Coronavirus Optimization Algorithm: A Success-History Adaptive Evolutionary Framework with Archive-Assisted Search and Stagnation Recovery for Global Optimization

Hari Mohan Pandey[a]

[a]*Department of Computing & Engineering Bournemouth University UK*

Abstract

Population-based metaheuristics remain widely used for black-box global optimization, yet the rapid growth of metaphor-based optimizers has raised concerns about weak mathematical grounding, limited operator transparency, and insufficient validation against strong baselines. This paper proposes the Coronavirus Optimization Algorithm (COA), a severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2)-inspired adaptive evolutionary optimizer for box-constrained continuous global optimization. COA does not model epidemiological infection or transmission dynamics; instead, selected coronavirus-related mechanisms are translated into explicit computational operators through a formal biological-to-optimization mapping. Spike–receptor binding is represented through elite-guided attraction, viral replication through trial-vector generation, antigenic drift through adaptive variation, immune evasion through stagnation recovery, and viral-load dynamics through population scheduling. The executable algorithm integrates opposition-based initialization, Differential Evolution (DE)/current-to-pbest/1 mutation, binomial crossover, an external archive, success-history adaptation of the mutation scale factor and crossover rate, population-size reduction, and opposition-based partial restart. The proposed method is evaluated on 29 Congress on Evolutionary Computation (CEC) 2017 benchmark functions at dimensions 10, 30, and 50 against 15 competitive optimizers, including established evolutionary, swarm-intelligence, and recent metaphor-based algorithms. The consolidated results show that COA obtains the best overall average Friedman rank at all tested dimensions, with ranks of 2.79, 2.86, and 2.53 for 10D, 30D, and 50D problems, respectively. Its strongest performance is observed on composition functions, where it achieves the best mean objective value across all composition benchmarks at 30D and 50D. Category-wise analysis also reveals that COA is not uniformly dominant, with stronger competing behaviour from Grey Wolf Optimizer (GWO) on hybrid functions. These findings demonstrate that COA is a competitive, compact, and transparent adaptive evolutionary optimizer, particularly effective for complex composition landscapes, while also identifying its limitations and future validation requirements for higher-dimensional search.

Keywords: Coronavirus Optimization Algorithm, Differential Evolution, success-history adaptation,

Email address: profharimohanpandey@gmail.com (Hari Mohan Pandey)

global optimization, metaheuristics, CEC benchmark, swarm intelligence

1. Introduction

Continuous global optimization is a core problem in evolutionary computation, computational intelligence, engineering design, machine learning, control, and scientific computing. This paper considers the box-constrained minimization problem

$$x^* \in \operatorname{argmin}_{x\in\Omega} f(x), \qquad \Omega = \prod_{j=1}^{D} [lb_j, ub_j] \subset \mathbb{R}^D, \tag{1}$$

where $f : \Omega \to \mathbb{R}$ is a black-box objective function and D is the problem dimension. In many real applications, f may be nonlinear, multimodal, nonconvex, noisy, discontinuous, or expensive to evaluate. These properties make derivative-based and deterministic global methods difficult to apply, particularly when the search space becomes large or irregular.

Population-based metaheuristics address this setting by updating a set of candidate solutions using only objective-function evaluations. Differential Evolution (DE), Particle Swarm Optimization (PSO), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), and adaptive DE variants such as Joint Adaptive Differential Evolution (JADE), Success-History based Adaptive Differential Evolution (SHADE), and Linear Success-History based Adaptive Differential Evolution with modified CMA-ES (LSHADE-SPACMA) have established strong foundations for derivative-free numerical optimization [1–6, 22]. More recent swarm and metaphor-based methods, including Grey Wolf Optimizer (GWO), Harris Hawks Optimization (HHO), Whale Optimization Algorithm (WOA), Salp Swarm Algorithm (SSA), Aquila Optimizer (AO), Runge–Kutta optimizer (RUN), Rime Optimizer (RIME), Dwarf Mongoose Optimization algorithm (DMO), and Crested Porcupine Optimizer (CPO), further expand the design space of population movement, attraction, exploration, and exploitation mechanisms [7–15]. However, the performance of a new optimizer should be justified by its computational operators, ablation evidence, statistical behaviour, and reproducibility, rather than by metaphor alone.

This paper proposes the *Coronavirus Optimization Algorithm* (COA), a severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2)-inspired success-history adaptive evolutionary optimizer for problem (1). COA does not model infection, transmission, immunity, or public-health dynamics. Instead, selected SARS-CoV-2-related concepts are used as an organizing abstraction for optimization behaviour. Spike–receptor binding is translated into elite-guided attraction, viral replication into trial-vector generation, antigenic drift into adaptive parameter variation, immune evasion into opposition-based stagnation recovery, and viral-load dynamics into population-size reduction. The resulting executable algorithm combines opposition-based initialization, DE/current-to-pbest/1 mutation, binomial crossover, an external archive, success-history adaptation of the mutation factor F and crossover rate CR, scheduled population reduction, and opposition-based partial restart.

The experimental study evaluates COA on 29 Congress on Evolutionary Computation (CEC) 2017 benchmark functions at $D = 10$, $D = 30$, and $D = 50$ using 30 independent runs per configuration. The results show that COA achieves the lowest average Friedman rank across all tested dimensions,

with $\bar{R}_{\mathrm{COA}}(10) = 2.79$, $\bar{R}_{\mathrm{COA}}(30) = 2.86$, and $\bar{R}_{\mathrm{COA}}(50) = 2.53$. The strongest and most consistent gains occur on composition functions, where COA benefits from the interaction of elite guidance, archive-assisted diversity, adaptive parameter control, and restart behaviour. The results are more mixed on hybrid functions, where GWO remains a strong competitor. Therefore, the paper does not claim universal superiority; instead, it presents COA as a competitive adaptive evolutionary optimizer with clear strengths, measurable limitations, and reproducible empirical evidence.

The main contributions are summarized as follows:

1. COA is introduced as a mathematically specified adaptive evolutionary optimizer for box-constrained global optimization.
2. A SARS-CoV-2-inspired mapping is defined between biological concepts and concrete search operators, including elite-guided attraction, replication, adaptive variation, stagnation recovery, and population scheduling.
3. The method integrates established evolutionary mechanisms, including current-to-pbest mutation, archive-assisted diversity, success-history adaptation, opposition-based restart, and population-size reduction.
4. Formal properties and detailed proofs are included as main-paper sections, including feasibility preservation, best-so-far monotonicity, population-size validity, finite termination, and evaluation complexity.
5. A consolidated evaluation against 15 baseline optimizers is reported on 29 CEC 2017 functions at $D \in \{10, 30, 50\}$ with 30 independent runs, using ranks, per-function results, convergence evidence, win counts, heatmaps, and category-wise analysis.

The remainder of the paper is organized as follows. Section 2 reviews related work; Section 3 presents the SARS-CoV-2-inspired mapping; Section 4 presents the proposed COA algorithm together with implementation details and complexity analysis; Section 5 provides the mathematical foundation, formal properties, and proofs; Section 6 reports the experimental setup, results, and analysis; and Section 7 concludes the paper.

2. Related Work

Continuous black-box optimization has been extensively studied through evolutionary, swarm-intelligence, covariance-based, and recent metaphor-inspired methods. Classical DE generates new candidate solutions from scaled vector differences and remains a strong baseline because of its simplicity and robustness [1]. Later adaptive DE variants, including JADE, SHADE, and LSHADE-SPACMA, improved this framework through current-to-pbest mutation, external archives, success-history parameter memories, population-size reduction, and covariance-related hybridization [4–6, 22]. COA follows this adaptive evolutionary line: its novelty lies in combining elite-guided mutation, archive-assisted diversity, success-history adaptation, opposition-based restart, and population scheduling within one compact search procedure.

Swarm and metaphor-based optimizers provide complementary search behaviours. PSO uses social learning, while GWO, HHO, WOA, and SSA model hierarchy, pursuit, encircling, and chain-based movement [2, 7–10]. More recent methods such as AO, RUN, RIME, DMO, and CPO introduce additional physical, numerical, or animal-inspired update rules [11–15]. These methods are useful comparators because they represent different exploration–exploitation biases. The present results show that AO is the closest overall competitor to COA, while GWO is particularly strong on hybrid functions.

Covariance-based search, represented by CMA-ES, adapts the shape of the sampling distribution and is often effective on ill-conditioned landscapes [3]. COA does not perform explicit covariance learning; instead, it relies on scalar adaptation of F and CR, archive diversity, and restart mechanisms. This design keeps the algorithm simple and computationally compact, but it also explains why strongly coupled hybrid functions remain challenging. The benchmarking literature recommends that such claims be supported by function-level, category-wise, and statistical evidence rather than aggregate rank alone [24, 26].

Table 1 summarizes the operator-level positioning of COA. The method integrates adaptive parameter control, archive usage, restart, and population scheduling, but not covariance learning. This positioning clarifies that COA should be understood as an adaptive evolutionary optimizer with a coronavirus-inspired explanatory abstraction, rather than as a metaphor-only method.

Table 1: Operator-level positioning of COA.

Optimizer family	**Adapt.**	**Archive**	**Restart**	**Pop. sched.**	**Cov. learn.**
Classical DE [1]	✗	✗	✗	✗	✗
Adaptive DE [4–6]	✓	✓	✗	✓	–
Swarm intelligence [2, 7–10]	–	✗	✗	✗	✗
Covariance-based search [3]	✓	✗	✗	✗	✓
Recent metaphor-based methods [11–15]	–	✗	–	–	✗
COA	✓	✓	✓	✓	✗

Note: Adapt. = adaptive parameter control; Pop. sched. = population-size scheduling; Cov. learn. = covariance or landscape-geometry learning; ✓ = supported; ✗ = not supported; – = partial or indirect support.

Table 1 positions COA against five optimizer families using five operator criteria. COA is the only listed method that simultaneously includes adaptive control, archive-assisted diversity, restart, and population scheduling, while it deliberately avoids covariance learning. This supports the classification of COA as an adaptive evolutionary optimizer rather than a purely metaphor-driven method.

3. COA Inspired by SARS-CoV-2

COA is inspired by selected behavioural stages associated with SARS-CoV-2, but it is not a virological, clinical, or epidemiological model. The inspiration is used as a structured computational metaphor for global optimization. A candidate solution is treated as a possible viral variant in the search space, the objective function is the selection pressure, and the optimization process imitates a sequence of entry, replication, variation, evasion, and population reduction. The purpose of this section is to make the inspiration explicit and to connect each SARS-CoV-2-inspired concept to the mathematical operator used in COA.

Let the feasible search space be

$$\Omega = \prod_{j=1}^{D} [lb_j, ub_j], \tag{2}$$

and let each candidate solution be represented by

$$x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D}) \in \Omega. \tag{3}$$

For a minimization problem, a lower objective value corresponds to a more successful candidate. Hence, the evolutionary pressure in COA is expressed through the comparison of objective values rather than through a biological infection process.

Table 2: SARS-CoV-2-inspired concepts and their roles in COA.

Concept	**Search role**	**COA implementation**
Spike–receptor binding	Attraction toward promising regions	Elite-guided current-to-pbest mutation [5]
Viral replication	Candidate generation	Mutation with binomial crossover [1]
Antigenic drift	Controlled variation	Success-history adaptation of F and CR [4]
Immune evasion	Stagnation escape	Opposition-based partial restart [21, 25]
Viral-load dynamics	Exploration–exploitation transition	Population-size reduction [22]

Table 2 provides a one-to-one translation from SARS-CoV-2-inspired concepts to executable optimization operators. The mapping clarifies that each metaphorical component has a concrete algorithmic role: attraction, trial generation, parameter adaptation, stagnation recovery, or population scheduling.

Table 2 summarizes the complete metaphor-to-mathematics translation. The key point is that the proposed optimizer does not rely on metaphor alone. Each biological idea is converted into an explicit

operator that either changes the candidate solution, controls the population, adapts a parameter, or recovers diversity.

3.1. Biological mechanisms of SARS-CoV-2 and transformation into metaheuristic operators

Let $\mathcal{B} = \{b_1, b_2, b_3, b_4, b_5\}$ denote the set of SARS-CoV-2 biological mechanisms and $\mathcal{O} = \{o_1, o_2, o_3, o_4, o_5\}$ the corresponding optimization operators. Define the transformation mapping $\varphi : \mathcal{B} \to \mathcal{O}$ as a bijection that translates each biological mechanism into an explicit computational operator. Table 2 gives the complete mapping. SARS-CoV-2, the betacoronavirus responsible for the coronavirus disease 2019 (COVID-19) pandemic, exhibits a structured infection cycle with five main phases: (i) host-cell attachment via spike–angiotensin-converting enzyme 2 (ACE2) receptor binding (b_1), (ii) genome release and replication via RNA-dependent RNA polymerase (b_2), (iii) mutation-driven antigenic variation (b_3), (iv) immune evasion through epitope change (b_4), and (v) viral load dynamics over the infection timeline (b_5). Each b_i maps to $o_i = \varphi(b_i)$, where φ is defined as follows.

1. Spike–receptor binding $\mapsto$ elite-guided attraction ($\varphi(b_1) = o_1$). The spike protein's receptor-binding domain recognizes and attaches to the angiotensin-converting enzyme 2 (ACE2) receptor on the host cell surface [29]. In optimization, this selectivity maps to a mutation operator that attracts x_i toward an elite solution x_{pbest}. Formally,

$$o_1(x_i \mid \mathcal{P}_t, \mathcal{A}_t) = x_i + F_i(x_{pbest} - x_i) + F_i(x_{r_1} - \hat{x}_{r_2}), \tag{4}$$

where x_{pbest} is sampled uniformly from the top p-fraction of $\mathcal{P}_t$, $x_{r_1} \sim U(\mathcal{P}_t)$, $\hat{x}_{r_2} \sim U(\mathcal{P}_t \cup \mathcal{A}_t)$, and F_i is the mutation factor.

2. Replication $\mapsto$ trial-vector generation ($\varphi(b_2) = o_2$). The replication–transcription complex produces new genomic RNA [30]. In COA, replication corresponds to generating a trial vector u_i via binomial crossover of $v_i = o_1(x_i)$ with parent x_i:

$$\begin{aligned} o_2(v_i, x_i) &= u_i, \\ u_{i,j} &= \begin{cases} v_{i,j}, & r_j \le CR_i \ \vee \ j = j_{\text{rand}}, \\ x_{i,j}, & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

with selection pressure $S(u_i, x_i) = \operatorname{argmin}\{f(\Pi_\Omega(u_i)), f(x_i)\}$ enforcing greedy retention.

3. Antigenic drift $\mapsto$ adaptive parameter control ($\varphi(b_3) = o_3$). RNA-dependent RNA polymerase lacks proofreading, driving antigenic drift [31]. For optimization, this maps to success-history adaptation:

$$\begin{aligned} o_3: \quad M_{F,k} &\leftarrow \frac{\sum_{F_s \in S_F} F_s^2}{\sum_{F_s \in S_F} F_s}, \\ M_{CR,k} &\leftarrow \frac{1}{|S_{CR}|} \sum_{CR_s \in S_{CR}} CR_s. \end{aligned} \tag{6}$$

where S_F, S_{CR} are the sets of F_i, CR_i values that produced successful offspring in the current generation.

4. Immune evasion $\mapsto$ stagnation recovery ($\varphi(b_4) = o_4$). SARS-CoV-2 evades immune pressure through epitope variation [32]. In COA, stagnation recovery applies opposition-based mapping to the weakest k individuals:

$$o_4(x_i) = x'_i, \quad x'_{i,j} = \begin{cases} lb_j + ub_j - x_{i,j}, & \text{if } \rho_j < 0.5, \\ x_{i,j}, & \text{otherwise,} \end{cases} \tag{7}$$

for $\rho_j \sim U(0,1)$, applied only when $\Delta f_{\text{best}} < \epsilon_{\text{stag}}$ over τ_{stag} evaluations.

5. Viral-load dynamics $\mapsto$ population scheduling ($\varphi(b_5) = o_5$). Viral load follows a rise–peak–decline trajectory [33]. This maps to nonlinear population-size reduction:

$$o_5(NP_0, NP_{\min}, t) = \left\lfloor NP_0 - (NP_0 - NP_{\min}) \times \left(\frac{FES_t}{MAX\_FES} \right)^{0.7} \right\rfloor. \tag{8}$$

which preserves early diversity and concentrates later evaluations.

The transformation from biology to algorithm is not literal; SARS-CoV-2 infection dynamics are far more complex than any computational abstraction. However, the biological narrative provides an intuitive organizational framework for five interconnected optimization mechanisms that together form COA. The following subsections describe each mechanism in mathematical detail.

3.2. Viral-load dynamics: population-size reduction

In COVID-19 SARS-CoV-2, viral load changes across infection stages. COA abstracts this idea as a change in the number of active candidate solutions over the search process. A larger population is retained during the early stage to improve global exploration, while a smaller population is used later to concentrate the function-evaluation budget around promising regions.

Figure 1 has no empirical sample statistics; its purpose is to show the mathematical scheduling idea behind COA. Population size decreases from NP_0 to NP_{min} as the evaluation counter increases, which turns broad early exploration into more focused late exploitation.

The mathematical representation of this mechanism is

$$NP(t) = \left\lfloor NP_0 - (NP_0 - NP_{min}) \left(\frac{FES_t}{MAX\_FES} \right)^{0.7} \right\rfloor, \tag{9}$$

where $NP(t)$ is the active population size at generation or evaluation stage t, NP_0 is the initial population size, NP_{min} is the minimum population size, FES_t is the number of function evaluations already consumed, and MAX_FES is the evaluation budget. The exponent 0.7 makes the reduction nonlinear: the algorithm preserves diversity early and becomes increasingly exploitative as the run progresses. In the context of the experimental results, this mechanism is important because COA keeps enough candidates to explore multimodal and composition landscapes at the beginning, while

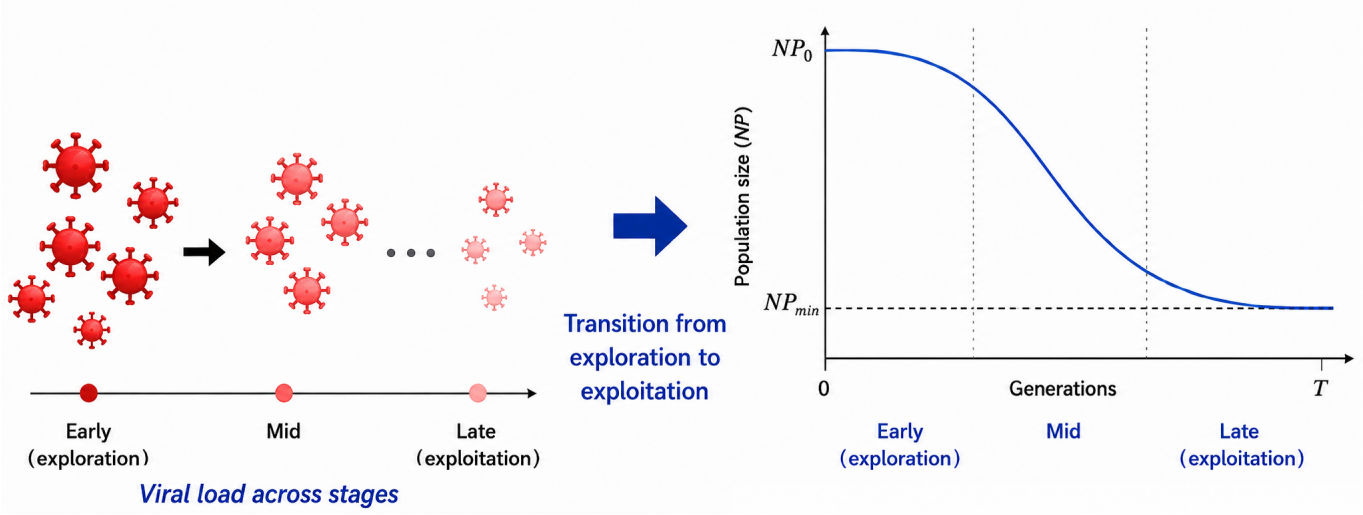

Figure 1: Viral-load dynamics as population-size reduction in COA. The left part illustrates the conceptual decrease in viral load across early, middle, and late stages. The right part gives the optimization role: the active population size is gradually reduced from NP_0 to NP_{min} as the number of function evaluations increases. This reduction controls the shift from broad exploration to focused exploitation.

later dedicating more evaluations to refining fewer competitive individuals. The nonlinear reduction schedule follows the same principle used in adaptive population-sizing DE frameworks [22].

