Revisiting CEC 2022 ranking: A new ranking method and influence of parameter tuning[⋆]

Rafał Biedrzycki[*]

Warsaw University of Technology, Institute of Computer Science

Abstract

Optimization competitions give an impulse to develop optimization algorithms. However, there is no common agreement on how to rank the contestants. This paper proposes a method of assessing the performance of the algorithms. The proposed mark is easily interpretable by humans and can be compared with previously published marks. The proposed method was used to create a ranking for contestants of the CEC 2022 competition on single objective bound constrained numerical optimization. The resulting ranking is different from the official one. The ranks of some algorithms differ by up to five. As the proposed ranking is more focused on the results at the end of the budget, winning algorithms are better suited for most real-world applications. Since the tuning effort influences the algorithm's results, the paper also examines the influence of parameter tuning on the place achieved in both rankings by the top four algorithms. The parameters to tune were extracted from the papers that introduced the algorithms and from the source codes. The results showed that all considered algorithms were not carefully tuned for CEC. Generally, tuning using the target performance metric helps, but tuning using the other metric is harmful. The profit of tuning is up to a 33% increase in the number of trails that found the global optimum. The ablation analyses of the algorithms' parameters showed that only a few parameters strongly influence the results. Frequently, parameters not listed in the papers are among the most important.

Keywords: CEC 2022, ranking, tuning, benchmarking

1. Introduction

Optimization competitions give an impulse to develop optimization algorithms. Competition results also make it easier to find state-of-the-art algo-

[⋆]©2024. This manuscript version is made available under the CC-BY-NC-ND 4.0 license http://creativecommons.org/licenses/by-nc-nd/4.0/. This is self-archiving version of an accepted article. DOI: 10.1016/j.swevo.2024.101623

[*]Corresponding author

Email address: `rafal.biedrzycki@pw.edu.pl` (Rafał Biedrzycki) rithms. To be able to order the contestants from the best to the worst, each competition requires some objective function. Still, no common agreement exists on how this function should be composed. There are two complementary approaches for single objective optimization: fixed cost and fixed target. In a fixed target scenario, the target fitness level is assumed, and algorithms are compared by the number of fitness evaluations (FES) needed to achieve the target. In a fixed-cost approach, the budget of fitness evaluations is assumed, and algorithms are compared by the error value achieved after reaching the budget. In practice, these approaches are more complicated because there are cases when specified fitness cannot be reached, and achieving extremely small error values is not practical. One of the families of the single objective optimization competitions is connected with the Congress of Evolutionary Computation (CEC). The first CEC competition was held in 1996 [1]. Since then, the competition has been organized nearly regularly, but its newer versions usually use other performance measures than the older ones. As was noticed in [2], CEC performance measures do not allow us to infer how much the new competition's winners improved over the older ones. Moreover, finding a place for a new algorithm in published ranking is impossible without recalculating all scores from raw data. Without context (e.g., taken from the publication), CEC scores alone say nothing about the algorithm.

The most recent CEC competitions [3, 4] reward for speed of the algorithms without preference for problem-solving ability over speed. From a practical point of view, solving more problems is more important than improving the algorithm's performance. The score achieved by an algorithm in the aforementioned competitions can be interpreted as the number of its wins when all of its trials are compared to all trials from all other algorithms. In practice, the score is hardly interpretable when many algorithms and independent runs are involved, e.g., 123456789 vs. 123457789 – is the difference meaningful in practice?

The second most popular benchmark is the Black Box Optimization Benchmark (BBOB) [5]. The BBOB does not provide a ranking of the algorithms; it focuses on visualization of the performance of the algorithms using plots of Empirical Cumulative Distribution Functions (ECDF).

Each contemporary algorithm has many parameters. Both CEC 2017 and CEC 2022 require participants not to search for distinct parameters for each problem/dimension. Contestants should also provide dynamic ranges of the parameters, guides on how to tune them, and the estimated cost of the tuning. Similar rules were applied during Real-Parameter Black-Box Optimization Benchmarking [6]. Having those loose requirements, one contestant can use minimal tuning by hand, and the other can use tools for automatic parameter tuning. It was previously noticed [7] that comparison of the algorithms with the different levels of tuning is not fair and should be avoided. The importance of parameter tuning before benchmarking is also discussed in [8]. Since knowledge of the tuning effort of CEC 2022 participants is not available, the published ranking may reflect time spent on tuning rather than the performance of the algorithms.

For further development of evolutionary algorithms, it is necessary to know which parameters or components of the top-rated methods strongly influence their position in the ranking. Unfortunately, no such analysis exists in the publications presenting the CEC 2022 winners.

The main contributions of the paper are:

- The proposal of a new method of assessing the performance of metaheuristic optimization algorithms. The proposed performance metric is humanly interpretable; it allows for easy comparison of the results of a new algorithm to the published results, and it has a built-in preference for problem-solving over speed.
- Presentation of the alternative ranking for CEC 2022 bound constrained single objective optimization competition. The resulting new ranking is compared to the old one and to the rankings based on the area under ECDF curves and based on values of ECDF curves achieved at the end of the budget.
- Experimental examination of the influence of tuning the parameters of the top four algorithms on both rankings. Tuning by the same method and using the same budget makes the ranking more trustworthy.
- Extraction of hidden parameters from the source codes of the top four algorithms and making the algorithms configurable from the command line. The parametrized codes are made available to the community.
- Examination of the influence of the parameter values on the results and ranks of the top four algorithms. The most important parameters of the examined algorithms were identified.

The article is composed in the following way. Section 2 provides a literature survey. Section 3 describes the proposed ranking method. The ranking is calculated for CEC 2022 bound constrained single objective optimization contestants and compared to the official one. Section 4 examines the influence of parameter tuning on both rankings. Section 5 summarizes the observations and concludes the paper.

2. Related work

2.1. Competitions and rankings

When a new algorithm is proposed, it should be compared with state-of-the-art. The state-of-the-art can be easily identified thanks to the competitions and rankings they provide. As mentioned earlier, one of the families of the competitions is connected with the Congress of Evolutionary Computation (CEC). The first CEC competition was held in 1996 [1]. At that time, the primary performance measure used the expected number of function evaluations per success (ENES), where success means that the method achieved a specified error value. The ENES was renamed to Expected Runtime (ERT) and used as a performance measure for the Comparing Continuous Optimizers (COCO) benchmarking platform [6]. The COCO is used by The Black Box Optimization Benchmark (BBOB) competition series [5]. As some methods may not reach the specified error level, COCO defines 51 error targets ranging from 10^2 to 10^{-8} [9]. The fractions of target fitness values achieved in consecutive iterations by the best-so-far solution observed in that run are compiled into an Empirical Cumulative Distribution Function (ECDF) [9]. The ECDF curves can be averaged over many independent runs of the algorithm and many different optimization problems. Thus, they are a convenient tool for results aggregation and presentation. The BBOB competition does not provide a ranking of algorithms, but some authors propose ranking them by the area under their ECDF curves [10]. This ranking favors algorithms that are efficient at the beginning of the search over methods that are oriented toward exploitation at the end of the search, as discussed in Section 3.3.

The CEC 2017 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization (SOBC) [11] used two combined measures of function error: $50(SE_{min}/SE + SR_{min}/SR)$, where SE is dimension weighted sum of errors of all functions, SE_{min} is the minimal sum of errors of all algorithms, SR is the dimension-weighted sum of ranks, where ranks were based on mean error values, and SR_{min} is the minimal sum of ranks from all algorithms. The benchmark assumed weight of 0.1 for 10 dimensions, 0.2 for 30, 0.3 for 50, and 0.4 for 100. CEC 2020 [12] and CEC 2021 [13] adopted the same idea, but error scores were normalized so that each function contributed more equitably. The quality measure used in this method does not show how many functions were solved. It also does not prefer a faster method when an identical objective function error level is found.

During CEC 2005 [14], the algorithm was considered better when it had more success. This rule was applied when at least one algorithm had at least one success. On the other hand, when there were no success trials, the algorithm with a lower median error was preferred. This approach does not account for error levels when at least one success occurred. It also does not promote faster algorithms.

The CEC 2022 Competition on Single Objective Bound Constrained Numerical Optimization (SOBC) [3] takes into consideration both accuracy and speed without making any assumptions about their relative importance. The ranking method is based on the Wilcoxon rank-sum test [15] (Mann-Whitney U-test [16]). As usual, each algorithm has n trials on each function. To compare m algorithms on one function, FES orders the results of all trials that achieved the desired error level. The rest of the trials are ordered by the error level achieved at the end of the budget. The best trial gets the highest rank $(n \cdot m)$. For each function, score of the algorithm is a sum of the ranks of its trials, minus the correction term $n(n+1)/2$. The final score of the algorithm is the sum of its function scores. The elaborate explanation of the rationale behind the CEC 2022 ranking together with the survey of ranking methods used in CEC competitions is available in [17]. This method does not show how many functions were solved. Its other deficiencies were discussed in Section 1.

In [18], a chess rating system is applied to create a ranking of evolutionary algorithms. The rating system is based on the Glicko-2 [19] chess rating system. The evolutionary algorithms are treated as chess players. Each algorithm is described by rating R, rating deviation RD, and rating volatility σ. The σ indicates expected rating fluctuations. The parameters of the new algorithm are initialized to $R = 1500$, $RD = 350$, $\sigma = 0.06$. Specific equations update these parameters after each tournament. In the case of evolutionary algorithms, the tournament consists of k algorithms, N problems, and n independent runs. For each problem, each run, for each pair of the algorithms, the algorithm with the result closer to the optimum wins. If the difference between algorithms is below the specified threshold, the game's result is a draw. The authors identified the benefits of their approach, like robustness to outliers, but also identified its limitations, like the change of the ranking list after the addition of the algorithm and the fact that already published results cannot be used to create the proposed ranking. It can also be observed that the ratings do not show how many functions were solved. This system also does not prefer a faster method when an identical objective function error level is found.

