima, independent of monotonicity [13].
5. Reinforcement learning: Uses policies for dynamic environments with rewards based on utility differences [22].

Approaches such as fixed budgets and marginal value assume monotonicity, limiting their application. Monitoring policies and reinforcement learning are promising, whereas fixed-target values require prior knowledge of the optimum.

2.3. Performance Metrics for AAs on Noisy Optimization Problems

According to [13], a suitable performance metric should be quantitative, interpretable, reproducible, comparable across problems, and applicable to most algorithms and optimization problems while maintaining simplicity and practical relevance. Furthermore, it should support differentiation across a large range of values and remain independent of experimental settings. Reference [13] argued that only the number of function evaluations meets all these criteria, as measuring runtime in CPU or wall-clock time depends on machine performance, making it less reliable [4]. For AAs, both the solution quality and runtime can serve as performance metrics. A fixed runtime budget uses the function value for comparison, while fixing a target value uses runtime, with the number of function evaluations being the most reliable measure [13]. In noisy optimization, due to the uncertainty of solution quality, ref. [13] suggest a quality indicator based on the worst 1% of the last function values, ensuring a measure independent of data size which, however, remained untested. Section 4.5 describes the procedure of testing this quality indicator.

2.4. Delimitation of the Paradigms

We point out the differences between the two paradigms of how to solve noisy optimization problems with AAs. As identified in Section 2.1, noise always affects the objective function [6], forming a first paradigm: addressing noise sources through resampling, population control, meta models, and robust or dynamic optimization. Resampling, widely studied [17,23,24], exemplifies this paradigm by refining individual fitness evaluations to estimate the true fitness value.

Zilberstein's and his scientific community's research initially focused on AA behavior, yet later incorporated noise handling due to its prevalence in engineering. The standard became a combination of monitoring policies and performance profiles, forming the second paradigm [7,12,18,21,25–27]. Here, the fitness value is referred to as the loss value. Instead of resampling, the best-performing solution is identified from the sequence of candidate solutions observed during runtime. Thus, the objective is to find an adequate stopping criterion while monitoring the runtime of the algorithm. Table 1 illustrates the characteristics and differences in the two paradigms.

Table 1. Characteristics of the two paradigms for solving noisy optimization problems with AAs.

	Noisy Optimization with EAs	**Monitoring AAs**
Origin	Evolutionary computation	Engineering field
Objective	Reducing/eliminating noise	Trading off runtime vs. solution quality
Goal of the algorithm	Estimating the true fitness value	Finding a stopping criterion
Type of noise	Noise in the fitness function	Noise in the objective function
Type of algorithm	EAs	AAs
Assumptions	Each solution candidate is equally eligible	Assumes monotonicity [1] to a certain degree
Noise handling vanilla version	Resampling and averaging the solution candidates	Applying a policy correlating a feature with the solution quality to estimate the solution quality at runtime

[1] The condition of monotonicity assumes continuous solution quality improvement the longer the algorithm runs.

The key difference between the paradigms lies in the use of resampling in EAs to handle noise, where the true quality of solution candidates is uncertain [28], whereas AAs assume convergence and improve solution quality over time [21]. These differing assumptions result in distinct approaches, although EAs may also be AAs. This divergence may explain the limited research on EAs in noisy optimization with anytime behavior analysis [22,29], despite their strong performance [8].

2.5. Benchmarking

Empirical experiments are vital for developing, assessing, and selecting algorithms, making benchmarking crucial for cost-effective optimization [30]. However, research faces a conflict between competitive testing and understanding algorithm behavior [4,9]. Competitive testing focuses on results, not processes, while simplified benchmarks fail to reflect real-world complexity. Ultimately, understanding an algorithm's behavior is essential for handling real-world problems [31]. Thus, benchmarking algorithms plays an important role in solving optimization problems at the lowest cost possible [30].

2.6. Research Gap

Benchmarking frameworks establish standards to ensure meaningful, comparable insights [3]. Performance metrics are key, determining whether algorithms are superior, sufficient, or insufficient, highlighting the need for interpretability and comparability [13]. Noisy optimization is challenging because true solution values can only be estimated [7]. The different paradigms identified in Section 2.4 [6,21], the uncontrolled growth of studies [32], and varying stopping criteria [8,22] hinder a universal performance metric. Although [13] proposed a quality indicator for runtime, it remains untested. This research addresses this gap by developing a comprehensive metric for single-objective noisy optimization with AAs.

3. Experimental Design

This section covers experimental design decisions preparing the experiments to generate data for validating the performance metric.

3.1. Choosing the Performance Metric

Both solution quality and runtime are crucial metrics. Runtime, measured in function evaluations, reflects the cost of optimization and allows for cross-problem comparisons.

It is ratio-scaled and independent of the experimental settings, although noise reduction techniques can complicate interpretation [33]. Reporting the CPU time per evaluation can improve comparability [34]. Solution quality, measured by the loss value, is another common metric [3,35,36]. While continuous and reproducible, it can be unreliable in noisy optimization [13] and is tied to a fixed-budget approach. Despite its limitations, the loss value remains a key quality indicator [13]. The final choice of the performance metric is discussed in Section 3.2.

3.2. Choosing a Benchmarking Platform

To generate suitable data for evaluating the proposed performance metric, the benchmarking environment must provide a traceable performance measure, test functions with adjustable and distinguishable noise levels, and implementations of the selected test functions and algorithms. It should support the creation of customized test suites, parameterization of problem instances, integration of algorithms with the test suite, and the execution of reproducible experiments. Furthermore, the implementations should be transparent and accompanied by adequate documentation. Finally, the environment must offer performance profiles for result visualization and data-logging capabilities for subsequent statistical analysis.

Nevergrad and COCO are widely used open-source platforms for continuous, derivative-free black-box optimization, where solutions are accessed through loss or fitness values, and runtime is measured in function evaluations [9,37]. Both offer popular artificial test functions [9,38] and provide user guides [39,40]. They support noisy optimization, logging, and data visualization [9,38]. Table 2 summarizes the differences between the two platforms with respect to the specified criteria.

Table 2. Comparison between Nevergrad (version 1.0.1) and COCO (version 2.6.3): Features that are better supported by one platform than the other are marked as ++, while the inferior feature is indicated by only one +. Only one criterion is marked with –, indicating that the loss value is unavailable as a performance metric.

Criteria	Nevergrad	COCO
Performance Metric	++ The loss value is available within the fixed-budget approach.	– The number of function evaluations is implemented as the central performance metric.
Tunable noise-level	++ The noise level is tunable in the function parameters.	+ One must change the source code or add a customized test suite to tune the noise-level.
Open-source platform	++ Both are open-source platforms.	
Selected test functions and algorithms available	++ The functions and algorithms were selected according to Nevergrad.	+ Most selected functions are available but there is no algorithm library.
Usability	++ Python facilitates invoking functions and algorithms via a registry, and the modular library system provides great usability.	+ Interfaces in C/C++, Java, Matlab/Octave, and Python are available for connecting solvers and customized test suites. More programming skills are required.
Documentation	+ A user guide is available, but information about the functions exists only as docstrings.	++ Excellent documentation of all functions, incl. release notes and user guide, is available.
Visualization	++ Visualizations are available on both platforms.	
Logging facilities	++ Logging the data into a csv is available on both platforms.	

Both platforms allow adding a customized test suite and connecting algorithms [9,38]. In this regard, Nevergrad offers better usability than COCO with its parametrization interface and the ease of connecting algorithms via Python's modular system [40]. Although Nevergrad provides the number of function evaluations, it does not use it as the performance metric; instead, it relies on the fixed-budget approach [41]. However, COCO already uses the number of function evaluations as the central performance metric [13,42], and its noisy BBOB test suite has been extensively tested [42]. As COCO would not provide additional empirical insights, Nevergrad's implementation of the oppositional performance metric is selected, using a fixed-budget approach to evaluate the achieved loss values.

3.3. Choosing Noisy Optimization Problems

Two types of optimization problems are commonly used for benchmarking: complex real-world problems, often specific and non-generalizable, and artificial test functions, which are widely used on benchmarking platforms [3,43]. Artificial functions are less complex, support the assessment of algorithm characteristics, and provide a known global optimum unlike real-world problems [44]. Since they are considered the best practice for benchmarking [45], this work uses Nevergrad's artificial test functions for generating a test suite. However, researchers recommend that artificial test functions reflect real-world problem characteristics and cover increasing levels of difficulty [3,46]. Accordingly, the test suite used in this work covers several characteristics known to influence optimization difficulty. The number of input variables should be scalable, as increasing dimensionality causes the search space to grow exponentially [47]. Both separable and non-separable functions should be included to account for the presence or absence of variable interactions [44]. Furthermore, the suite should incorporate bias, which alters the objective-function landscape and increases problem difficulty [44], as well as varying levels of modality, ranging from unimodal functions with a single global optimum to multimodal functions with multiple local optima [44]. Ill-conditioning should also be represented, as high condition numbers render solutions highly sensitive to small perturbations which means that small data errors can cause large solution errors [48]. Finally, deceptive functions should be included because they can mislead optimization algorithms and penalize localized hill-climbing strategies [49]. The functions selected to balance these characteristics are presented in Appendix A.

Noise-levels: All functions are biased with Gaussian noise, defined as an additive term η [50]:

$$\eta = e * N(0,1) * (f(x + N(0,1)) - f(x))$$

where $N(0,1)$ is a Gaussian distribution and e is the tunable noise level. Independent Gaussian random draws are generated for each function evaluation; consequently, repeated evaluations at the same input point may yield different objective values. The conditional distribution of the noise depends on the perturbation of the input point. Conditional on a fixed input x and a realization of the Gaussian perturbation $N(0,1)$, the noise is normally distributed with zero mean and variance $e^2 * (f(x + N(0,1)) - f(x))^2$. When conditioning only on $f(x)$, the resulting noise distribution is generally a mixture of zero-mean Gaussian distributions with different variances. Its conditional mean is zero, while its variance depends on the objective function, the input point, and the noise level parameter e. Each evaluation of the noisy objective counts as one function evaluation, including all internal computations required to generate the noise. Noise levels $(0, 1, 10, 100)$ were explored using the Styblinski–Tang function and tested with the TBPSA algorithm, which performs best in noisy optimization using Nevergrad [51]. Explorative test experiments, conducted with three input variables, led to the following conclusions:

- Outliers appeared at all noise levels, becoming more extreme at higher noise levels. However, noise level 100 only shows a single outlier at 13,000. It is assumed that at such a high noise level, large fluctuations smooth out the effect of outliers.
- Convergence was observed at noise levels 0–10, but not at noise level 100, where only fluctuations were noted beyond 20,000 function evaluations.
- Loss values became negative as noise levels increased: −3 at noise level 1, −25 at noise level 10, and −400 at noise level 100.