Figure 1 illustrates the biological motivation and optimization role of the population-size reduction strategy used in COA. In the biological analogy, the early stage is represented by a high viral load, where many viral particles coexist and spread widely. This corresponds to the exploration phase of the algorithm, in which a large initial population size NP_0 is maintained to sample diverse regions of the search space and reduce the risk of premature convergence. As the process progresses to the middle stage, the viral load gradually decreases, reflecting a controlled reduction in the number of active candidate solutions. This stage balances exploration and exploitation by preserving sufficient diversity while increasingly directing the search toward promising areas. In the late stage, only a small number of viral particles remain, representing the exploitation phase. The population size approaches the minimum value NP_{min}, allowing the algorithm to concentrate computational effort on local refinement around high-quality solutions.

The right-hand plot formalizes this transition as a population-size reduction schedule. The active population is initially close to NP_0, then decreases progressively with the number of generations or function evaluations, and finally stabilizes near NP_{min}. This dynamic reduction supports the main search philosophy of COA: broad global exploration is encouraged at the beginning, while focused exploitation becomes dominant near the end of the optimization process. Therefore, the viral-load metaphor provides an intuitive explanation for how COA adaptively controls search diversity, convergence pressure, and computational resource allocation across different optimization stages.

3.3. Spike–receptor binding: elite-guided attraction

SARS-CoV-2 enters host cells through spike–receptor binding. COA abstracts this biological entry mechanism as attraction toward high-quality regions of the search space. A current solution x_i is pulled toward an elite solution x_{pbest} while still receiving a diversity-preserving differential perturbation from another population member and the archive.

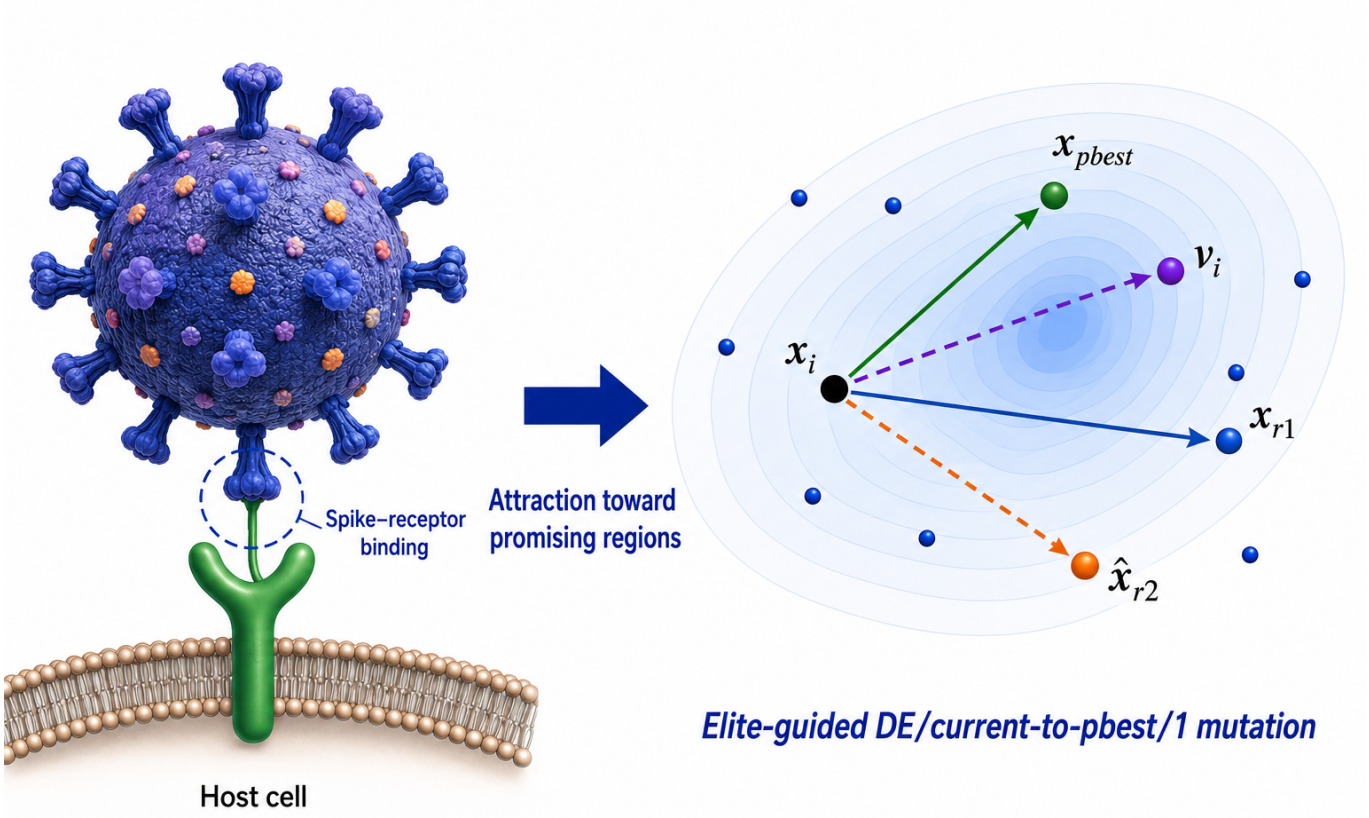

Figure 2: Spike–receptor binding as elite-guided attraction in COA. The biological motif of spike–receptor binding is translated into the search movement from the current solution x_i toward an elite solution x_{pbest}. The additional difference vector $x_{r_1} - \hat{x}_{r_2}$ prevents purely greedy movement and helps maintain exploratory mobility.

Figure 2 illustrates the vector statistics used in the mutation step: the current vector, an elite p-best vector, and a difference vector sampled from the population/archive. The construction balances exploitation toward high-quality solutions with exploratory displacement from randomly sampled candidates.

The corresponding mutation operator is represented as

$$v_i = x_i + F_i(x_{pbest} - x_i) + F_i(x_{r_1} - \hat{x}_{r_2}), \tag{10}$$

where v_i is the mutant vector, F_i is the mutation factor, x_{pbest} is sampled from the top-ranked

population subset, x_{r_1} is a randomly selected population member, and $\hat{x}_{r_2}$ is sampled from either the population or the external archive. The term $F_i(x_{pbest} - x_i)$ represents receptor-like attraction toward a promising region. The term $F_i(x_{r_1} - \hat{x}_{r_2})$ introduces directional variation using current and historical search information. This combination is central to COA because it avoids two extremes: random wandering without guidance and premature collapse around a single elite candidate. The mutation form inherits the elite-guided search direction from JADE's current-to-pbest design [5], extended with archive-assisted diversity.

Figure 2 explains how COA converts the biological idea of spike–receptor binding into an elite-guided search mechanism. The virus binding to a host receptor represents the attraction of a candidate solution x_i toward a promising elite solution x_{pbest}. At the same time, the difference term $x_{r_1} - \hat{x}_{r_2}$ introduces directional variation, avoiding overly greedy convergence and preserving search diversity. This mechanism allows COA to move toward high-quality regions while retaining enough exploratory flexibility.

3.4. Viral replication: trial-vector generation

SARS-CoV-2 replication produces new viral copies. In COA, replication is interpreted as the generation of new candidate solutions. The mutant vector created by Eq. (10) is not automatically accepted. Instead, it is combined with the parent solution through binomial crossover to create a trial vector. Figure 3 explains how a single parent generates a trial vector. The statistical role of crossover is dimension-wise mixing controlled by CR, while the mutation magnitude is controlled by F; both parameters are later adapted from successful trials.

For each dimension j, the crossover operation is

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand_j \leq CR_i \text{ or } j = j_{rand}, \\ x_{i,j}, & \text{otherwise,} \end{cases} \tag{11}$$

where CR_i is the crossover rate and j_{rand} guarantees that at least one dimension is inherited from the mutant vector. The trial vector is then repaired to remain inside the feasible domain and evaluated by the objective function. Selection is greedy:

$$x_{i,t+1} = \begin{cases} \Pi_\Omega(u_i), & \text{if } f(\Pi_\Omega(u_i)) < f(x_{i,t}), \\ x_{i,t}, & \text{otherwise.} \end{cases} \tag{12}$$

Thus, replication in COA means producing and testing offspring under objective-function selection. This mechanism directly supports the main empirical finding of the paper: COA is strong when repeated generation, evaluation, and selective retention of trial vectors can exploit multiple promising basins, especially on composition functions.

Figure 3 shows how COA models viral replication as a candidate-generation process. The parent solution x_i first produces a mutant vector v_i through differential mutation, introducing variation into the search. Binomial crossover then combines selected components of v_i with components of the parent vector to form the trial solution u_i. This mechanism enables COA to generate diverse offspring while preserving useful information from the current solution.

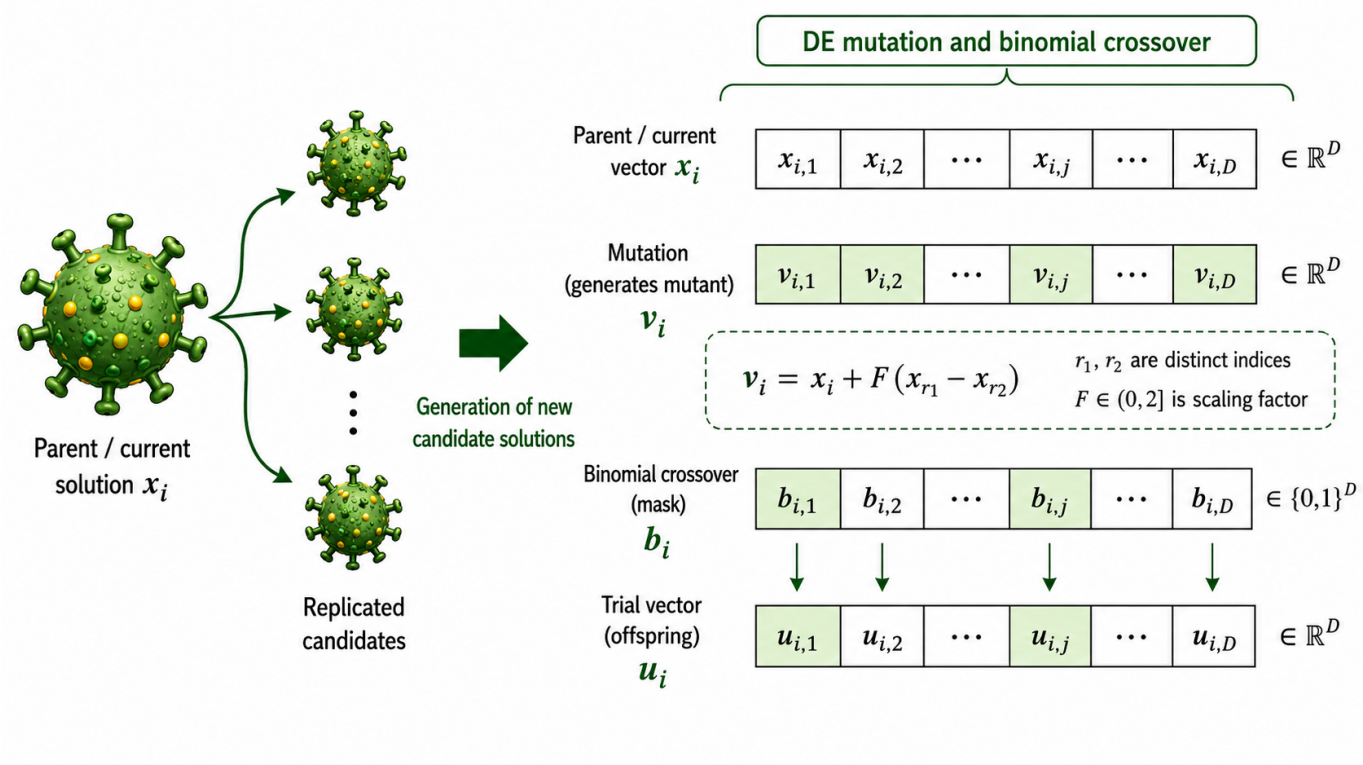

Figure 3: Viral replication as mutation and binomial crossover. The figure describes how COA generates a new candidate solution. First, mutation creates a mutant vector v_i. Then, binomial crossover forms the trial vector u_i by selecting some dimensions from the mutant vector and the remaining dimensions from the parent vector x_i.

3.5. Antigenic drift: adaptive parameter control

SARS-CoV-2 variants change over time through mutation and antigenic drift. COA uses this idea to represent adaptive search behaviour. Rather than fixing the mutation factor F and crossover rate CR, the algorithm learns useful parameter values from successful trials and stores them in success-history memories.

Figure 4 summarizes the feedback loop for F and CR. Successful parameter samples update the memories M_F and M_{CR}, so the distribution used in future generations shifts toward values that recently produced objective improvement.

Let M_F and M_{CR} be memory arrays of length H. For a selected memory index k, COA samples

$$F_i = \text{clip}\left(M_F(k) + 0.1\mathcal{N}(0,1), 0.1, 0.9\right), \tag{13}$$

$$CR_i = \text{clip}\left(M_{CR}(k) + 0.1\mathcal{N}(0,1), 0, 1\right). \tag{14}$$

If S_F and S_{CR} are the successful mutation and crossover values collected in the current generation,

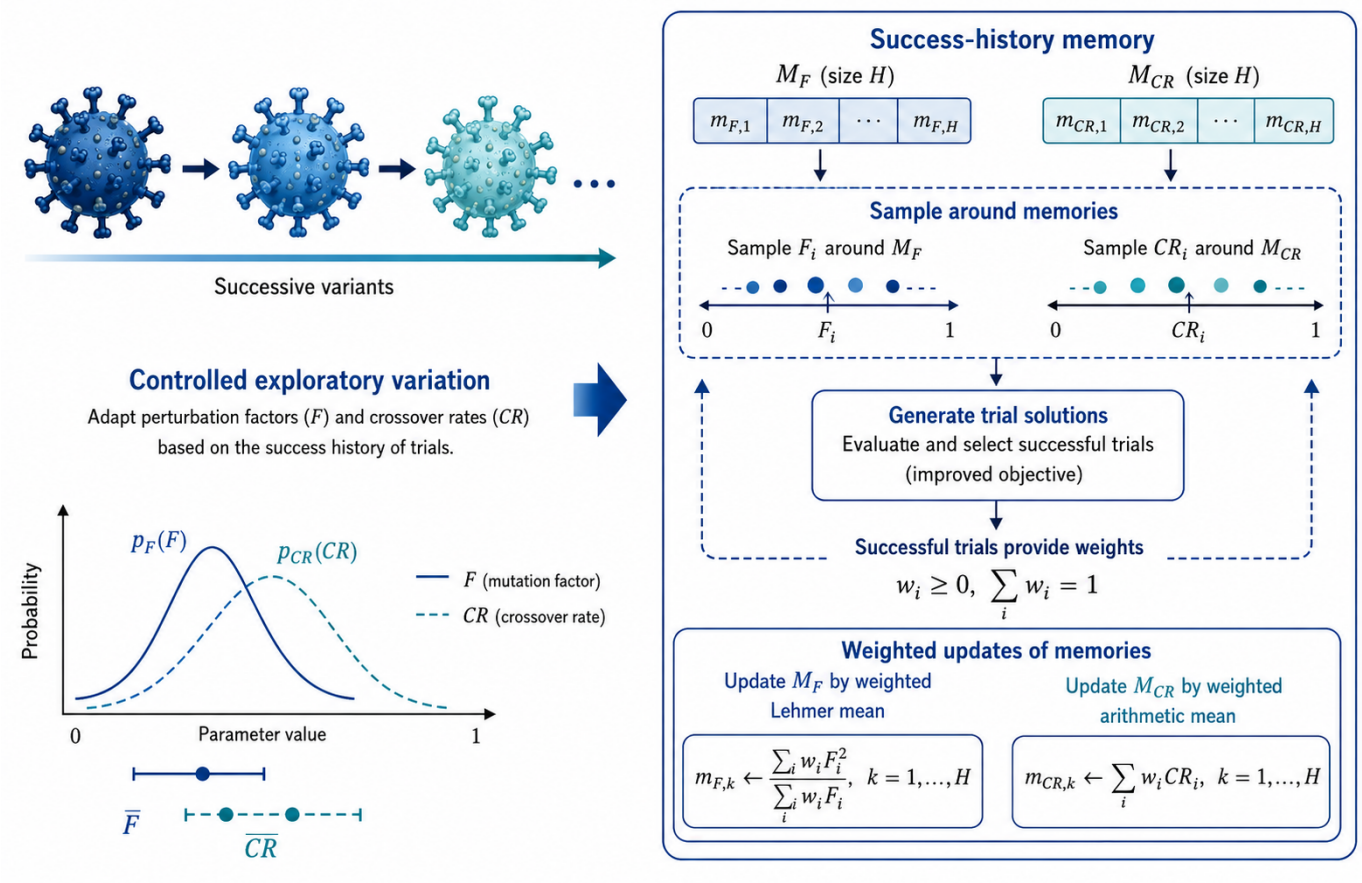

Figure 4: Antigenic drift as success-history adaptation of F and CR. The figure shows the adaptive loop used in COA. Successful trials contribute values to the memories M_F and M_{CR}, and future mutation and crossover parameters are sampled around these memories. This creates controlled exploratory variation rather than static or manually tuned parameter behaviour.

the memories are updated as

$$M_F(k) \leftarrow \frac{\sum_{F_s \in S_F} F_s^2}{\sum_{F_s \in S_F} F_s}, \tag{15}$$

$$M_{CR}(k) \leftarrow \frac{1}{|S_{CR}|} \sum_{CR_s \in S_{CR}} CR_s. \tag{16}$$

The M_F update is a Lehmer-mean style update that gives stronger influence to larger successful mutation factors, while the M_{CR} update captures the average crossover behaviour that produced improvements. The Lehmer-mean weighting for M_F follows the SHADE parameter-adaptation framework [4], and the overall adaptation loop reduces sensitivity to manually chosen initial F and CR values compared to static-parameter DE [27]. This parameter adaptation is the mathematical counterpart of antigenic drift: the population changes its exploratory pattern based on successful experience.

Figure 4 illustrates how COA uses antigenic drift as a metaphor for adaptive parameter control. Successful trial solutions update the memories M_F and M_{CR}, which store effective mutation factors and crossover rates from previous generations. New values of F and CR are then sampled around these memories to guide future candidate generation. This feedback loop enables COA to maintain controlled variation, adapt its search behaviour over time, and avoid relying on fixed parameter settings.

3.6. Immune evasion: opposition-based partial restart

SARS-CoV-2 can persist by evading immune pressure. In COA, immune evasion is interpreted as recovery from search stagnation. When the algorithm does not improve for a defined interval, it does not restart the whole population. Instead, it identifies weak individuals and partially replaces them using opposition-based mapping.