In [20], a fixed target approach is used to compare stochastic solvers. For each solver, the observed variable is runtime or the number of function evaluations required to achieve the desired objective function value. In this approach for specified dimensionality and objective function, n independent runs are performed. The results are analyzed to identify the best-matching statistical distribution. Knowing the distribution allows for using appropriate statistical tests to determine whether there is a significant difference between the solvers. The paper proposes to find distribution parameters for different dimensionalities and build a model for each distribution parameter. Having the models allows for the prediction of the algorithm's behavior for larger dimensionalities of the problem. For this approach, the target objective function level should be appropriately set to obtain a 100% hit ratio. Objective functions with variable dimensionality are required to obtain a predictive model. If this approach were used to create a ranking, then it would be strongly influenced by the speed of the algorithms.

2.2. Parameter tuning and ablation analysis

Besides educated guesses by the algorithm's author, there are more systematic ways of setting parameter values. One of the most obvious ways is the exhaustive search on the product of discretized parameter values (grid search).

Another possibility is the application of the Taguchi design method [21] to parameter tuning [22]. As for a grid search, a few levels should be proposed for each parameter. The Taguchi method requires fever experiments than the grid search. The decision on which combination of parameters should be evaluated is taken according to specially designed orthogonal arrays. As random factors influence results, this method maximizes signal-to-noise ratio.

There is also a group of parameter tuning methods that treat tuning as an optimization task. One of them is based on Bayesian optimization (BO) [23]. Typically, BO has two components: a Bayesian statistical model for mapping from hyperparameter values to the values of the objective function and an acquisition function, which decides where to sample the next set of observations. The optimization starts from a certain number of random samples of parameters values, which are evaluated by using an optimization algorithm. After that, the model is updated iteratively. It was shown [23] that BO obtains better results in fever evaluations than random search due to its ability to reason about the quality of the experiment before running it.

In [24], authors propose a Surrogate-based Bayesian Hyperparameter Optimizer (SUBHO). In SUBHO, each hyperparameter of EA is represented as a bound-constrained integer. The objective function to be minimized is defined as follows: $f(x_t) = q(x_0) + (100 - q(x_t))$, where q is the performance metric defined by CEC 2021 competition organizers [13], x_0 are the initial parameters and x_t are parameters at iteration t. SUBHO minimizes the f using Bayesian optimization with a limit of 128 function evaluations.

In [25], a parameter tuning method based on racing is proposed. The iterated racing repeats three steps: 1) sampling new configurations according to specified distribution; 2) rejecting weak configurations by racing; 3) changing distributions to maximize the chance of selecting the best configurations. The implementation of that approach is readily available in the package *irace* [26].

There are also methods that use specially designed metaheuristics for parameter tuning, like gender-based genetic algorithms [27] and ParamILS [28].

As many algorithms tend to be overcomplicated [29] and some parameters undergo adaptation, the influence of some parameters could be marginal. To order parameters from the most important to the least important, an ablation analysis [30] could be used. The input to ablation analysis is two configurations: source (not tuned) and target (tuned). The parameters that have identical values in the source and target are ignored. The rest of the process has as many iterations as the number of remaining parameters. Each iteration generates a sequence of configurations where one parameter is taken from the target and the rest from the source. Each configuration is tested using a specified number of independent runs. The best parameter is identified, and it is fixed in further iterations. The implementation of ablation is also available in the package *irace*.

3. Proposed ranking method

3.1. Introduction

As discussed earlier, CEC 2022 SOBC considers both the speed and the error level without prioritizing error. Therefore, an algorithm faster by one objective function evaluation may achieve a better rank than an algorithm that solves more problems. Table 1 shows an example of such a situation and Table 2 presents the resulting ranking (the more, the better). The example assumes that the global minimum has fitness equal to zero, and there are two competing algorithms, 'A' and 'B', each run three times. Table 1 presents achieved fitness and the number of used fitness function evaluations (in parentheses). Table 2 presents ranks according to CEC 2022 SOBC. Even though algorithm 'A' always

Table 1: Exemplary results of two hypothetical algorithms. The number of used objective function evaluations (FES) is shown in parentheses. Assumed budget is 20 FES.

Algorithm	run 1	run 2	run 3
A	0 (12)	0 (10)	0 (11)
B	1 (20)	0 (8)	0 (9)

Table 2: CEC 2022 SOBC ranking for results from Table 1 (the more the better).

Algorithm	run 1	run 2	run 3	sum	CEC score
A	2	4	3	9	3
B	1	6	5	**12**	**6**

quickly found the global optimum, algorithm 'B', which failed, won because it was slightly faster in two runs.

3.2. Proposed performance metric

From the point of view of practical application, e.g., [31, 32, 33], the best algorithm is the one that finds the global optimum. Of the algorithms that cannot do this, the best is the one that achieves the smallest error. The improved performance of the algorithm is a nice bonus. Based on the observations above, this paper proposes to use three ordered criteria to rank algorithms during competitions:

1. Percent of trails that found the global optimum. The optimum is considered to be found when the objective function error is less or equal to 10^{-8}.
2. Percent of thresholds achieved by all trials. The thresholds are set up like for creating ECDF curves [9]. There are 51 thresholds, which are evenly spaced on a logarithmic scale from 10^3 to 10^{-8}.
3. Percent of budget left. The run is interrupted when the algorithm finds the global optimum.

Since the criteria are ordered, and no lower-order criterion should be able to compensate for a significant loss in a higher-order criterion, these criteria can be combined by weighting.

$$q = c_1 + wc_2 + w^2c_3, \tag{1}$$

where c_1, c_2, c_3 are the results of criteria 1, 2, 3, and w is a weight. From the practical point of view, a weight smaller or equal to 0.01 is acceptable. All ratings (c_1, c_2, c_3) are calculated using many independent runs (30 for CEC 2022). They can be averaged for all functions in the benchmark and all dimensionalities to produce a single value for each algorithm. The value can be used to create ranking and can be understood by humans when w is a "nice" number (e.g., 100, 1000), but for better readability c_1, c_2, c_3 can be shown as three columns in tables. Like the CEC 2022 ranking method, the proposed method uses both

Table 3: An example of the application of the proposed performance metric for results from Table 1. Each triplet contains the percent of trials that found the global optimum (c_1), the percent of thresholds achieved by all trials (c_2), and the percent of budget left (c_3).

Algorithm	c_1	c_2	c_3	rank
A	100	100	45	1
B	67	76	38	2

a fixed target and fixed cost approach, but unlike CEC, the proposed method allows one to determine the place of new algorithms in previously published rankings.

As an example of the calculations involved in the application of the proposed method, the algorithms' marks and ranks for the data from Table 1 have been determined. It can be easily observed that 100% trials ($c_1 = 100$) finished successfully for algorithm 'A' but only 67% for 'B'. As 'A' always finds the global optimum, it also achieves 100% of the thresholds ($c_2 = 100$). On the other hand, 'B' achieved 76% of the thresholds ($(14 + 51 + 51)/(3 \cdot 51) \cdot 100$). Assuming that the budget was set to 20 evaluations of the objective function (FES), 'A' spared $8 + 10 + 9$ FES, which gives 45% of the budget ($c_3 = 45$). 'B' spared $0 + 12 + 11$, which gives 38% of the budget. 'A' scored better by about 33 percentage points, winning the ranking. The results presented as triplets (c_1, c_2, c_3) are presented in Table 3.

3.3. Application to the CEC 2022 SOBC competition

The proposed ranking was used to rank methods from the CEC 2022 competition. Raw results of the contestants were downloaded from the official repository [34]. The weight w was set to 0.01. The thresholds were calculated like for ECDF [9], but the sequence starts from 10^3 to be sure that at least one threshold is achieved. Table 4 presents the resulting ranking and the CEC 2022 SOBC official ranking [17]. Additionally, two rankings based on the ECDF curves are created. One uses the area under the curve (AUC), and the other uses the ECDF curve value taken at the budget's end. All ranking methods disagree. According to the proposed method, the best was S-LSHADE-DP [38], which was 4th in the official ranking. The second was IUMOEAII [41], which was 7th, and the third was EA4EigN100_10 [35], which was the official winner. The ranking based on ECDF value at the end of the budget is in better agreement with the proposed one than with the official. According to the averaged AUC, the best was EA4EigN100_10, MTT-SHADE [40] was the second, but it was 6th in the official ranking and 9th in the proposed one. All ranking methods agreed that SPHH-Ensemble [47] is the worst.

To better understand what is behind the rankings, Figure 1 shows ECDFs of the winners of each ranking and the second best according to AUC (for 20 dimensional problems). It can be observed that EA4EigN100_10 and MTT-SHADE, which were the first and the second according to AUC improved their results quickly before 10^5 evaluations of the objective function. On the other hand, S-LSHADE-DP, considered the best by the proposed ranking, improved

Table 4: Possible rankings of contestants of the CEC 2022 SOBC competition.

Algorithm	official	proposed	ECDF at end	AUC
EA4EigN100_10 [35]	**1**	3	4	**1**
NL-SHADE-LBC [36]	2	5	3	3
NL-SHADE-RSP-MID [37]	3	4	5	5
S-LSHADE-DP [38]	4	**1**	**1**	8
jSObinexpEig[39]	5	8	8	7
MTT-SHADE [40]	6	9	10	2
IUMOEAII [41]	7	2	2	9
IMPML-SHADE [42]	8	12	9	10
NLSOMACLP [43]	9	6	7	11
ZOCMAES [44]	10	11	12	4
OMCSOMA [45]	11	7	6	12
Co-PPSO [46]	12	10	11	6
SPHH-Ensemble [47]	13	13	13	13

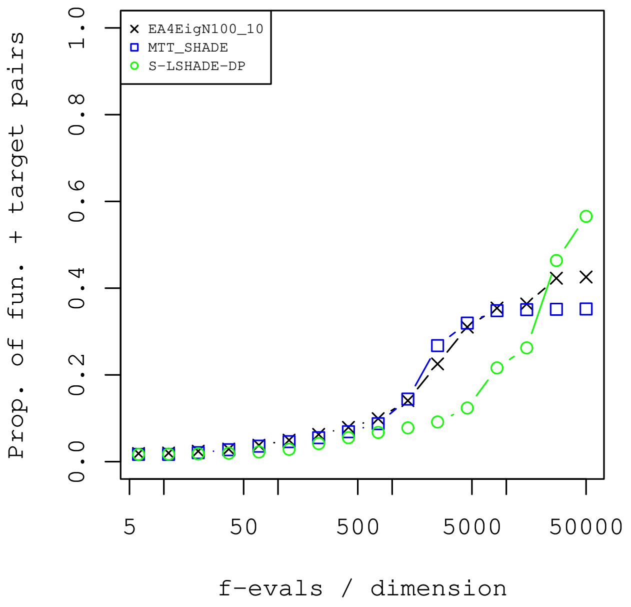

Figure 1: ECDFs of the winners of compared ranking method (EA4EigN100_10, S-LSHADE-DP) and the second best according to AUC (MTT-SHADE).