These conclusions are limited by the use of only three input variables and one low-difficulty function. However, the noise levels proved to influence the Styblinski–Tang function sufficiently to be applied to other functions. Noise level 100 showed no convergence after 20,000 evaluations and was excluded due to the diminishing benefit relative to its computational cost.

Parametrization of the Instances: Benchmark studies typically use input variables at levels like 2, 5, 10, 15, 20, 30, 50, and 100 [10,52,53]. For evaluating the performance metric, dimensions such as 5, 30, and 50 per noise level are sufficient.

Budget: Reference [41] recommends 128,000 function evaluations for noise level 10. Because our test suite includes higher dimensions (30, 50), this budget is doubled to 256,000 evaluations.

Search Space: All functions are optimized in an unbounded search space to avoid constraints that would add the complexity of multi-objective optimization [54].

Prior: In Nevergrad, the input variables are initialized at zero by default with a unitary standard deviation of one for the mutation rate [55]. This is unfavorable because most of the global optima of the artificial test functions are also located at zero. Therefore, the initial values of the input variables are randomly shifted by values sampled from the interval [0, 100], using a fixed random seed (42) to ensure reproducibility. The test functions and their global optima remain unchanged.

Target Precision Threshold: Based on initial experiments, a target precision threshold $\varepsilon = 10^{k}$ with $k_{min} = -1$ is sufficient, as noise increases the difficulty and budget requirements.

Condition Number: For ill-conditioned functions, three levels of conditioning (10^6, 10^4, 10^2) are applied based on explorative experiments with condition number 10^6 which is the highest one [56] worked with in their testbeds.

Thirteen artificial test functions (see Appendix A, Table A1), each with six instances (dimensions 5, 30, 50, noise levels 1 and 10), were selected. Ill-conditioned functions have 18 instances due to their three condition numbers, totaling 102 instances for the entire test suite.

3.4. Choosing Two AAs

Bandit algorithms have proven effective for noisy optimization and are widely used in fields such as operations research, healthcare, economics, and statistics to make decisions under uncertainty. Inspired by a gambler choosing between different slot machines (one-armed bandits) to maximize rewards, these algorithms balance exploration and exploitation [57,58]. Another approach for handling noise is the use of population-based EAs, which maintain diversity to mitigate noise [16]. A key example is the pcCMSA-ES algorithm, which outperforms traditional resampling by combining Covariance Matrix Adaptation (CMA) with population size adjustment [14]. A simpler but analogous implementation is the Test-Based Population Size Adaptation (TBPSA) algorithm in Nevergrad [59]. The platform also contains a bandit algorithm for noise handling called NoisyBandit. Both algorithms were selected because of their tried and tested noise handling features and their different search mechanisms: TBPSA is an EA that handles noise by adapting the

population size and its mutation rate [59], while the NoisyBandit works with probability distributions [51].

TBPSA: TBPSA showed the best performance among the noise handling algorithms (NoisyDE, NaiveTBPSA and NoisyBandit) in the Nevergrad benchmarking test suite for noisy optimization [60]. In the following, a subsequent description refers to the respective lines of the pseudo-code.

Pseudo-Code of TBPSA

Initialization: $g \leftarrow 0$; $\lambda \leftarrow 4d$; $d \leftarrow d^{(init)}$; $\sigma \leftarrow \frac{1}{\sqrt{d}}$; $\mu \leftarrow d$; $x \leftarrow x^{(init)}$

2: *repeat*
3: $\lambda \leftarrow |\mu|$
4: *for* $l \leftarrow 1$ *to* λ *do*
5: $\tilde{f}_l \leftarrow \tilde{f}(x_l)$
6 *rank* $\tilde{f}(x_l)$
7: *select* $\mu \leftarrow \tilde{f}(x_1), \ldots, \tilde{f}(x_\mu)$
8: $\langle x \rangle \leftarrow \frac{\sum_{m=1}^{\mu}(x_m)}{\mu}$
9: $\langle \sigma \rangle \leftarrow \frac{\log(\sum_{m=1}^{\mu}(\sigma_m))}{\mu}$
10: *end for*
11: $g \leftarrow g + 1$
12: $x^{g+1} \leftarrow x + N; (0, \langle \sigma \rangle)$
13: *if* $g \geq 5\lambda$
14: $\langle x_1 \rangle \leftarrow first\frac{\lambda}{5}$
15: $\langle x_2 \rangle \leftarrow last\frac{\lambda}{5}$
16: *if* $\langle x_1 \rangle - \langle x_2 \rangle < 2\sigma$
17: $\lambda \leftarrow 2\lambda$
18: $\mu \leftarrow 2\mu$
19: *else*
20: $\lambda \leftarrow 2^{\frac{-1}{4}}\lambda$
21: $\mu \leftarrow 2^{\frac{-1}{4}}\mu$
22: *if* $\lambda < \lambda^{(init)}$
23: $\lambda \leftarrow 4d$
24: $\mu \leftarrow d$

TBPSA is initialized with $\lambda = 4d$, $\mu = d$ and $\sigma = 1/\sqrt{d}$ while d is the dimension that is to be defined by the user (line 1). It implements the self-adaptive evolutionary strategy (ES) denoted as (μ/μ,λ)-ES as a recombination technique: λ is the population size. μ is the number of selected best individuals out of the population size λ. Each individual of the population is a solution candidate and denoted as x_i with ($i = 1, \ldots, \lambda$) while its step size σ_i is a positive number. At each generation g, the individuals' fitness values $f(x_i)$ are ranked (line 6) to select the μ best (line 7). To choose the next parent x, the mean of the μ best individuals is calculated (line 8). Their step sizes σ_i are log-averaged to define the next parental step size σ (line 9). The next generation $g + 1$ is instantiated (line 11) by sampling new λ individuals x_i^{g+1} according to the updated step size:

$$\sigma_i = \sigma e^{\mathcal{N}(0,1)}$$

They are placed from the centroid of the parental generation g to a new centroid according to the length and direction of the updated step size σ_i which serves as the standard deviation in the normal distribution of the new centroid (line 12).

$$x_i^{g+1} = x + \mathcal{N}_i(0, \sigma_i)$$

Likewise, the individuals should approximate the optimum with each generation by consistently updating the mutation strength.

To control population size, a sequence of five generations of λ individuals is kept and its first fifth (line 14) and its last fifth (line 15) are examined to determine whether they differ statistically. This means that the differences between their means are two or more standard deviations apart (line 16). If they do not differ, population size λ and μ are doubled (line 17 + 18). Otherwise, they are reduced by a factor of 0.84 (lines 20–21), without falling below either the initial value (lines 23–24) or the degree of parallelism. The factors governing population size adjustments are established constants in other areas of EC [61].

NoisyBandit: In the noisy real-world game007 problem with a simple 12-dimensional linear controller, NoisyBandit outperforms TBPSA. It also surpasses TBPSA in problems with asymmetric noise [58]. The NoisyBandit is based on the multi-armed bandit problem, originating from slot machines where a gambler must choose the machine with the highest reward [62].

In a broader sense, it involves a decision maker selecting from a fixed set of options $B = \{o_1, \ldots, o_k\}$, with each selection being a trial t. Each option has an unknown loss l distribution, revealed only after selection, and the goal is to minimize the average loss. The approach can be framed in terms of either rewards or losses. The algorithm's performance is measured by regret ρ defined as the difference between the average loss $\frac{\sum_{t=1}^{T} l_t}{T}$ over T trials and the minimal possible loss l_{min} defined by the following formula [63]:

$$\rho = T l_{min} - \sum_{t=1}^{T} \hat{l}_t$$

The optimal strategy for identifying the best option requires balancing exploration and exploitation. Exploitation involves selecting the option with the lowest observed loss, while exploration focuses on testing less-known options to discover potentially better ones. To achieve this balance, the Upper Confidence Bound (UCB) policy is used. In Nevergrad, UCB is adapted for loss minimization, as shown in the NoisyBandit pseudo-code [51]. Initialization involves t as the number of trials, l as the loss, B as a set with options o corresponding to the solution candidates and B_{best} as a set with the best options encountered so far.

Pseudo-Code of NoisyBandit

```
1:  Initialization: t ← 0; l ← 0; B_best ← 0
2:  if 20t > t_B^3
3:      B_new = N(0, 1)
4:  random choice (True, False)
5:      if True
6:          return random choice B_new(o_1, ..., o_i)
7:      else
8:          return o_best from B_best(o_1, ..., o_i)
```

Action 1: If 20 times the number of function evaluations is greater than the number of times this specific set was already chosen to the power of three (line 2), a new set with a normal distribution is added (line 3). If not, no new set is added.

Action 2: A random choice between true and false is made (line 4).

Exploration: If true, a random index selects an individual from the new set, which becomes the solution candidate for the current evaluation (line 6).

Exploitation: If false, an individual from the set that previously provided the best solution is chosen as the new candidate (line 8).

The algorithm tracks all evaluations, updating the best solution candidate and noting which set produced it at each trial t.

4. Results and Discussion

Based on the experimental data, the algorithms' performances were interpreted and compared using the best-achieved mean loss values and their corresponding standard deviations.

4.1. Execution of the Experiments

The experiments were conducted in Visual Studio 1.85.1 (Universal) on a MacBook Air with an Apple M1 processor and 16 GB RAM, using Nevergrad's Python code from GitHub [38] (details in Table 3). Each artificial test function is run for all specified budgets, with new seeds in each iteration to ensure reproducibility.