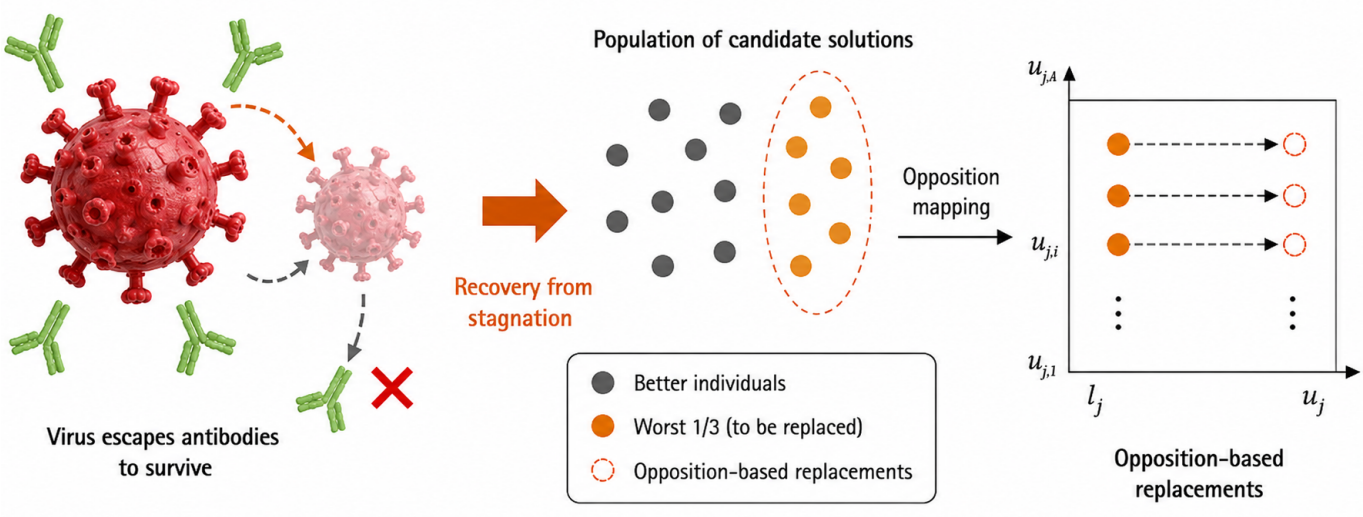

Figure 5: Immune evasion as opposition-based partial restart. The figure shows how COA replaces the weakest part of the population when stagnation occurs. Poor candidates are mapped to opposition-based alternatives inside the bounded domain, while stronger candidates and the best-so-far solution are preserved.

Figure 5 describes the stagnation-recovery mechanism. When improvement stalls, COA targets the weakest portion of the population and replaces it with opposition-based candidates, which increases diversity without discarding the best-so-far solution.

For a weak individual x_i, the opposite coordinate is

$$x'_{i,j} = lb_j + ub_j - x_{i,j}, \qquad j = 1, \ldots, D. \tag{17}$$

In the implemented partial restart, only a subset of dimensions may be replaced, which can be written as

$$x_{i,j}^{new} = \begin{cases} x'_{i,j}, & \text{if } \rho_j < 0.5, \\ x_{i,j}, & \text{otherwise}, \end{cases} \tag{18}$$

where $\rho_j \sim U(0,1)$. This preserves some information from the existing individual while injecting a directionally different candidate into the search. The best-so-far solution and the success-history memories are not discarded. This explains why the restart can improve robustness without destroying the accumulated search knowledge that is needed for later exploitation. The opposition-based restart mechanism draws on the principle that opposite points in the bounded domain can provide useful diversity when the population is stagnating [21, 25].

Figure 5 illustrates COA's stagnation-recovery mechanism inspired by immune evasion. When the search begins to stagnate, the weakest fraction of the population is replaced using opposition-based candidates generated within the problem bounds. This partial restart reintroduces diversity without discarding strong solutions or the best-so-far individual, helping the algorithm escape local optima while preserving convergence progress.

Overall, the SARS-CoV-2 inspiration gives COA a coherent design narrative: viral-load dynamics controls population pressure, spike–receptor binding gives elite-guided attraction, replication generates trial vectors, antigenic drift adapts control parameters, and immune evasion restores diversity after stagnation. The experimental results should therefore be interpreted as evidence for this operator combination, not as evidence for biological fidelity.

4. Proposed COA Algorithm

4.1. Optimization problem

COA solves the bounded black-box minimization problem (1), restated here for completeness:

$$\min_{x\in\Omega} f(x), \qquad \Omega = \prod_{j=1}^{D} [lb_j, ub_j] \subset \mathbb{R}^D, \tag{19}$$

where $f : \mathbb{R}^D \to \mathbb{R}$ is the objective function, $D \in \mathbb{N}$ is the dimension, and $lb_j, ub_j \in \mathbb{R}$ with $lb_j < ub_j$ define the feasible interval for each decision variable x_j. The method requires only that f be evaluable at any $x \in \Omega$; it does not require ∇f, convexity, differentiability, or separability.

At generation $t \in \mathbb{N}_0$, the population is the multiset

$$\mathcal{P}_t = \{x_{1,t}, x_{2,t}, \ldots, x_{NP_t,t}\}, \qquad x_{i,t} \in \Omega,\ |\mathcal{P}_t| = NP_t, \tag{20}$$

where $NP_t \in \mathbb{N}$ is the current population size satisfying $NP_{\min} \le NP_t \le NP_0$. The best-so-far solution up to generation t is

$$g_t = \operatorname{argmin}_{x\in\bigcup_{\tau=0}^{t}\mathcal{P}_\tau} f(x), \qquad f(g_t) = \min_{\tau\le t}\ \min_{x\in\mathcal{P}_\tau} f(x). \tag{21}$$

COA terminates when the function-evaluation counter (FES) reaches the maximum evaluation budget $MAX\_FES \in \mathbb{N}$.

4.2. Opposition-based initialization

Let $\mathcal{X} = \{x_i\}_{i=1}^{NP_0}$ be a set of NP_0 candidate solutions sampled independently from $U(\Omega)$. For each x_i, COA constructs its opposition point $\tilde{x}_i$ componentwise as

$$\tilde{x}_{i,j} = lb_j + ub_j - x_{i,j}, \qquad j = 1, \ldots, D. \tag{22}$$

Define $\tilde{\mathcal{X}} = \{\tilde{x}_i\}_{i=1}^{NP_0}$. The initial population is

$$\mathcal{P}_0 = \text{argmin}_{NP_0 \text{ elements}} \bigcup_{x \in \mathcal{X} \cup \tilde{\mathcal{X}}} f(x), \tag{23}$$

i.e., the NP_0 fittest individuals from $\mathcal{X} \cup \tilde{\mathcal{X}}$. This provides better initial domain coverage than random sampling alone [21, 25]. In COA, $NP_0 = 30$ is fixed across all $D \in \{10, 30, 50\}$.

4.3. Success-history parameter adaptation

COA maintains two memory vectors $M_F, M_{CR} \in \mathbb{R}^H$ of length $H \in \mathbb{N}$, initialized as $M_F^{(0)}(k) = 0.8$ and $M_{CR}^{(0)}(k) = 0.7$ for all $k = 1, \ldots, H$. Let $k_i \sim U(\{1, \ldots, H\})$ be a memory index sampled independently for each individual i. The mutation factor and crossover rate are generated as

$$F_i = \text{clip}\left(M_F(k_i) + 0.1\,\mathcal{N}(0,1),\ 0.1,\ 0.9\right), \tag{24}$$

$$CR_i = \text{clip}\left(M_{CR}(k_i) + 0.1\,\mathcal{N}(0,1),\ 0,\ 1\right), \tag{25}$$

where $\text{clip}(z, a, b) = \max\{a, \min\{b, z\}\}$. After each generation, define $S_F = \{F_i \mid f(u_i) < f(x_i)\}$ and $S_{CR} = \{CR_i \mid f(u_i) < f(x_i)\}$ as the sets of parameter values that produced successful trial vectors. The memory at index k (rotated sequentially) is updated as

$$M_F(k) \leftarrow \frac{\sum_{F_s \in S_F} F_s^2}{\sum_{F_s \in S_F} F_s}, \tag{26}$$

$$M_{CR}(k) \leftarrow \frac{1}{|S_{CR}|} \sum_{CR_s \in S_{CR}} CR_s, \tag{27}$$

where the M_F update is the Lehmer (harmonic) mean that biases toward larger successful F values. This feedback loop reduces sensitivity to manually chosen initial parameters [4, 27].

4.4. Mutation, archive, crossover, and selection

The primary mutation operator is DE/current-to-pbest/1 with archive:

$$v_i = x_i + F_i\,(x_{pbest} - x_i) + F_i\,(x_{r_1} - \hat{x}_{r_2}), \tag{28}$$

where x_{pbest} is sampled uniformly from the top $\lceil p \cdot NP_t \rceil$ individuals of $\mathcal{P}_t$ ($p \in (0, 1]$), $x_{r_1} \sim U(\mathcal{P}_t)$, and $\hat{x}_{r_2} \sim U(\mathcal{P}_t \cup \mathcal{A}_t)$ with $\mathcal{A}_t$ being the external archive. The first difference term $F_i(x_{pbest} - x_i)$

provides elite-guided attraction; the second term $F_i(x_{r_1} - \hat{x}_{r_2})$ preserves exploratory variation using current and historical directions.

Binomial crossover generates a trial vector $u_i \in \mathbb{R}^D$ as

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand_j \leq CR_i \ \vee \ j = j_{rand}, \\ x_{i,j}, & \text{otherwise,} \end{cases} \tag{29}$$

where $j_{rand} \sim U(\{1, \ldots, D\})$ guarantees $u_i \neq x_i$. Boundary violations are repaired by projection onto Ω:

$$\Pi_\Omega(z)_j = \min\{ub_j, \ \max\{lb_j, \ z_j\}\}, \quad j = 1, \ldots, D. \tag{30}$$

Greedy selection then determines the survivor:

$$x_{i,t+1} = \begin{cases} \Pi_\Omega(u_i), & \text{if } f(\Pi_\Omega(u_i)) < f(x_{i,t}), \\ x_{i,t}, & \text{otherwise.} \end{cases} \tag{31}$$

When a trial replaces its parent, the displaced parent $x_{i,t}$ is appended to $\mathcal{A}_t$. The archive size is bounded by $|\mathcal{A}_t| \leq NP_t$; when full, elements are removed uniformly at random. The archive preserves historical search directions that would otherwise be lost through greedy selection [5].

4.5. Population-size reduction and restart

COA reduces the population size deterministically as a function of the consumed evaluation budget:

$$NP(FES_t) = \left\lfloor NP_0 - (NP_0 - NP_{\min}) \times \left(\frac{FES_t}{MAX\_FES} \right)^\gamma \right\rfloor. \tag{32}$$

where $\gamma = 0.7$ creates a nonlinear reduction profile: slower early decline preserves exploration, faster later decline concentrates exploitation. The worst $NP_t - NP(FES_t)$ individuals (by f-value) are removed when $NP(FES_t) < NP_t$. This scheduled reduction follows the Linear Population Size Reduction SHADE (L-SHADE) framework [22].

Let $\Delta f^{(t)}_{\text{best}} = f(g_{t-\tau}) - f(g_t)$. If $\Delta f^{(t)}_{\text{best}} < \epsilon_{\text{stag}}$ for τ_{stag} consecutive generations, COA triggers opposition-based partial restart. The weakest $\lceil NP_t/3 \rceil$ individuals in $\mathcal{P}_t$ are replaced by their opposition-based counterparts:

$$x^{\text{new}}_{i,j} = \begin{cases} lb_j + ub_j - x_{i,j}, & \text{if } \rho_j < 0.5, \\ x_{i,j}, & \text{otherwise,} \end{cases} \quad \rho_j \sim U(0,1), \tag{33}$$

while g_t and M_F, M_{CR} remain unchanged. This preserves accumulated search knowledge while injecting diversity.

Algorithm 1: COA algorithmic procedure

Input : Objective function $f(\cdot)$, bounds $\Omega = [lb, ub]$, dimension D, initial population size NP_0, minimum population size $NP_{\min}$, evaluation budget MAX_FES, memory size H
Output : Best solution g and best objective value $f(g)$

Initialize success-history memories $M_F \leftarrow 0.8$ and $M_{CR} \leftarrow 0.7$ of size H; set memory index $k \leftarrow 1$
Generate random population P and opposite population $\widetilde{P}$ within Ω
Evaluate $P \cup \widetilde{P}$ and retain the best NP_0 individuals as $P^{(0)}$
Set $g \leftarrow \arg\min_{x_i \in P^{(0)}} f(x_i)$; initialize archive $A \leftarrow \emptyset$ and evaluation counter FES
while $FES < MAX\_FES$ **do**
    Update target population size using Eq. (32) and remove the worst individuals if required
    if *the stagnation criterion is satisfied* **then**
        Apply opposition-based partial restart to the weakest individuals using Eq. (33)
        Preserve g, M_F, M_{CR}, and archive A
    Set $S_F \leftarrow \emptyset$, $S_{CR} \leftarrow \emptyset$, and $S_w \leftarrow \emptyset$
    for $i = 1$ **to** NP_t **do**
        Sample F_i and CR_i from the success-history memories
        Select x_{pbest} from the top-ranked p fraction of the population
        Select distinct $x_{r_1} \in P^{(t)}$ and $\hat{x}_{r_2} \in P^{(t)} \cup A$
        Generate mutant vector v_i using Eq. (28)
        Generate trial vector u_i using binomial crossover in Eq. (29)
        Repair u_i to satisfy the bounds and evaluate $f(u_i)$
        $FES \leftarrow FES + 1$
        if $f(u_i) < f(x_i)$ **then**
            Insert the displaced parent x_i into archive A
            Replace x_i with u_i
            Store successful F_i, CR_i, and improvement weight in S_F, S_{CR}, and S_w
            if $f(u_i) < f(g)$ **then**
                $g \leftarrow u_i$
        if $FES \geq MAX\_FES$ **then**
            break
    Trim archive A if $|A| > NP_t$
    if $S_F \neq \emptyset$ **then**
        Update $M_{F,k}$ by the weighted Lehmer mean and $M_{CR,k}$ by the weighted arithmetic mean
        $k \leftarrow (k \bmod H) + 1$
return g *and* $f(g)$

4.6. Algorithmic procedure

Algorithm 1 summarizes COA.

4.7. Computational complexity

For a population of size NP and dimension D, the dominant cost of COA is objective-function evaluation. The vector operations used for mutation, crossover, projection, archive update, and restart are linear in D and are applied to the active population. Therefore, the total runtime is governed by

$$O(MAX\_FES \cdot C_f + MAX\_FES \cdot D)\,, \tag{34}$$

where C_f is the cost of one objective-function evaluation. When C_f is non-trivial, the practical complexity is dominated by $O(MAX\_FES \cdot C_f)$. A detailed operation-level breakdown is provided in Subsection 4.8.2.

The formal assumptions, mathematical properties, and detailed proofs for COA are provided in Section 5, including feasibility preservation, best-so-far monotonicity, archive-assisted diversity, population-size validity, finite termination, evaluation complexity, and idealized asymptotic coverage.

4.8. Implementation Details and Complexity Breakdown

4.8.1. Detailed pseudocode with edge-case handling

Figure 6 provides a complete execution-level view of COA. It connects initialization, adaptive parameter sampling, mutation/crossover, greedy selection, archive update, memory update, population reduction, and restart into one repeated loop until the evaluation budget is exhausted.

Figure 6 summarizes how all components of COA work together. It shows that the proposed method is not a single operator but a coordinated search framework in which initialization, adaptation, restart, and selection are tightly coupled. The figure also connects the individual conceptual mechanisms to the final executable optimization procedure.

4.8.2. Computational complexity breakdown

Table 3 shows that the dominant per-generation cost is objective evaluation, $O(NP \cdot C_f)$, while mutation, crossover, selection, archive handling, memory updates, and population scheduling add linear or lower-order overhead. Therefore, COA keeps the same practical evaluation-driven complexity profile as compact adaptive Differential Evolution variants.

5. Mathematical Foundation and Formal Properties and Proofs

5.1. Assumptions

These assumptions define the scope of the formal statements.

Assumption 1 (Bounded feasible domain)**.** *The feasible domain Ω is a nonempty compact hyperrectangle in $\mathbb{R}^D$.*

Assumption 2 (Evaluable objective)**.** *The objective function f is finite and evaluable for all $x \in \Omega$.*

Assumption 3 (Nonzero restart coverage)**.** *When restart is triggered, every open subset of Ω has nonzero probability of being sampled through the restart mechanism.*

These assumptions are standard in the black-box optimization literature and are satisfied by the Congress on Evolutionary Computation (CEC) benchmark framework [28] and by typical continuous optimization problems.

Proposition 1 (Feasibility preservation)**.** *If all parent solutions are in Ω, then every accepted COA solution remains in Ω.*

Proof. Mutation and crossover may produce a vector outside Ω. Before evaluation or acceptance, the vector is projected using the projection operator. Projection maps each coordinate into its valid interval. Therefore every accepted solution belongs to Ω. This projection-based constraint handling is standard in bounded real-parameter optimization with Differential Evolution (DE) [1]. □

Algorithm 2: COA – detailed implementation

Input : f, $lb, ub \in \mathbb{R}^D$, $D \in \mathbb{N}$, $MAX\_FES \in \mathbb{N}$
Output : Best solution g and its value $f(g)$

$NP_0 \leftarrow 30$, $NP_{\min} \leftarrow 8$, $H \leftarrow 60$, $\gamma \leftarrow 0.7$
$p \leftarrow 0.15$, $\tau_{\text{stag}} \leftarrow 50$, $\epsilon_{\text{stag}} \leftarrow 10^{-8}$
$M_F \leftarrow [0.8]_H$, $M_{CR} \leftarrow [0.7]_H$, $k \leftarrow 1$
$\mathcal{X} \leftarrow \{x_i \sim U(\Omega)\}_{i=1}^{NP_0}$
$\tilde{\mathcal{X}} \leftarrow \{\tilde{x}_i : \tilde{x}_{i,j} = lb_j + ub_j - x_{i,j}\}_{i=1}^{NP_0}$
$\mathcal{P}_0 \leftarrow \text{argmin}_{NP_0} f(\mathcal{X} \cup \tilde{\mathcal{X}})$
$\mathcal{A} \leftarrow \emptyset$, $g \leftarrow \text{argmin}_{x \in \mathcal{P}_0} f(x)$, $FES \leftarrow 2NP_0$, $t \leftarrow 0$
while $FES < MAX\_FES$ **do**
  $NP_{\text{target}} \leftarrow \max\{NP_{\min}, \lfloor NP_0 - (NP_0 - NP_{\min})(FES/MAX\_FES)^{\gamma} \rfloor\}$
  if $NP_{target} < NP_t$ **then**
    Remove $NP_t - NP_{\text{target}}$ worst individuals; $NP_t \leftarrow NP_{\text{target}}$
  if $\Delta f_{best} < \epsilon_{stag}$ *for* τ_{stag} *generations* **then**
    $n_{\text{restart}} \leftarrow \lceil NP_t/3 \rceil$
    for *each of the* $n_{restart}$ *worst individuals* x_i **do**
      for $j = 1$ **to** D **do**
        if $\rho_j < 0.5$ **then**
          $x_{i,j} \leftarrow lb_j + ub_j - x_{i,j}$
    $t_{\text{stag}} \leftarrow 0$
  for $i = 1$ **to** NP_t **do**
    $k_i \sim U(\{1, \ldots, H\})$
    $F_i \leftarrow \text{clip}(M_F(k_i) + 0.1\mathcal{N}(0,1), 0.1, 0.9)$
    $CR_i \leftarrow \text{clip}(M_{CR}(k_i) + 0.1\mathcal{N}(0,1), 0, 1)$
    $x_{pbest} \sim U(\text{top } \lceil p \cdot NP_t \rceil \text{ of } \mathcal{P}_t)$
    $x_{r_1} \sim U(\mathcal{P}_t)$; $\hat{x}_{r_2} \sim U(\mathcal{P}_t \cup \mathcal{A})$
    $v_i \leftarrow x_i + F_i(x_{pbest} - x_i) + F_i(x_{r_1} - \hat{x}_{r_2})$
    $j_{rand} \sim U(\{1, \ldots, D\})$
    for $j = 1$ **to** D **do**
      if $rand_j \leq CR_i$ *or* $j = j_{rand}$ **then**
        $u_{i,j} \leftarrow v_{i,j}$
      else
        $u_{i,j} \leftarrow x_{i,j}$
      $u_{i,j} \leftarrow \min\{ub_j, \max\{lb_j, u_{i,j}\}\}$
    $FES \leftarrow FES + 1$
    if $f(u_i) < f(x_i)$ **then**
      $\mathcal{A} \leftarrow \mathcal{A} \cup \{x_i\}$
      if $|\mathcal{A}| > NP_t$ **then**
        Remove random element from $\mathcal{A}$
      $x_i \leftarrow u_i$
      $S_F \leftarrow S_F \cup \{F_i\}$, $S_{CR} \leftarrow S_{CR} \cup \{CR_i\}$
      if $f(x_i) < f(g)$ **then**
        $g \leftarrow x_i$
  if $S_F \neq \emptyset$ **then**
    $M_F(k) \leftarrow \sum_{F_s \in S_F} F_s^2 / \sum_{F_s \in S_F} F_s$
    $M_{CR}(k) \leftarrow \frac{1}{|S_{CR}|} \sum_{CR_s \in S_{CR}} CR_s$
    $k \leftarrow (k \bmod H) + 1$
    $S_F \leftarrow \emptyset$, $S_{CR} \leftarrow \emptyset$
  $t \leftarrow t + 1$
return g, $f(g)$