Table 5: The proposed performance metric for contestants of the CEC 2022 SOBC competition. Each triplet contain the percent of trials that found the global optimum, the percent of thresholds achieved by all trials, and the percent of budget left.

Algorithm	10 D	20 D	All
S-LSHADE-DP	59 69 23	48 57 28	54 63 27
IUMOEAII	54 65 29	26 36 17	40 50 19
EA4EigN100_10	49 61 28	31 43 22	40 52 23
NL-SHADE-RSP-MID	48 60 32	28 40 21	38 50 23
NL-SHADE-LBC	50 61 36	25 37 22	37 49 24
NLSOMACLP	40 56 18	28 42 15	34 49 16
OMCSOMA	41 58 10	24 42 13	33 50 12
jSObinexpEig	38 54 21	25 37 18	31 46 19
MTT-SHADE	34 48 28	25 35 22	29 42 23
Co-PPSO	32 45 24	14 29 07	23 37 10
ZOCMAES	21 41 18	20 32 18	20 37 18
IMPML-SHADE	27 52 15	13 37 06	20 45 07
SPHH-Ensemble	0 26 00	0 11 00	0 18 00

strongly at the end of the budget and achieved much better final results. When the budget is known, it is quite reasonable for the algorithm to focus on exploration for most of the budget and exploitation at the end, which can result in low AUC.

As previously discussed, the proposed performance metric can be shown as triplets containing the percent of trials that found the global optimum, the percent of thresholds achieved by all trials, and the percent of budget left. The results of the contestants of the CEC 2022 SOBC competition are presented in Table 5. Algorithms are ordered according to results on the whole benchmark (column 'All'). According to the proposed ranking scheme, S-LSHADE-DP is the winner. It found global optimum in 14 percentage points more trials than the official winner. It also passed 11 percentage points more thresholds, with four percentage points more budget left. Now, it can be easily observed that SPHH-Ensemble found optimum in 0% trials, which suggests the existence of an error in the implementation. When we compare results for 10 and 20 dimensions, we can observe that the second method, i.e., IUMOEAII, is more than two times better in 10 D than in 20 D. NL-SHADE-RSP-MID [37] and NL-SHADE-LBC [36] have the same core, but NL-SHADE-RSP-MID won because it was better in 20 D. All algorithms except ZOCMAES [44] perform much worse in 20 D.

3.4. Summary and discussion

The proposed performance metric:

- Allows for creating a ranking that agrees with the practitioner's needs.
- Can be calculated for a single algorithm and compared with others previously published to create the ranking. Each algorithm's score calculated

according to CEC 2022 SOBC rules depends on all algorithms in competition.

- Is easily interpretable by humans.

The proposed approach matches the needs of the CEC competition series well. For other benchmarks, when all benchmark functions are too complex to solve even once, the mark will degrade to its second component, but it will still be interpretable and allow the ranking of the algorithms. On the other hand, if all functions are solved, the mark will degrade to its last component, which equals the fixed target approach.

The proposed mark aggregates well. It can be used to compare algorithms on one function, all functions, and all functions and dimensionalities or even on a whole family of benchmarks like CEC competitions of single objective bound constrained search. The mark can be extended by adding components that match the needs of the future benchmarks.

The influence of outliers on the proposed mark is small for CEC benchmarks. For a specific dimensionality in CEC 2022 one outlier changes the mark by 0.28 percentage points. The change is 0.065 percentage points for CEC 2017. The influence can be noticeable for very small benchmarks (few functions, few independent runs), e.g., 3.3 percentage points for a benchmark with 3 functions and 10 runs.

One of the limitations of the proposed approach is that the fitness at the global optimum should be known. For the proposed method to work as intended, the results at the end of the budget should be within the expected range (by default, from 10^3 to 0).

4. Influence of parameter tuning on ranking

As mentioned earlier, rankings can be artificially skewed if only a subset of the contestants are carefully tuned for the competition. To check whether this was the case for CEC 2022 SOBC competition, the top four algorithms (according to the official ranking) were tuned, and changes in the performance metrics of each were analyzed. Since automatic tuning requires source code adaptation and large computational effort, only implementations made in C++, i.e., NL-SHADE-LBC, NL-SHADE-RSP-MID, S-LSHADE-DP, were tuned by *irace* [26]. As EA4EigN100_10 was implemented in Matlab, it was tuned by hand by turning off its internal components.

The source codes of the aforementioned top four algorithms were downloaded from the repository [34]. The codes were analyzed, and internal numerical parameters were found and named. For the automatic tuning, the codes were changed to allow for setting parameters' values during the start of the program. The source codes of the parametrized algorithms are available on [48].

The tuning was performed two times; once, the aim was to improve the algorithm's ranks in the official CEC ranking, and the second time, the aim was to improve the algorithm's place in the proposed ranking. The budget of *irace* was set to 5000 experiments. The tuning was performed collectively, i.e., the whole set of 12 functions run in both 10 and 20 dimensions is treated as an instance in *irace*. Instances differ by used seed, which influences the algorithm's starting points and random numbers. The instance result seen by *irace* comes from 5 independent runs of the algorithm under tuning. A larger number of runs would make the experiments too time-consuming. Fewer runs would make marks used in both ranking methods too sensitive to random fluctuations. Since competing algorithms are needed to calculate the CEC 2022 rating, the NL-SHADE-RSP [49], RB_IPOP [50], and SADE [51] were used.

After tuning, the resulting configuration was tested using 30 independent runs as required by the CEC 2022 competition. During the test, all algorithms started from the same starting points when possible, i.e., when some algorithm uses a larger population than the other, all additional points were not seen by the other.

4.1. NL-SHADE-RSP

In the following sections, the parameters of several algorithms will be named and tuned. Most parameters were not named in the original articles. They were just numeric constants in the source codes. This section aims not to describe algorithms in detail but to facilitate understanding of parameter names and their roles. For this purpose, the NL-SHADE-RSP [49] algorithm will be presented and briefly described. The NL-SHADE-RSP is the predecessor to NL-SHADE-RSP-MID and NL-SHADE-LBC, and it is similar to S-LSHADE-DP. The pseudocode of NL-SHADE-RSP is presented in Figure 2. To calculate population size *pop_size_mult* parameter is multiplied by problem dimensionality N. The memory size for F and CR is a product of *mem_size_mult* and problem dimensionality. The archive size of the individuals is the product of *arch_size_mult* and population size. Parameters *initial_CR* and *initial_F* are used at the beginning of the search to initialize CR and F. The parameter *arch_prob* sets the probability of archive usage during mutation (line 16). Parameters *per_of_the_best_considered_at_start* and *per_of_the_best_considered_at_end* are used to calculate *p_best_rate*, which controls greediness of current-to-pbest strategy. The parameter *norm_sigma* sets standard deviation (σ) for the normal distribution random number generator used to generate CR (line 6). The parameter *cauchy_sigma* sets the scale parameter of the Cauchy distribution random number generator, which is used to generate F (line 8). The parameter *prob_use_exp* sets the probability of the usage of exponential crossover (line 26). Variable A is an archive that is filled in line 32 and used during mutation (line 23) when the condition in line 16 is not satisfied. Variable k is a counter used to select a place for the update in memory of F and CR (line 36).

The algorithm works iteratively till the budget of the objective function evaluations is exhausted (line 2). At the beginning of each iteration, the memory of F and CR is cleaned. Lines 4-11 are responsible for generating and normalizing CR an F. Lines 13-21 are responsible for selecting indexes of individuals that will be used to perform mutation in line 23. Line 25 updates *CRtoUse*. It is set to 0 during *when_start_2_use_CRtoUse* fraction of the budget. After that,

```
1: $pop\_size\_mult = 30$; $pop\_size = pop\_size\_mult \cdot N$; $mem\_size\_mult = 20$;
   $mem\_size = mem\_size\_mult \cdot N$; $A = \emptyset$; $k = 1$; $initial\_F = 0.2$; $MF_r =$
   $initial\_F$; $initial\_CR = 0.2$; $MCR_r = initial\_CR$; $arch\_size\_mult = 2.1$;
   $arch\_size = arch\_size\_mult \cdot pop\_size$; $arch\_prob = 0.5$; $norm\_sigma =$
   $0.1$; $cauchy\_sigma = 0.1$; $per\_of\_the\_best\_considered\_at\_start = 20$;
   $p\_best\_rate = per\_of\_the\_best\_considered\_at\_start/100$; $CRtoUse = 0$;
   $prob\_use\_exp = 0.5$
2: while $fes < fes_{max}$ do
3:   $SF = \emptyset$; $SCR = \emptyset$;
4:   for all $i \in \{1, 2, ..., pop\_size\}$ do
5:     $r = \text{randInt}(1, mem\_size + 1)$
6:     $CR_i = \text{randNorm}(MCR_r, norm\_sigma)$; $CR_i = \min(1, \max(0, CR_i))$
7:     repeat
8:       $F_i = \text{randCauchy}(MF_r, cauchy\_sigma)$
9:     until $F_i \geq 0$
10:    $F_i = \min(1, F_i)$
11:  end for
12:  for all $i \in \{1, 2, ..., pop\_size\}$ do
13:    repeat
14:      $pbest = \text{randInt}(1, pop\_size \cdot p\_best\_rate)$
15:      $r_1 = \text{randInt}(1, pop\_size)$
16:      if $rand(0, 1) > arch\_prob$ then
17:        $r_2 = \text{randInt}(1, pop\_size)$
18:      else
19:        $r_2 = \text{randInt}(1, arch\_size)$
20:      end if
21:    until $i \neq pbest \neq r_1 \neq r_2$
22:    for all $j \in \{1, 2, ..., N\}$ do
23:      $v_{i,j} = x_{i,j} + F_i \cdot (x_{pbest,j} - x_{i,j}) + F_i \cdot (x_{r_1,j} - x_{r_2,j})$
24:    end for
25:    Update $CRtoUse$
26:    if $\text{rand}(0, 1) > prob\_use\_exp$ then
27:      $u_i$ =binomial crossover with $CRtoUse$
28:    else
29:      $u_i$ =exponential crossover with $CR_i$
30:    end if
31:    if q($u_i$)<q($x_i$) then                    // q is the objective function
32:      $a = \text{randInt}(1, arch\_size)$; $A_a = x_i$; $x_i = u_i$
33:      $SF = SF \cup F_i$; $SCR = SCR \cup CR_i$
34:    end if
35:    Update $pop\_size$, $arch\_size$, adjust archive content by removing ran-
       dom individuals, adjust population by removing worst individuals
36:    Update $MF_k, MCR_k, arch\_prob$; $k = \text{modulo}(k, mem\_size) + 1$
37:  end for
38: end while
```

Figure 2: The pseudocode of NL-SHADE-RSP.