Table 3. Subordinated test sets and their specifications.

Algorithms	Dimensions	Noise-Level	Budgets	Repetitions	Reproducibility
TBPSA, NoisyBandit	5, 30, 50	1, 10	500, 1000, 2000, 4000, 8000, 16,000, 32,000, 64,000, 128,000, 256,000	50	seed = 42

In Nevergrad, a decision maker sets multiple budgets for a benchmarking test suite. When a budget is reached, the loss value recommended by the optimizer is logged in a CSV file, and the optimization restarts for the next budget, beginning with the first function evaluation. The reported loss reflects the recommended solution's performance on the underlying objective, without the additive observation noise. For each instance-algorithm combination, 50 repetitions are performed. The mean loss values are plotted as points and connecting them creates a noise-reduced graph that highlights the optimization trend and illustrates the relationship between the available evaluation budget and average terminal performance, rather than the anytime trajectory of a single optimization run.

4.2. Descriptive Results and Algorithm Comparison

Algorithm performance is compared using performance profiles, loss values and percentage improvements of one algorithm over the other. Statistical tests are used to assess whether observed differences are supported by the experimental data. Detailed statistical results are provided in Appendix C (Table A3).

NoisyBandit: NoisyBandit typically evaluates an initial solution and then stagnates on most artificial test functions, particularly at noise level 1. Across most evaluated instances, it achieves higher loss values than TBPSA. The main exceptions are the Ackley instances and the Bentcigar instances with condition numbers of 10,000 and 1,000,000 at noise level 10 in five dimensions. On Ackley, NoisyBandit shows limited convergence and achieves loss values up to 3.5% lower than those of TBPSA. On Bentcigar with condition number 10,000, the apparent 99.9% improvement for NoisyBandit is associated with an exponential increase in TBPSA's loss. Despite this relative difference, both algorithms remain far from the known optimum: NoisyBandit's loss value is 254,142,539 and TBPSA's is 4.2×10^{16}. A similar pattern occurs for Bentcigar with condition number 1,000,000: TBPSA's loss initially increases sharply before decreasing, while NoisyBandit's loss remains 99.9% lower at the evaluated budget.

For Genzcornerpeak and Deceptive Path, NoisyBandit produced infinite loss values which are interpreted as failures to obtain finite loss values under the tested configurations. At noise level 1, NoisyBandit generally stagnates across the tested dimensions. At noise level 10, some improvement is visible in dimensions 30 and 50. This suggests that its observed convergence behavior varies with noise level, while its response to dimensionality is limited at noise level 1 within the range tested.

TBPSA: TBPSA's performance varies with dimensionality, noise level, and function difficulty. Its mean loss did not reach the optimum in most evaluated cases; however, at noise level 1 on two instances, it meets the target precision threshold $\varepsilon = 10^{-1}$ (see Table 4).

Table 4. Best mean loss values achieved by TBPSA on noise level 1 meeting the target precision threshold $\varepsilon = 10^{-1}$.

Function	Dimension	Mean Loss Value	Standard Deviation
Sphere	5	0.172	0.145
Styblinski–Tang	30	0.46	1.00

TBPSA generally exhibits decreasing loss trends, with exceptions on Genzcornerpeak and Deceptive Path, where infinite loss values and stalling were also observed. In dimensions 30 and 50 at noise level 1, its behavior on challenging functions follows two broad patterns: stagnation followed by a sharp decrease, or an initial decrease followed by stagnation. In five dimensions, the standard deviation exceeds the mean loss on eight functions, indicating substantial dispersion relative to the mean. This observation is consistent with the variability reported in [64] for Nevergrad's real-world problem Shotgun007, where TBPSA performed better in higher dimensions under simple noise models. TBPSA also exhibits substantial variability on Rosenbrock and ill-conditioned functions in five dimensions. Hence, outliers allow the NoisyBandit to outperform TBPSA on the Bentcigar functions at noise level 10 in five dimensions in our experiments. This specific function seems to systematically mislead TBPSA. However, overall, the descriptive results favor TBPSA on most evaluated instances, but not uniformly across all functions and noise conditions.

4.3. Statistical Analysis of Algorithm Performance

The proposed performance metric is evaluated using a benchmarking test suite to demonstrate its application in a regular algorithm benchmarking procedure. Statistical tests assess whether the observed loss values of the two algorithms differ significantly from the known optimum and from each other. These analyses illustrate the practical application of the metric and highlight challenges in benchmarking, but do not establish loss as a reliable indicator of solution quality. This aspect is examined separately in Section 4.6 through correlation analysis, which investigates the relationship between loss and solution quality. The metric's overall suitability is then assessed based on the combined experimental findings and their contribution to existing knowledge.

For each combination of noise level, dimension, test function, algorithm and budget, 50 loss values were obtained from independent runs with different random seeds. For each configuration, the budget yielding the lowest mean loss was selected for statistical analysis. This resulted in 90 evaluable configurations for comparing the two algorithms. Genzcornerpeak and Deceptive Path were excluded from inferential comparisons since infinite loss values prevented the calculation of the relevant statistics. The algorithms' runs were not paired; therefore, independent-sample tests were used for between-algorithm comparisons.

Three hypotheses were evaluated, with normality tested using either the Shapiro–Wilk test for small samples or the Kolmogorov–Smirnov test for larger samples. For H1 and H2, either a one-sample T-test or a one-sample Wilcoxon test was applied.

H1: *TBPSA's loss values significantly differ from the optimum loss value for the respective functions.*

H2: *The NoisyBandit's loss values significantly differ from the optimum loss value for the respective functions.*

H3: *TBPSA's and the NoisyBandit's loss values are significantly different from each other.*

For H3, either a two-sample Welch's T-test or a two-sample Wilcoxon rank sum test was applied. Cliff's delta was used as a non-parametric effect-size measure because some configurations showed non-normal distributions and substantial outliers. The exact test used for each configuration, together with p-values, effect sizes, and confidence intervals, is reported in Appendix C (Table A3).

H1 is confirmed for all functions and its instances except Styblinski–Tang at noise level 10 in five dimensions. In that configuration, TBPSA had a mean loss of 2.7 and a standard deviation of 10.7; the test did not establish a significant difference from the optimum with a p-value = 0.08. This result should not be taken as evidence that TBPSA reliably reached the optimum, given the large variability.

For H2, the tests indicated that NoisyBandit's loss values differed significantly from the known optimum in every evaluated case. This supports the conclusion that NoisyBandit did not reliably attain the optimum under the tested configurations.

For H3, the algorithm comparison was statistically significant in most evaluable configurations. Two Bentcigar cases at noise level 10 in five dimensions were exceptions to the general pattern. For condition number 10,000, the difference was not statistically significant ($p = 0.20$; Cliff's delta = 0.14, a weak effect). For condition number 1,000,000, the result favored NoisyBandit but the effect was weak ($p = 0.03$; Cliff's delta = -0.24). On Ackley, NoisyBandit performed better, but the effect was smaller in five dimensions.

Overall, the descriptive and inferential results indicate that TBPSA achieved lower loss values in most evaluable configurations, while NoisyBandit performed better on Ackley and on the specified Bentcigar cases. This conclusion applies to the benchmark suite and experimental settings studied here; it does not establish universal superiority.

4.4. Discussion of the Performance Metric

This section examines how the loss value and number of function evaluations meet the eight validity criteria from Section 2.3, using empirical data. It also compares the fixed-budget and fixed-target approaches based on the characteristics of the observed data.

Ratio-Statements: The loss value is quantitative but not inherently ratio-scaled; percentages are needed for ratio comparisons. Faster convergence cannot be fully expressed by the mean loss alone. In contrast, the number of function evaluations provides a clearer and more expressive measure for comparing algorithm performance.

Approximation: Instead of Target Hitting: In all experiments, the global optimum remained unreached; however, the difference between the achieved mean loss and the optimal loss indicates how closely each algorithm approaches the target. This aligns with [7]'s property for AAs of measurable quality. Three issues, all observed in this project, can hinder reaching the global optimum:

- Insufficient budget: On the Sphere function, TBPSA converges but does not stall, suggesting that more evaluations could achieve the optimum.
- Algorithm unsuitability: NoisyBandit is insensitive to noise and stalls on most functions.
- Problem difficulty: Both algorithms fail on Deceptive Path, get stuck in local optima, and perform poorly on Genzcornerpeak.

These challenges highlight the need for a performance metric that works even when the global optimum is unknown. The number of function evaluations, often unavailable or varying across algorithms, is uninformative if the target is unmet. Reference [13] recommended using the final quality indicator value as an additional metric. In noisy optimization, this requires verifying how the loss is computed rather than relying on the final loss value.

Monotonicity: When approximating, a ratio statement can compare function evaluations, e.g., one algorithm requiring half as many evaluations as another to reach a certain distance to the optimum. However, this implies monotonicity, a property desired for AAs [7]. However, more function evaluations do not always yield better solutions. The fixed-target approach addresses non-monotonicity by focusing on achieving a specific target value regardless of the process trends.

Stagnation: The fixed-target approach becomes ineffective if optimization stagnates, such as when NoisyBandit gets trapped in local optima on Sphere, Griewank, and Styblinski–Tang. Detecting stagnation under this approach wastes time. Reference [13] emphasized the need for a well-defined termination condition when using the fixed-target approach. In real-world scenarios, where time is limited, the termination condition is often set by a maximum time budget, reverting to the fixed-budget approach.

Performance Profiles and Monitoring: Addressing non-monotonicity and stagnation requires an early recognition of optimization trends, which are best identified through performance profiles. However, these profiles are only available after optimization [19]. In a fixed-budget scenario, they can serve as intermediate results to decide whether to extend the runtime or stop. This manual monitoring resembles [21]'s monitoring strategy, which allows premature termination if sufficient solution quality or the target is reached, effectively combining fixed-budget and fixed-target approaches. Early stagnation detection strategies include analyzing intermediate performance profiles or defining a termination condition based on a set number of consecutive equal loss values. However, setting this number is challenging because stagnation can unexpectedly lead to convergence. Anticipating such shifts requires extensive preliminary experiments. In cases of stagnation, the fixed-budget approach is preferable, enabling incremental decision-making and runtime control.