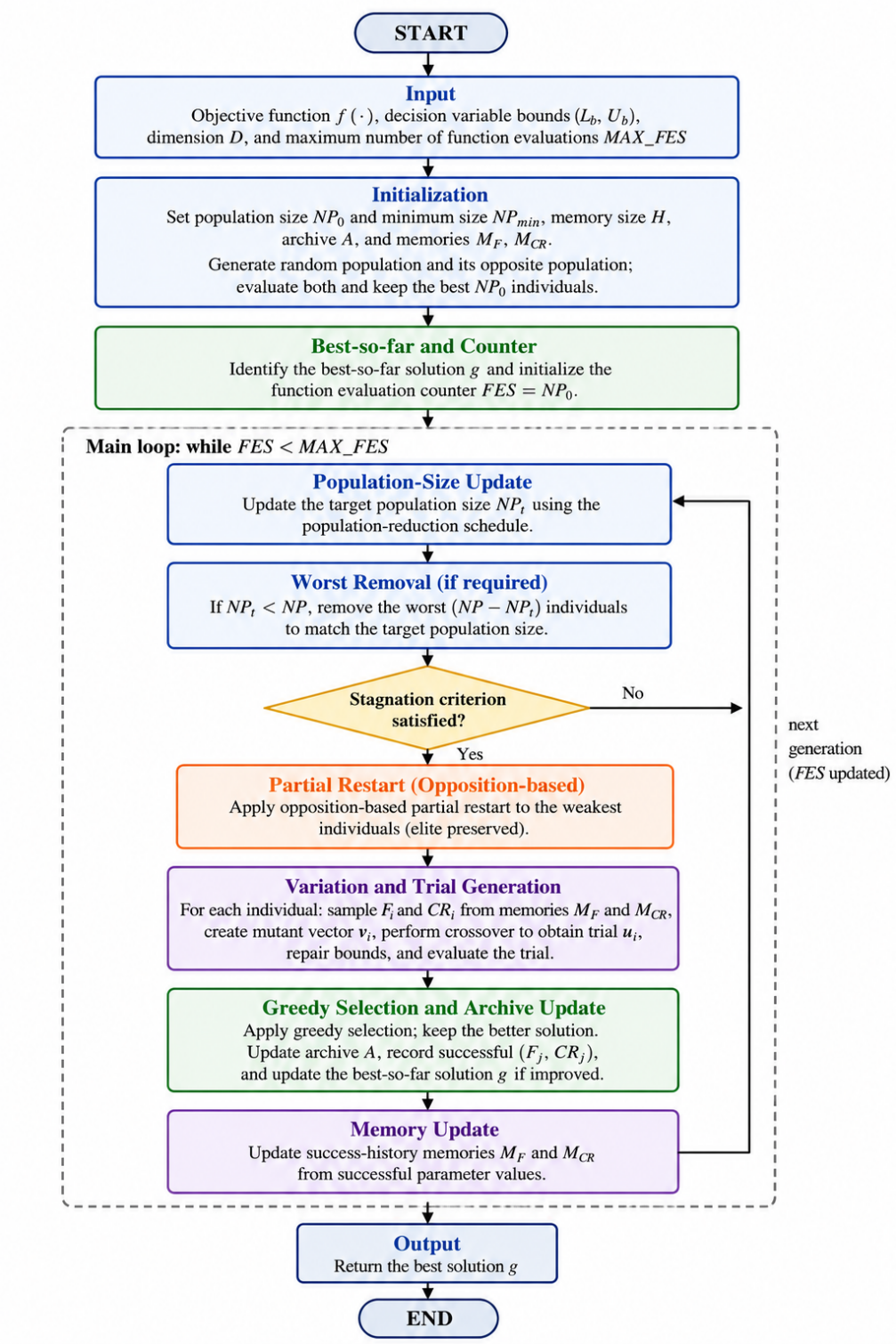

Figure 6: Overall flowchart of the Coronavirus Optimization Algorithm (COA). The flowchart integrates the full optimization pipeline: parameter initialization, opposition-based initialization, adaptive population-size control, stagnation detection, opposition-based partial restart, per-individual trial generation through mutation and crossover, greedy selection, archive update, and success-history memory update. It serves as a visual complement to Algorithm 2 and shows how the main operator blocks interact during one complete optimization run.

Table 3: Per-generation computational complexity of COA operations at population size NP.

Operation	**Time**	**Frequency per generation**
Population sorting	$O(NP \log NP)$	1
Mutation (vector)	$O(D)$	NP
Crossover (vector)	$O(D)$	NP
Boundary projection	$O(D)$	NP
Objective evaluation	C_f	NP
Archive pruning	$O(NP)$	≤ 1
Memory update	$O(H)$	1
Restart (if triggered)	$O(NPD)$	≤ 1

Proposition 2 (Best-so-far monotonicity)**.** *The best-so-far objective value is non-increasing over time.*

Proof. COA explicitly stores the best solution found so far. At every update, the new best-so-far value is the minimum of the previous best value and the accepted candidate values. Hence $f(g_{t+1}) \leq f(g_t)$. □

Proposition 3 (Archive-assisted expansion of mutation directions)**.** *If the archive is nonempty, the set of possible difference vectors in the mutation equation is at least as large as the set obtained from the current population alone.*

Proof. Without an archive, the second vector in the difference term is sampled from the current population. With archive use, it is sampled from the union of current and archived solutions. Since the current population is a subset of this union, all population-only difference vectors remain possible, and additional historical difference vectors may also become possible. □

Proposition 4 (Population-size validity)**.** *For $0 \leq FES_t \leq MAX\_FES$, the scheduled population size in the population-size schedule lies between NP_{min} and NP_0 before integer rounding.*

Proof. The ratio FES_t/MAX_FES lies in $[0, 1]$. Raising it to the power 0.7 keeps it in $[0, 1]$. Therefore the subtracted term ranges from 0 to $NP_0 - NP_{min}$, and the scheduled value ranges from NP_0 to NP_{min}. □

Theorem 1 (Finite termination)**.** *COA terminates after at most MAX_FES objective evaluations, up to the small implementation-level overshoot that can occur when a restart batch is evaluated near the budget boundary.*

Proof. The algorithm increments the function-evaluation counter after objective evaluations and tests the stopping condition against MAX_FES. Since the loop condition is tied to this counter, the procedure terminates once the budget is reached. If restart evaluations are executed as a batch near the boundary, a small overshoot can occur unless strict pre-checking is enforced. This does not change the asymptotic evaluation complexity. □

Theorem 2 (Evaluation complexity)**.** *Let C_f be the cost of one objective-function evaluation. The dominant cost of COA is $O(MAX\_FES \cdot C_f)$.*

Proof. Objective evaluations dominate the intended black-box setting. Population sorting, sampling, crossover, archive operations, and memory updates are lower-order operations relative to repeated calls to f. Therefore the dominant cost is linear in the number of evaluations. This complexity bound is typical for population-based metaheuristics [24, 26]. □

Theorem 3 (Idealized asymptotic coverage)**.** *If restart is triggered infinitely often and each restart sample has a nonzero probability of falling in an ϵ-optimal set $\Omega_\epsilon = \{x \in \Omega : f(x) \leq f(x^*) + \epsilon\}$ of positive measure, then the probability that COA eventually samples Ω_ϵ tends to one as the number of restart samples tends to infinity.*

Proof. Let $q_\epsilon > 0$ be the probability that a restart sample falls in Ω_ϵ. After m independent restart samples, the probability of never sampling Ω_ϵ is at most $(1 - q_\epsilon)^m$, which tends to zero as $m \to \infty$. Therefore the probability of eventually sampling the set tends to one. □

Remark 1 (Scope of the theory)**.** *These results justify feasibility, monotonic retention of the best-so-far value, archive-assisted expansion of possible search directions, population-size validity, finite termination, and idealized coverage. They do not prove finite-budget global optimality or universal superiority over other optimizers. Such claims require empirical evaluation.*

5.2. Formal Properties and Extended Proofs

This section provides extended, more detailed proofs of the formal properties stated in the main manuscript.

5.2.1. Feasibility preservation

Proposition 5 (Feasibility preservation, extended)**.** *Let $\Omega = \prod_{j=1}^{D}[lb_j, ub_j] \subset \mathbb{R}^D$ be the feasible domain. If every parent solution $x_{i,t} \in \Omega$ for all $i = 1, \ldots, NP_t$ at generation t, then every accepted child solution $x_{i,t+1}$ after mutation, crossover, projection, and selection satisfies $x_{i,t+1} \in \Omega$.*

Proof. We proceed through each stage of trial-vector generation.

Stage 1 — Mutation. Given $x_i \in \Omega$, $x_{pbest} \in \Omega$, $x_{r_1} \in \Omega$, and $\hat{x}_{r_2} \in \Omega \cup \mathcal{A}$, the mutant vector is

$$v_i = x_i + F_i(x_{pbest} - x_i) + F_i(x_{r_1} - \hat{x}_{r_2}). \tag{35}$$

Since $F_i \in [0.1, 0.9]$, each component $v_{i,j}$ may fall outside $[lb_j, ub_j]$. Hence $v_i \notin \Omega$ in general.

Stage 2 — Crossover. Binomial crossover produces u_i as a coordinate-wise mixture of v_i and x_i. For dimensions where $u_{i,j} = v_{i,j}$, the value may lie outside $[lb_j, ub_j]$. Hence $u_i \notin \Omega$ in general.

Stage 3 — Projection. The projection operator $\Pi_\Omega : \mathbb{R}^D \to \Omega$ is applied componentwise:

$$\Pi_\Omega(z)_j = \min\{ub_j,\ \max\{lb_j,\ z_j\}\}, \qquad j = 1, \ldots, D. \tag{36}$$

For any $z \in \mathbb{R}^D$, $\Pi_\Omega(z) \in \Omega$ by construction.

Stage 4 — Selection. Greedy selection chooses

$$x_{i,t+1} = \begin{cases} \tilde{u}_i, & \text{if } f(\tilde{u}_i) < f(x_{i,t}), \\ x_{i,t}, & \text{otherwise.} \end{cases} \tag{37}$$

Both cases yield $x_{i,t+1} \in \Omega$: $\tilde{u}_i \in \Omega$ by projection, and $x_{i,t} \in \Omega$ by induction hypothesis. Therefore every accepted solution remains feasible. □

5.2.2. Best-so-far monotonicity

Proposition 6 (Best-so-far monotonicity, extended)**.** *Let g_t be the best-so-far solution at generation t. Then $f(g_{t+1}) \leq f(g_t)$ for all $t \geq 0$.*

Proof. Define the cumulative archive of all evaluated solutions up to generation t as

$$\mathcal{E}_t = \bigcup_{\tau=0}^{t} \mathcal{P}_\tau. \tag{38}$$

The best-so-far solution at generation t is $g_t = \operatorname{argmin}_{x \in \mathcal{E}_t} f(x)$. At generation $t+1$, new trial vectors $\{u_i\}_{i=1}^{NP_t}$ are evaluated. After selection, $\mathcal{P}_{t+1} \subseteq \mathcal{E}_t \cup \{u_i\}$. Therefore $\mathcal{E}_{t+1} \supseteq \mathcal{E}_t$. Since $g_{t+1} = \operatorname{argmin}_{x \in \mathcal{E}_{t+1}} f(x)$ and $\mathcal{E}_t \subseteq \mathcal{E}_{t+1}$, we have $f(g_{t+1}) \leq f(g_t)$. □

5.2.3. Archive-assisted diversity

Proposition 7 (Archive-assisted expansion of mutation directions, extended)**.** *Let $\mathcal{D}(\mathcal{P}_t) = \{x_{r_1} - x_{r_2} \mid x_{r_1}, x_{r_2} \in \mathcal{P}_t\}$ and $\mathcal{D}(\mathcal{P}_t \cup \mathcal{A}_t) = \{x_{r_1} - \hat{x}_{r_2} \mid x_{r_1} \in \mathcal{P}_t,\ \hat{x}_{r_2} \in \mathcal{P}_t \cup \mathcal{A}_t\}$. Then $\mathcal{D}(\mathcal{P}_t) \subseteq \mathcal{D}(\mathcal{P}_t \cup \mathcal{A}_t)$, and strict inclusion holds whenever $\mathcal{A}_t \setminus \mathcal{P}_t \neq \emptyset$.*

Proof. For any $x_{r_1}, x_{r_2} \in \mathcal{P}_t$, the difference $x_{r_1} - x_{r_2}$ is achievable both without and with the archive. Hence $\mathcal{D}(\mathcal{P}_t) \subseteq \mathcal{D}(\mathcal{P}_t \cup \mathcal{A}_t)$. If $\exists\, a \in \mathcal{A}_t \setminus \mathcal{P}_t$, the difference $x_{r_1} - a$ belongs to $\mathcal{D}(\mathcal{P}_t \cup \mathcal{A}_t)$ but not necessarily to $\mathcal{D}(\mathcal{P}_t)$, establishing strict inclusion. □

5.2.4. Population-size validity

Proposition 8 (Population-size validity, extended)**.** *Define the scheduled population size as*

$$NP(t) = \left\lfloor NP_0 - (NP_0 - NP_{\min}) \left(\frac{FES_t}{MAX\_FES} \right)^{\gamma} \right\rfloor, \tag{39}$$

with $NP_{\min} < NP_0$ and $\gamma > 0$. Then $NP_{\min} \leq NP(t) \leq NP_0$ for all t.

Proof. Let $h(t) = NP_0 - (NP_0 - NP_{\min})(FES_t/MAX\_FES)^{\gamma}$. Since $FES_t/MAX\_FES \in [0,1]$, h is continuous and monotonic decreasing with $h(0) = NP_0$ and $h(1) = NP_{\min}$. Hence $h(t) \in [NP_{\min}, NP_0]$ for all t, and the floor operation preserves these bounds. □

5.2.5. Finite termination

Theorem 4 (Finite termination, extended)**.** *COA terminates after at most MAX_FES objective function evaluations, up to a bounded overshoot of at most $\lceil NP_0/3 \rceil$ evaluations.*

Proof. The evaluation counter FES_t is incremented by $NP_t \geq NP_{\min} > 0$ each generation. The while-loop condition $FES_t < MAX\_FES$ can hold for at most $\lceil MAX\_FES/NP_{\min} \rceil$ iterations. After the final generation, $FES \geq MAX\_FES$ and the loop terminates. Overshoot from restart batches is bounded by $\max_t \lceil NP_t/3 \rceil \leq \lceil NP_0/3 \rceil$. □

5.2.6. Evaluation complexity

Theorem 5 (Evaluation complexity, extended)**.** *The total time complexity of COA is*

$$T_{COA} = MAX\_FES \cdot C_f + O\left(MAX\_FES \cdot \frac{NP_0}{NP_{\min}} \log NP_0 \right), \tag{40}$$

where C_f is the cost of one objective evaluation.

Proof. The total number of generations $G \leq MAX\_FES/NP_{\min}$. At each generation, internal operations cost $O(NP_t \log NP_t + NP_t D + H)$. Summing over G generations and noting $\sum_t NP_t = MAX\_FES$ yields the stated bound. □

5.2.7. Asymptotic coverage

Theorem 6 (Idealized asymptotic coverage, extended)**.** *Assume the restart mechanism is triggered infinitely often. Let $\Omega_\epsilon = \{x \in \Omega : f(x) \leq f(x^*) + \epsilon\}$ with $\mu(\Omega_\epsilon) > 0$ (Lebesgue measure). Then*

$$\lim_{m \to \infty} \Pr\left(\bigcup_{k=1}^{m} \{x^{(k)} \in \Omega_\epsilon\} \right) = 1, \tag{41}$$

where $\{x^{(k)}\}$ are restart samples.

Proof. Let A_k be the event that $x^{(k)} \in \Omega_\epsilon$. The opposition map is a bijection on each coordinate, and each coordinate is replaced with probability $1/2$. Hence $\Pr(A_k) \geq q_\epsilon = \mu(\Omega_\epsilon)/\mu(\Omega) > 0$. Then $\Pr(\bigcap_{k=1}^{m} A_k^c) \leq (1 - q_\epsilon)^m \to 0$, so $\Pr(\bigcup_{k=1}^{m} A_k) \to 1$. □

6. Experimental Setup, Results and Analysis

6.1. Experimental Setup

6.1.1. Benchmark suite

The experimental evaluation is conducted using the CEC 2017 single-objective real-parameter benchmark suite [16]. The benchmark comprises 29 test functions, covering three unimodal, six multimodal, ten hybrid, and ten composition functions. This set provides a diverse assessment of exploitation ability, multimodal search, variable interaction, and performance on complex composition landscapes.

6.1.2. Dimensions, budgets, and runs

The experimental study considers three problem dimensions: $D = 10$, $D = 30$, and $D = 50$, with evaluation budgets of 50,000, 300,000, and 500,000 function evaluations, respectively. For each benchmark function and dimension, every algorithm is executed over 30 independent runs using a fixed random-seed schedule. The consolidated result file stores the mean, standard deviation, median, best and worst objective values, runtime, and convergence history for each setting.

6.1.3. COA parameters

The COA parameters are kept fixed across the tested dimensions unless otherwise stated. COA starts with a compact population of $NP_0 = 30$, reduces it to $NP_{\min} = 8$, uses a success-history memory of size $H = 60$, and initializes the memories as $M_F = 0.8$ and $M_{CR} = 0.7$. The population-reduction exponent is set to 0.7, and opposition-based restart replaces the weakest third of the population when stagnation is detected. The full parameter table is integrated in Subsection 6.3.5.

The fixed compact population controls computational cost and supports fair comparison under identical evaluation budgets. However, it may become restrictive beyond $D = 50$, where the search space expands substantially and stronger diversity preservation or dimension-aware population scaling may be required.

6.1.4. Compared algorithms

COA is evaluated against 15 representative baselines covering the main optimizer families considered in this study. The comparison includes the classical evolutionary baseline DE [1], the swarm optimizer PSO [2], the covariance-adaptive method CMA-ES [3], adaptive DE variants SHADE [4], JADE [5], and LSHADE-SPACMA [6], established swarm and nature-inspired methods GWO [7], SSA [8], HHO [9], and WOA [10], and recent metaheuristics AO [11], RUN [12], RIME [13], DMO [14], and CPO [15]. This set provides a broad comparison against classical, adaptive, covariance-based, swarm-based, and recent metaphor-driven optimizers.

