RCMAES: A Robust CMA-ES Variant for CEC2026 Competition

Khoirul Faiq Muzakka*†, Sören Möller*, Martin Finsterbusch*
* Institute of Energy Materials and Devices (IMD-2), Forschungszentrum Jülich GmbH, Germany
† Email: k.muzakka@fz-juelich.de

***Abstract*—This paper proposes RCMAES, a novel variant of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) for CEC benchmark optimization. RCMAES integrates a dimension-dependent nonlinear population-size reduction strategy with an adaptive restart mechanism within a pure CMA-ES framework. RCMAES is evaluated on three benchmark suites (CEC2017, CEC2020, and CEC2022) and compared with state-of-the-art DE algorithms as well as its closely related counterpart, BIPOP-aCMAES. Experimental results show that RCMAES achieves competitive and robust performance across all benchmarks.**

***Index Terms*—Evolutionary Computation, Evolutionary algorithms, CMA-ES, Population Size Reduction**

I. Introduction

The IEEE Congress on Evolutionary Computation (CEC) competitions on bound-constrained single-objective optimization provide a standard testbed for evaluating evolutionary algorithms under fixed function evaluation budgets. In recent editions, Differential Evolution (DE) and its variants have been the most frequently adopted approaches.

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [1] follows a distinct optimization paradigm based on learning second-order information via covariance matrix adaptation. Although CMA-ES is a well-established and widely studied algorithm, pure CMA-ES variants are less commonly used as competitive submissions in recent CEC bound-constrained competitions.

Motivated by competition-oriented advances in DE, this work revisits CMA-ES from a CEC competition perspective. Inspired by the Adaptive Restart–Refine Differential Evolution (ARRDE) algorithm [2], we incorporate a restart mechanism with nonlinear, dimension-dependent population-size reduction into a pure CMA-ES framework. Unlike IPOP- and BIPOP-aCMA-ES [3], [4], which increase population size after restarts, the proposed strategy favors repeated restarts with the same or reduced population sizes and initializes new search phases away from previously converged regions.

Based on these ideas, we propose *RCMAES* (population-reduction Active CMA-ES with Restart), a CMA-ES variant tailored for bound-constrained optimization in the CEC competition setting. The algorithm is evaluated comprehensively on the CEC2017, CEC2020, and CEC2022 benchmark suites, which together cover a wide range of problem dimensionalities and evaluation budgets. In particular, CEC2017 includes higher-dimensional settings ($D = 10, 30, 50,$ and 100), while CEC2020 and CEC2022 focus on lower to medium dimensions ($D \leq 20$) with larger relative evaluation budgets, with CEC2020 providing the largest evaluation budget among the three suites.

To support reproducibility, all algorithms and CEC benchmark problems used in this work, including RCMAES, are implemented within the Minion framework [5] and are publicly available. Experimental results generated in this study can be shared upon reasonable request.

II. CMA-ES

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [1] is a stochastic, derivative-free evolutionary algorithm designed for continuous black-box optimization. CMA-ES performs search by iteratively sampling candidate solutions from a multivariate normal distribution and adapting its parameters to capture the underlying structure of the objective-function landscape.

A. Sampling and Mean Update

At generation t, CMA-ES samples λ offspring according to

$$\mathbf{x}_k^{(t)} \sim \mathcal{N}(\mathbf{m}^{(t)}, {\sigma^{(t)}}^2 \mathbf{C}^{(t)}), \tag{1}$$

where $\mathbf{m}^{(t)}$ is the distribution mean, $\sigma^{(t)}$ is the global step size, and $\mathbf{C}^{(t)}$ is the covariance matrix.

After evaluating all offspring, the mean is updated by weighted recombination of the best μ individuals:

$$\mathbf{m}^{(t+1)} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}^{(t)}, \tag{2}$$

where $\mathbf{x}_{i:\lambda}^{(t)}$ denotes the i-th best solution and w_i are positive weights satisfying $\sum_i w_i = 1$.

B. Covariance Matrix Adaptation

The covariance matrix is updated using a combination of rank-one and rank-μ updates:

$$\mathbf{C}^{(t+1)} = (1 - c_1 - c_\mu)\mathbf{C}^{(t)} + c_1 \mathbf{p}_c \mathbf{p}_c^\top + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_i \mathbf{y}_i^\top, \tag{3}$$

where $\mathbf{p}_c$ is the evolution path for covariance adaptation, $\mathbf{y}_i = (\mathbf{x}_{i:\lambda}^{(t)} - \mathbf{m}^{(t)})/\sigma^{(t)}$, and c_1 and c_μ are learning rates.