Complementary Paradigms: Reference [21] states that the more accurately a solution's quality can be estimated, the more valuable monitoring becomes. While the loss value serves as a quality indicator, it is inaccurate in noisy optimization. Algorithms address this by employing noise handling techniques like resampling or population control. For instance, TBPSA uses the center of the Gaussian noise distribution [64], enhancing the accuracy of the loss value as a quality indicator. This shows that monitoring and noise handling, as discussed in Section 2.4, are complementary: noise handling reduces noise, making the loss value more reliable and monitoring more effective.

Reproducibility and Comparability: Using heuristic algorithms requires seeds to ensure reproducibility. This allows the loss value to be reproduced for a given function with the same algorithm, seed, initialization, and loss function. However, function evaluations pose issues when considering the actual computational time [33]. Equal numbers of function evaluations can be misleading if performed on machines with different computational speeds. Similarly, comparing two algorithms with the same number of evaluations to reach a target value can be deceptive if one takes longer in real time. For instance, on Ackley, TBPSA and NoisyBandit achieved the same loss value after 500 evaluations, but TBPSA was a few milliseconds faster (see Table 5). While this seems minor, over 256,000 evaluations it results in nearly 20 s more.

Table 5. Number of function evaluations compared in seconds between TBPSA and the NoisyBandit.

Ackley				
Number of Function Evaluations	**TBPSA**		**NoisyBandit**	
	Loss Value	**Seconds**	**Loss Value**	**Seconds**
500	21.7	0.088	21.7	0.092
256,000	21.5	54.2	21.5	71.4

Decision makers prioritize the actual clock time required to reach a viable solution. Similarly, an algorithm's performance is not directly comparable across different optimization problems as shown in Table 6.

Table 6. Number of function evaluations compared across the functions Rastrigin and Sphere in seconds.

TBPSA				
Number of Function Evaluations	**Rastrigin**		**Sphere**	
	Loss Value	**Seconds**	**Loss Value**	**Seconds**
500	195.6	0.13	195.7	0.083
256,000	0.45	80.72	0.45	50.72

For nearly identical loss values, the runtime in seconds can vary significantly depending on the function. This questions the comparability of function evaluations, making it essential to report runtime in seconds. Loss value comparability is ensured if the algorithms share the same starting conditions. Since the objective functions are tailored to specific optimization problems, the optimum loss may vary. For a meaningful comparison, the optimum loss must be consistent. If not, it should be standardized across all functions in a test suite. Although the benchmark functions in Nevergrad have consistent optima of zero, their objective values may differ substantially in scale. Consequently, absolute loss differences are not necessarily comparable across different functions, as the same numerical deviation from the optimum may have different implications for solution quality depending on the objective function. Therefore, algorithms are compared using function-specific rankings rather than by aggregating raw loss differences across the test suite. This approach avoids allowing differences in numerical scale to disproportionately influence the overall comparison and preserves the problem-specific interpretation of solution quality. Such rankings are complemented by the separate reporting of function evaluations and wall-clock time, allowing decision makers to assess solution quality and computational cost in the context of each optimization problem.

Interpretability: When using AAs, decision makers prioritize achieving acceptable solution quality within a reasonable time [19]. Solution quality or runtime can serve as performance metrics for real-world applications. However, since the loss value is computed using a problem-specific loss function, it may be less intuitive. In contrast, the number of function evaluations is more straightforward, clearly reflecting the runtime cost. The implications discussed are found in Appendix B (Table A2).

A challenge in noisy optimization is that the number of function evaluations depends on reaching the target value. However, if only the noisy loss value is available as a quality indicator, it is unclear whether the target has been reached. In this case, the loss value must be verified as a reliable indicator. Nevertheless, knowing whether the target is reached without understanding the time taken is insufficient to assess an algorithm's performance. Therefore, this study recommends using both the loss value and the number of function evaluations as a combined performance metric. When combining runtime and loss, the role of the loss as a quality indicator must be clarified.

4.5. Quality Indicator

As discussed in Section 2.3, ref. [13] defined a formula for noisy optimization to find a loss value serving as a quality indicator and representing the recent optimization state of the algorithm. This formula is applied to different optimization states of the Styblinski–Tang and Ellipsoid function, chosen for their varied optimization behaviors with TBPSA. For the NoisyBandit, no optimization occurs due to stagnation, so its performance is not analyzed.

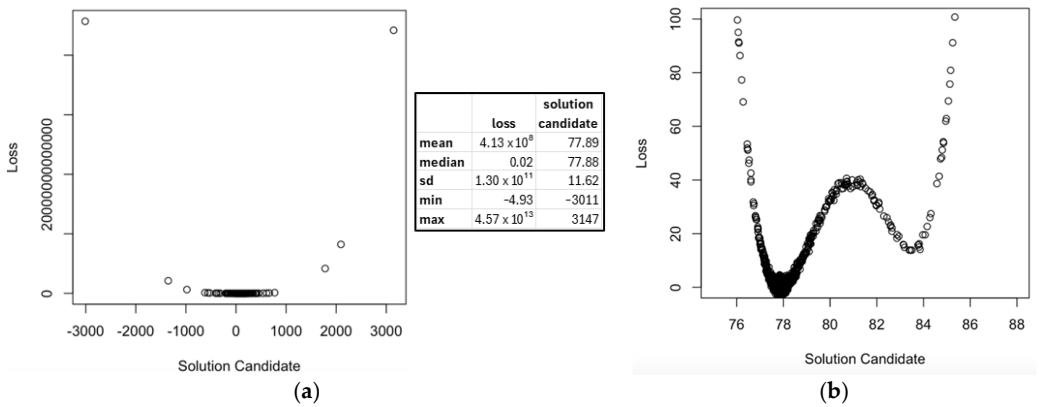
One dimension is used because the mapping from the loss value to a single solution is better interpretable. The formula $ln^2\left(\frac{t+3}{2}\right)$ calculates the number of last values in the sequence while t corresponds to the current number of function evaluations. The quality indicator is computed using this formula, with t varying based on the optimization state budget. The formula can only be reasonably evaluated if the relationship between the loss and solution candidates is defined, which is tested statistically in Section 4.6. If a correlation exists between the loss and solution candidates, searching for a reliable quality indicator within the sequence helps counteract non-monotonicity, as it uses multiple values rather than just one. However, the analysis of the three designated optimization states is not conclusive.

- Extreme fluctuation: When outliers are present, selecting the worst 1% of the values in the sequence results in an outlier being used as the quality indicator.
- Convergence: If the convergence state is stagnant near the optimum, the loss and solution candidates have nearly identical values, so the choice of quality indicator becomes less relevant. However, the sensitivity to differences depends on the optimization problem.
- Mediocre fluctuation: In the case of early fluctuation before convergence, higher loss values correspond to smaller solution candidates and vice versa. Picking the worst 1% often results in the smallest solution candidate, which is not an outlier but a marginal value, unrepresentative of the algorithm's state, especially when sequence differences are large early in the optimization [7].

Therefore, selecting the worst 1% may lead to extreme or marginal candidates. A better approach could be to use the median of the sequence to more accurately reflect the algorithm's state. Reference [13]'s quality indicator should be tested on other algorithms and test functions to draw broader conclusions, yet this is left for further research.

4.6. Best Achieved Loss Value as a Quality Indicator

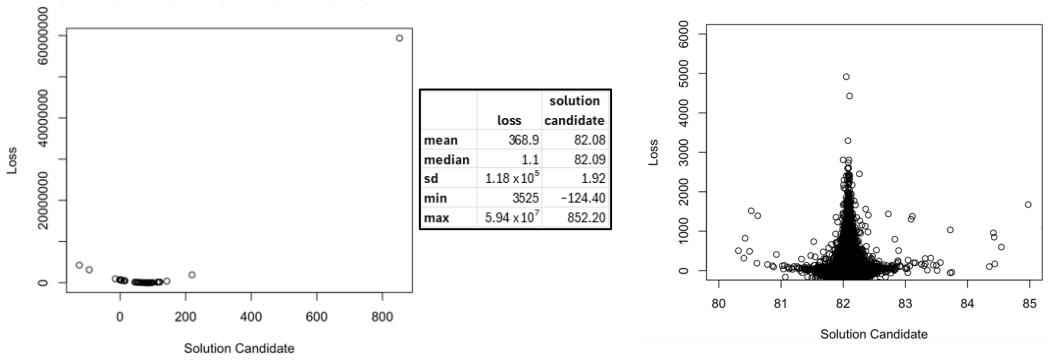
The correlation between the loss values and solution candidates was tested on three functions of varying difficulty: Styblinski–Tang (easy), Discus100 (medium), and Bentcigar (condition number 1,000,000) (hard) to assess the loss as a reliable quality indicator. For TBPSA, none of the data was normally distributed or linearly correlated. Using the adaptive local linear correlation, the non-linear relationship was tested. In Styblinski–Tang and Discus100, a parabolic shape showed that the loss increases as the solution deviates more from the optimum (see Figure 1).



	loss	solution candidate
mean	4.13×10^8	77.89
median	0.02	77.88
sd	1.30×10^{11}	11.62
min	-4.93	-3011
max	4.57×10^{13}	3147

Figure 1. Styblinski–Tang on noise level 1 optimized by TBPSA: Loss outliers drastically affect the mean, which is extremely large compared to the median (**a**). The zoomed-in scatterplot shows a W-shaped distribution (**b**).

On Discus100, the datapoints were further away from the optimum solution, but still, TBPSA may have reached it. Its correlation on noise level 1 is particularly high due to its anchor-shaped datapoint distribution, which becomes visible when zooming in (see Figure 2).