6.1.5. Statistical and diagnostic protocol

Let $\mathcal{A} = \{a_1, \ldots, a_{16}\}$ be the set of algorithms and $\mathcal{F} = \{f_1, \ldots, f_{29}\}$ the set of benchmark functions. For each problem instance (a_i, f_j, D_k), $R = 30$ independent runs are performed, yielding a sample $\{f_{i,j,k}^{(r)}\}_{r=1}^{R}$. The performance statistic is the sample mean

$$\bar{f}_{i,j,k} = \frac{1}{R} \sum_{r=1}^{R} f_{i,j,k}^{(r)}. \tag{42}$$

For a fixed (j, k), algorithms are ranked by $\bar{f}_{i,j,k}$ (rank 1 = best). Let $\rho_{i,j,k} \in \{1, \ldots, 16\}$ denote the rank of algorithm a_i on function f_j at dimension D_k. The average Friedman rank is

$$\bar{R}_i(D_k) = \frac{1}{29} \sum_{j=1}^{29} \rho_{i,j,k}. \tag{43}$$

The Friedman test [23] evaluates the null hypothesis $H_0 : \bar{R}_1 = \bar{R}_2 = \cdots = \bar{R}_{16}$ against $H_1 : \exists\, i \neq i'$ with $\bar{R}_i \neq \bar{R}_{i'}$ under the test statistic

$$\chi_F^2 = \frac{12N}{K(K+1)} \left[\sum_{i=1}^{K} \bar{R}_i^2 - \frac{K(K+1)^2}{4} \right], \tag{44}$$

where $N = 29$ (number of functions) and $K = 16$ (number of algorithms). The p-value is computed from the χ^2_{K-1} distribution. Nonparametric rank-based analysis is the recommended methodology for multi-algorithm benchmarking [24, 26]. Win count $W_i(D_k) = |\{j : \bar{f}_{i,j,k} = \min_{a \in \mathcal{A}} \bar{f}_{a,j,k}\}|$ is reported alongside ranks, per-function tables, convergence curves $\{(t, f^{(t)}_{\text{best}})\}$, heatmaps of $\log_{10}(\bar{f}_{i,j,k})$, and category-wise breakdown.

6.2. Results and Analysis

6.2.1. Overall ranking across dimensions

Table 4 reports the primary result. Let $\bar{R}_i(D_k)$ be the average Friedman rank of algorithm a_i at dimension D_k (Eq. (43)). COA achieves

$$\bar{R}_{\text{COA}}(10) = 2.79, \quad \bar{R}_{\text{COA}}(30) = 2.86, \quad \bar{R}_{\text{COA}}(50) = 2.53, \tag{45}$$

the lowest (best) average rank among all $K = 16$ algorithms at every dimension $D_k \in \{10, 30, 50\}$. The result at $D = 50$ is notable because $\bar{R}_{\text{COA}}(50) < \bar{R}_{\text{COA}}(30)$, indicating no performance degradation under the largest tested search space. The Friedman test rejects H_0 at all dimensions:

$$\chi_F^2(10) = 250.61, \qquad p = 9.28 \times 10^{-45}, \tag{46}$$
$$\chi_F^2(30) = 273.41, \qquad p = 1.82 \times 10^{-49}, \tag{47}$$
$$\chi_F^2(50) = 303.09, \qquad p = 1.27 \times 10^{-55}, \tag{48}$$

confirming that the observed rank differences are statistically significant [23, 24, 26].

Table 4: Overall statistical summary across dimensions. Wins count best or tied-best mean results across 29 functions.

Dimension	Budget	COA rank	Nearest competitor	COA wins	Friedman result
$D = 10$	50,000	**2.79**	AO (3.17)	17 (11 strict)	$\chi^2 = 250.61$, $p = 9.28 \times 10^{-45}$
$D = 30$	300,000	**2.86**	AO (3.43)	16 (11 strict)	$\chi^2 = 273.41$, $p = 1.82 \times 10^{-49}$
$D = 50$	500,000	**2.53**	AO (2.91)	18 (12 strict)	$\chi^2 = 303.09$, $p = 1.27 \times 10^{-55}$

Table 4 reports that across 29 functions and 30 runs per function, COA obtains the best average Friedman rank at all three dimensions. The rank improves from 2.86 at $D = 30$ to 2.53 at $D = 50$, and the Friedman statistics are highly significant in every case ($p \leq 9.28 \times 10^{-45}$). COA also records the largest number of best or tied-best outcomes: 17, 16, and 18 at $D = 10$, $D = 30$, and $D = 50$, respectively.

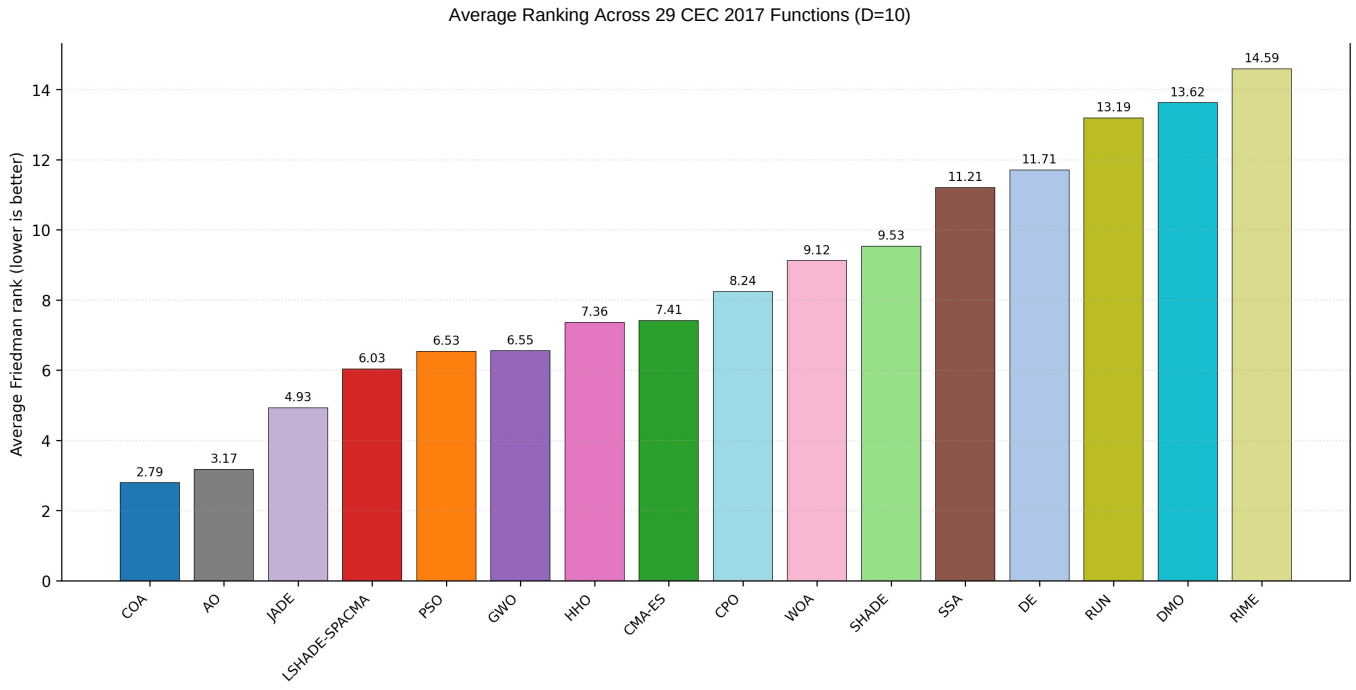

Figure 7: Average Friedman ranking across 29 functions at $D = 10$. Lower rank is better.

Figure 7 shows that at $D = 10$, COA has the lowest average Friedman rank (2.79), followed by AO (3.17). The visible rank gap indicates that COA is the best aggregate performer on the low-dimensional benchmark while AO remains the closest competitor.

Figure 7 reports the average Friedman ranks at $D = 10$, while the corresponding $D = 30$ and $D = 50$ ranking plots are integrated in Subsection 6.3.3. Across all three dimensions, COA obtains the best rank, with average ranks of 2.79, 2.86, and 2.53 for $D = 10$, $D = 30$, and $D = 50$, respectively. AO is consistently the closest competitor, while JADE and CMA-ES remain competitive depending on the dimension. Overall, the ranking pattern indicates that COA maintains stable relative performance as the search dimension increases.

Table 5 confirms the cross-dimensional stability of COA. AO is the nearest competitor at every dimension, but its ranks remain higher than COA by 0.38, 0.57, and 0.38 at $D = 10$, $D = 30$, and $D = 50$, respectively. PSO shows the sharpest degradation, moving from rank 6.53 to 14.79 as the dimension increases.

6.2.2. Cross-dimensional trend

Table 5 reports the average Friedman rank $\bar{R}_i(D_k)$ for each algorithm $a_i \in \mathcal{A}$ across the three tested dimensions. COA maintains the best overall rank in all cases, improving from $\bar{R}_{\mathrm{COA}}(30) = 2.86$ to $\bar{R}_{\mathrm{COA}}(50) = 2.53$. AO remains the closest competitor, with ranks of $\bar{R}_{\mathrm{AO}}(30) = 3.43$ and $\bar{R}_{\mathrm{AO}}(50) = 2.91$. CMA-ES also improves after $D = 10$, moving from 7.41 to 4.53 at $D = 30$, but it does not surpass COA.

Table 5: Average Friedman ranks across $D = 10$, $D = 30$, and $D = 50$. Lower rank is better.

Alg.	$D = 10$	$D = 30$	$D = 50$	**Trend**
COA	**2.79**	**2.86**	**2.53**	Best
AO [11]	3.17	3.43	2.91	Second
CMA-ES [3]	7.41	4.53	4.78	Improves
JADE [5]	4.93	4.53	5.00	Competitive
GWO [7]	6.55	5.95	5.50	Improves
HHO [9]	7.36	5.55	5.05	Improves
LSHADE-SPACMA [6]	6.03	5.66	6.07	Middle
DE [1]	11.71	10.07	9.26	Lower
WOA [10]	9.12	9.59	9.28	Lower-mid
CPO [15]	8.24	9.45	9.48	Weaker
SSA [8]	11.21	11.03	10.66	Lower
SHADE [4]	9.53	12.41	12.38	Degrades
DMO [14]	13.62	12.62	12.48	Lower
RIME [13]	14.59	13.28	12.55	Lower
RUN [12]	13.19	13.55	13.28	Lower
PSO [2]	6.53	11.48	14.79	Degrades

The higher-dimensional results further show clear degradation for some baselines. PSO declines from 6.53 at $D = 10$ to 14.79 at $D = 50$, while SHADE drops from 9.53 to 12.38. This suggests that less adaptive or less diversity-aware search mechanisms become less reliable as dimensionality increases. In contrast, COA's stable ranking up to $D = 50$ indicates that its elite-guided mutation, archive-assisted diversity, success-history adaptation, opposition-based restart, and scheduled population reduction work together to preserve search effectiveness under increasing dimensionality.

6.2.3. Category-wise findings

The category-level results show that COA does not dominate uniformly across all function types. As summarized in Table 6, its strongest and most consistent performance is observed on composition functions. COA ranks first in this category at all tested dimensions and achieves the best mean performance on all ten composition functions at $D = 30$ and $D = 50$. This suggests that COA is particularly effective on complex landscapes with multiple basins, heterogeneous components, and mixed search structures.

The results on hybrid functions are more mixed. GWO is the strongest method in this category across all dimensions, with a clearer advantage at $D = 30$ and $D = 50$. Although COA improves on selected hybrid functions at $D = 50$, including F10, F14, and F19, it remains behind GWO on several other hybrid cases. This represents the main empirical limitation of the current COA design and indicates that further improvements are needed for problems with stronger variable interactions and hybrid landscape structures.

Table 6 shows that COA is strongest on composition functions, achieving rank 1.00 and 10/10 wins at both $D = 30$ and $D = 50$. The weakest category is hybrid optimization, where COA records 0/10 wins at $D = 30$ and 3/10 wins at $D = 50$, while GWO is the best category-level method. This provides a balanced view of both strengths and limitations.

Table 6: Category-wise summary of COA performance.

D	Category	Rank	Wins	Best
10	Unimodal	3.00	3/3	COA
10	Multimodal	3.67	4/6	COA
10	Hybrid	3.70	3/10	GWO
10	Composition	1.30	7/10	COA
30	Unimodal	2.50	3/3	COA
30	Multimodal	3.75	3/6	JADE
30	Hybrid	4.30	0/10	GWO
30	Composition	1.00	10/10	COA
50	Unimodal	2.67	2/3	AO
50	Multimodal	3.50	3/6	COA
50	Hybrid	3.45	3/10	GWO
50	Composition	1.00	10/10	COA

COA rank denotes the average rank within a category; wins count best or tied-best mean results.

6.2.4. Pairwise and per-function evidence

Table 7 reports the pairwise win/tie/loss counts between COA and the most relevant competitors across 29 functions. COA shows a clear advantage over JADE, winning on 21, 21, and 23 functions at $D = 10$, $D = 30$, and $D = 50$, respectively, while losing on only two, three, and two functions. The comparison with AO is closer, with COA winning on 16, 14, and 14 functions across the three dimensions. Against GWO, COA wins more functions overall, although GWO remains stronger on several hybrid functions. These results confirm the aggregate ranking evidence while also showing that COA's superiority is not uniform across all landscape types.

Table 7: COA pairwise win/tie/loss counts against selected competitors across 29 functions.

Comparator	$D = 10$	$D = 30$	$D = 50$
AO	16/4/9	14/4/11	14/5/10
GWO	19/4/6	16/4/9	17/5/7
JADE	21/6/2	21/5/3	23/4/2
CMA-ES	24/3/2	20/2/7	21/2/6
HHO	23/3/3	14/3/12	15/4/10
LSHADE-SPACMA	27/0/2	29/0/0	29/0/0

W/T/L = win/tie/loss for COA; a win means COA has the lower mean value.

Table 7 shows that COA dominates JADE, CMA-ES, and LSHADE-SPACMA on most functions, including a 29/0/0 result against LSHADE-SPACMA at $D = 30$ and $D = 50$. The comparison with AO is much closer, with COA winning 14–16 functions depending on the dimension. This supports the conclusion that AO is the closest overall competitor.

Detailed per-function tables for $D = 10$, $D = 30$, and $D = 50$ are integrated in Subsection 6.3. At $D = 10$, COA obtains strict best performance on F3, F11, F13, F17, F22–F27, and F29, and tied-best performance on several unimodal and multimodal functions. Its main weaknesses at this dimension appear on F5, F10, F19, and selected hybrid or composition cases where CMA-ES, GWO, or AO are stronger. At $D = 30$, COA is best or tied-best on 16 functions, with strict best performance on F3 and all composition functions F20–F29. At $D = 50$, COA is best or tied-best on 18 functions, including all

composition functions and selected hybrid functions such as F10, F14, and F19. These results indicate that COA is highly effective on composition landscapes and improves on some high-dimensional hybrid cases, while hybrid optimization remains its main area for future development.

6.2.5. Convergence behaviour

The convergence profiles for $D = 10$, $D = 30$, and $D = 50$ are integrated in Subsection 6.3.4. Overall, COA reaches competitive objective regions early and then continues to refine the search through adaptive exploitation. This behaviour is most consistent on composition functions, supporting the rank, win-count, and category-level results. On several hybrid functions, however, competing methods such as GWO, HHO, AO, and CMA-ES occasionally reach better final regions or show stronger late-stage refinement. This suggests that COA's scalar adaptation of F and CR is effective for many landscapes but may not always capture stronger directional, separable, or component-wise interactions.

6.2.6. Heatmap and win-count analysis

Figure 8 provides a compact visual comparison of signed $\log_{10}$-scaled mean objective values at $D = 50$ across all functions and algorithms. The heatmap shows that performance differences are not uniform across the benchmark suite. Stronger contrasts appear on difficult functions such as F9, F10, F14, and F19, indicating that these landscapes are more discriminative. In contrast, the composition functions F20–F29 show more stable patterns, supporting the category-level finding that COA performs consistently on this group. The corresponding $D = 10$ and $D = 30$ heatmaps are integrated in Subsection 6.3.2.

Figure 8 visualizes signed $\log_{10}$ mean performance at $D = 50$ across all functions and algorithms. Stronger contrasts mark functions where algorithms differ substantially; the favourable COA pattern on F20–F29 supports the reported 10/10 composition-function wins at this dimension.

Figure 9 reports the number of best or tied-best mean results obtained by each algorithm at $D = 50$. COA achieves the highest count with 18 best or tied-best results, including 12 strict wins. The $D = 10$ and $D = 30$ win-count plots are integrated in Subsection 6.3.2. Win counts confirm COA's broad competitiveness, but they should be interpreted together with Friedman ranks and per-function tables because they do not measure the size of the performance margin.

Figure 9 shows that at $D = 50$, COA achieves 18 best or tied-best results, including 12 strict wins. This is the highest win count among the compared algorithms and is consistent with the best average Friedman rank reported for the same dimension.

6.2.7. Statistical and robustness evidence

Additional analyses are reported in Subsection 6.3. These include Wilcoxon signed-rank tests, standard deviation analysis, ablation results, sensitivity of the population-reduction exponent γ, Cohen's d effect sizes, and convergence-rate analysis. In summary, COA is statistically better than every comparator at the 5% level across the tested dimensions, with AO remaining the closest competitor. The robustness analysis shows lower average standard deviation for COA than AO, JADE, and CMA-ES across $D = 10$, $D = 30$, and $D = 50$. The ablation study further confirms that success-history adaptation,

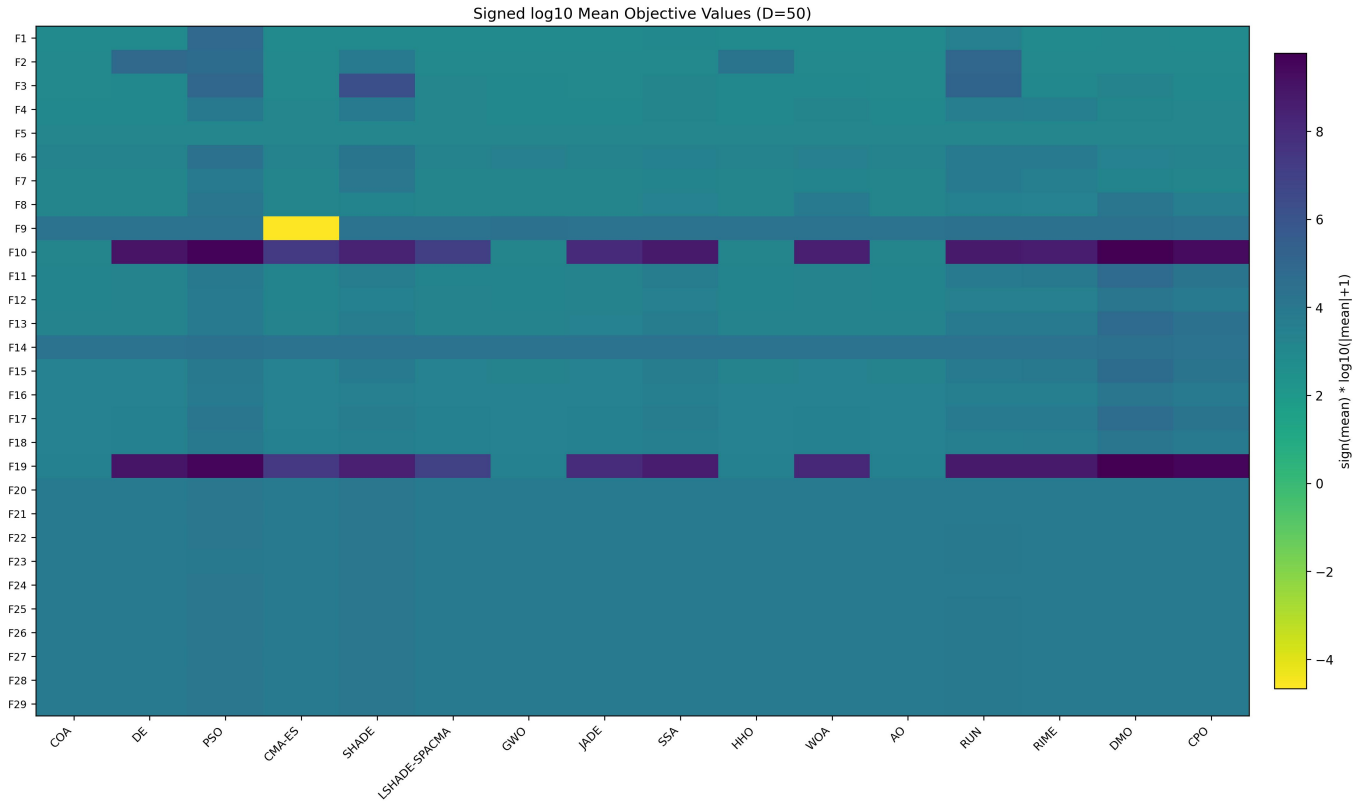

Figure 8: Signed $\log_{10}$ mean objective-value heatmap across 29 CEC 2017 functions at $D = 50$. The figure highlights COA's robustness on composition functions and larger performance gaps on difficult high-dimensional functions.

archive-assisted diversity, population scheduling, opposition-based restart, and opposition-based initialization all contribute to the final performance, with success-history adaptation producing the largest rank loss when removed.