This update allows CMA-ES to learn correlations between variables and adapt the search distribution to the local topology of the objective function.

Algorithm 1 RCMAES (Restart CMA-ES with Population Reduction)

Input: f, D, bounds $[L, U]$, budget $N_{\max}$
$N_{\text{evals}} \leftarrow 0$; $(x^\star, f^\star) \leftarrow (\emptyset, +\infty)$; $\varepsilon \leftarrow 10^{-12}$
Compute N_0 by (6) and exponent r by (9)
while $N_{\text{evals}} < N_{\max}$ **do** ▷ restart loop
  Sample mean $\mathbf{m}$ uniformly in $[L, U]$, avoiding previous convergence regions if any
  Initialize $\sigma \leftarrow 0.3$, $\mathbf{C} \leftarrow \mathbf{I}$, $\mathbf{p}_c, \mathbf{p}_\sigma \leftarrow \mathbf{0}$
  while $N_{\text{evals}} < N_{\max}$ **do** ▷ one CMA-ES run
    $t \leftarrow N_{\text{evals}}/N_{\max}$
    $N_p \leftarrow N_0 - (N_0 - D)\,[1 - (1 - t)^r]$
    Sample and evaluate N_p offspring using CMA-ES
    Apply bound-constraint handling
    $N_{\text{evals}} \leftarrow N_{\text{evals}} + N_p$
    Update $(x^\star, f^\star)$
    Update $\mathbf{m}$, σ, and $\mathbf{C}$ using active CMA-ES
    if $(f_{\max} - f_{\min})/\max(|f_{\text{mean}}|, \varepsilon) \leq 10^{-8}$ **then**
      Store converged mean
      break ▷ terminate current run and restart
    end if
  end while
end while
return $x^\star$

C. Step-Size Adaptation

CMA-ES employs cumulative step-size adaptation (CSA) to control the global step size:

$$\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma\|}{E\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1\right)\right), \tag{4}$$

where $\mathbf{p}_\sigma$ is the evolution path for step-size control, and c_σ and d_σ are learning parameters. This mechanism enables self-adaptive control of the exploration–exploitation trade-off without manual tuning.

D. Active Covariance Matrix Adaptation

Instead of using (3), one can also use active covariance adaptation [6], which extends the standard covariance update by incorporating information from unsuccessful search directions. Negative recombination weights are assigned to poorly performing solutions, yielding the modified update:

$$\begin{aligned}\mathbf{C}^{(t+1)} = {} & (1 - c_1 - c_\mu^+ - c_\mu^-)\mathbf{C}^{(t)} + c_1 \mathbf{p}_c \mathbf{p}_c^\top \\ & + c_\mu^+ \sum_{i=1}^{\mu} w_i^+ \mathbf{y}_i \mathbf{y}_i^\top - c_\mu^- \sum_{j=\mu+1}^{\lambda} |w_j^-| \mathbf{y}_j \mathbf{y}_j^\top, \end{aligned} \tag{5}$$

where $w_i^+ > 0$ and $w_j^- < 0$ denote positive and negative recombination weights, respectively. Active covariance adaptation accelerates learning by explicitly decreasing variance in unpromising directions and has been shown to improve convergence speed and robustness[6].

III. The Proposed RCMAES Algorithm

The design of RCMAES is motivated by the trade-off between population size and evaluation budget. While larger populations improve exploration in high-dimensional problems, they quickly exhaust the available budget, requiring population-size reduction. Unlike many DE variants (e.g., LSHADE [7]) that use linear reduction, RCMAES adopts a nonlinear, dimension-dependent strategy. More aggressive reduction is applied in lower dimensions, where large populations are less critical; however, this increases the risk of premature convergence, which is mitigated by an explicit restart mechanism.

Given a maximum evaluation budget $N_{\max}$ and problem dimensionality D, RCMAES initializes the population by sampling individuals from a Gaussian distribution centered at a randomly selected mean within the search bounds. The initial population size N_0 is defined as

$$N_0 = D \times \max[2,\ 10\eta - 20]\,, \tag{6}$$

$$\eta = \log_{10}(N_{\max}/D), \tag{7}$$

making it dependent on both the available evaluation budget and the problem dimension. This functional form allows larger initial populations when sufficient evaluations are available, while remaining conservative under tight budgets.