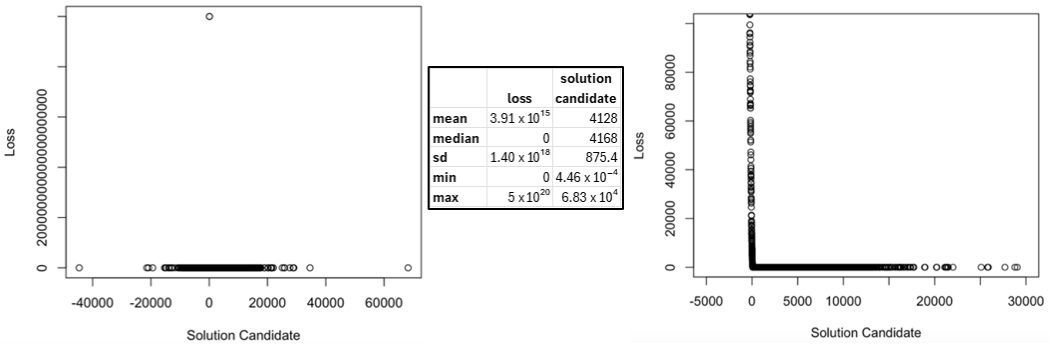


	loss	solution candidate
mean	368.9	82.08
median	1.1	82.09
sd	1.18×10^{5}	1.92
min	3525	−124.40
max	5.94×10^{7}	852.20

(**a**) (**b**)

Figure 2. Discus100 on noise level 1 optimized by TBPSA: The loss' standard deviation is huge compared to the mean and median because of one extreme outlier (**a**). The zoomed-in scatterplot shows an anchor-shaped distribution (**b**).

On Bentcigar, on noise level 1, TBPSA returned a loss value of zero for many different solution candidates; thus, the non-linear correlation is only 1.3%, returning a loss value randomly (see Figure 3).



	loss	solution candidate
mean	3.91×10^{15}	4128
median	0	4168
sd	1.40×10^{18}	875.4
min	0	4.46×10^{-4}
max	5×10^{20}	6.83×10^{4}

(**a**) (**b**)

Figure 3. Bentcigar on noise level 1 optimized by TBPSA: Mean and standard deviation of the loss are extremely large while the median is zero (**a**). The zoomed-in scatterplot reveals a 90° angle (**b**).

Table 7 reports all non-linear correlations per function and noise level.

Table 7. Nonlinear correlation scores (0–100%) for the Styblinski–Tang, Discus100, and Bent Cigar (1,000,000-dimensional) functions, with higher values indicating stronger correlations.

	Styblinski–Tang	Discus 100	Bentcigar
noise level 1	80.7%	94.1%	1.3%
noise level 10	53.4%	66.3%	no correlation

Evaluating only three functions is insufficient for a definitive conclusion, but it suggests that the reliability of the loss as a quality indicator depends on the algorithm's and function's characteristics. This aligns with [2]'s NFLT, which states that no universal optimization algorithm exists, only tailored ones for specific problems. For Styblinski–Tang and Discus100, the loss reliably indicated the solution quality, while for more complex functions such as

Bentcigar and Rosenbrock, it was ineffective. The loss reliability also depends on the noise level, and higher dimensions are likely to reduce reliability. Further research is needed to determine when the loss is a reliable indicator. For the formula by [13], it is essential to first check the correlation between the loss and solution candidates; if no correlation exists, applying the formula is unnecessary. If a correlation is present, the formula must be refined to select a value that accurately represents the optimization state.

4.7. Contribution to the Existing Knowledge Base

A simple, interpretable performance metric for noisy optimization cannot capture all relevant information in a single value. Noise affects the reliability of loss values and can lead to non-monotonic optimization behavior. This research therefore proposes a standardized performance report that combines complementary information about solution quality and computational effort. It extends [13]'s criteria for valid performance metrics with three requirements:

1. The report must be informative under target approximation, non-monotonicity, and stagnation.
2. It must be simple and comprehensible while providing sufficient detail.
3. It must support both fixed-budget and fixed-target approaches, allowing flexibility in prioritizing solution quality or runtime.

The proposed report includes four quantitative inputs as components of the performance metric: the best loss value of the run, the number of function evaluations, its actual wall-clock time and the correlation between the loss and its solution candidates. Loss is reported numerically, function evaluations as a count, wall-clock time in seconds and the correlation as a numerical value between 0 and 1. These values are reported separately rather than combined through a utility function, as proposed by [8], and jointly serve as the basis for interpreting algorithm performance on a noisy optimization problem. The performance of an algorithm (A) on a problem (P) is represented by the metric vector

$$\mathrm{M(A,P)} = \begin{pmatrix} L_{best}(A,\ P) \\ N_{FE}(A,\ P) \\ T_{wall}(A,P) \\ r_{noise}(A,\ P) \end{pmatrix}$$

where L_{best} denotes the best loss value of the run, N_{FE} the number of function evaluations, T_{wall} the measured wall-clock time in seconds and r_{noise} the correlation between loss and solution candidates.

A single loss value does not reveal non-monotonic behavior or stagnation. Therefore, performance profiles are needed to identify these patterns. Such profiles provide information for decisions about allocating smaller budgets to preliminary runs and defining evaluation thresholds for detecting stagnation before rerunning an algorithm with an adjusted budget. Performance profiles visualize the trajectory of loss values, enabling early detection of trends. Although performance profiles cannot be condensed into a single scalar metric, they are included in the proposed report as a visual complement, providing a more comprehensive assessment of an algorithm's performance on a given problem. Furthermore, ref. [13]'s formula for the quality indicator requires further investigation, as it may not adequately represent algorithm performance.

The novelty of this research lies in proposing a standardized reporting framework that presents the information needed to interpret algorithm performance and supports comparisons between algorithms on the same problem. Its effectiveness depends on consistent adoption across benchmarking studies. Thus, the proposed framework improves

transparency and comparability without claiming that loss, function evaluations, wall-clock time or the correlation between loss and solution candidates are themselves new metrics.

Validity of this research: This research addresses the gap of a standardized performance metric in noisy optimization with AAs. The conclusions of this research are based on a literature review combined with experiments including 13 different artificial test functions and two AAs for continuous optimization. For all-encompassing research, real-world problems must be optimized with a variety of AAs with different noise handling features. However, artificial test functions exhibiting isolated real-world features are the state-of-the-art way to benchmark algorithms [3,36,56,65]. The research acknowledges the following limitations:

- Only Gaussian noise distributions were analyzed, with TBPSA designed for such noise [64].
- The experiments were limited to three dimensions and two noise levels.
- A budget of 256,000 was used due to hardware limits but longer runs might yield different results.

Extending the research on these points may provide insights into algorithm performance, yet this is not the main objective of this work.

- Only 13 test functions were used, fewer than the recommended 20 [5].

Since algorithm performances are far from the global optimum for most functions, additional iterations would likely not alter the results from 50 runs.

- The correlation between the loss and solution candidates was calculated for three functions, but more extensive testing is needed across all 13 functions.
- Correlation was calculated for one dimension, limiting the conclusions. Testing in higher dimensions is necessary.
- Reference [13]'s formula for quality indicators was tested only on the Styblinski–Tang and Ellipsoid. It should be applied to other optimization states.
- The correlation and quality indicator formula were not tested on the NoisyBandit's performance.

The last four points are limited by time constraints. While testing more functions and algorithms could support existing findings, it may also challenge them. Therefore, this research does not claim universal validity, and its insights apply within the scope of the reviewed theories, benchmarking practices, and selected functions and algorithms.

Despite the limitations, this work fulfills the research statement with a caveat on comparability across optimization problems:

It is possible to develop an overall performance metric for benchmarking AAs solving single-objective noisy optimization problems, as supported by theoretical arguments and empirical experiments.

Both runtime and loss value are widely used in BBOB [3,13]. The overall performance report combines these two in addition with the respective wall-clock time addressing differences in computational cost of the function evaluations and the correlation between loss and solution candidates which accounts for the loss' reliability, as it depends on the problem and algorithm's noise handling technique. The loss value is not universally reliable, which limits comparability. Combining these four quantitative measures with the corresponding performance profile provides a comprehensive assessment of algorithm performance in noisy optimization, albeit at the cost of additional reporting effort. This standardized reporting framework can be integrated into existing benchmarking guidelines, and further research is needed to confirm or challenge these findings. This work contributes to a broader body of knowledge without being definitive.

5. Conclusions

Our study addresses this research gap by developing a standardized reporting framework for algorithms in single-objective noisy optimization, based on the literature and benchmarking experiments. The benchmark community around [13] advocated runtime in function evaluations as the standard metric in BBOB, with the COCO environment built around it [9]. COCO focuses on standardization and is widely used [34,66,67]. Alongside runtime, the loss value is also commonly used [35,36,53]. Following [13]'s empirical tests for runtime, this work tests the loss value in the Nevergrad environment. Two AAs with noise handling features were tested on a test suite of 13 artificial functions designed to balance real-world characteristics.

5.1. Key Findings

The experiments showed that only one function instance reached the global optimum, highlighting the need for an approximation measure. The loss value is useful for approximating the optimum if available a priori, while runtime cannot be approximated without knowing the optimum. Non-monotonicity was another issue: the fixed-target approach focuses solely on reaching the target, while the fixed-budget approach relies on monotonicity. If the target is not reached within the budget, more runtime may be allocated, especially if the process stalls. In such cases, the fixed-target approach becomes ineffective. Ratio statements for loss and runtime can be made to compare the algorithm performance, but the loss is affected by noise, making its reliability as a quality indicator uncertain. In contrast, the runtime is interpretable as the optimization cost but depends on the noisy loss reaching the target. Thus, considering the correlation between the loss and solution candidates is crucial for assessing the loss as a quality indicator. Additionally, the loss function is tailored to each optimization problem, complicating comparability because the optimum loss values vary. The runtime is also affected by the computational cost of function evaluations and the machine performance, so it must be reported in seconds for comparison. Loss and runtime are complementary; combining them provides a more comprehensive performance metric. Reporting wall-clock time complements the cost for the function evaluations and the correlation between loss and solution candidates accounts for the loss' reliability. While reporting all four values reduces simplicity, it is necessary due to the inherent complexity and uncertainty of noisy optimization. Despite advancements in noise handling and quality indicators, noise remains a challenge and requires ongoing research.