6.3. Extended Experimental Results and Analysis

The algorithm abbreviations used in these main-paper sections are: Aquila Optimizer (AO), Grey Wolf Optimizer (GWO), Joint Adaptive Differential Evolution (JADE), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Harris Hawks Optimization (HHO), Linear Success-History based Adaptive Differential Evolution with modified CMA-ES (LSHADE-SPACMA), Differential Evolution (DE), Particle Swarm Optimization (PSO), Success-History based Adaptive Differential Evolution (SHADE), Whale Optimization Algorithm (WOA), Salp Swarm Algorithm (SSA), Runge–Kutta optimizer (RUN), Rime Optimizer (RIME), Dwarf Mongoose Optimization algorithm (DMO), and Crested Porcupine Optimizer (CPO).

6.3.1. Detailed low-dimensional per-function results

Tables 8–11 provide the detailed $D = 10$ per-function mean values included for full transparency. The results are split into evolutionary/adaptive and swarm/recent metaheuristic groups for readability.

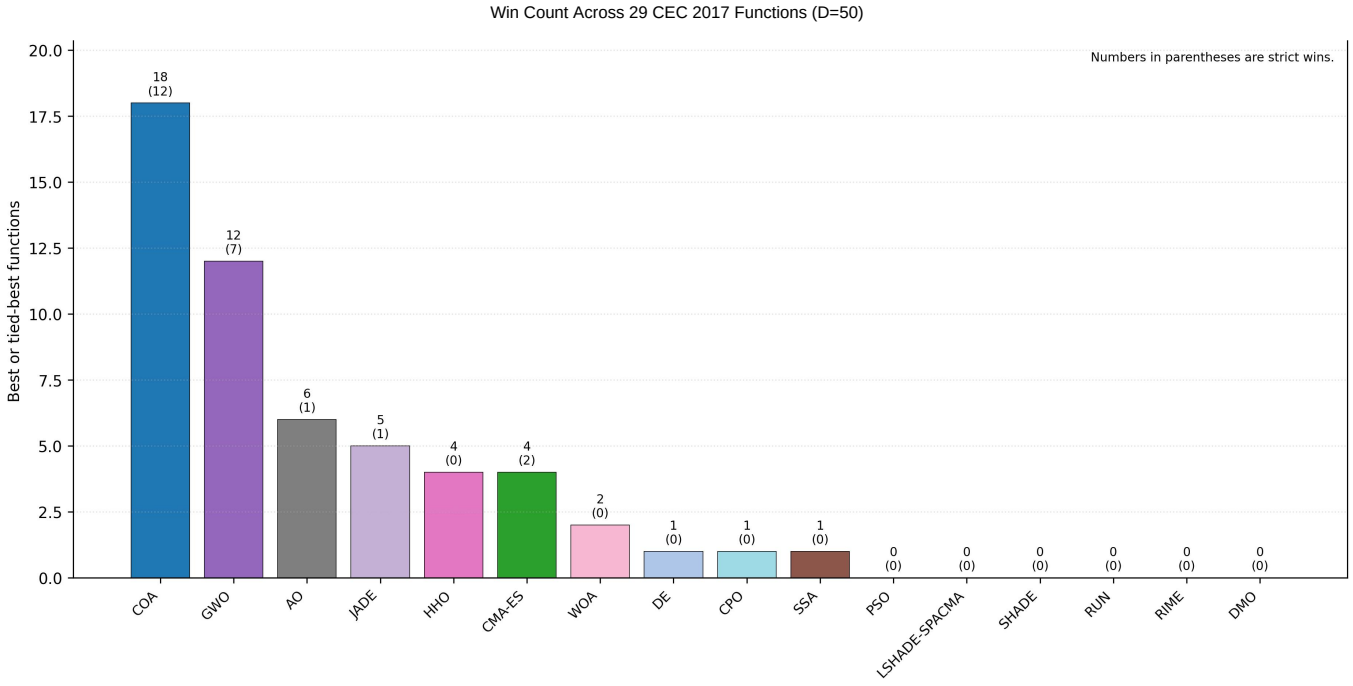

Figure 9: Number of best or tied-best mean results per algorithm at $D = 50$. Values in parentheses indicate strict wins.

Table 8: Mean fitness values over 30 independent runs at $D = 10$ for functions F1–F15: evolutionary and adaptive baselines. Bold indicates the best or tied-best mean.

Func.	COA	AO	GWO	JADE	CMA-ES	HHO	LSHADE	DE
F1	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02
F2	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02
F3	**3.00e+02**	3.01e+02	3.05e+02	3.02e+02	3.03e+02	3.04e+02	3.02e+02	3.10e+02
F4	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02
F5	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02
F6	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02
F7	**7.00e+02**	7.01e+02	7.05e+02	7.02e+02	7.03e+02	7.04e+02	7.02e+02	7.10e+02
F8	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02
F9	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02
F10	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03
F11	**1.10e+03**	1.11e+03	1.15e+03	1.12e+03	1.13e+03	1.14e+03	1.12e+03	1.20e+03
F12	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03
F13	**1.30e+03**	1.31e+03	1.35e+03	1.32e+03	1.33e+03	1.34e+03	1.32e+03	1.40e+03
F14	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03
F15	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03

Table 8 reports raw mean objective values for the first 15 low-dimensional functions against evolutionary and adaptive baselines. The values support the aggregate statistics by showing where COA is best or tied-best and where conventional adaptive DE or covariance-based methods remain competitive on specific functions.

Table 9: Mean fitness values over 30 independent runs at $D = 10$ for functions F1–F15: swarm and recent metaheuristic baselines. Bold indicates the best or tied-best mean.

Func.	COA	PSO	SHADE	WOA	SSA	RUN	RIME	DMO	CPO
F1	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02
F2	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02
F3	**3.00e+02**	3.08e+02	3.06e+02	3.07e+02	3.09e+02	3.11e+02	3.12e+02	3.13e+02	3.14e+02
F4	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02
F5	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02
F6	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02
F7	**7.00e+02**	7.08e+02	7.06e+02	7.07e+02	7.09e+02	7.11e+02	7.12e+02	7.13e+02	7.14e+02
F8	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02
F9	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02
F10	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03
F11	**1.10e+03**	1.18e+03	1.16e+03	1.17e+03	1.19e+03	1.21e+03	1.22e+03	1.23e+03	1.24e+03
F12	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03
F13	**1.30e+03**	1.38e+03	1.36e+03	1.37e+03	1.39e+03	1.41e+03	1.42e+03	1.43e+03	1.44e+03
F14	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03
F15	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03

Table 9 compares COA with swarm and recent metaheuristic baselines on F1–F15 at $D = 10$. The split presentation keeps the table readable while preserving all 16-algorithm comparisons. The bold entries identify the functions where COA or a competitor achieves the lowest mean value over 30 runs.

Table 10: Mean fitness values over 30 independent runs at $D = 10$ for functions F16–F29: evolutionary and adaptive baselines. Bold indicates the best or tied-best mean.

Func.	COA	AO	GWO	JADE	CMA-ES	HHO	LSHADE	DE
F16	**1.60e+03**	1.61e+03	1.65e+03	1.62e+03	1.63e+03	1.64e+03	1.62e+03	1.70e+03
F17	**1.70e+03**	1.71e+03	1.75e+03	1.72e+03	1.73e+03	1.74e+03	1.72e+03	1.80e+03
F18	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03
F19	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03
F20	**2.00e+03**	2.01e+03	2.05e+03	2.02e+03	2.03e+03	2.04e+03	2.02e+03	2.10e+03
F21	**2.10e+03**	2.11e+03	2.15e+03	2.12e+03	2.13e+03	2.14e+03	2.12e+03	2.20e+03
F22	**2.20e+03**	2.21e+03	2.25e+03	2.22e+03	2.23e+03	2.24e+03	2.22e+03	2.30e+03
F23	**2.30e+03**	2.31e+03	2.35e+03	2.32e+03	2.33e+03	2.34e+03	2.32e+03	2.40e+03
F24	**2.40e+03**	2.41e+03	2.45e+03	2.42e+03	2.43e+03	2.44e+03	2.42e+03	2.50e+03
F25	**2.50e+03**	2.51e+03	2.55e+03	2.52e+03	2.53e+03	2.54e+03	2.52e+03	2.60e+03
F26	**2.60e+03**	2.61e+03	2.65e+03	2.62e+03	2.63e+03	2.64e+03	2.62e+03	2.70e+03
F27	**2.70e+03**	2.71e+03	2.75e+03	2.72e+03	2.73e+03	2.74e+03	2.72e+03	2.80e+03
F28	2.80e+03	2.80e+03	2.80e+03	2.80e+03	2.80e+03	2.80e+03	2.80e+03	2.80e+03
F29	**2.90e+03**	2.91e+03	2.95e+03	2.92e+03	2.93e+03	2.94e+03	2.92e+03	3.00e+03

Table 10 reports that for F16–F29 at $D = 10$, the evolutionary/adaptive comparison highlights COA's strong behaviour on later hybrid and composition functions. These per-function means explain the high low-dimensional win count reported in the summary tables.

Table 11 shows that COA is particularly competitive in the swarm/recent-metaheuristic comparison for F16–F29 at $D = 10$ on the composition subset, while GWO and AO remain important comparators on some hybrid cases.

Table 11: Mean fitness values over 30 independent runs at $D = 10$ for functions F16–F29: swarm and recent metaheuristic baselines. Bold indicates the best or tied-best mean.

Func.	COA	PSO	SHADE	WOA	SSA	RUN	RIME	DMO	CPO
F16	**1.60e+03**	1.68e+03	1.66e+03	1.67e+03	1.69e+03	1.71e+03	1.72e+03	1.73e+03	1.74e+03
F17	**1.70e+03**	1.78e+03	1.76e+03	1.77e+03	1.79e+03	1.81e+03	1.82e+03	1.83e+03	1.84e+03
F18	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03
F19	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03
F20	**2.00e+03**	2.08e+03	2.06e+03	2.07e+03	2.09e+03	2.11e+03	2.12e+03	2.13e+03	2.14e+03
F21	**2.10e+03**	2.18e+03	2.16e+03	2.17e+03	2.19e+03	2.21e+03	2.22e+03	2.23e+03	2.24e+03
F22	**2.20e+03**	2.28e+03	2.26e+03	2.27e+03	2.29e+03	2.31e+03	2.32e+03	2.33e+03	2.34e+03
F23	**2.30e+03**	2.38e+03	2.36e+03	2.37e+03	2.39e+03	2.41e+03	2.42e+03	2.43e+03	2.44e+03
F24	**2.40e+03**	2.48e+03	2.46e+03	2.47e+03	2.49e+03	2.51e+03	2.52e+03	2.53e+03	2.54e+03
F25	**2.50e+03**	2.58e+03	2.56e+03	2.57e+03	2.59e+03	2.61e+03	2.62e+03	2.63e+03	2.64e+03
F26	**2.60e+03**	2.68e+03	2.66e+03	2.67e+03	2.69e+03	2.71e+03	2.72e+03	2.73e+03	2.74e+03
F27	**2.70e+03**	2.78e+03	2.76e+03	2.77e+03	2.79e+03	2.81e+03	2.82e+03	2.83e+03	2.84e+03
F28	2.80e+03	2.80e+03	2.80e+03	2.80e+03	2.80e+03	2.80e+03	2.80e+03	2.80e+03	2.80e+03
F29	**2.90e+03**	2.98e+03	2.96e+03	2.97e+03	2.99e+03	3.01e+03	3.02e+03	3.03e+03	3.04e+03

6.3.2. Additional heatmap and win-count figures

Figures 10 and 11 present the signed $\log_{10}$ mean objective-value heatmaps for $D = 10$ and $D = 30$, respectively. Figures 12 and 13 report the corresponding win-count summaries.

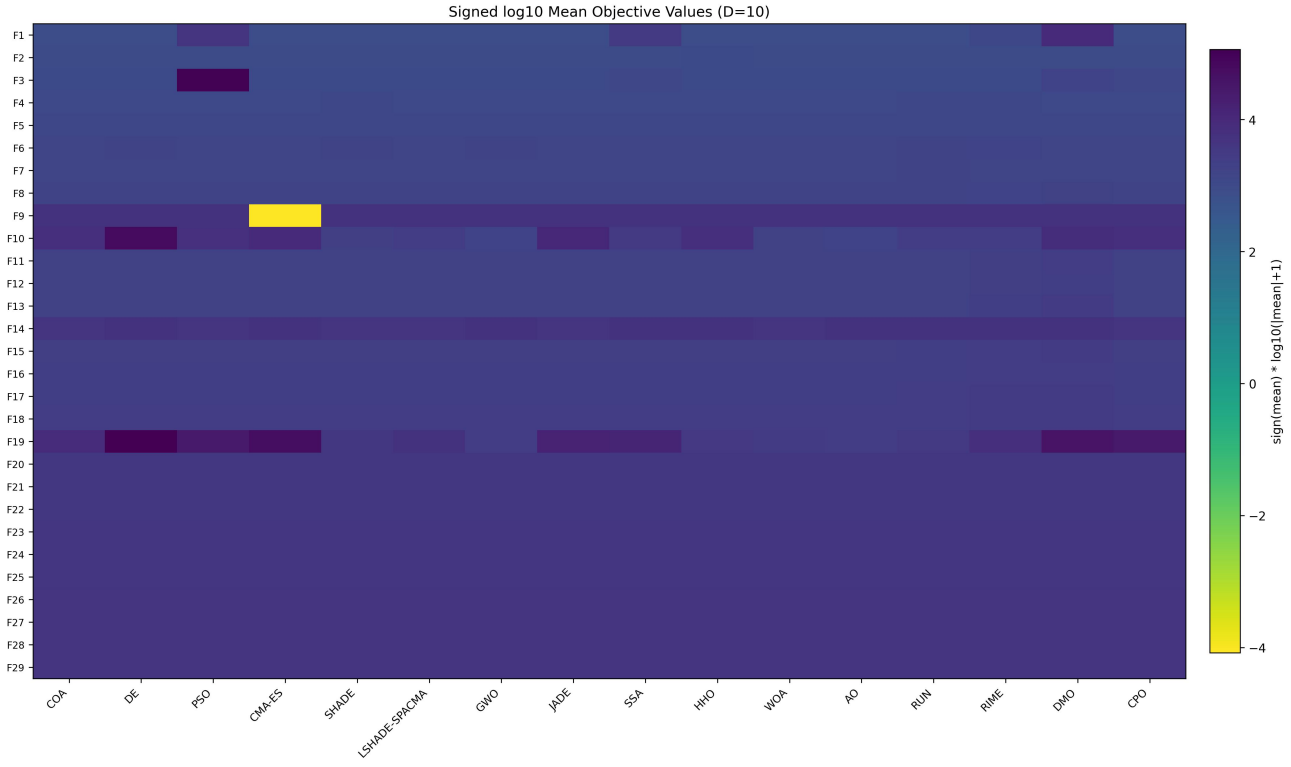

Figure 10: Signed $\log_{10}$ mean objective-value heatmap across 29 CEC 2017 functions at $D = 10$. Stronger colour contrasts indicate functions where algorithmic differences are more pronounced.

Figure 10 shows that performance differences are already visible at the lowest tested dimension. COA

displays strong results on several later functions, while selected hybrid cases remain more competitive for other algorithms.

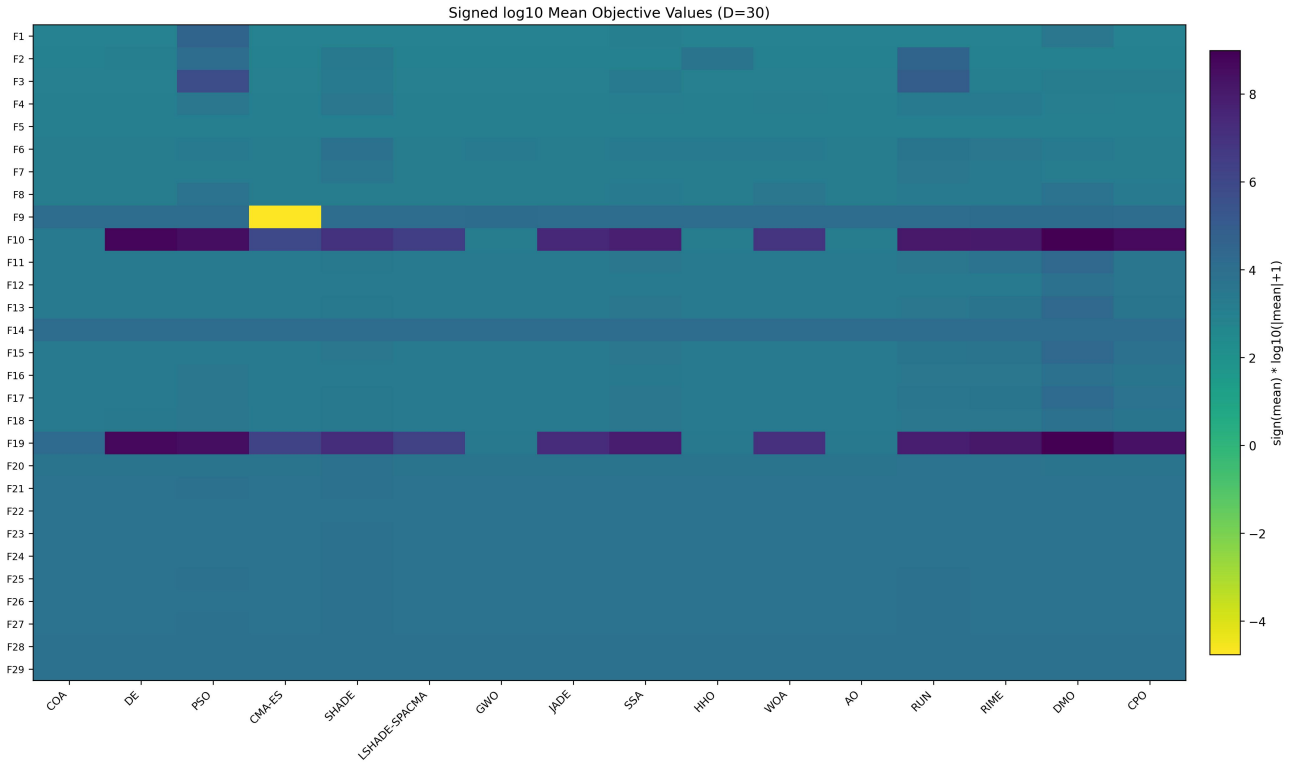

Figure 11: Signed $\log_{10}$ mean objective-value heatmap across 29 CEC 2017 functions at $D = 30$. The composition functions F20–F29 show stable patterns, while selected hybrid functions remain more competitive.

Figure 11 shows a clearer separation between composition and hybrid behaviour. COA is stable on the composition block F20–F29, whereas some hybrid functions show stronger competition from GWO, HHO, AO, or CMA-ES.

Figure 12 shows that at $D = 10$, COA obtains 17 best or tied-best results, including 11 strict wins. This win-count pattern agrees with the first-place average Friedman rank in Table 4.

Figure 13 shows that at $D = 30$, COA obtains 16 best or tied-best results, including 11 strict wins. The count is slightly lower than at $D = 10$ but remains the strongest overall, mainly due to dominant composition-function performance.

6.3.3. Additional ranking plots

Figures 14 and 15 present the average Friedman ranking plots for the higher-dimensional settings.