CRtoUse is proportional to the budget spent. This parameter is only used when the algorithm executes binomial crossover (line 27). If the result of the crossover (u_i) is better than the current individual x_i, then x_i is stored in the memory A, and its place in the population is given to u_i. Values of F and CR that lead to current success are stored in SF and SCR collections (line 33). In 36, expected values of F and CR are updated based on the Lehmer mean on SF or SCR, accordingly. The *arch_prob* update is based on objective function improvements achieved using the archive and the number of archive usages.

4.2. Modifying EA4EigN100_10

EA4EigN100_10 was proposed in [35]. It is the official winner of the CEC 2022 competition. The algorithm uses four well-known optimization methods, namely: CoBiDE [52], IDEbd [53], CMA-ES [54], and jSO [55]. At each iteration the algorithm stochastically selects which internal algorithm will be used. Additionally, all algorithms except CMA-ES use Eigen crossover with probability 0.4. Unfortunately, the paper does not include an analysis of the influence of each component on the quality of the results. This section will fill this gap. During the experiments, each of the internally used algorithms was either turned off or used alone, e.g., the version named here IDE uses only IDEbd as a search engine, but the version called NO_IDE uses all components except IDEbd. As Eigen crossover is used stochastically, two additional versions of the algorithm were tested. In the first of them (called EIG) Eigen crossover is always used. In the second version, Eigen crossover is disabled (NO_EIG). As NO_COBIDE and NO_CMA versions turned out good, the version with both components disabled was also examined.

During the experiments, it was noticed that one of the decision paths in EA4EigN100_10 allows it to ask for fitness values outside the bounds, and the official implementation of the benchmark answers such queries. This decision path is active when the algorithm uses jSO and Eigen crossover. The error was corrected, and the resulting version is named CORRECTED_BOUNDS. The original implementation was used in all modifications of the algorithm mentioned above. The source code of the discussed versions of the algorithm is available on [48].

The results of the experiments are shown in Table 6. As can be observed, the official implementation, which sometimes uses information outside the search boundary, is better than the corrected version according to both ranking schemes. Considering the CEC 2022 SOBC ranking, the official implementation is in 9th place, and its error-free version is 10th. The best is the version that does not use CoBiDE and CMA-ES. Even the version that uses only IDEbd is better than the whole algorithm. Eigen crossover is beneficial. Using it always improved results; disabling it deteriorated results.

Considering the proposed ranking scheme, the best version does not use CoBiDE, and the second best does not use IDEbd. The best version improved three percentage points on the top of the default.

Table 6: Analysis of the influence of EA4EigN100_10 components on its position in ranking.

Alg. version	CEC rank	CEC points	proposed rank	prop. perf. metric
NO_COBIDE_NO_CMA	1	157606	5	40 51 29
NO_COBIDE	2	150943	1	43 53 29
NO_CMA	3	137550	4	40 51 28
IDE	4	123947	6	38 48 26
EIG	5	121193	9	37 52 26
NO_IDE	6	121064	2	42 53 27
NO_jSO	7	117080	7	37 49 25
jSO	8	115005	10	37 50 28
official	9	112593	3	40 52 23
CORRECTED_BOUNDS	10	110666	11	37 48 24
NO_EIG	11	97744	8	37 48 24
COBIDE	12	60206	12	36 48 21

4.3. Tuning NL-SHADE-LBC

NL-SHADE-LBC took second place during the CEC 2022 competition. It is the modification of NL-SHADE-RSP [49]. The most important novelties of NL-SHADE-LBC include: 1) the usage of generalized Lehmer mean [56]; 2) linear change of power used in generalized Lehmer mean; 3) usage of the resampling [57] as bound constraints handling. The list of parameters to tune was created based on the source code analysis. Parameters *default_CR* and *default_F* are used during memory updates when there are no successful individuals in the current iteration. The *MWLp1* and *MWLp2* set the initial values of power (p) used in generalized Lehmer mean to calculate F and CR, respectively. The power is linearly reduced during the search to achieve *LBC_fin* at the end. The *MWLm* sets bias parameter (m) of generalized Lehmer mean. The *min_pop_size* set minimal population size at the end of the budget. The assumed ranges and types of the parameters are provided in Table A.1. The source code of the parametrized algorithm is available on [48].

The tuning results are provided in Table 7. It can be observed that the results of tuning for CEC score and for the proposed measure are different. An ablation analysis was performed to determine which parameters are most important. The maximal number of experiments for ablation was set to 5000. The results for configuration tuned for CEC score are provided in Figure 3, and for the proposed measure in Figure 4. Both performance measures should be maximized, but *irace* performs minimalization. Therefore, the sign of performance measures was changed. As it can be observed, *initial_F* is important for tuning for both ranking methods. It was increased for CEC but strongly decreased for the proposed measure. Even though modern algorithms adapt their parameters, their initial values are still important. The probability of archive usage was also considered important for both rankings. Again, changes for CEC and for the proposed method have opposite directions. The parameters introduced in

Table 7: Default and tuned parameters of NL-SHADE-LBC.

Parameter	default	tuned for CEC	tuned for proposed score
pop_size_mult	23	12	30
mem_size_mult	20	11	6
arch_size_mult	1	1.91	0.67
initial_CR	0.9	0.93	0.57
initial_F	0.5	0.65	0.12
min_pop_size	4	11	9
arch_prob	0.5	0.46	0.57
per_of_the_best_considered_at_start	20	37	33
per_of_the_best_considered_at_end	30	25	27
norm_sigma	0.1	0.22	0.18
cauchy_sigma	0.1	0.21	0.1
default_CR	0.5	0.82	0.43
default_F	0.5	0.8	0.53
MWLp1	3.5	2.49	2.29
MWLp2	1	2	1.06
MWLm	1.5	2.55	0.53
LBC_fin	1.5	1.88	2.49

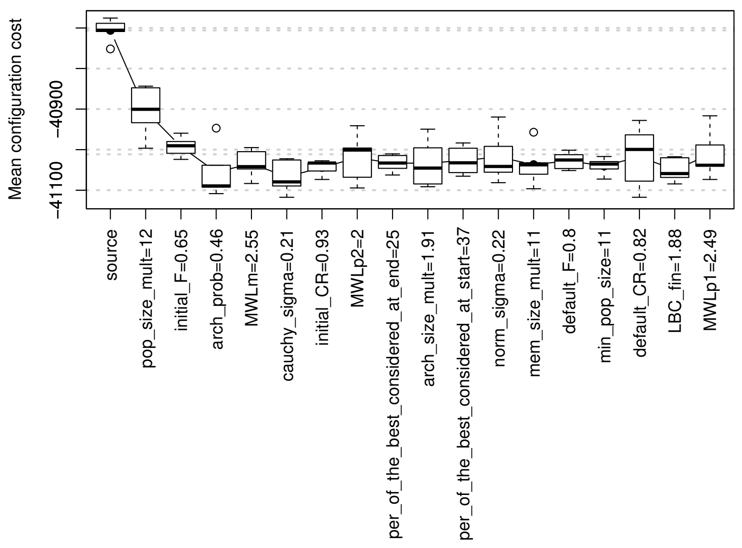

Figure 3: Ablation analysis of NL-SHADE-LBC parameters tuned for CEC 2022 ranking.

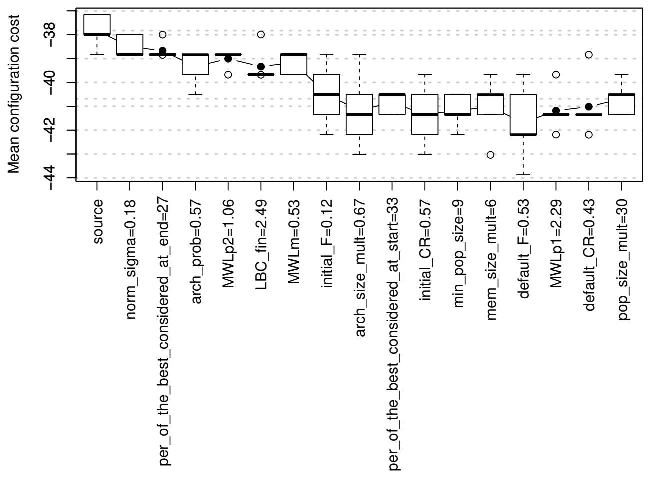

Figure 4: Ablation analysis of NL-SHADE-LBC parameters tuned for the proposed ranking.

NL-SHADE-LBC (*MWLp1*, *MWLp2*, *LBC_fin, MWLm*) are not important for CEC 2022, but all except *MWLp1* are important for the proposed measure.

The decrease of the population size (by *pop_size_mult*) was the most important change for CEC. For the proposed measure, the most important change was the increase of σ (*norm_sigma*), which results in higher *CR* variability.