5.2. Implications

Acceptance in the Community: Developing a performance metric for single-objective noisy optimization with AAs by analyzing the strengths and limitations of two established approaches and complementing it with supplementary metrics still leaves the challenge of achieving widespread acceptance as a standard. While loss and runtime are accepted individually and can be reported together (e.g., budget and loss in the fixed-budget approach, or target value and function evaluations in the fixed-target approach), this approach does not fully utilize the available information for interpreting benchmarking results. Performance should be compared in relation to an expected value or another algorithm's performance, and ratio statements help make these comparisons interpretable. While comparisons across different optimization problems are possible, they require additional calculations for both loss and runtime. Moreover, the reliability of loss as an indicator of solution quality remains uncertain under noise. Including a correlation measure between loss and solution quality helps assess this relationship, while reporting wall-clock time complements function evaluation counts by accounting for the computational cost of the optimization process and differences in computational efficiency. Together, these four quantitative measures

and the corresponding performance profile provide a more comprehensive assessment of algorithm performance, albeit at the cost of increased reporting effort. Establishing a standard would therefore require considerable implementation effort. A comprehensive benchmarking platform with predefined visualizations, as demonstrated in [9], would be an ideal solution but is beyond the scope of this research.

Advantage of the Combination: Combining loss, function evaluations, wall-clock time, and the correlation between loss and solution quality provides a more comprehensive basis for evaluating algorithm performance in noisy optimization. Loss and runtime capture complementary aspects of performance, allowing flexible trade-offs between solution quality and computational cost. At the start of any benchmarking experiment with AAs, it is crucial to discuss this trade-off. Based on the decision maker's priorities, either the fixed-budget or fixed-target approach is applied. In a fixed-budget scenario, the solution quality might not meet expectations when the budget is exhausted, potentially requiring an extension or a shift to the fixed-target approach. Conversely, in a fixed-target scenario, it might take too long to achieve the desired quality, prompting a decision maker to adjust the target and set a budget as a termination criterion. In both cases, monitoring helps identify when the acceptable target is reached before the budget is exhausted. Reporting wall-clock time and the correlation measure adds further context to these decisions by quantifying computational effort and clarifying the extent to which loss values can be trusted as indicators of solution quality. Thus, combining the two approaches with supplementary metrics supports a more informed assessment of both optimization efficiency and solution quality.

5.3. Further Research

Limitations referring to shortcomings in the research design are already discussed in Section 4.7, where this research's contribution to the existing knowledge base is assessed. Here, several research directions to enhance noisy optimization and performance metrics are elucidated. First, replicating the experiments for other noisy optimization problems, especially real-world scenarios, is crucial to apply the conclusions to practical settings. While this research focused on the fixed-budget and fixed-target approaches, ref. [22]'s reinforcement learning under uncertainty is a promising area not explored here. Further investigation could contribute valuable insights to the BBOB community. Additionally, ref. [13]'s quality indicator formula needs refinement; the 1% worst selection did not effectively represent the optimization state. This research proposes using the median instead, which requires empirical testing. Overall, researching the conditions under which the loss value is a reliable quality indicator would be beneficial, as this could enable estimating the loss' reliability in advance.

Author Contributions: Conceptualization, R.B.; Methodology, R.B.; Software, R.B.; Validation, R.B. and T.H.; Formal analysis, R.B.; Writing—original draft, R.B.; Writing—review and editing, T.H.; Supervision, T.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

EC	Evolutionary Computation
EA	Evolutionary Algorithm

NFLT	No Free Lunch Theorem
AA	Anytime Algorithm
CMA	Covariance Matrix Adaptation
TBPSA	Test-Based Population Size Adaptation
ES	Evolutionary Strategy
UCB	Upper Confidence Bound

Appendix A

Table A1. Overview of the test functions.

Test Function	Properties	Purpose	Global Optimum
Sphere	Unimodal, separable, rotation-invariant, symmetric	Simplest but classic function to test and compare the convergence velocity	$x = 81$, $f(x) = 0$
Griewank	Highly multimodal, non-separable, symmetric, and adequate global structure This function becomes easier to solve in higher dimensionality!	How to escape regularly distributed local optima?	$x = 81$, $f(x) = 0$
Ackley	Weakly multimodal, non-separable, funnel shape	How to find the funnel's lowest point in a landscape of local optima?	$x = 81$, $f(x) = 0$
Styblinski–Tang	Multimodal with 3 local optima and 1 global optimum, non-separable, classic function for noisy optimization	How to escape the local optima under noisy conditions?	$x = 78.096466$, $f(x) = -0.0001757$
Rastrigin	Highly multimodal, separable, symmetric, and adequate global structure	How to escape regularly distributed local optima?	$x = 81$, $f(x) = 0$
Büche–Rastrigin	Highly multimodal, separable, asymmetric placement of the local optima	How to escape asymmetrically distributed local optima compared to the Rastrigin function?	$x = 81$, $f(x) = 0$
Ellipsoid	Unimodal, separable, ill-conditioned $(10^0 - 10^6)$	How to find the global optimum in a highly ill-conditioned landscape?	$x = 81$, $f(x) = 0$
Discus	Unimodal, separable, ill-conditioned $(10^0 - 10^6)$	How to find the global optimum in a highly ill-conditioned landscape?	$x = 81$, $f(x) = 0$
Rosenbrock	Unimodal, non-separable It becomes multimodal if the dimension > 3!	Can the algorithm follow a bending valley?	$x = 83$, $f(x) = 0$
Bentcigar	Unimodal, non-separable, ill-conditioned $(10^0 - 10^6)$, non-quadratic function due to the symmetry-breaking transformation of the input variables	How to find the global optimum in a highly ill-conditioned, asymmetric ridge landscape?	$x = 81$, $f(x) = 0$
Lunacek	Highly multimodal with two funnels, separable, and deceptive for EAs with large population size	How to not fall for deception and what is a suitable population size?	$x = 83.5$, $f(x) = 0$
Genzcorner-peak	Unimodal, no information about separability found, integration test problem	How to find the highest point instead of getting stuck on the variety of x-values for the rising peak?	$x = 81$, $f(x) = 1$
Deceptive Path	Unimodal, no information about separability found, deceptive	How to follow a long path that becomes narrower the closer it gets to the optimum?	$x = 81$, $f(x) = 0$

Appendix B

Table A2. Summary of the discussion: ++ denotes that a criterion is fulfilled, + denotes that a criterion can be fulfilled with additional effort, − denotes that it can only be partly fulfilled and −− denotes that the criterion is not fulfilled and there is no current solution for this problem.

Criteria [13] and Peculiarities	**Loss Value**	**Number of Function Evaluations (NFE)**
Quantitative and ratio-scaled	+ The percental improvement must be calculated to make ratio comparisons.	++ The NFE can be directly used for calculating ratios since it is ratio-scaled.
Variety of values	++ The loss value can adopt all continuous and interval-scaled floating-point numbers.	++ The NFE can adopt all continuous and interval scaled integers ≥ 0.
Relevant and meaningful	++ The loss value defines the solution quality according to a specific loss function.	++ The NFE defines the cost to achieve a certain solution quality.
Available for algorithms and optimization problems	++ The loss value and the NFE are always available for anytime algorithms and problems solved by them.	
Reproducible	++ If the seed, initialization, and loss function are the same, the loss value is reproducible.	+ A function evaluation may differ in an actual clock's time depending on a machine's capacity.
Comparable	+ Loss functions are tailored to the respective optimization problems. Thus, it must be ensured that the same loss value is achieved across different optimization problems. Otherwise, the optimum losses must be standardized.	− One function evaluation may be of different computational costs for different algorithms due to their operating features and on different functions according to their difficulty. The runtime in seconds must be reported for comparison.
Comprehensible and interpretable	− If the optimum loss is known, the achieved loss value is expressive in terms of an approximation to it. However, understanding the loss function may not be that easy; especially if different functions have tailored loss functions.	++ The NFE is simple and understandable in terms of counting the trials until the global optimum is found.
Target approximation	++ If the optimum loss is known, the difference to the achieved loss value can be computed.	−− There is no such thing as approximation with the NFE without relying on the loss value.
Non-monotonicity	+ The fixed-budget approach relies on monotonicity. However, the global optimum may be found within the budget. If it is not found, additional runtime can be allocated.	++ The fixed-target approach is independent of monotonicity. Ratio statements based on NFEs suggest monotonicity, which is not granted in noisy optimization.
Stagnation	++ The loss value is available and an approximation to the target value can be computed. For the early detection of stagnation, the fixed-budget approach is suitable since a decision whether to allocate more runtime or to stop the optimization process can be made.	−− The fixed-target approach becomes useless. The NFE is not expressive because the same solution quality is achieved no matter how many function evaluations are executed.
Monitoring	++ It is a combination of the fixed-budget and the fixed-target approach, but it is only then advantageous if the target can be found within the fixed budget. In this case, the loss value and the NFE can be equally used as performance metrics.	

Appendix C

Cliff's delta is a relatively rarely used, non-parametric effect-size metric suitable for ordinal data, regardless of their underlying distribution [68,69]. It quantifies the degree to which values in one group tend to be larger than those in the other.

$$Cliff's\ delta(\delta) = \frac{\#\left(x_{i1} > x_{j2}\right) - \#\left(x_{i1} < x_{j2}\right)}{n_1 n_2}$$

The formula calculates the difference between the number of pairwise comparisons denoted by # in which an observation from group 1 (x_{i1}) is smaller than an observation from group 2 (x_{j2}) and the number of comparisons (#) in which the opposite is true. This

difference is then divided by the product of the two groups' sample sizes n_1 and n_2. Thus, every observation in one group is compared with every observation in the other. Cliff's delta ranges from −1 to +1, with 0 indicating no tendency for either group to have larger values than the other. The magnitude of the effect reflects the degree of separation between the two distributions. For the calculation of Cliff's delta, we used [69]'s shiny app.

Table A3. Statistical evaluation of the test runs with the budget containing the best loss value.