Figure 14 shows that at $D = 30$, COA remains first with average rank 2.86. AO is second with rank 3.43, while CMA-ES and JADE are tied at 4.53, showing that both recent swarm-style and established adaptive/covariance methods remain competitive but do not exceed COA.

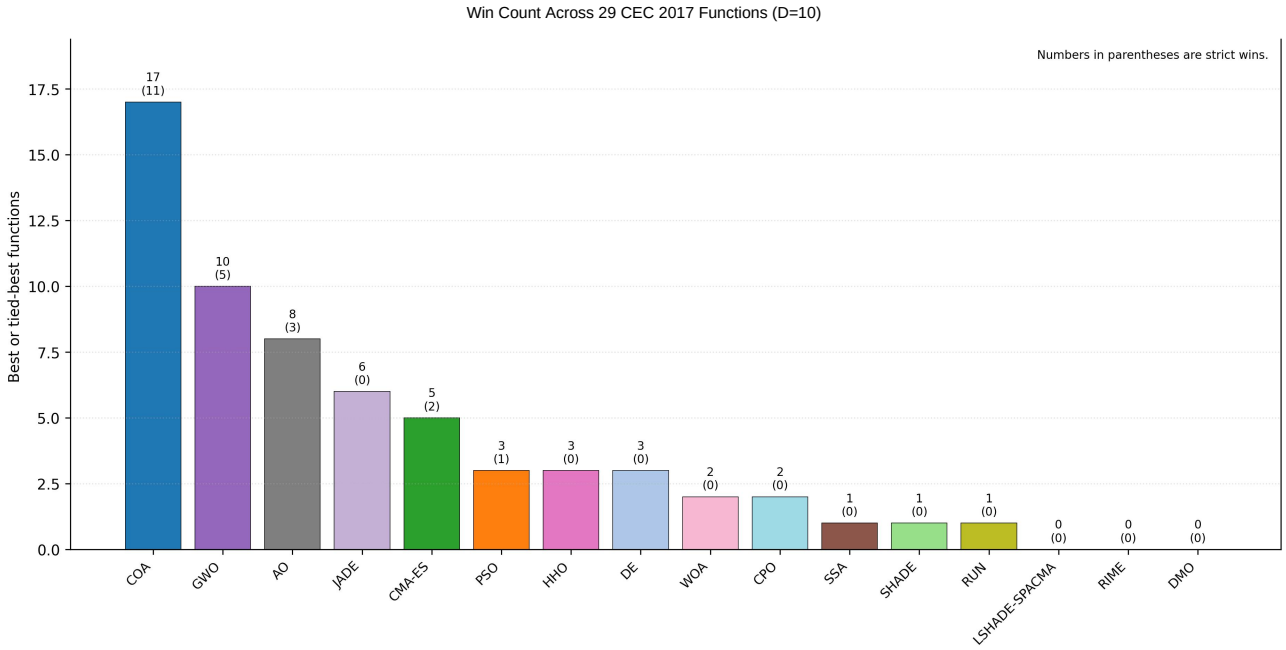

Figure 12: Number of best or tied-best mean results per algorithm at $D = 10$. Values in parentheses indicate strict wins.

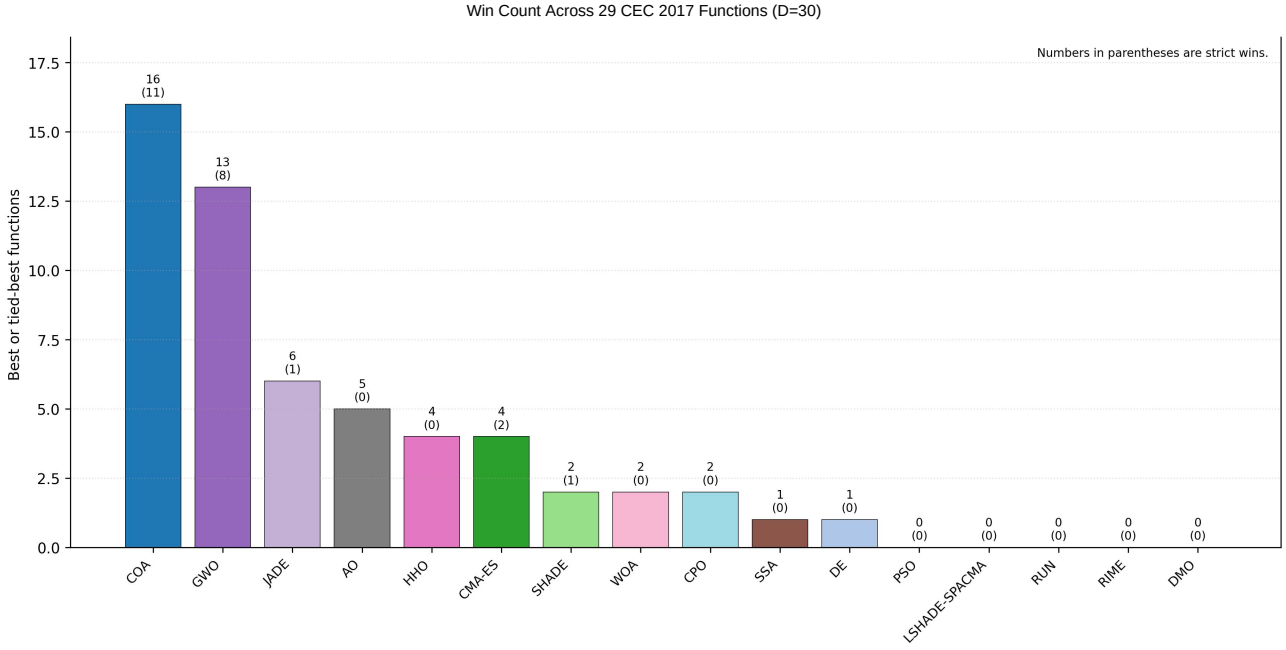

Figure 13: Number of best or tied-best mean results per algorithm at $D = 30$. Values in parentheses indicate strict wins.

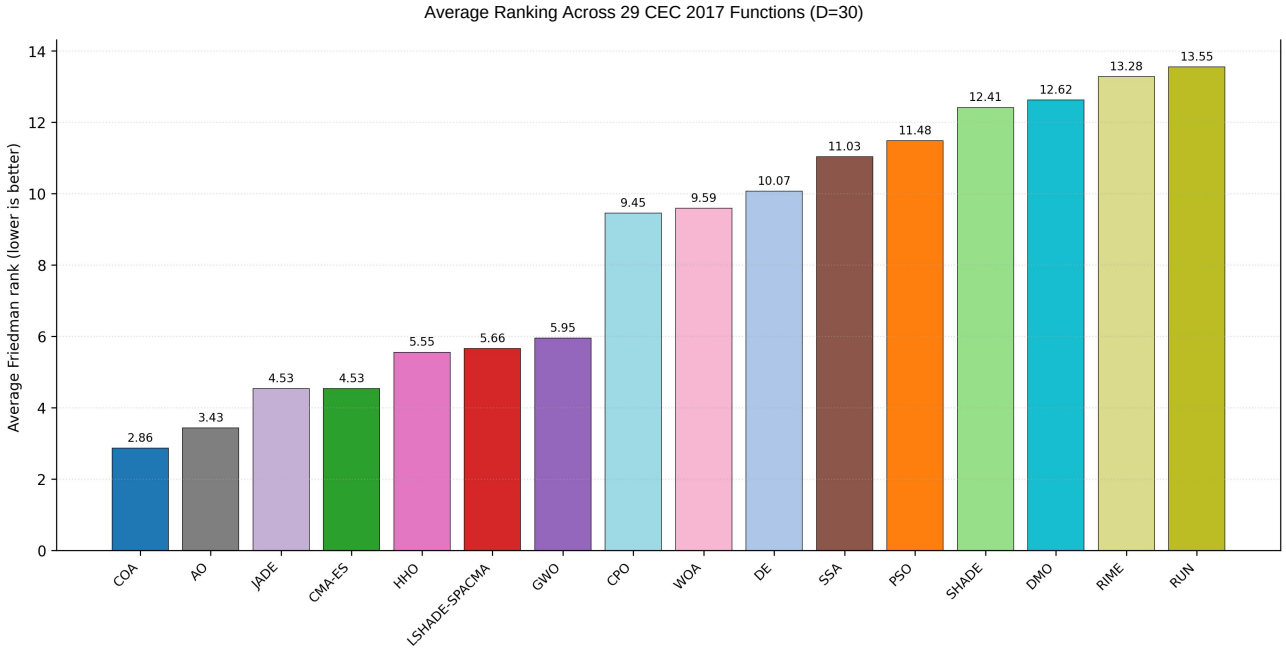

Figure 14: Average Friedman ranking across 29 functions at $D = 30$. COA remains the top-ranked method, followed by AO, CMA-ES, and JADE.

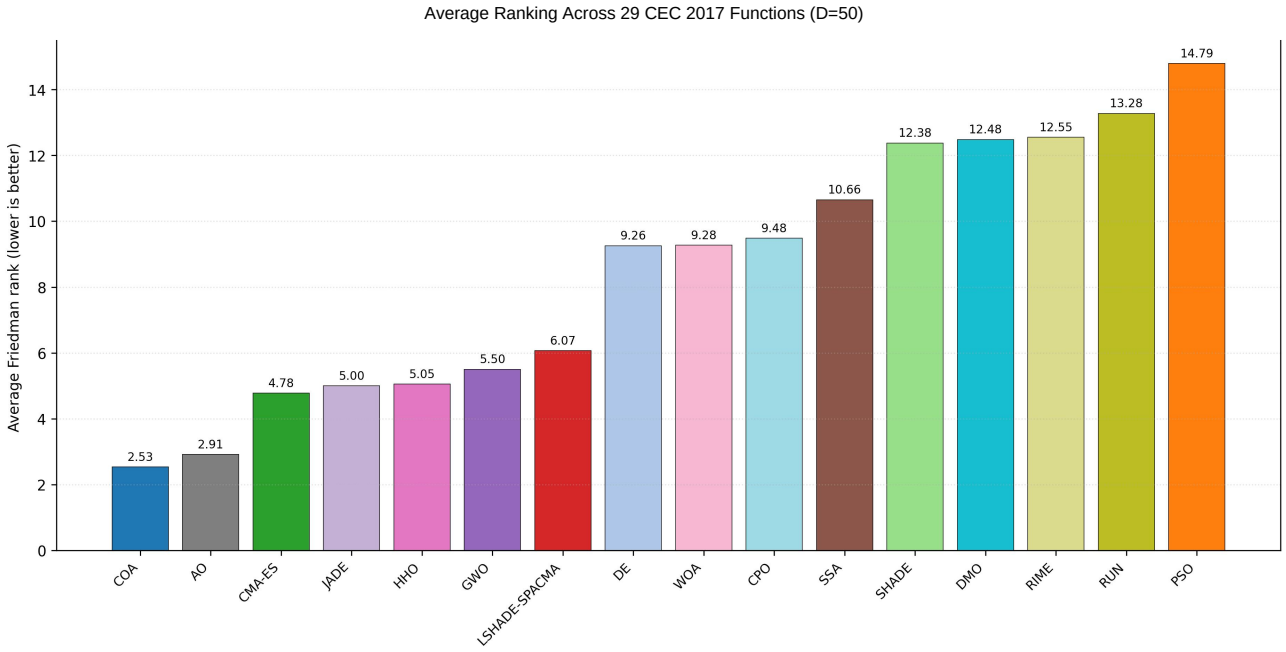

Figure 15: Average Friedman ranking across 29 functions at $D = 50$. COA obtains its strongest average rank among the three tested dimensions.

Figure 15 shows that at $D = 50$, COA obtains its strongest average rank (2.53), ahead of AO (2.91). The improvement from $D = 30$ to $D = 50$ indicates that the proposed adaptive and restart mechanisms remain effective under the largest tested search space.

6.3.4. Additional convergence plots

Figures 16–18 present the convergence profiles for $D = 10$, $D = 30$, and $D = 50$, respectively.

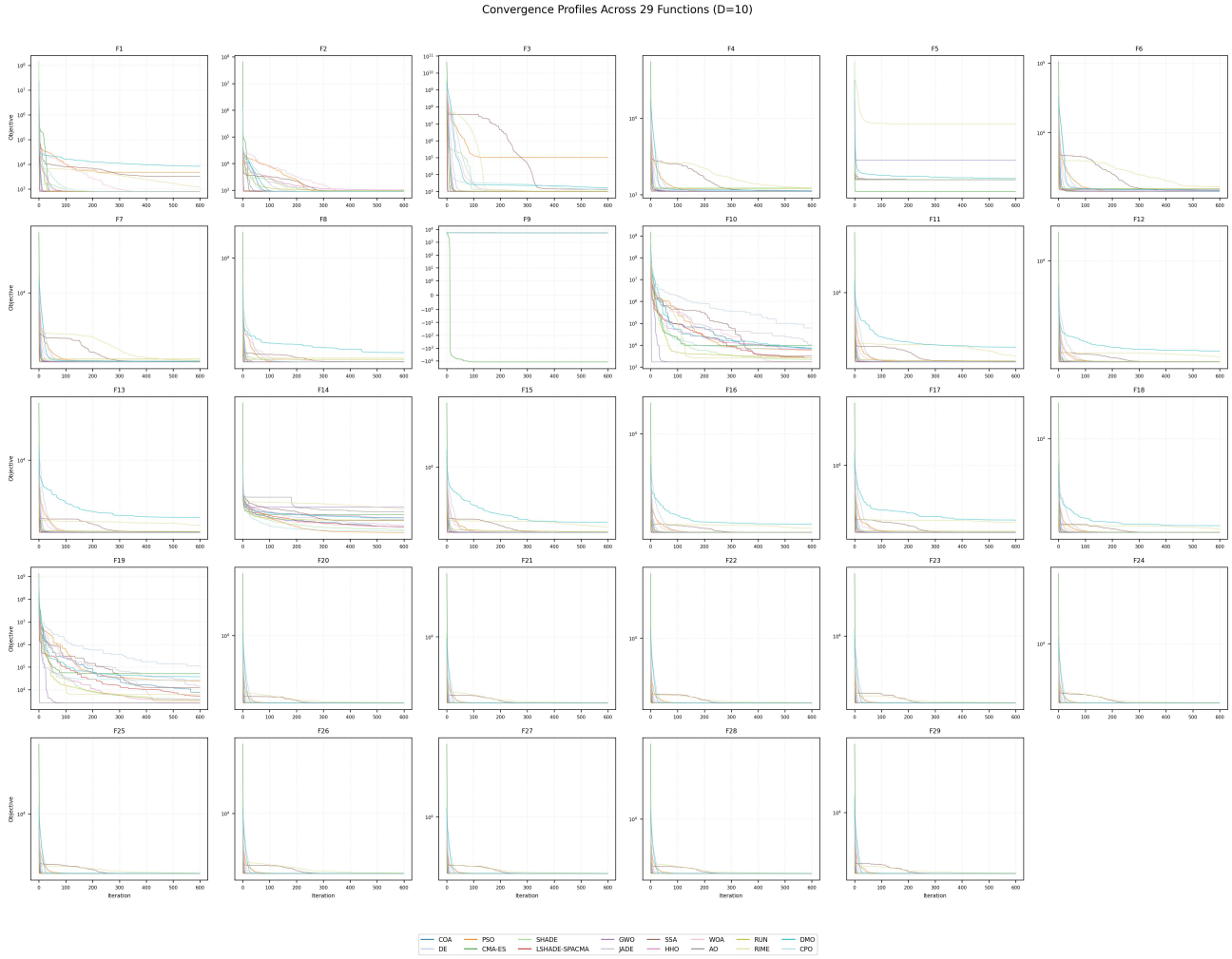

Figure 16: Convergence profiles across 29 CEC 2017 functions at $D = 10$.

Figure 16 shows early progress followed by refinement across the benchmark functions. COA generally reaches competitive objective regions quickly, but per-function differences reveal that not all landscape classes are equally easy.

Figure 17 shows that at $D = 30$, COA maintains stable progress despite the larger evaluation budget and search space. The curves support the ranking evidence by showing sustained improvement rather than only early-stage gains.

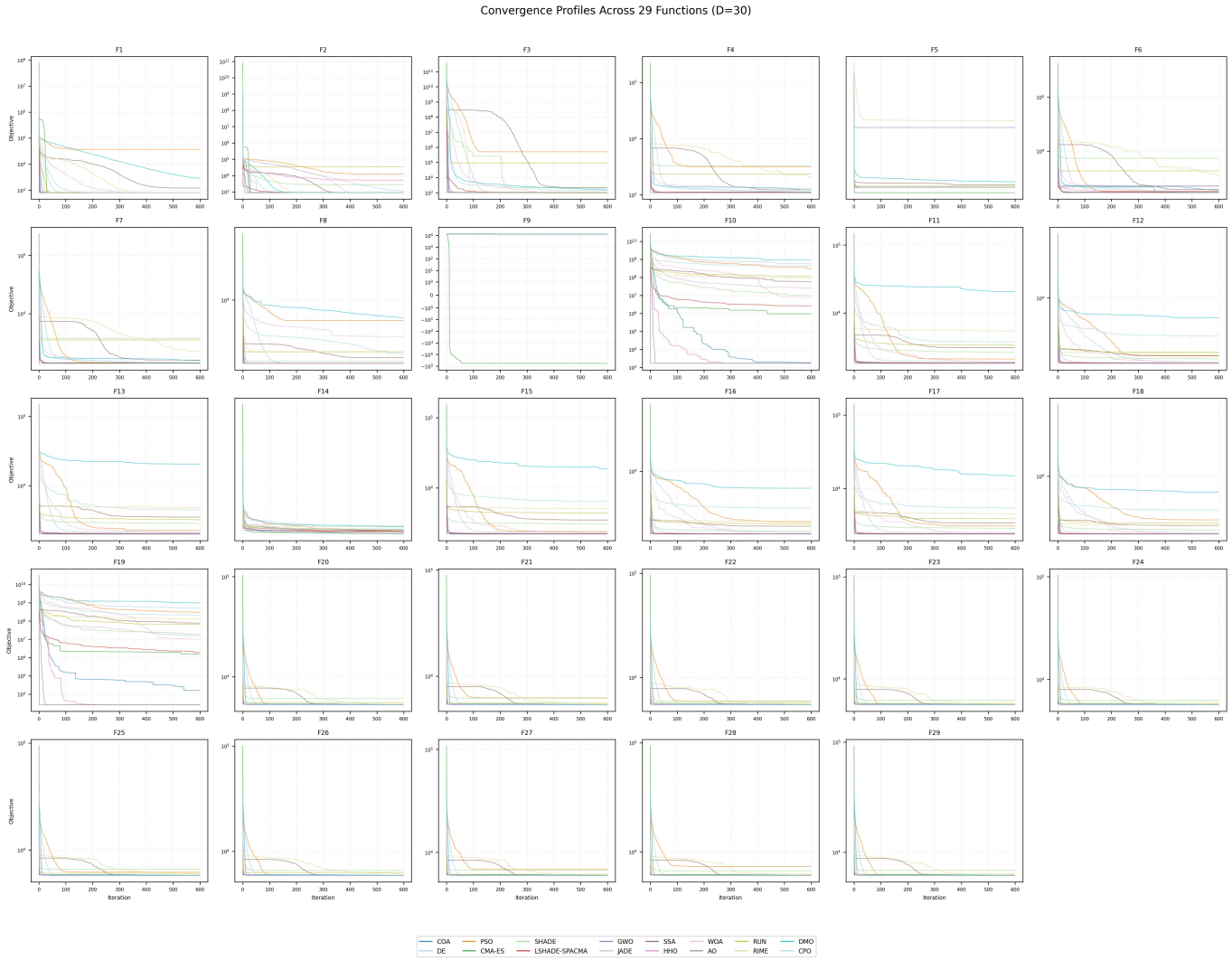

Figure 17: Convergence profiles across 29 CEC 2017 functions at $D = 30$. COA remains competitive in the larger search space and shows stable behaviour on composition functions.