4.4. Tuning NL-SHADE-RSP-MID

The NL-SHADE-RSP-MID [37] took third place during the CEC 2022 SOBC competition. The most important novelties introduced by NL-SHADE-RSP-MID include: 1) calculating and using population midpoint; 2) splitting population into two groups by k-means algorithm to find better midpoints; 3) introducing restarts to the algorithm; 4) using resampling [57] as bound constrain handling. As NL-SHADE-RSP-MID was built on top of NL-SHADE-RSP, during the first tuning series, the influence of parameters and components added to NL-SHADE-RSP was tested. The second series of tuning also includes parameters of NL-SHADE-RSP.

4.4.1. Tuning parameters introduced by NL-SHADE-RSP-MID

As mentioned earlier, the first series of experiments focused on tuning parameters added on top of NL-SHADE-RSP parameters. All parameters of the core algorithm were left untouched except the initial population size. Categorical parameters, like *k-means*, allow turning components introduced in NL-SHADE-RSP-MID on or off. To calculate population size *pop_size_mult* parameter is multiplied by problem dimensionality. The population size after restart of the algorithm is set by *pop_size_reset*. The minimal population size after restart is set by *min_pop_size_aft_restart*. The restart is triggered when at least one individual has at least one dimension on bounds for more than *min_its_on_bounds*

Table 8: Default and tuned parameters of NL-SHADE-RSP-MID.

Parameter	default	tuned for CEC	tuned for proposed score
k-means	1	1	1
resampling	1	0	1
count_lim	1	0	1
min_its_on_bounds	9	-	9
pop_size_mult	5	4	13
min_num_stag_it	8	12	10
min_pop_size_aft_restart	20	19	44
pop_size_reset	400	300	210
shape_const	0.1	0.08	0.11
min_silhouette_mult	0.025	0.01	0.02

iterations or when the distance between the current mean of the population and mean *min_num_stag_it*+1 older is less than 10^{-9}. The rate of decline in population size is affected by *shape_const*. The split of the population into two groups is rejected when the silhouette score calculated for that grouping is smaller than the product of *min_silhouette_mult* and $\sqrt{N}$, where N is problem dimensionality. The types and assumed ranges of the parameters are provided in Table A.2. The source code of the parametrized algorithm is available on [48].

Table 8 provides the default and tuned parameter values. It can be observed that the algorithm tuned for the proposed score uses all new components introduced in NL-SHADE-RSP-MID, but the algorithm tuned for CEC 2022 uses only the k-means component. Form changes of the *pop_size_mult* and *min_pop_size_aft_restart*, it can be inferred that larger populations are preferred for the proposed score and smaller than the default are preferred for the CEC score.

The results of the ablation analysis for parameters tuned for CEC 2022 are provided in Figure 5, and the results for parameters tuned for the proposed performance metric are in Figure 6. When tuning is performed for CEC ranking, the most important is to turn off resampling, followed by decreasing population size (*pop_size_mult* and *min_pop_size_aft_restart* parameters). On the other hand, when using the proposed measure, the most important is to increase population size.

When analyzing Figure 6 it can be noticed that using a tuned values of the two least important parameters deteriorated results, suggesting that tuned parameters are still far from optimal. This hypothesis was verified by an additional experiment in which only the top 5 tuned parameters were used, and the rest were default. For that setup 30 independent runs were performed. The results are similar to the results when all tuned parameters were used. The Wilcoxon test does not rejected null hypothesis at a confidence level of 0.95. Due to the large computational cost, the ablation analyses use data from only five independent runs. Therefore, some results may be distorted by randomness.

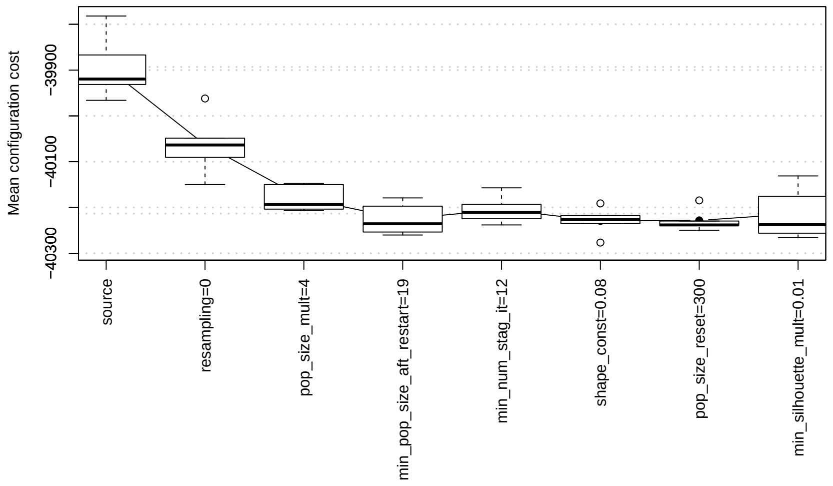

Figure 5: Ablation analysis of NL-SHADE-RSP-MID parameters tuned for CEC 2022 ranking.

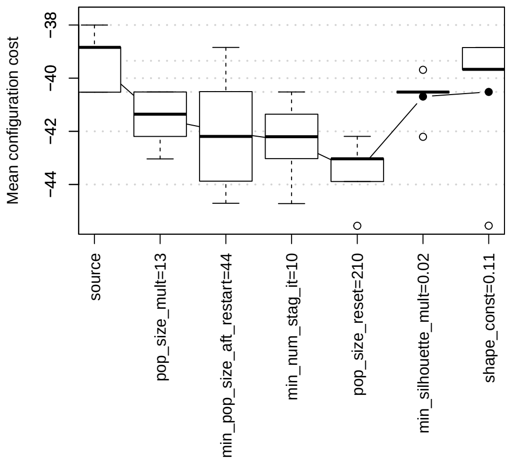

Figure 6: Ablation analysis of NL-SHADE-RSP-MID parameters tuned for the proposed ranking.

Table 9: Default and tuned full set of parameters of NL-SHADE-RSP-MID.

Parameter	default	tuned for CEC	tuned for proposed score
k-means	1	1	1
resampling	1	1	1
count_lim	1	1	1
min_its_on_bounds	9	10	13
pop_size_mult	5	3	7
min_num_stag_it	8	5	10
min_pop_size_aft_restart	20	24	21
pop_size_reset	400	62	272
shape_const	0.1	0.07	0.08
min_silhouette_mult	0.025	0.02	0.02
min_pop_size	4	5	43
mem_size_mult	20	19	26
arch_size_mult	2.1	1.69	2.83
initial_CR	0.2	0.9	0.12
initial_F	0.2	0.54	0.2
arch_prob	0.5	0.28	0.59
prob_use_exp	0.5	0.87	0.09
when_start_2_use_CRtoUse	0.5	0.24	0.44
per_of_the_best_considered_at_start	20	24	33
per_of_the_best_considered_at_end	60	32	22
norm_sigma	0.1	0.24	0.22
cauchy_sigma	0.1	0.25	0.26
default_CR	0.5	0.98	0.48
default_F	0.5	0.51	0.32

4.5. Tuning parameters of NL-SHADE-RSP

The base algorithm for NL-SHADE-RSP-MID, i.e., NL-SHADE-RSP, also has many parameters. In addition to parameters shown in its introductory paper [49] it has constants that were found in the source code. They were named and also tuned. The list of the NL-SHADE-RSP-MID parameters not listed in Table 8, together with their assumed ranges of the values, are provided in Table A.3. The meaning of all parameters was discussed earlier.

The results of the tuning are provided in Table 9. Unlike when tuning only new parameters of NL-SHADE-RSP-MID for CEC 2022, this time *resampling* and *count_lim* components are enabled. Again, tuning for CEC reduced population size, but tuning for the proposed measure increased it. Tuning for the proposed measure also increased the minimal population size by more than ten times. The results of ablation analyses for the CEC measure are provided in Figure 7, and for the proposed measure in Figure 8. For both considered ranking methods, the set of the most important parameters includes *initial_CR*, *per_of_the_best_considered_at_start*, and *cauchy_sigma*. Ini-

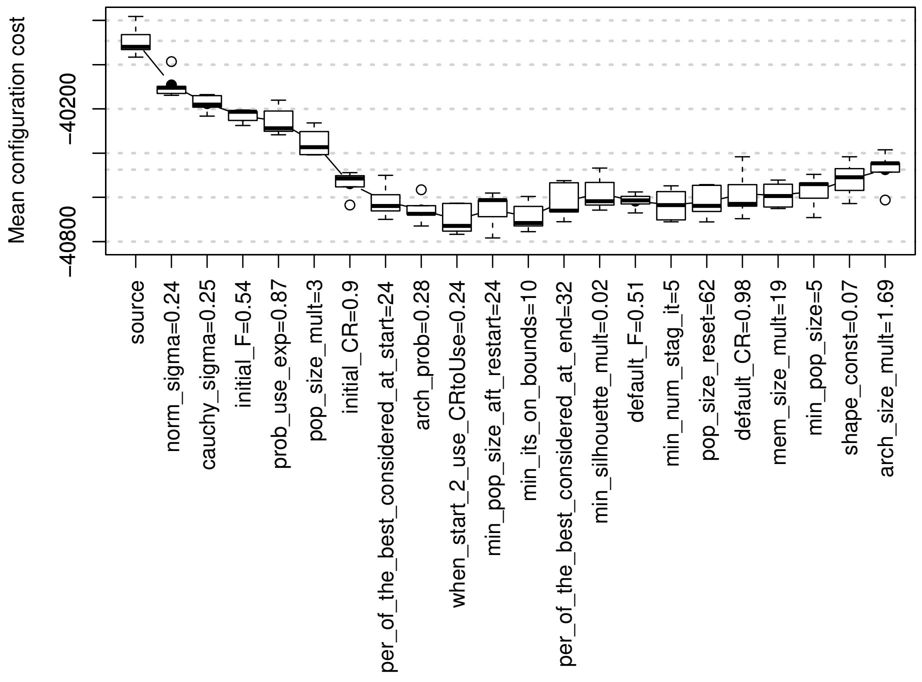

Figure 7: Ablation analysis of the full set of NL-SHADE-RSP-MID parameters tuned for the CEC 2022.