Function	Noise/ Dimension	Tested Budgets TBPSA	Tested Budgets NB	Test on Central Tendency	*p*-Value	Effect Size Cliff's δ	Confidence Intervals for Cliff's δ		Winner
							Lower	Upper	
Sphere	n1/d5	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d30	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d5	128,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	0.97	0.9	0.99	TBPSA
	n10/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
Griewank	n1/d5	16,000	256,000	Wilcoxon	$<3 \times 10^{-16}$	0.95	0.88	0.98	TBPSA
	n1/d30	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d5	256,000	256,000	Wilcoxon	$=1 \times 10^{-16}$	0.56	0.31	0.74	TBPSA
	n10/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
Ackley	n1/d5	256,000	256,000	Wilcoxon	=0.007	−0.31	−0.54	−0.05	Noisy Bandit
	n1/d30	500	256,000	Wilcoxon	$=3 \times 10^{-15}$	−0.92	−0.98	−0.74	Noisy Bandit
	n1/d50	256,000	256,000	Wilcoxon	$=7 \times 10^{-16}$	−0.94	−0.98	−0.79	Noisy Bandit
	n10/d5	256,000	256,000	Wilcoxon	$=4 \times 10^{-8}$	−0.64	−0.8	−0.39	Noisy Bandit
	n10/d30	64,000	256,000	Wilcoxon	$=3 \times 10^{-16}$	−0.95	−0.99	−0.82	Noisy Bandit
	n10/d50	32,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	−1	−1	−0.98	Noisy Bandit
Styblinski–Tang	n1/d5	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d30	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Welch's T-test	$=12 \times 10^{-13}$	1	1	1	TBPSA
	n10/d5	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
Rastrigin	n1/d5	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d5	256,000	256,000	Wilcoxon	$=1 \times 10^{-15}$	0.93	0.77	0.98	TBPSA
	n10/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	0.99	1	TBPSA
Büche–Rastrigin	n1/d5	128,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d5	128,000	256,000	Wilcoxon	$=4 \times 10^{-13}$	0.84	0.63	0.94	TBPSA
	n10/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
Ellipsoid	n1/d5	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d30	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d5	256,000	256,000	Wilcoxon	$=1 \times 10^{-15}$	0.93	0.78	0.98	TBPSA
	n10/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA

Table A3. *Cont.*

Function	Noise/ Dimension	Tested Budgets TBPSA	Tested Budgets NB	Test on Central Tendency	*p*-Value	Effect Size Cliff's δ	Confidence Intervals for Cliff's δ		Winner
							Lower	Upper	
Discus100	n1/d5	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d5	256,000	256,000	Wilcoxon	$=3 \times 10^{-16}$	0.95	0.82	0.99	TBPSA
	n10/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	0.99	1	TBPSA
	n10/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
Discus10000	n1/d5	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d30	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d5	256,000	256,000	Wilcoxon	$=1 \times 10^{-14}$	0.9	0.73	0.96	TBPSA
	n10/d30	256,000	256,000	Wilcoxon	$=1 \times 10^{-14}$	1	1	1	TBPSA
	n10/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
Discus	n1/d5	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d30	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d5	256,000	256,000	Wilcoxon	$=7 \times 10^{-15}$	0.9	0.72	0.97	TBPSA
	n10/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
Rosenbrock	n1/d5	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d5	256,000	256,000	Wilcoxon	$=9 \times 10^{-14}$	0.86	0.68	0.95	TBPSA
	n10/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	0.99	1	TBPSA
	n10/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
Bentcigar100	n1/d5	128,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d30	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d5	256,000	256,000	Wilcoxon	$<3 \times 10^{-8}$	−0.65	−0.81	−0.39	TBPSA
	n10/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
Bentcigar 10,000	n1/d5	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d5	16,000	256,000	Wilcoxon	=0.2	0.14	−0.12	0.39	none
	n10/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
Bentcigar 1,000,000	n1/d5	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d30	256,000	256,000	Welch's T-test	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d5	32,000	256,000	Wilcoxon	=0.03	−0.24	−0.48	0.03	Noisy Bandit
	n10/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
Lunacek	n1/d5	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n1/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d5	256,000	256,000	Wilcoxon	$=6 \times 10^{-15}$	0.9	0.76	0.96	TBPSA
	n10/d30	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA
	n10/d50	256,000	256,000	Wilcoxon	$<2 \times 10^{-16}$	1	1	1	TBPSA

References

1. De Jong, K.; Fogel, D.B.; Schwefel, H.-P. A History of Evolutionary Computation. In *Handbook of Evolutionary Computation*; Bäck, T., Fogel, D.B., Michalewicz, Z., Eds.; Oxford University: Oxford, UK; CRC Press: Boca Raton, FL, USA, 1997; p. 1130. [CrossRef]
2. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]

3. Bartz-Beielstein, T.; Doerr, C.; Bossek, J.; Chandrasekaran, S.; Eftimov, T.; Fischbach, A.; Kerschke, P.; López-Ibáñez, M.; Malan, K.; Moore, J.; et al. Benchmarking in Optimization: Best Practice and Open Issues. *arXiv* **2020**, arXiv:2007.03488. [CrossRef]
4. Hooker, J.N. Testing heuristics: We have it all wrong. *J. Heuristics* **1995**, *1*, 33–42. [CrossRef]
5. Beiranvand, V.; Hare, W.; Lucet, Y. Best practices for comparing optimization algorithms. *Optim. Eng.* **2017**, *18*, 815–848. [CrossRef]
6. Rakshit, P.; Konar, A.; Das, S. Noisy Evolutionary Optimization Algorithms–A Comprehensive Survey. *Swarm Evol. Comput.* **2017**, *33*, 18–45. [CrossRef]
7. Zilberstein, S. Using Anytime Algorithms in Intelligent Systems. *AI Mag.* **1996**, *17*, 3. [CrossRef]
8. Jesus, A.D.; Liefooghe, A.; Derbel, B.; Paquete, L. Algorithm selection of Anytime Algorithms. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*; ACM: New York, NY, USA, 2020; pp. 850–858. [CrossRef]
9. Hansen, N.; Auger, A.; Ros, R.; Mersmann, O.; Tušar, T.; Brockhoff, D. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *Optim. Methods Softw.* **2021**, *36*, 114–144. [CrossRef]
10. Hellwig, M.; Beyer, H.-G. Benchmarking Evolutionary Algorithms for Single Objective Real-valued Constrained Optimization—A Critical Review. *Swarm Evol. Comput.* **2019**, *44*, 927–944. [CrossRef]
11. Bossek, J.; Kerschke, P.; Trautmann, H. Anytime Behavior of Inexact TSP Solvers and Perspectives for Automated Algorithm Selection. In *2020 IEEE Congress on Evolutionary Computation (CEC)*; IEEE: Piscataway, NJ, USA, 2020; pp. 1–8. [CrossRef]
12. Carlin, A.; Zilberstein, S. Decentralized monitoring of distributed AAs. In *The 10th International Conference on Autonomous Agents and Multiagent Systems—Volume 1*; IFAAMAS: Richland, SC, USA, 2011; pp. 157–164. Available online: https://dl.acm.org/doi/abs/10.5555/2030470.2030493 (accessed on 13 August 2026).
13. Hansen, N.; Auger, A.; Brockhoff, D.; Tušar, T. Anytime Performance Assessment in Blackbox Optimization Benchmarking. *IEEE Trans. Evol. Comput.* **2022**, *26*, 1293–1305. [CrossRef]
14. Hellwig, M.; Beyer, H.-G. 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*; Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; Volume 9921, pp. 26–36. [CrossRef]
15. Jin, Y.; Sendhoff, B. Trade-Off between Performance and Robustness: An Evolutionary Multiobjective Approach. In *Evolutionary Multi-Criterion Optimization*; Fonseca, C.M., Fleming, P.J., Zitzler, E., Thiele, L., Deb, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2632, pp. 237–251. [CrossRef]
16. Jin, Y.; Branke, J. Evolutionary optimization in uncertain environments-a survey. *IEEE Trans. Evol. Comput.* **2005**, *9*, 303–317. [CrossRef]
17. Branke, J.; Schmidt, C. Selection in the Presence of Noise. In *Genetic and Evolutionary Computation—GECCO 2003*; Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L.D., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R., Kendall, G., Wilson, S., et al., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 766–777. [CrossRef]
18. Zilberstein, S.; Russell, S. Optimal composition of real-time systems. *Artif. Intell.* **1996**, *82*, 181–213. [CrossRef]
19. Boddy, M.; Dean, T. Solving Time-Dependent Planning Problems. In Proceedings of the International Joint Conference on Artificial Intelligence, Detroit, MI, USA, 20–25 August 1989. Available online: https://www.semanticscholar.org/paper/Solving-Time-Dependent-Planning-Problems-Boddy-Dean/fc078a34b3b854602ba932fe3afc4b00060b2a48 (accessed on 13 August 2026).
20. Russell, S.; Wefald, E. Principles of metareasoning. *Artif. Intell.* **1991**, *49*, 361–395. [CrossRef]
21. Hansen, E.; Zilberstein, S. Monitoring the Progress of Anytime Problem-Solving. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI/IAAI)*; AAAI Press: Menlo Park, CA, USA, 1996; pp. 1229–1234.
22. Svegliato, J.; Sharma, P.; Zilberstein, S. A Model-Free Approach to Meta-Level Control of AAs. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*; IEEE: Piscataway, NJ, USA, 2020; pp. 11436–11442. [CrossRef]
23. Bui, L.T.; Abbass, H.A.; Essam, D. Fitness inheritance for noisy evolutionary multi-objective optimization. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*; ACM: New York, NY, USA, 2005; pp. 779–785. [CrossRef]
24. Syberfeldt, A.; Ng, A.; John, R.I.; Moore, P. Evolutionary optimisation of noisy multi-objective problems using confidence-based dynamic resampling. *Eur. J. Oper. Res.* **2010**, *204*, 533–544. [CrossRef]
25. Grass, J.; Zilberstein, S. A Value-Driven System for Autonomous Information Gathering. *J. Intell. Inf. Syst.* **2000**, *14*, 5–27. [CrossRef]
26. Hansen, E.A.; Zilberstein, S. Monitoring and control of Anytime Algorithms: A dynamic programming approach. *Artif. Intell.* **2001**, *126*, 139–157. [CrossRef]
27. Zilberstein, S. Operational Rationality through Compilation of Anytime Algorithms. Ph.D. Dissertation, University of California at Berkeley, Berkeley, CA, USA, 1993.
28. Qian, C.; Yu, Y.; Jin, Y.; Zhou, Z.-H. On the Effectiveness of Sampling for Evolutionary Optimization in Noisy Environments. In *Parallel Problem Solving from Nature—PPSN XIII*; Bartz-Beielstein, T., Branke, J., Filipič, B., Smith, J., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2014; pp. 302–311. [CrossRef]