The population is then evolved according to the standard CMA-ES update rules. During optimization, the population size is reduced in a nonlinear, time-dependent manner. Let N_{evals} denote the number of function evaluations used so far, and define $t = N_{\text{evals}}/N_{\max}$. The population size at time t is given by

$$N_p(t) = N_0 - (N_0 - D)[1 - (1 - t)^r]\,, \tag{8}$$

$$r = 1.7 - 0.01D. \tag{9}$$

This formulation yields more aggressive population reduction for low-dimensional problems and more conservative reduction for higher-dimensional ones. In particular, when $r = 1$, the schedule reduces to a linear scheme.

The coefficients in (6) and (9) were obtained through empirical tuning on the CEC2017, CEC2020, and CEC2022 benchmark suites. Notably, the proposed formulation implies that $r \geq 1$ (i.e., faster-than-linear reduction) occurs when $D \leq 70$, whereas for $D > 70$, the reduction becomes slower than linear. For example, at $D = 100$, $r = 0.7$, resulting in a significantly more gradual reduction.

We observed that this dimension-dependent schedule provides a favorable balance between exploration and exploitation. In lower-dimensional settings, more aggressive reduction leads to more frequent restarts, which enhances the ability to escape local optima. Empirical observations (from an ablation study, omitted due to space constraints) indicate that this behavior is particularly beneficial for complex, multimodal problems in the CEC2020 benchmark suite.

A restart is triggered when population convergence is detected, defined as $\delta = (f_{\max} - f_{\min})/\max(|f_{\text{mean}}|, \varepsilon) \leq 10^{-8}$, where $\varepsilon = 10^{-12}$ and $f_{\max}$, $f_{\min}$, and f_{mean} denote the

maximum, minimum, and mean objective values in the current population, respectively. Upon restart, the population size continues to follow (8), while the covariance matrix, step size, and evolution paths are reset. The new mean is sampled outside a local exclusion region defined as a hyper-rectangular box centered at the previous converged mean, with side lengths equal to 10% of the variable bounds. This mechanism encourages exploration of previously unexplored regions of the search space.

To handle bound constraints, RCMAES adopts a stochastic repair strategy, as the original CMA-ES is designed for unconstrained optimization and does not provide native support for bound constraints. When a candidate violates the search bounds, the infeasible coordinate is re-sampled uniformly within a truncated interval adjacent to the violated boundary. The width of this interval is limited by the violation magnitude, ensuring small corrective steps while preserving diversity near the boundary.

The algorithm terminates when the evaluation budget $N_{\max}$ is exhausted. Pseudocode for RCMAES is provided in Algorithm 1.

IV. Numerical Experiments and Discussions

A. Experimental Setup

The proposed RCMAES is evaluated on three benchmark suites: CEC2017 [8], CEC2020 [9], and CEC2022 [10]. For all experiments, the maximum number of function evaluations ($N_{\max}$) follows the official competition specifications. Each algorithm is executed for 51 independent runs, using the run index as the random seed to ensure reproducibility. All experiments are conducted within the Minion framework [5], implemented in C++ and compiled with MSVC on a Windows 11 workstation. All function evaluations are performed using standard double-precision floating-point arithmetic. Reported results are presented with standard numerical formatting for readability, which may affect the ranking in cases where performance differences are extremely small.

RCMAES is compared against representative state-of-the-art DE and CMA-ES algorithms, including BIPOP-aCMA-ES [4], ARRDE [2], jSO [11], j2020 [12], LSRTDE [13], and NL-SHADE-RSP [14]. All algorithms use the parameter settings recommended in their original publications.

B. Scoring Methodology

Algorithm performance is evaluated using both *rank-based* and *accuracy-based* metrics, following standard CEC evaluation practices. Let k denote the algorithm index, j the benchmark problem, i the independent run, D the problem dimension, N_p the number of benchmark problems, and N_{runs} the number of independent runs.

For rank-based assessment, Friedman ranking [15] is employed. For each problem and run, algorithms are ranked according to their final objective values, with smaller values indicating better performance; ties are assigned the average rank. Let $r_{i,j,k}$ denote the rank of algorithm k on problem j in run i. The overall Friedman score is defined as

$$R_k = \frac{1}{N_p\, N_{\text{runs}}} \sum_{j=1}^{N_p} \sum_{i=1}^{N_{\text{runs}}} r_{i,j,k}.$$

Smaller values of R_k indicate superior overall performance.