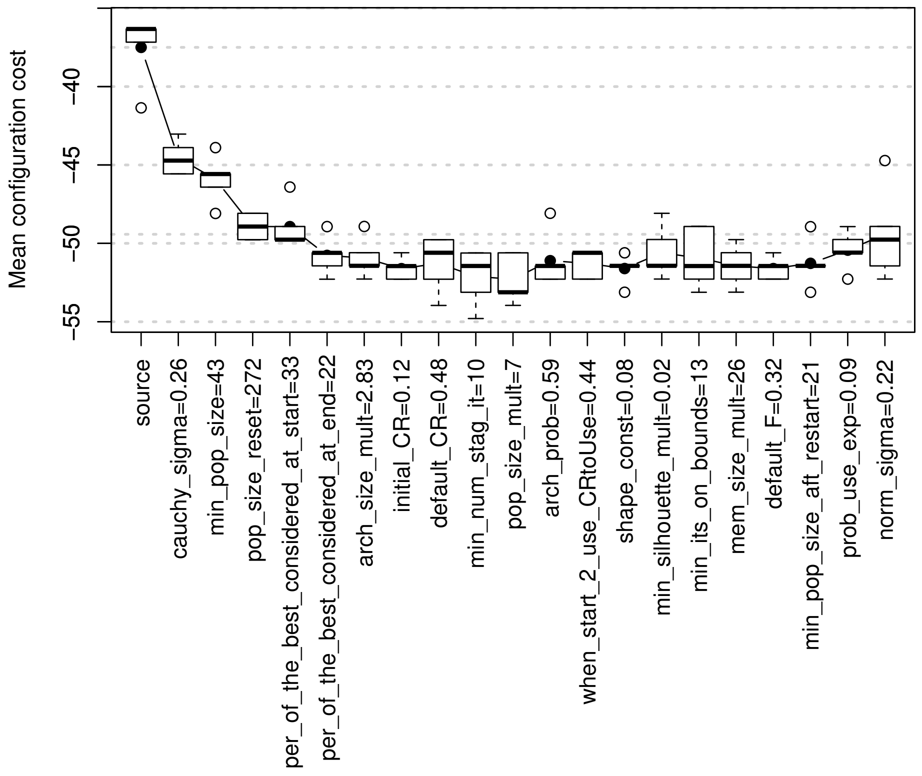

Figure 8: Ablation analysis of the full set of NL-SHADE-RSP-MID parameters tuned for the proposed ranking.

tial CR was strongly increased for CEC but decreased for the proposed measure. The *per_of_the_best_considered_at_start* was increased for both ranking methods, but for the proposed measure, it was also important to decrease *per_of_the_best_considered_at_end*. In both cases, *cauchy_sigma* was increased more than two times. It seems that a stronger variability of F is desired. From the set of other parameters important for CEC, *norm_sigma* was also increased more than two times, resulting in more substantial CR variability. The *initial_F* was also increased more than two times. The probability of using exponential crossover (*prob_use_exp*) was increased to nearly 0.9. The probability of archive usage was reduced nearly two times. From the set of other parameters important for the proposed measure, the population size after the restart was strongly reduced, and the archive size was increased.

4.6. Tuning S-LSHADE-DP

S-LSHADE-DP [38] took fourth place during the CEC 2022 competition, but according to the proposed ranking, it is the winner. The most important novelties introduced by the algorithm include: 1) stagnation detection and additional perturbation of stagnated individuals; 2) using two mutation strategies (DE/target/1, DE/current-to-pbest/1) switched by additional heuristics; 3) modified adaptation mechanism for CR.

The list of parameters to tune was created based on the source code analysis. Parameter *gamma* sets the probability of current-to-pbest mutation instead DE/target/1. The number of the best solutions used in p-best mutation is the product of *p_best_rate* and population size. The selection of the mutation operator is made every *upd_mut_op_iters*. The *stagnation_threshold* sets the number of iterations after which a stagnated individual undergoes dynamic perturbation. The *CR_switch_at* sets at what point of the budget algorithm will switch CR from 0 to a uniform random number U(0,1). The *mut_switch_at* is the probability of performing dynamic perturbation when stagnation is detected. The *stag_detect_at* sets at what point of the budget the stagnation detection mechanism will be enabled. The types and assumed ranges of the parameters are provided in Table A.4. The source code of the parametrized algorithm is available on [48].

The results of parameter tuning are provided in Table 10. The results of ablation analysis for CEC is provided in Figure 9 and for the proposed measure in Figure 10. It can be observed that for both ranking methods, population size is the most important parameter. Like for other previously analyzed algorithms, tuning for CEC requires a smaller population size, and tuning for the proposed measure requires larger populations. When tuning for CEC it is important to switch CR from 0 to a larger number much earlier. Another interesting aspect is a more than two-fold reduction of the probability of current-to-pbest mutation.

4.7. The influence of the tuning on the rankings

This section examines the influence of the tuning on both rankings. Every tuned algorithm was run 30 times to calculate its score for each ranking method.

Table 10: Default and tuned parameters of S-LSHADE-DP.

Parameter	default	tuned for CEC	tuned for proposed score
pop_size	100	47	114
mem_size	6	12	7
arch_size_mult	2.6	1.88	1.86
p_best_rate	0.11	0.1	0.12
gamma	0.3	0.12	0.37
upd_mut_op_iters	20	32	37
min_pop_size	4	13	17
cauchy_sigma	0.1	0.21	0.27
initial_F	0.5	0.45	0.56
stagnation_threshold	100	134	120
CR_switch_at	0.5	0.13	0.27
mut_switch_at	0.5	0.78	0.29
stag_detect_at	0.5	0.44	0.82

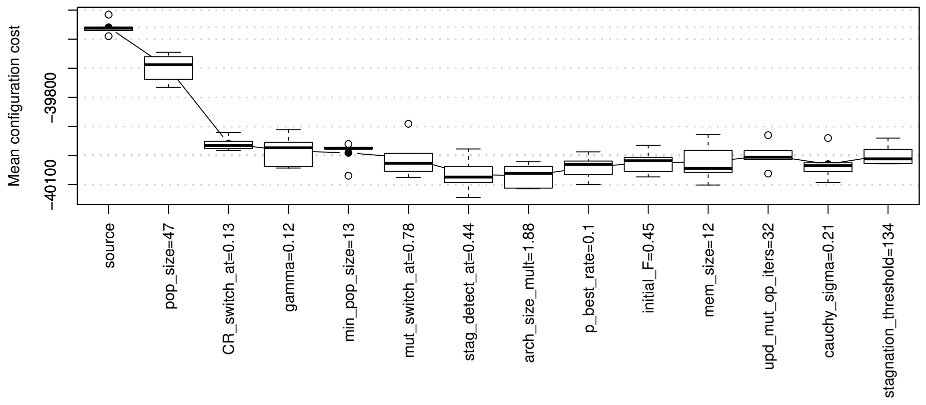

Figure 9: Ablation analysis of the S-LSHADE-DP parameters tuned for CEC 2022 SOBC.

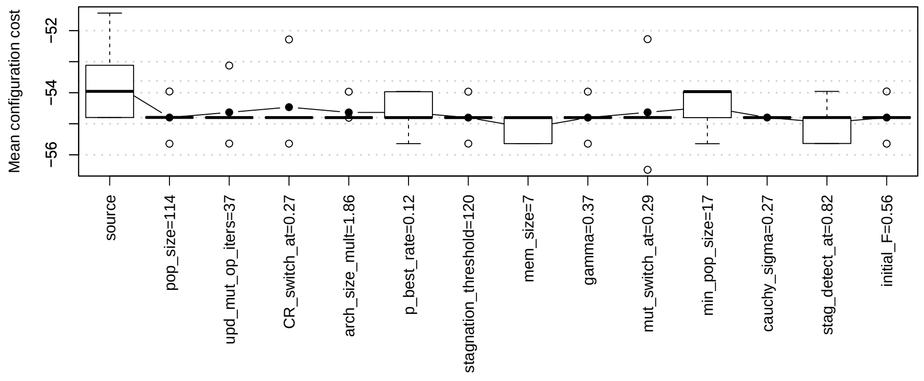

Figure 10: Ablation analysis of the S-LSHADE-DP parameters tuned for the proposed ranking.

The results of the experiments are shown in Table 11. The table includes the results of all tuned algorithms and IUMOEAII, as it was the second in the proposed ranking. As other algorithms are worse than even not-tuned considered methods, they are omitted in this table. It can be observed that generally, tuning using target measure helps, but tuning using the other measure is harmful, e.g., NL-SHADE-RSP-MID tuned for the proposed ranking is the last one in CEC 2022 SOBC ranking, NL-SHADE-LBC tuned for CEC is the last one for the proposed ranking. The first for the proposed ranking is the eleventh for CEC.

Analysis of the CEC ranking points shows that all improvements by tuning are large. Tuning all parameters of NL-SHADE-RSP-MID gave better results than tuning only new ones. If only NL-SHADE-LBC was tuned, and the other contestants were not, it would win the competition. The same holds for S-LSHADE-DP. If only NL-SHADE-RSP-MID was tuned, it would be the second. If all considered algorithms were tuned, EA4Eig and NL-SHADE-LBC would maintain their order, but NL-SHADE-RSP-MID and S-LSHADE-DP would switch their positions.

When considering the proposed ranking, tuned versions of S-LSHADE-DP were not better than the base version. If only one algorithm were tuned, NL-SHADE-RSP-MID would advance two positions to the second place. EA4Eig and NL-SHADE-LBC could also be the second. If all considered algorithms were tuned, NL-SHADE-RSP-MID would be the second, EA4EigN100_10 would be the third, and NL-SHADE-LBC would be the fourth. Untuned IUMOEAII would drop from the second to the fifth place. The profit of tuning EA4Eig is three percentage points. The same holds for NL-SHADE-LBC. For the NL-SHADE-RSP-MID, it is 13 percentage points.