29. Akimoto, Y.; Astete-Morales, S.; Teytaud, O. Analysis of runtime of optimization algorithms for noisy functions over discrete codomains. *Theor. Comput. Sci.* **2015**, *605*, 42–50. [CrossRef]
30. Hellwig, M.; Beyer, H.-G. A Linear Constrained Optimization Benchmark for Probabilistic Search Algorithms: The Rotated Klee-Minty Problem. In *Theory and Practice of Natural Computing*; Fagan, D., Martín-Vide, C., O'Neill, M., Vega-Rodríguez, M.A., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 139–151. [CrossRef]
31. Ruml, W. The Logic of Benchmarking: A Case Against State-of-the-Art Performance. *Proc. Int. Symp. Comb. Search* **2010**, *1*, 142–143. [CrossRef]
32. Goldwasser, M.; Johnson, D.; McGeoch, C. (Eds.) *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*; American Mathematical Society: Providence, RI, USA, 2002; Volume 59. [CrossRef]
33. Weise, T.; Chiong, R.; Lassig, J.; Tang, K.; Tsutsui, S.; Chen, W.; Michalewicz, Z.; Yao, X. Benchmarking Optimization Algorithms: An Open Source Framework for the Traveling Salesman Problem. *IEEE Comput. Intell. Mag.* **2014**, *9*, 40–52. [CrossRef]
34. Hansen, N. Benchmarking a BI-population CMA-ES on the BBOB-2009 noisy testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*; ACM: New York, NY, USA, 2009; pp. 2397–2402. [CrossRef]
35. Liu, C.; Sun, F.; Ni, Q.; Wang, L.; Yang, B. A Novel Graphic Bending Transformation on Benchmark. *arXiv* **2020**, arXiv:2004.10042. [CrossRef]
36. Plevris, V.; Solorzano, G. A Collection of 30 Multidimensional Functions for Global Optimization Benchmarking. *Data* **2022**, *7*, 46. [CrossRef]
37. Bennet, P.; Doerr, C.; Moreau, A.; Rapin, J.; Teytaud, F.; Teytaud, O. Nevergrad: Black-box optimization platform. *ACM SIGEVOlution* **2021**, *14*, 8–15. [CrossRef]
38. Nevergrad—A Gradient-Free Optimization Platform. Available online: https://github.com/facebookresearch/nevergrad/tree/main/nevergrad (accessed on 5 November 2023).
39. Finck, S.; Ros, R. COCO (COmparing Continuous Optimizers) Software: User Documentation. 2011. Available online: https://inria.hal.science/inria-00597334/document (accessed on 8 August 2026).
40. How to Perform Optimization—Nevergrad Documentation. Available online: https://facebookresearch.github.io/nevergrad/optimization.html (accessed on 8 November 2023).
41. Nevergrad—Frozen Experiments. GitHub. Available online: https://github.com/facebookresearch/nevergrad/blob/e1f92d7b33a5260682318245c3c1ce42f8730f54/nevergrad/benchmark/frozenexperiments.py (accessed on 10 November 2023).
42. Bbob-Noisy Data Archive. Coco-Data-Archive. Available online: https://numbbo.github.io/data-archive/bbob-noisy/ (accessed on 10 November 2023).
43. Doerr, C.; Wang, H.; Ye, F.; van Rijn, S.; Bäck, T. IOHprofiler: A Benchmarking and Profiling Tool for Iterative Optimization Heuristics. *arXiv* **2018**, arXiv:1810.05281. [CrossRef]
44. Huband, S.; Hingston, P.; Barone, L.; While, L. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evol. Comput.* **2006**, *10*, 477–506. [CrossRef]
45. Brockhoff, D.; Auger, A.; Hansen, N.; Tušar, T. Using Well-Understood Single-Objective Functions in Multiobjective Black-Box Optimization Test Suites. *Evol. Comput.* **2022**, *30*, 165–193. [CrossRef]
46. Bäck, T.; Michalewicz, Z. Test Landscapes. In *Handbook of Evolutionary Computation*, 1st ed.; Bäck, T., Fogel, D.B., Michalewicz, Z., Eds.; CRC Press: Boca Raton, FL, USA, 1997; pp. 14–20. [CrossRef]
47. Schwefel, H. *Evolution and Optimum Seeking*; Sixth-Generation Computer Technology Series; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1995.
48. Datta, B.N. *Numerical Methods for Linear Control Systems*, 1st ed.; Datta, B.N., Ed.; Academic Press: Cambridge, MA, USA, 2003. [CrossRef]
49. Sastry, K.; Pelikan, M.; Goldberg, D.E. Decomposable Problems, Niching, and Scalability of Multiobjective Estimation of Distribution Algorithms. *arXiv* **2005**, arXiv:cs/0502057. [CrossRef]
50. Nevergrad Artificial Functions. GitHub. Available online: https://github.com/facebookresearch/nevergrad/tree/e1f92d7b33a5260682318245c3c1ce42f8730f54/nevergrad/functions (accessed on 10 November 2023).
51. Nevergrad—Optimizerlibrary. GitHub. Available online: https://github.com/facebookresearch/nevergrad/blob/e1f92d7b33a5260682318245c3c1ce42f8730f54/nevergrad/optimization/optimizerlib.py (accessed on 10 November 2023).
52. Dieterich, J.M.; Hartke, B. Empirical review of standard benchmark functions using evolutionary global optimization. *arXiv* **2012**, arXiv:1207.4318. [CrossRef]
53. Hayashida, T.; Nishizaki, I.; Sekizaki, S.; Koto, S. Cooperative Particle Swarm Optimization in Distance-Based Clustered Groups. *J. Softw. Eng. Appl.* **2017**, *10*, 143–158. [CrossRef]
54. Helwig, S.; Branke, J.; Mostaghim, S. Experimental Analysis of Bound Handling Techniques in Particle Swarm Optimization. *IEEE Trans. Evol. Comput.* **2013**, *17*, 259–271. [CrossRef]

55. Nevergrad—Parametrization. GitHub. Available online: https://github.com/facebookresearch/nevergrad/tree/e1f92d7b33a5260682318245c3c1ce42f8730f54/nevergrad (accessed on 11 November 2023).
56. Hansen, N.; Finck, S.; Ros, R.; Auger, A. Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. R-6829, version 2, INRIA. 2019. ⟨inria-00362633v2⟩. Available online: https://inria.hal.science/inria-00362633/document/ (accessed on 11 November 2023).
57. Heidrich-Meisner, V.; Igel, C. Hoeffding and Bernstein races for selecting policies in evolutionary direct policy search. In *Proceedings of the 26th Annual International Conference on Machine Learning*; ACM: New York, NY, USA, 2009; pp. 401–408. [CrossRef]
58. Oquab, M.; Rapin, J.; Teytaud, O.; Cazenave, T. Parallel Noisy Optimization in Front of Simulators: Optimism, Pessimism, Repetitions Population Control. In *Proceedings of the Data-Driven Optimization and Applications at CEC 2019*; IEEE: Piscataway, NJ, USA, 2019.
59. Cauwet, M.-L.; Teytaud, O. Population Control meets Doob's Martingale Theorems: The Noise-free Multimodal Case. *arXiv* **2020**, arXiv:2005.13970. [CrossRef]
60. Examples of Benchmarks—Nevergrad Documentation. Available online: https://facebookresearch.github.io/nevergrad/benchmarks.html (accessed on 11 November 2023).
61. Reference for TBPSA? · Issue #273 · Facebookresearch/Nevergrad. GitHub. Available online: https://github.com/facebookresearch/nevergrad/issues/273 (accessed on 11 November 2023).
62. Robbins, H. Some aspects of the sequential design of experiments. *Bull. Am. Math. Soc.* **1952**, *58*, 527–535. [CrossRef]
63. Kleinberg, R. Anytime Algorithms for multi-armed bandit problems. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm—SODA '06*; ACM: New York, NY, USA, 2006; pp. 928–936. [CrossRef]
64. Khalidov, V.; Oquab, M.; Rapin, J.; Teytaud, O. Consistent population control: Generate plenty of points, but with a bit of resampling. In *Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*; ACM: New York, NY, USA, 2019; pp. 116–123. [CrossRef]
65. Nevergrad—Corefuncs.py. GitHub. Available online: https://github.com/facebookresearch/nevergrad/blob/e1f92d7b33a5260682318245c3c1ce42f8730f54/nevergrad/functions/corefuncs.py (accessed on 13 November 2023).
66. Loshchilov, I.; Schoenauer, M.; Sebag, M. Black-box optimization benchmarking of IPOP-saACM-ES and BIPOP-saACM-ES on the BBOB-2012 noiseless testbed. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*; ACM: New York, NY, USA, 2012; pp. 175–182. [CrossRef]
67. Nishida, K.; Akimoto, Y. Evaluating the Population Size Adaptation Mechanism for CMA-ES on the BBOB Noisy Testbed. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*; ACM: New York, NY, USA, 2016; pp. 1193–1200. [CrossRef]
68. Cliff, N. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychol. Bull.* **1993**, *114*, 494–509. [CrossRef]
69. Meissel, K.; Yao, E.S. Using Cliff's Delta as a Non-Parametric Effect Size Measure: An Accessible Web App and R Tutorial. *Pract. Assess. Res. Eval.* **2024**, *29*, 2.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.