To assess whether one algorithm outperforms another in terms of win, tie, or loss outcomes, pairwise win/tie/loss (W/T/L) statistics are employed. For each pair of algorithms, results from N_{runs} independent runs are compared using the Mann–Whitney U test [16] at a significance level of $\alpha = 0.05$. If no statistically significant difference is detected, the comparison is recorded as a tie. Otherwise, the direction of preference is determined according to the rank-sum statistic, yielding either a win or a loss.

Accuracy-based performance is measured using the relative error. For algorithm k on problem j, the relative error is defined as

$$\epsilon_{k,j} = \frac{1}{N_{\text{runs}}} \sum_{i=1}^{N_{\text{runs}}} \frac{f_j(x_{i,k,j}) - f_j(x_j^*)}{f_j(x_j^*)},$$

where $x_{i,k,j}$ is the solution obtained by algorithm k in run i, $f_j(\cdot)$ is the objective function, and x_j^* is the known global optimum. Note that for the considered benchmark suites (CEC2017, CEC2020, and CEC2022), all function values are strictly positive, ensuring that the relative error is well-defined.

To limit the influence of outlier problems, the relative error is mapped to a bounded interval via

$$\mathcal{E}_{k,j} = \frac{\epsilon_{k,j}}{1 + \epsilon_{k,j}},$$

which restricts values to $[0, 1)$. This transformation retains sensitivity for small errors while preventing large errors from dominating the aggregated score. The overall accuracy-based score $\mathcal{E}_k$ of algorithm k is obtained by averaging $\mathcal{E}_{k,j}$ over all benchmark problems for the given dimension.

C. Results

a) CEC2017: Table I summarizes the results on the CEC2017 benchmark suite for $D = 10$, 30, 50, and 100 using W/T/L statistics, Friedman rank R, and the accuracy-based metric $\mathcal{E}$. Since these metrics capture different aspects of performance, they should be interpreted jointly.

Across all dimensions, RCMAES exhibits a strong pairwise W/T/L profile. At $D = 10$, RCMAES performs competitively against all algorithms, with the exception of ARRDE and NL-SHADE-RSP. At higher dimensions ($D = 30$, 50, and 100), RCMAES demonstrates pairwise superiority, winning substantially more often against most competing algorithms.

Notably, RCMAES outperforms LSRTDE in direct pairwise comparisons, despite LSRTDE being a particularly strong baseline. In its original publication[13], LSRTDE was shown to exhibit decisive and systematic dominance over the CEC2017 winning algorithms, such as jSO, establishing a clear performance gap rather than marginal improvements. The W/T/L statistics reported in Table I are consistent with

TABLE I
CEC2017 RESULTS AT $D = 10, 30, 50, 100$ SHOWING ACCURACY $\mathcal{E}$, FRIEDMAN RANK R, AND WIN/TIE/LOSS (W/T/L) STATISTICS OF RCMAES AGAINST COMPETING ALGORITHMS. VALUES IN PARENTHESES DENOTE RANKS; BOLDFACE INDICATES THE BEST PERFORMANCE.

Algorithm	10D			30D			50D			100D		
	$\mathcal{E}$	R	W/T/L	$\mathcal{E}$	R	W/T/L	$\mathcal{E}$	R	W/T/L	$\mathcal{E}$	R	W/T/L
ARRDE	**0.031 (1)**	**3.349 (1)**	5/11/13	0.094 (4)	3.445 (3)	13/13/3	0.160 (4)	3.402 (3)	20/8/1	0.252 (3)	3.438 (3)	22/1/6
jSO	0.036 (4)	3.620 (2)	10/9/10	0.092 (3)	3.452 (4)	19/8/2	0.157 (3)	3.419 (4)	18/9/2	0.260 (4)	3.523 (4)	21/2/6
j2020	0.037 (5)	4.577 (6)	13/10/6	0.200 (7)	6.016 (7)	25/4/0	0.403 (7)	6.139 (7)	26/2/1	0.596 (7)	6.318 (7)	25/3/1
NL-SHADE-RSP	0.035 (2)	3.894 (5)	11/6/12	0.168 (6)	5.144 (6)	23/3/3	0.362 (6)	5.886 (6)	26/2/1	0.546 (6)	6.035 (6)	27/1/1
LSRTDE	0.063 (7)	3.716 (3)	10/9/10	**0.068 (1)**	**2.396 (1)**	12/11/6	**0.101 (1)**	**2.083 (1)**	14/9/6	0.199 (2)	**2.125 (1)**	16/4/9
RCMAES	0.035 (3)	3.731 (4)	–	0.071 (2)	2.436 (2)	–	0.129 (2)	2.139 (2)	–	**0.190 (1)**	2.261 (2)	–
BIPOP-aCMAES	0.051 (6)	5.114 (7)	21/8/0	0.164 (5)	5.111 (5)	25/3/1	0.262 (5)	4.931 (5)	24/3/2	0.354 (5)	4.300 (5)	22/2/5