When using the proposed measure, it is possible to calculate the mark on the whole benchmark on both dimensionalities for each run separately. Thanks to that, it is possible to compare two algorithms in a rather standard way using statistical tests. As both compared methods start from common starting

Table 11: Results and ranking of the tuned algorithms. For better readability suffix N100_10 was removed from EA4EigN100_10 and prefix NL-SHADE was removed from NL-SHADE-RSP-MID and NL-SHADE-LBC.

Alg. version	CEC rank	CEC points	proposed rank	prop. quality measure
EA4Eig for CEC	1	204591	10	40 51 29
EA4Eig for prop.	2	195415	5	43 53 29
LBC for CEC	3	183983	15	37 49 25
S-LSHADE-DP for CEC	4	180997	3	52 61 38
EA4Eig	5	172320	9	40 52 23
RSP-MID all for CEC	6	169964	8	41 51 30
RSP-MID new for CEC	7	164483	14	37 50 23
LBC	8	162954	13	38 49 25
LBC for proposed	9	132412	7	41 52 23
RSP-MID	10	128856	12	39 51 23
S-LSHADE-DP	11	124787	1	54 63 27
S-LSHADE-DP for prop.	12	123656	2	54 62 30
IUMOEAII	13	114675	11	40 50 19
RSP-MID all for prop.	14	108404	4	52 61 22
RSP-MID new for prop.	15	100499	6	42 52 17

points, the Wilcoxon signed-rank test was used. Tuning NL-SHADE-RSP-MID significantly improved its results (p-value: 9e-10). The same holds for NL-SHADE-LBC (p-value 6e-06).

5. Conclusions

This paper proposes a new method of assessing the performance of metaheuristic optimization algorithms. The method was used to create an alternative ranking for CEC 2022 SOBC competition contestants. The new ranking was compared to the official one. The influence of parameter tuning on both rankings was also investigated.

The proposed performance metric is human-interpretable and allows for finding a place for new algorithms in published rankings. As resulting rankings are more focused on the results at the end of the budget, winning algorithms are better suited for real-world applications than winners of rankings more focused on convergence rate.

The results showed that automatically tuned algorithms significantly improved their ranks when the tuning used the same performance metric as the target ranking. On the other hand, when tuning used the other metric, the rank deteriorated. The observed improvements indicate that none of the examined algorithms was carefully tuned for CEC 2022 SOBC. The tuning results showed that larger populations are usually preferred for the proposed score, and smaller than the default are preferred for the CEC score.

The ablation analyses of the algorithms' parameters showed that only a few of them strongly affect the results. Some of them are not documented in the introductory papers and can only be found in the source codes. Even though modern algorithms adapt their parameters, their initial values are still important.

Since tuning impacts ranking significantly, future competitions should specify the tuning procedure. The procedure should determine the tuning budget and the tuning method. The latter is essential to reduce the variability of the results that come from personal preferences on tuning methods. As some authors wish to make their algorithms as universal as possible, they propose initial values of the parameters that should work for many optimization problems. Therefore, it would be interesting to have and compare two rankings, one for not-tuned algorithms and the second for the tuned ones. The comparison of the results of these rankings could give additional knowledge about the algorithms, e.g., a significant change in the ranking position suggests that built-in adaptation of a few parameters could be improved or extended. The high tuning potential may also indicate that the algorithm is overcomplicated, i.e., it has too many modules and parameters.

Future work aims to find out how to optimally select thresholds for a given benchmark and how to handle functions with unknown global optimum.

Appendix A. Types and ranges of the parameters of tuned algorithms

This section provides types and ranges of the parameters used for tuning by *irace* method. Table A.1 contains parameters of NL-SHADE-LBC, Table A.2 parameters of NL-SHADE-RSP-MID, Table A.3 contains parameters of NL-SHADE-RSP-MID inherited from NL-SHADE-RSP, and Table A.4 contains parameters of S-LSHADE-DP.

References

[1] H. Bersini, M. Dorigo, S. Langerman, G. Seront, L. Gambardella, Results of the first international contest on evolutionary optimisation, in: T. Bäck, T. Fukuda, Z. Michalewicz (Eds.), Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96), IEEE Press, Piscataway, NJ, 1996, pp. 611–615.

[2] A. P. Piotrowski, J. J. Napiorkowski, A. E. Piotrowska, How much do swarm intelligence and evolutionary algorithms improve over a classical heuristic from 1960?, IEEE Access 11 (2023) 19775–19793. `doi:10.1109/ACCESS.2023.3247954`.

[3] A. Kumar, K. V. Price, A. W. Mohamed, A. A. Hadi, P. Suganthan, Problem definitions and evaluation criteria for the CEC 2022 special session and competition on single objective bound constrained numerical optimization, Tech. rep., Nanyang Technol. Univ., Singapore (2022).

Table A.1: Types and ranges of the parameters of NL-SHADE-LBC. 'i' stands for integer, 'r' for real number.

Parameter	type	range
pop_size_mult	i	1 – 30
mem_size_mult	i	5 – 25
arch_size_mult	r	0.5 – 3
initial_CR	r	0.1 – 1
initial_F	r	0.1 – 0.9
min_pop_size	i	4 – 100
arch_prob	r	0.3 – 0.8
per_of_the_best_considered_at_start	r	10 – 50
per_of_the_best_considered_at_end	r	20 – 90
norm_sigma	r	0.05 – 0.3
cauchy_sigma	r	0.05 – 0.3
default_CR	r	0.1 – 0.9
default_F	r	0.1 – 0.9
MWLp1	r	2.0 – 5.0
MWLp2	r	0.5 – 2.0
MWLm	r	0.5 – 3.0
LBC_fin	r	0.5 – 3.0

Table A.2: Types and ranges of basic parameters of NL-SHADE-RSP-MID. 'c' stands for categorical, 'i' for integer, 'r' for real number, 'log' enables logarithmic scaling. If some parameter is depended on the other this dependency is shown after '|'.

Parameter	type	range
k-means	c	0, 1
resampling	c	0, 1
count_lim	c|resampling	0, 1
min_its_on_bounds	i|count_lim	5 – 15
pop_size_mult	i	1 – 20
min_num_stag_it	i	0 – 15
min_pop_size_aft_restart	i	10 – 50
pop_size_reset	i,log	50 – 400
shape_const	r	0.05 – 0.2
min_silhouette_mult	r|k_means	0.01 – 0.034

Table A.3: Types and ranges of the parameters of NL-SHADE-RSP-MID inherited from NL-SHADE-RSP.

Parameter	type	range
min_pop_size	i	4 – 100
mem_size_mult	i	10 – 30
arch_size_mult	r	1.5 – 3
initial_CR	r	0.1 – 1
initial_F	r	0.1 – 0.9
arch_prob	r	0.1 – 0.8
prob_use_exp	r	0 – 1
when_start_2_use_CRtoUse	r	0.1 – 0.9
per_of_the_best_considered_at_start	r	10 – 40
per_of_the_best_considered_at_end	r	20 – 80
norm_sigma	r	0.05 – 0.3
cauchy_sigma	r	0.05 – 0.3
default_CR	r	0.1 – 1
default_F	r	0.1 – 0.9

Table A.4: Types and ranges of parameters of S-LSHADE-DP. 'i' stands for integer, 'r' for real number.

Parameter	type	range
pop_size	i	20 – 400
mem_size	i	4 – 12
arch_size_mult	r	1.5 – 3
p_best_rate	r	0.05 – 0.2
gamma	r	0.1 – 0.7
upd_mut_op_iters	i	10 – 40
min_pop_size	i	4 – 100
cauchy_sigma	r	0.05 – 0.3
initial_F	r	0.1 – 0.9
stagnation_threshold	i	50 – 200
CR_switch_at	r	0.1 – 0.9
mut_switch_at	r	0.1 – 0.9
stag_detect_at	r	0.1 – 0.9

[4] K. Qiao, X. Wen, X. Ban, P. Chen, K. V. Price, P. N. Suganthan, J. Liang, G. Wu, C. Yue, Evaluation criteria for CEC 2024 competition and special session on numerical optimization considering accuracy and speed, Tech. rep., Zhengzhou University, Zhengzhou, China (2024).

[5] N. Hansen, S. Finck, R. Ros, A. Auger, Real-parameter black-box optimization bench-marking 2009: Noiseless functions definitions., Tech. rep., Inria (RR-6829) (2009).

[6] N. Hansen, T. Tusar, O. Mersmann, A. Auger, D. Brockhoff, COCO: The experimental procedure (2016). `arXiv:1603.08776`.

[7] T. Bartz-Beielstein, M. Preuss, The future of experimental research, in: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, GECCO '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 3185–3226. `doi:10.1145/1570256.1570417`.

[8] J. Brownlee, A note on research methodology and benchmarking optimization algorithms, Tech. rep., Complex Intelligent Systems Laboratory (CIS), Centre for Information Technology Research (CITR), Faculty of Information and Communication Technologies (ICT), Swinburne University of Technology, Victoria, Australia (2007).

[9] N. Hansen, A. Auger, D. Brockhoff, D. Tusar, T. Tusar, COCO: Performance assessment, arXiv:1605.03560 (2016).

[10] N. Hansen, et al., Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009, in: GECCO, ACM, 2010, pp. 1689–1696. `doi:10.1145/1830761.1830790`.

[11] N. H. Awad, M. Ali, J. Liang, B. Qu, P. N. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 special session and competition on real-parameter optimization, Tech. rep., Nanyang Technol. Univ., Singapore and Jordan Univ. Sci. Technol. and Zhengzhou Univ., China (2016).

[12] C. T. Yue, K. V. Price, P. N. Suganthan, J. J. Liang, M. Z. Ali, B. Y. Qu, N. H. Awad, P. Biswas, Problem definitions and evaluation criteria for the CEC 2020 special session and competition on single objective bound constrained numerical optimization, Tech. rep., Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Nanyang Technological University, Singapore (2020).

[13] A. Wagdy, A. A. Hadi, A. K. Mohamed, P. Agrawal, A. Kumar, P. N. Suganthan, Problem definitions and evaluation criteria for the CEC 2021 special session and competition on single objective bound constrained numerical optimization, Tech. rep., Nanyang Technological University, Singapore (2020).