this characterization, as LSRTDE continues to significantly outperform these earlier winners. Against this strong reference, RCMAES nonetheless achieves more frequent wins in direct comparisons.

In terms of Friedman ranking, RCMAES ranks second at $D = 30$, 50, and 100, and third at $D = 10$, while LSRTDE attains the top rank for $D \geq 30$. The apparent discrepancy between the strong W/T/L results and the Friedman ranks can be attributed to the fundamentally different nature of these measures. Friedman ranking aggregates relative ordering across all algorithms, such that poor performance on a small number of functions can disproportionately affect the overall rank. In the case of RCMAES, this effect is primarily due to function F29, on which RCMAES performs relatively poorly. In contrast, W/T/L analysis reflects pairwise statistical significance. Indeed, when restricting the comparison to RCMAES and LSRTDE only, or when excluding F29, we verified that RCMAES achieves a better Friedman rank than LSRTDE. Overall, these results indicate that the performance of RCMAES and LSRTDE is very close, with differences largely attributable to isolated functions rather than systematic superiority.

The accuracy-based metric $\mathcal{E}$ provides complementary insight into solution quality. LSRTDE achieves the lowest error values at $D = 30$ and $D = 50$, with RCMAES ranking closely behind and attaining the lowest error at $D = 100$. The accuracy differences between these two algorithms are small, while both clearly outperform the remaining methods. At $D = 10$, ARRDE achieves the best accuracy, with RCMAES yielding comparable error values, whereas LSRTDE ranks last under this metric.

b) CEC2020: Table II summarizes the results on the CEC2020 benchmark suite for $D = 5$, 10, 15, and 20. Compared to its performance on CEC2017, RCMAES is less competitive on this benchmark. However, all three performance indicators—Friedman rank R, accuracy $\mathcal{E}$, and W/T/L statistics—exhibit consistent improvement as the problem dimension increases. Notably, at $D = 15$ and $D = 20$, RCMAES performs competitively relative to j2020, despite j2020 being specifically designed for the CEC2020 benchmark.

A marked contrast with the CEC2017 results is observed when examining overall algorithm behavior. LSRTDE, which achieved strong rankings on CEC2017, ranks last or near last across all dimensions on CEC2020. In contrast, algorithms that performed less favorably on CEC2017, such as j2020 and NL-SHADE-RSP, achieve substantially improved rankings on CEC2020, with NL-SHADE-RSP consistently ranking among the top methods. These observations are consistent with the findings reported in [17], [18].

Although RCMAES experiences a degradation in overall ranking on CEC2020, this degradation is relatively moderate, indicating better robustness compared to the competing algorithms.

c) CEC2022: Table III summarizes the results on the CEC2022 benchmark suite for $D = 5$, 10, 15, and 20. RCMAES achieves the best performance at $D = 20$, ranking first according to both the accuracy metric $\mathcal{E}$ and the Friedman rank R. At $D = 10$, however, RCMAES ranks fourth in terms of $\mathcal{E}$ and fifth according to R.

D. CEC2026

The IEEE CEC2026 competition on single-objective bound-constrained optimization adopts the CEC2017 benchmark suite at $D = 30$. The competition emphasizes both fast convergence toward the global optimum and high final solution accuracy. While the final accuracy has been reported in Table I, this subsection focuses on convergence behavior, illustrated in Fig. 1, which shows the average error $f - f^*$ across independent runs as a function of the normalized evaluation budget N_{evals}/D.

From the convergence curves, RCMAES demonstrates strong performance on basic functions, particularly F4, F6, F7, and F9, where it converges rapidly toward high-quality solutions. In contrast, LSRTDE shows clear strengths on hybrid functions, such as F11, F13, and F15.