[14] P. N. Suganthan, et al., Problem definitions and evaluation criteria for the CEC 2005 Special Session on Real Parameter Optimization, Tech. rep., Nanyang Technological University, Singapore, and KanGAL, IIT Kanpur, India (2005).

[15] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin 1 (6) (1945) 80–83.

[16] H. B. Mann, D. R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, The Annals of Mathematical Statistics 18 (1) (1947) 50 – 60. doi:10.1214/aoms/1177730491.

[17] K. V. Price, A. Kumar, P. Suganthan, Trial-based dominance for comparing both the speed and accuracy of stochastic optimizers with standard nonparametric tests, Swarm and Evolutionary Computation 78 (2023) 101287. doi:10.1016/j.swevo.2023.101287.

[18] N. Veček, M. Mernik, M. Črepinšek, A chess rating system for evolutionary algorithms: A new method for the comparison and ranking of evolutionary algorithms, Information Sciences 277 (2014) 656–679. doi:10.1016/j.ins.2014.02.154.

[19] M. E. Glickman, Example of the Glicko-2 system, Tech. rep., Boston University (2022).

[20] J. Herzog, J. Brest, B. Bošković, Analysis based on statistical distributions: A practical approach for stochastic solvers using discrete and continuous problems, Information Sciences 633 (2023) 469–490. doi:10.1016/j.ins.2023.03.081.

[21] G. Taguchi, System of Experimental Design: Engineering Methods to Optimize Quality and Minimize Costs, no. t. 1, UNIPUB/Kraus International Publications, 1987.

[22] A. Mozdgir, I. Mahdavi, I. S. Badeleh, M. Solimanpur, Using the Taguchi method to optimize the differential evolution algorithm parameters for minimizing the workload smoothness index in simple assembly line balancing, Mathematical and Computer Modelling 57 (1) (2013) 137–151. doi:10.1016/j.mcm.2011.06.056.

[23] J. Snoek, H. Larochelle, R. P. Adams, Practical Bayesian optimization of machine learning algorithms, arXiv e-prints (Jun. 2012). doi:10.48550/arXiv.1206.2944.

[24] S. Biswas, D. Saha, S. De, A. D. Cobb, S. Das, B. A. Jalaian, Improving differential evolution through bayesian hyperparameter optimization, in: 2021 IEEE Congress on Evolutionary Computation (CEC), IEEE Press, 2021, p. 832–840. doi:10.1109/CEC45853.2021.9504792.

[25] M. Birattari, Z. Yuan, P. Balaprakash, T. Stützle, F-Race and Iterated F-Race: An Overview, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 311–336. doi:10.1007/978-3-642-02538-9_13.

[26] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, M. Birattari, The irace package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (2016) 43–58. doi:10.1016/j.orp.2016.09.002.

[27] C. Ansótegui, M. Sellmann, K. Tierney, A gender-based genetic algorithm for the automatic configuration of algorithms, in: Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, Vol. 5732 of Lecture Notes in Computer Science, Springer, 2009, pp. 142–157. doi:10.1007/978-3-642-04244-7_14.

[28] F. Hutter, H. H. Hoos, K. Leyton-Brown, T. Stützle, ParamILS: An automatic algorithm configuration framework., J. Artif. Intell. Res. 36 (2009) 267–306.

[29] A. P. Piotrowski, J. J. Napiorkowski, Some metaheuristics should be simplified, Information Sciences 427 (2018) 32–62. doi:10.1016/j.ins.2017.10.039.

[30] C. Fawcett, H. H. Hoos, Analysing differences between algorithm configurations through ablation, Journal of Heuristics 22 (4) (2016) 431–458. doi:10.1007/s10732-014-9275-9.

[31] R. Biedrzycki, K. Kwiatkowski, P. Cichosz, Compressor schedule optimization for a refrigerated warehouse using metaheuristic algorithms, in: 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 201–208. doi:10.1109/CEC45853.2021.9504924.

[32] R. Biedrzycki, J. Arabas, A. Jasik, et al., Application of evolutionary methods to semiconductor double-chirped mirrors design, in: T. Bartz-Beielstein, et al. (Eds.), Parallel Problem Solving from Nature – PPSN XIII, Vol. 8672 of Lecture Notes in Computer Science, Springer, 2014, pp. 761–770. doi:10.1007/978-3-319-10762-2_75.

[33] R. Biedrzycki, D. Jackiewicz, R. Szewczyk, Reliability and efficiency of differential evolution based method of determination of Jiles-Atherton model parameters for X30Cr13 corrosion resisting martensitic steel, Journal of Automation, Mobile Robotics and Intelligent Systems 8 (4) (2014) 63–68. doi:10.14313/JAMRIS_4-2014/39.

[34] P. Suganthan, Repository for CEC 2022 special session and competition on single objective bound constrained numerical optimization, https://github.com/P-N-Suganthan/2022-SO-BO/tree/main, (Accessed: 1 October 2023).

[35] P. Bujok, P. Kolenovsky, Eigen crossover in cooperative model of evolutionary algorithms applied to CEC 2022 single objective numerical optimisation, in: 2022 IEEE Congress on Evolutionary Computation (CEC), IEEE Press, 2022, p. 1–8. doi:10.1109/CEC55065.2022.9870433.

[36] V. Stanovov, S. Akhmedova, E. Semenkin, NL-SHADE-LBC algorithm with linear parameter adaptation bias change for CEC 2022 numerical optimization, in: 2022 IEEE Congress on Evolutionary Computation (CEC), 2022. doi:10.1109/CEC55065.2022.9870295.

[37] R. Biedrzycki, J. Arabas, E. Warchulski, A version of NL-SHADE-RSP algorithm with midpoint for CEC 2022 single objective bound constrained problems, in: 2022 IEEE Congress on Evolutionary Computation (CEC), 2022. doi:10.1109/CEC55065.2022.9870220.

[38] L. Van Cuong, N. N. Bao, N. K. Phuong, H. T. T. Binh, Dynamic perturbation for population diversity management in differential evolution, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 391–394. doi:10.1145/3520304.3529075.

[39] P. Kolenovsky, P. Bujok, An adaptive variant of jSO with multiple crossover strategies employing Eigen transformation, in: 2022 IEEE Congress on Evolutionary Computation (CEC), 2022. doi:10.1109/CEC55065.2022.9870378.

[40] B. Sun, Y. Sun, W. Li, Multiple topology SHADE with tolerance-based composite framework for CEC2022 single objective bound constrained numerical optimization, in: 2022 IEEE Congress on Evolutionary Computation (CEC), 2022. doi:10.1109/CEC55065.2022.9870395.

[41] K. M. Sallam, M. Abdel-Basset, M. El-Abd, A. Wagdy, IMODEII: an improved IMODE algorithm based on the reinforcement learning, in: 2022 IEEE Congress on Evolutionary Computation (CEC), 2022. doi:10.1109/CEC55065.2022.9870420.

[42] T.-R. Tseng, Improvement of multi-population ML-SHADE, Tech. rep., Submission to GECCO 2022 - Competition on Single Objective Bound Constrained Numerical Optimization (2022).

[43] H. Ding, Y. Gu, H. Wu, J. Zhou, NL-SOMA-CLP for real parameter single objective bound constrained optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 5–6. doi:10.1145/3520304.3534051.

[44] Y. Ning, D. Jian, H. Wu, J. Zhou, Zeroth-order covariance matrix adaptation evolution strategy for single objective bound constrained numerical optimization competition, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '22, Association

for Computing Machinery, New York, NY, USA, 2022, p. 9–10. doi:10.1145/3520304.3534053.

[45] Y. Gu, H. Ding, H. Wu, J. Zhou, Opposite learning and multi-migrating strategy-based self-organizing migrating algorithm with the convergence monitoring mechanism, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 7–8. doi:10.1145/3520304.3534052.

[46] B. Sun, W. Li, Y. Huang, Performance of composite PPSO on single objective bound constrained numerical optimization problems of CEC 2022, in: 2022 IEEE Congress on Evolutionary Computation (CEC), 2022. doi:10.1109/CEC55065.2022.9870369.

[47] N. Pillay, M. Gerber, GECCO 2022 - competition on single objective bound constrained numerical optimization - submission, Tech. rep., Department of Computer Science, University of Pretoria Gauteng, South Africa (2022).

[48] R. Biedrzycki, Supplementary materials for the paper: "Revisiting CEC 2022 ranking: a new ranking method and influence of parameter tuning", https://staff.elka.pw.edu.pl/~rbiedrzy/publ/CEC22revisiting.zip, (Accessed: 1 October 2023).

[49] V. Stanovov, S. Akhmedova, E. Semenkin, NL-SHADE-RSP algorithm with adaptive archive and selective pressure for CEC 2021 numerical optimization, in: 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 809–816. doi:10.1109/CEC45853.2021.9504959.

[50] R. Biedrzycki, A version of IPOP-CMA-ES algorithm with midpoint for CEC 2017 single objective bound constrained problems, in: IEEE Congr. Evol. Comput., 2017, pp. 1489–1494. doi:10.1109/CEC.2017.7969479.

[51] A. K. Qin, V. L. Huang, P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417. doi:10.1109/TEVC.2008.927706.

[52] Y. Wang, H.-X. Li, T. Huang, L. Li, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (C) (2014) 232–247. doi:10.1016/j.asoc.2014.01.038.

[53] P. Bujok, J. Tvrdík, Enhanced individual-dependent differential evolution with population size adaptation, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE Press, 2017, p. 1358–1365. doi:10.1109/CEC.2017.7969462.

[54] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation 9 (2) (2001) 159–195. doi:10.1162/106365601750190398.

[55] J. Brest, M. S. Maučec, B. Bošković, Single objective real-parameter optimization: Algorithm jSO, in: IEEE Congr. Evol. Comput., 2017, pp. 1311–1318. doi:10.1109/CEC.2017.7969456.

[56] P. S. Bullen, Handbook of Means and Their Inequalities, Springer Dordrecht, 2010. doi:10.1007/978-94-017-0399-4.

[57] R. Biedrzycki, J. Arabas, D. Jagodziński, Bound constraints handling in differential evolution: An experimental study, Swarm and Evolutionary Computation 50 (2019) 100453. doi:10.1016/j.swevo.2018.10.004.