Algorithms that perform well on CEC2020, such as NL-SHADE-RSP and j2020, exhibit comparatively slow convergence on CEC2017 at $D = 30$. Meanwhile, BIPOP-aCMAES converges rapidly in the early stages but frequently stagnates at local optima, resulting in final solutions that remain far from the global optimum.

For composition functions, most algorithms display similar convergence trends. RCMAES, however, occasionally achieves faster convergence than competing methods, as observed on functions F22 and F23.

TABLE II

CEC2020 RESULTS AT $D = 5, 10, 15, 20$ SHOWING ACCURACY $\mathcal{E}$, FRIEDMAN RANK R, AND WIN/TIE/LOSS (W/T/L) STATISTICS OF RCMAES AGAINST COMPETING ALGORITHMS. VALUES IN PARENTHESES DENOTE RANKS; BOLDFACE INDICATES THE BEST PERFORMANCE.

Algorithm	5D			10D			15D			20D		
	$\mathcal{E}$	R	W/T/L	$\mathcal{E}$	R	W/T/L	$\mathcal{E}$	R	W/T/L	$\mathcal{E}$	R	W/T/L
ARRDE	**0.008 (1)**	3.372 (2)	1/3/6	**0.009 (1)**	3.425 (3)	2/3/5	**0.018 (1)**	3.463 (2)	3/3/4	**0.023 (1)**	**3.378 (1)**	4/1/5
jSO	0.018 (4)	4.097 (4)	1/4/5	0.029 (5)	4.397 (6)	4/2/4	0.039 (7)	4.934 (6)	6/2/2	0.036 (6)	4.961 (6)	6/2/2
j2020	0.010 (2)	3.837 (3)	2/2/6	0.011 (2)	3.374 (2)	2/2/6	0.027 (4)	3.885 (5)	4/3/3	0.039 (7)	3.786 (5)	3/3/4
NL-SHADE-RSP	0.010 (3)	**2.875 (1)**	0/2/8	0.018 (3)	**3.154 (1)**	1/4/5	0.022 (2)	**3.103 (1)**	1/3/6	0.032 (3)	3.393 (2)	4/2/4
LSRTDE	0.018 (5)	4.560 (6)	2/7/1	0.032 (7)	5.579 (7)	8/2/0	0.038 (6)	5.458 (7)	8/2/0	0.035 (5)	5.481 (7)	9/1/0
RCMAES	0.019 (6)	4.519 (5)	–	0.023 (4)	4.111 (5)	–	0.024 (3)	3.680 (4)	–	0.024 (2)	3.527 (4)	–
BIPOP-aCMAES	0.020 (7)	4.741 (7)	3/6/1	0.030 (6)	3.961 (4)	3/2/5	0.028 (5)	3.476 (3)	2/3/5	0.032 (4)	3.473 (3)	2/4/4

TABLE III

CEC2022 RESULTS FOR $D = 10$ AND $D = 20$, REPORTING ACCURACY $\mathcal{E}$, FRIEDMAN RANK R, AND WIN/TIE/LOSS (W/T/L) STATISTICS OF RCMAES AGAINST COMPETING ALGORITHMS. VALUES IN PARENTHESES DENOTE RANKS; BOLDFACE INDICATES THE BEST PERFORMANCE.

Algorithm	10D			20D		
	$\mathcal{E}$	R	W/T/L	$\mathcal{E}$	R	W/T/L
ARRDE	0.014 (3)	**3.516 (1)**	2/4/6	0.017 (2)	3.320 (2)	3/6/3
jSO	0.016 (5)	3.992 (4)	2/6/4	0.035 (6)	4.227 (5)	5/6/1
j2020	**0.013 (1)**	3.660 (2)	2/4/6	0.026 (4)	4.696 (6)	7/4/1
NL-SHADE-RSP	0.014 (2)	3.790 (3)	2/6/4	0.038 (7)	4.876 (7)	7/4/1
LSRTDE	0.016 (6)	4.233 (6)	4/6/2	0.034 (5)	4.069 (4)	5/7/0
RCMAES	0.016 (4)	4.029 (5)	–	**0.016 (1)**	**3.170 (1)**	–
BIPOP-aCMAES	0.021 (7)	4.780 (7)	4/7/1	0.023 (3)	3.642 (3)	5/3/4

Following the CEC2026 competition guidelines, time complexity is evaluated using two measures: T_1, the average time required to evaluate each of the 29 benchmark functions in 30D for 10 000 evaluations, and T_2, the average time required for the algorithm to run with 10 000 function evaluations. For RCMAES, the measured values are $T_1 = 0.0453$, $T_2 = 0.0802$, resulting in a normalized computational overhead of $(T_2 - T_1)/T_1 = 0.7705$.

V. CONCLUSION

This paper introduced RCMAES, a CMA-ES variant that integrates a dimension-dependent nonlinear population-size reduction strategy with an adaptive restart mechanism. Experimental results on the CEC2017, CEC2020, and CEC2022 benchmark suites demonstrate that RCMAES achieves competitive and robust performance compared with state-of-the-art algorithms. Notably, RCMAES exhibits stable behavior across benchmark suites with differing characteristics.

ACKNOWLEDGMENTS

The authors acknowledge financial support from the German Federal Ministry of Education and Research (BMBF) under Grant No. 13XP0445 (For-Analytik). K.F.M., S.M., and M.F. are supported by this grant.

The authors acknowledge the use of artificial intelligence tools (ChatGPT, OpenAI) for language refinement and clarity improvement during manuscript preparation. All technical content, experiments, and conclusions were developed and verified by the authors.

REFERENCES

[1] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[2] Khoirul Faiq Muzakka, Ahsani Hafizhu Shali, Haris Suhendar, Sören Möller, and Martin Finsterbusch. Robust differential evolution via nonlinear population size reduction and adaptive restart: The arrde algorithm. https://arxiv.org/abs/2511.18429, 2026.

[3] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1769–1776 Vol. 2, 2005.

[4] Nikolaus Hansen. Benchmarking a BI-Population CMA-ES on the BBOB-2009 Function Testbed. In *ACM-GECCO Genetic and Evolutionary Computation Conference*, Montreal, Canada, July 2009.

[5] Khoirul Faiq Muzakka, Sören Möller, and Martin Finsterbusch. Minion : a high-performance derivative-free optimization library designed for solving complex optimization problems., February 2025. GitHub repository: https://github.com/khoirulmuzakka/Minion.

[6] Nikolaus Hansen and Raymond Ros. Benchmarking a weighted negative covariance matrix update on the bbob-2010 noisy testbed. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '10, page 1681–1688, New York, NY, USA, 2010. Association for Computing Machinery.

[7] Ryoji Tanabe and Alex S. Fukunaga. Improving the search performance of shade using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1658–1665, 2014.

[8] N.H.Awad, M.Z.Ali, P.N.Suganthan, J.J.Liang, , and B.Y.Qu. Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Technical report, Nanyang Technological University, Jordan University of Science and Technology, and Zhengzhou University, 2016. Available at https://github.com/P-N-Suganthan/CEC2017-BoundContrained.

[9] C. T. Yue, K. V. Price, P. N. Suganthan, J. J. Liang, M. Z. Ali, B. Y. Qu, N. H. Awad, and P. P. Biswas. Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization. Technical report, Zhengzhou University and Nanyang Technological University, 2019. Available at https://github.com/P-N-Suganthan/2020-Bound-Constrained-Opt-Benchmark.

[10] A. Kumar, K. V. Price, A. W. Mohamed, A. A. Hadi, and P. N. Suganthan. Problem definitions and evaluation criteria for the cec 2022 special session and competition on single objective bound constrained numerical optimization. Technical report, Nanyang Technological University, Singapore, 2021. Available: https://github.com/P-N-Suganthan/2022-SO-BO.

[11] Janez Brest, Mirjam Sepesy Maučec, and Borko Bošković. Single

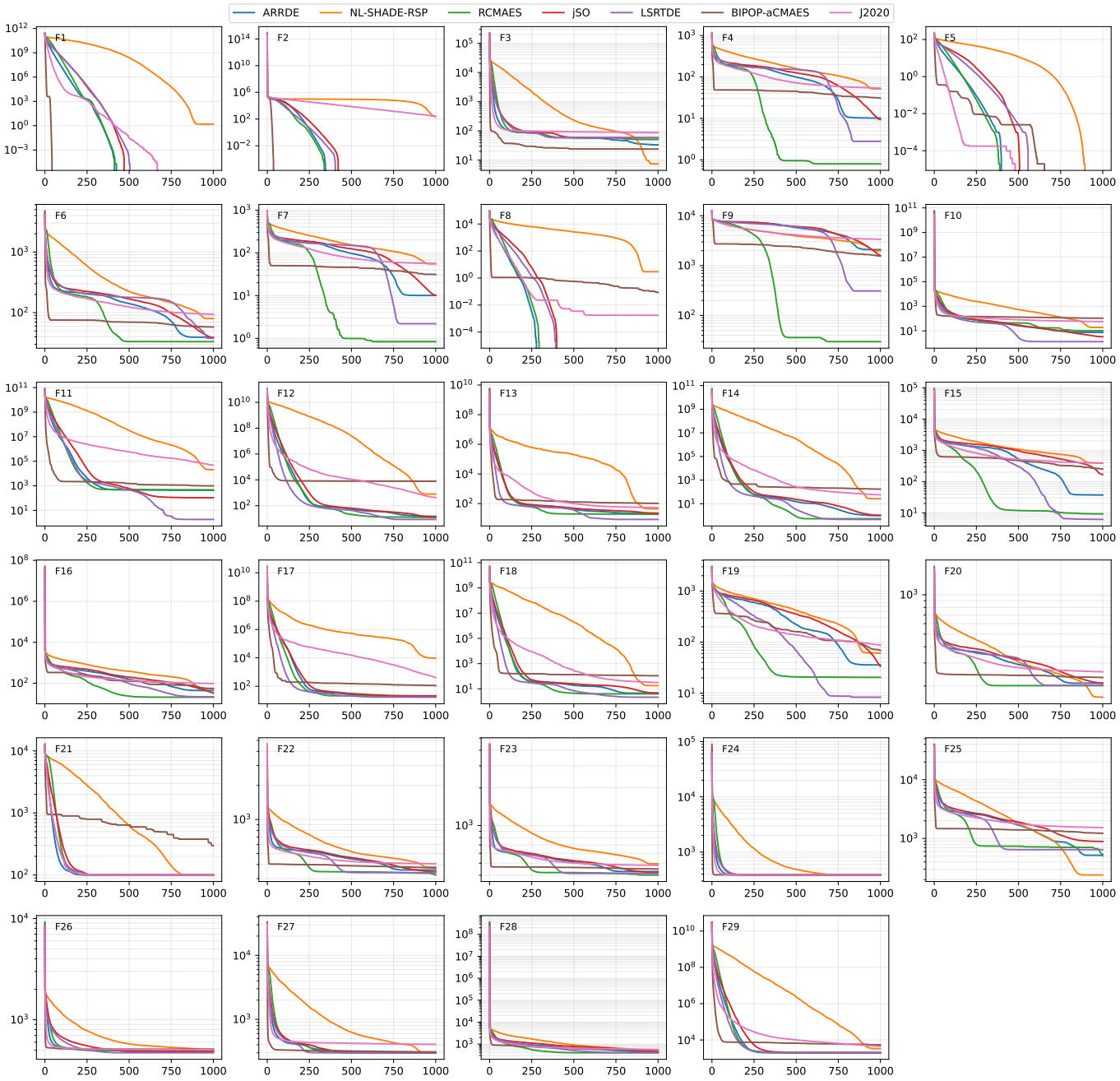

Fig. 1. Average convergence curves on the CEC2017 benchmark at $D = 30$. The plots show the mean error $f - f^*$ over all runs as a function of the normalized number of function evaluations N_{evals}/D.

objective real-parameter optimization: Algorithm jso. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1311–1318, 2017.

[12] Janez Brest, Mirjam Sepesy Maučec, and Borko Bošković. Differential evolution algorithm for single objective bound-constrained optimization: Algorithm j2020. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020.

[13] Vladimir Stanovov and Eugene Semenkin. Success rate-based adaptive differential evolution l-srtde for cec 2024 competition. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2024.

[14] Vladimir Stanovov, Shakhnaz Akhmedova, and Eugene Semenkin. Nl-shade-rsp algorithm with adaptive archive and selective pressure for cec 2021 numerical optimization. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 809–816, 2021.

[15] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[16] H. B. Mann and 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):50 – 60, 1947.

[17] Adam P. Piotrowski, Jaroslaw J. Napiorkowski, and Agnieszka E. Piotrowska. Choice of benchmark optimization problems does matter. *Swarm and Evolutionary Computation*, 83:101378, 2023.

[18] Adam P. Piotrowski, Jaroslaw J. Napiorkowski, and Agnieszka E. Piotrowska. Metaheuristics should be tested on large benchmark set with various numbers of function evaluations. *Swarm and Evolutionary Computation*, 92:101807, 2025.