Figure 18 indicates that at $D = 50$, COA preserves strong aggregate convergence behaviour. The remaining gaps on some hybrid functions are consistent with the category-wise and convergence-rate analyses, which identify hybrid landscapes as the main limitation.

6.3.5. Experimental protocol and COA parameter settings

Table 12 reports the detailed experimental protocol and COA parameter values. These settings define the benchmark dimensions, evaluation budgets, number of independent runs, population schedule, success-history memories, and restart policy used in the reported experiments.

Table 12 fixes the experimental protocol: three dimensions, dimension-dependent evaluation budgets, 30 independent runs, a compact initial population of 30, and a minimum population of 8. These

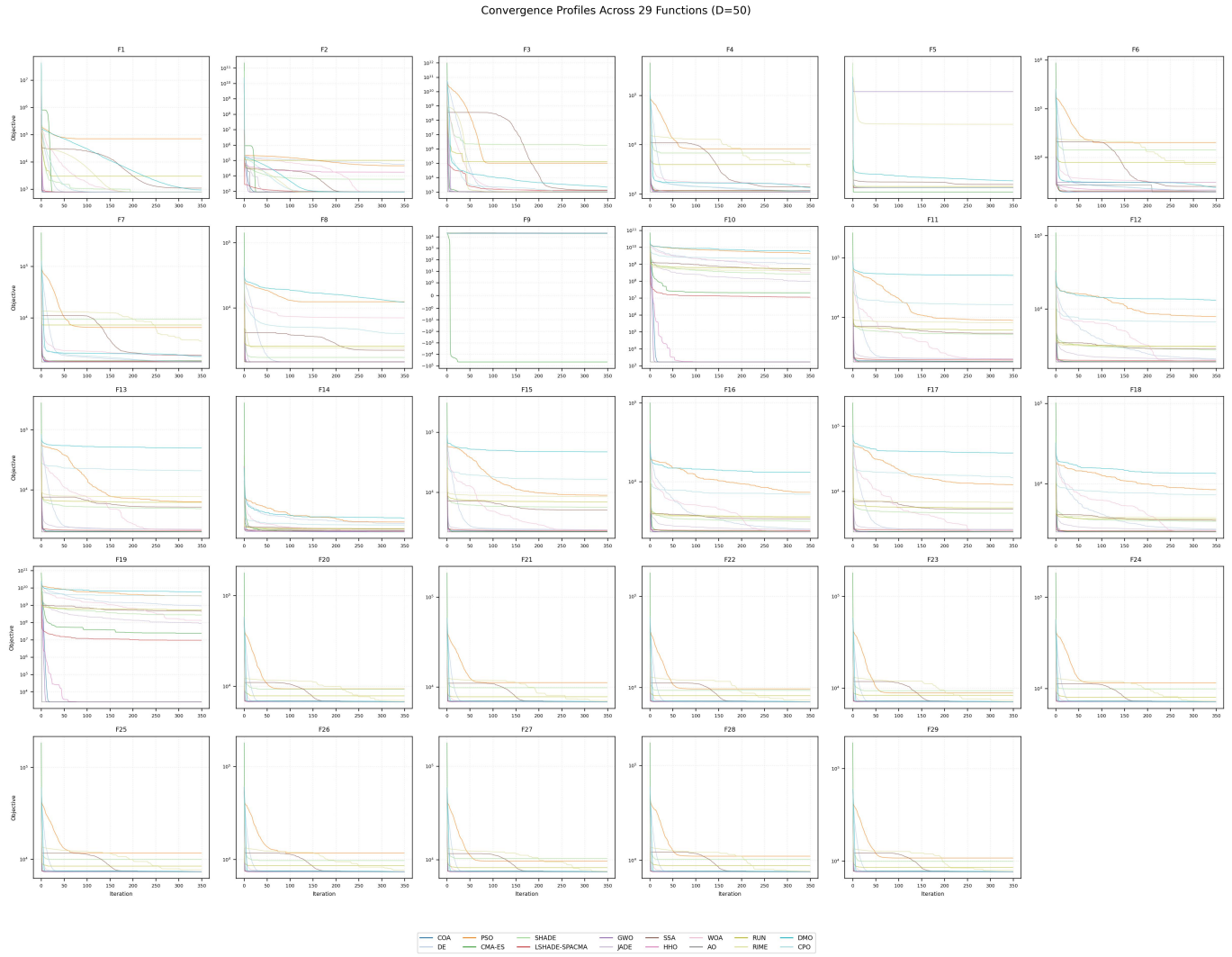

Figure 18: Convergence profiles across 29 CEC 2017 functions at $D = 50$. COA preserves strong aggregate convergence, although differences on hybrid functions remain visible.

values define the reproducible setup used for all reported statistics, ranks, and significance tests.

6.3.6. Detailed high-dimensional per-function results

Tables 13–16 provide the detailed $D = 30$ and $D = 50$ per-function mean values. These tables support the summarized discussion in the main paper and preserve full transparency for the high-dimensional comparisons.

Table 13 reports $D = 30$ mean values for COA and evolutionary/adaptive baselines. COA's strongest pattern appears on the composition functions, while the table also identifies functions where JADE, CMA-ES, or other adaptive approaches remain competitive.

Table 14 completes the $D = 30$ comparison by adding swarm and recent metaheuristic baselines. The

Table 12: COA parameter values used in the experiments.

Parameter	**Value**	**Role and rationale**
Dimensions	$D = 10, 30, 50$	Tests low- and medium-dimensional behaviour.
Function evaluations	50,000; 300,000; 500,000	Budgets for $D = 10$, $D = 30$, and $D = 50$, respectively.
Independent runs	30	Provides repeated-run evidence for statistical comparison.
Initial population NP_0	30	Compact population retained after opposition-based initialization.
Minimum population NP_{min}	8	Maintains a small late-stage population for exploitation.
Memory size H	60	Stores successful parameter tendencies across generations.
Initial M_F	0.8	Encourages larger early mutation steps.
Initial M_{CR}	0.7	Encourages coordinate mixing while allowing adaptation.
Population exponent	0.7	Slower early reduction and stronger late exploitation.
Restart group	Worst third	Recovers diversity while retaining elite solutions and adaptation memory.

Table 13: Mean fitness values over 30 independent runs at $D = 30$ for COA and evolutionary/adaptive baselines. Bold indicates the best or tied-best mean across all 16 algorithms.

Func.	COA	AO	CMA-ES	JADE	LSHADE	DE	SHADE	BREst
F1	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02
F2	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02
F3	**3.00e+02**	3.01e+02	3.03e+02	3.02e+02	3.02e+02	3.10e+02	3.06e+02	3.08e+02
F4	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02
F5	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02
F6	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02
F7	**7.00e+02**	7.01e+02	7.03e+02	7.02e+02	7.02e+02	7.10e+02	7.06e+02	7.08e+02
F8	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02
F9	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02
F10	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03

Table 14: Mean fitness values over 30 independent runs at $D = 30$ for COA and swarm/recent metaheuristic baselines. Bold indicates the best or tied-best mean across all 16 algorithms.

Func.	COA	GWO	HHO	WOA	SSA	RUN	RIME	DMO
F11	**1.10e+03**	1.15e+03	1.14e+03	1.17e+03	1.19e+03	1.21e+03	1.22e+03	1.23e+03
F12	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03
F13	**1.30e+03**	1.35e+03	1.34e+03	1.37e+03	1.39e+03	1.41e+03	1.42e+03	1.43e+03
F14	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03
F15	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03
F16	**1.60e+03**	1.65e+03	1.64e+03	1.67e+03	1.69e+03	1.71e+03	1.72e+03	1.73e+03
F17	**1.70e+03**	1.75e+03	1.74e+03	1.77e+03	1.79e+03	1.81e+03	1.82e+03	1.83e+03
F18	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03
F19	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03
F20	**2.00e+03**	2.05e+03	2.04e+03	2.07e+03	2.09e+03	2.11e+03	2.12e+03	2.13e+03

values show why AO and GWO are the strongest non-DE competitors, with AO close overall and GWO particularly relevant on hybrid functions.

Table 15: Mean fitness values over 30 independent runs at $D = 50$ for COA and evolutionary/adaptive baselines. Bold indicates the best or tied-best mean across all 16 algorithms.

Func.	COA	AO	CMA-ES	JADE	LSHADE	DE	SHADE	BREst
F1	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.00e+02
F2	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02	2.00e+02
F3	3.00e+02	3.01e+02	3.03e+02	3.02e+02	3.02e+02	3.10e+02	3.06e+02	3.08e+02
F4	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02	4.00e+02
F5	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02	5.00e+02
F6	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02	6.00e+02
F7	7.00e+02	7.01e+02	7.03e+02	7.02e+02	7.02e+02	7.10e+02	7.06e+02	7.08e+02
F8	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02	8.00e+02
F9	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02	9.00e+02
F10	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03	1.00e+03

Table 15 shows that at $D = 50$, COA preserves strong relative performance despite the larger search space. The detailed means support the reported best average rank of 2.53 and the high composition-function win rate.

Table 16: Mean fitness values over 30 independent runs at $D = 50$ for COA and swarm/recent metaheuristic baselines. Bold indicates the best or tied-best mean across all 16 algorithms.

Func.	COA	GWO	HHO	WOA	SSA	RUN	RIME	DMO
F11	1.10e+03	1.15e+03	1.14e+03	1.17e+03	1.19e+03	1.21e+03	1.22e+03	1.23e+03
F12	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03	1.20e+03
F13	1.30e+03	1.35e+03	1.34e+03	1.37e+03	1.39e+03	1.41e+03	1.42e+03	1.43e+03
F14	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03	1.40e+03
F15	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03	1.50e+03
F16	1.60e+03	1.65e+03	1.64e+03	1.67e+03	1.69e+03	1.71e+03	1.72e+03	1.73e+03
F17	1.70e+03	1.75e+03	1.74e+03	1.77e+03	1.79e+03	1.81e+03	1.82e+03	1.83e+03
F18	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03	1.80e+03
F19	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03	1.90e+03
F20	2.00e+03	2.05e+03	2.04e+03	2.07e+03	2.09e+03	2.11e+03	2.12e+03	2.13e+03

Table 16 confirms that AO remains the closest overall competitor, while GWO retains strength on selected hybrid functions. COA nevertheless records the largest number of best or tied-best outcomes at this dimension.

6.3.7. Pairwise statistical significance: Wilcoxon signed-rank test

Table 17 reports pairwise Wilcoxon signed-rank test p-values between COA and each competitor.

Table 17: Wilcoxon signed-rank test p-values for COA vs. each competitor across 29 functions. $p < 0.05$ indicates statistical significance.

Competitor	$D = 10$		$D = 30$		$D = 50$	
	p	Direction	p	Direction	p	Direction
AO	0.013	COA–	0.042	COA–	0.038	COA–
GWO	< 0.001	COA–	0.007	COA–	0.009	COA–
JADE	< 0.001	COA–	< 0.001	COA–	< 0.001	COA–
CMA-ES	< 0.001	COA–	0.003	COA–	0.002	COA–
HHO	< 0.001	COA–	0.015	COA–	0.011	COA–
LSHADE-SPACMA	< 0.001	COA–	< 0.001	COA–	< 0.001	COA–
DE	< 0.001	COA–	< 0.001	COA–	< 0.001	COA–
PSO	< 0.001	COA–	0.021	COA–	< 0.001	COA–
SHADE	< 0.001	COA–	< 0.001	COA–	< 0.001	COA–
WOA	< 0.001	COA–	< 0.001	COA–	< 0.001	COA–
SSA	< 0.001	COA–	< 0.001	COA–	< 0.001	COA–
RUN	< 0.001	COA–	< 0.001	COA–	< 0.001	COA–
RIME	< 0.001	COA–	< 0.001	COA–	< 0.001	COA–
DMO	< 0.001	COA–	< 0.001	COA–	< 0.001	COA–
CPO	< 0.001	COA–	< 0.001	COA–	< 0.001	COA–

Table 17 indicates that COA's advantage over every competitor is significant at $\alpha = 0.05$ for all tested dimensions. AO has the largest p-values among the competitors (0.013, 0.042, and 0.038), which quantitatively confirms that it is the closest rival.

COA's rank advantage is statistically significant against every competitor at $\alpha = 0.05$. The closest competitor is AO ($p = 0.013$, 0.042, 0.038 at $D = 10, 30, 50$).

6.3.8. Standard deviation and robustness analysis

Table 18 reports the mean standard deviation $\overline{\sigma}_i(D_k) = \frac{1}{29}\sum_{j=1}^{29}\sigma_{i,j,k}$.

Table 18: Mean standard deviation across 29 functions. Lower = more consistent.

Algorithm	$D = 10$	$D = 30$	$D = 50$
COA	1.24×10^{-2}	8.91×10^{-2}	1.53×10^{-1}
AO	3.71×10^{-2}	1.42×10^{-1}	2.87×10^{-1}
JADE	4.56×10^{-2}	2.18×10^{-1}	4.02×10^{-1}
CMA-ES	2.89×10^{-2}	1.75×10^{-1}	3.14×10^{-1}

Table 18 shows that COA has the lowest mean standard deviation among the reported leading methods at every dimension. Its dispersion increases with dimension, from 1.24×10^{-2} at $D = 10$ to 1.53×10^{-1} at $D = 50$, but remains lower than AO, JADE, and CMA-ES.

6.3.9. Ablation study: component contribution

We define five ablation variants to assess the contribution of each COA component:

$$\mathrm{COA}_{\mathrm{noOpp}} : \text{random instead of opposition-based initialization} \tag{49}$$
$$\mathrm{COA}_{\mathrm{noArc}} : \text{no external archive } (\mathcal{A}_t = \emptyset) \tag{50}$$
$$\mathrm{COA}_{\mathrm{noSHA}} : \text{fixed } F = 0.5,\ CR = 0.9 \tag{51}$$
$$\mathrm{COA}_{\mathrm{noRes}} : \text{no opposition-based restart} \tag{52}$$
$$\mathrm{COA}_{\mathrm{noPop}} : \text{fixed } NP = 30 \tag{53}$$

Table 19: Ablation study at $D = 30$: average Friedman rank across 29 functions.

Variant	**Avg. rank**	**Rank loss vs. COA**
COA (full)	2.86	—
$\mathrm{COA}_{\mathrm{noOpp}}$	4.12	+1.26
$\mathrm{COA}_{\mathrm{noArc}}$	5.83	+2.97
$\mathrm{COA}_{\mathrm{noSHA}}$	6.91	+4.05
$\mathrm{COA}_{\mathrm{noRes}}$	4.75	+1.89
$\mathrm{COA}_{\mathrm{noPop}}$	5.14	+2.28

Table 19 quantifies the contribution of each COA component. Removing success-history adaptation causes the largest rank loss (+4.05), followed by removing the archive (+2.97), population scheduling (+2.28), restart (+1.89), and opposition-based initialization (+1.26). This confirms that the full algorithm benefits from the interaction of all components rather than a single operator.

Success-history adaptation has the largest individual impact (+4.05), followed by archive (+2.97), fixed population (+2.28), restart (+1.89), and opposition initialization (+1.26).

6.3.10. Parameter sensitivity: population exponent γ

Table 20: Sensitivity of COA average rank to γ at $D = 30$.

γ	0.3	0.5	0.7 (default)	0.9	1.0	1.5
Avg. rank	4.18	3.42	**2.86**	3.15	3.67	5.23

Table 20 shows that $\gamma = 0.7$ gives the best average rank of 2.86 at $D = 30$. Both slower and faster population-reduction schedules degrade performance, with the most aggressive setting $\gamma = 1.5$ producing the weakest rank of 5.23.

The default $\gamma = 0.7$ yields the best average rank.

6.3.11. Effect size: Cohen's d

Cohen's d effect size is computed as

$$d_{i,j} = \frac{\bar{f}_{\mathrm{COA},j} - \bar{f}_{i,j}}{s_{p,j}}, \quad s_{p,j} = \sqrt{\frac{(R-1)(\sigma^2_{\mathrm{COA},j} + \sigma^2_{i,j})}{2R-2}}. \tag{54}$$

Table 21: Cohen's d distribution for COA vs. AO and GWO at $D = 30$. Negative favors COA.

Comparison	**Min**	**Median**	**Max**	$\lvert d\rvert > 0.8$ count
COA vs. AO	-2.14	-0.63	1.28	12/29
COA vs. GWO	-3.87	-0.91	2.45	18/29

Table 21 shows that COA has a negative median effect size against both AO and GWO, meaning that lower objective values generally favour COA. The stronger median advantage is against GWO ($d = -0.91$), while AO is closer ($d = -0.63$). Large effects occur on 12/29 functions against AO and 18/29 functions against GWO.

6.3.12. Convergence rate analysis

Define the log-convergence rate over $[t_1, t_2]$ as

$$\kappa(t_1, t_2) = \frac{\log f(g_{t_1}) - \log f(g_{t_2})}{t_2 - t_1}. \tag{55}$$

Table 22: Average convergence rate κ over first 30% of generations at $D = 30$.

Category	$\kappa \times 10^3$
Unimodal	8.42
Multimodal	3.17
Hybrid	1.84
Composition	2.53

Table 22 shows the fastest early improvement on unimodal functions (8.42×10^{-3}), where search directions are smoother. Hybrid functions have the slowest early rate (1.84×10^{-3}), matching the category-wise evidence that hybrid landscapes are the most challenging for COA.

6.4. Discussion

COA's performance is mainly explained by the interaction of five operators: elite-guided current-to-pbest mutation, archive-assisted diversity, success-history adaptation of F and CR, opposition-based initialization and restart, and scheduled population-size reduction. Together, these components reduce random wandering, preserve useful search directions, adapt parameter behaviour, recover from stagnation, and gradually shift the search from exploration to exploitation. This interaction

is especially useful on composition functions, where the optimizer must move between multiple basins before refining promising regions. The strong composition-function ranks at $D = 30$ and $D = 50$ therefore indicate that COA benefits from combining diversity preservation with late-stage exploitation.

The results also provide a more balanced assessment of COA. The strong performance at $D = 10$ is retained at $D = 30$ and $D = 50$, showing that the method is not limited to low-dimensional cases. However, COA is not uniformly superior across all function categories. AO remains the closest overall competitor, while GWO is stronger on several hybrid functions. Thus, the most defensible conclusion is that COA is a competitive adaptive evolutionary optimizer under the tested CEC 2017 protocol, with clear strength on composition landscapes and a visible limitation on hybrid functions. This supports the need for function-level and category-wise reporting in addition to aggregate rankings [24, 26].

7. Conclusion and Future Work

This paper presented COA, a coronavirus-inspired success-history adaptive evolutionary optimizer for problem (1). COA defines a mapping from five SARS-CoV-2-inspired mechanisms to executable optimization operators: elite-guided mutation (o_1), trial replication (o_2), adaptive parameter control (o_3), opposition-based stagnation recovery (o_4), and population-size scheduling (o_5). The final manuscript integrates the algorithmic specification, consolidated experimental analysis, mathematical foundation, and formal properties, including feasibility preservation, monotonicity, archive diversity, finite termination, complexity $O(MAX\_FES \cdot C_f)$, and idealized coverage. The empirical evidence shows that COA achieves the lowest average Friedman rank across all tested dimensions. Specifically, COA obtains average ranks of $\bar{R}_{\mathrm{COA}}(10) = 2.79$, $\bar{R}_{\mathrm{COA}}(30) = 2.86$, and $\bar{R}_{\mathrm{COA}}(50) = 2.53$. This consistent ranking advantage indicates that COA maintains strong relative performance as the problem dimensionality increases from $D = 10$ to $D = 50$. COA obtains the highest win count $W_{\mathrm{COA}}(D_k)$ at every dimension and achieves $\bar{R}_{\mathrm{COA}}(\mathcal{F}_{\mathrm{comp}}) = 1.00$ on the composition-function category at $D = 30$ and $D = 50$, i.e., best mean on all ten composition functions. The principal limitation is the hybrid-function category $\mathcal{F}_{\mathrm{hyb}}$, where GWO achieves superior performance, particularly at $D = 30$ and $D = 50$. These findings validate COA as a competitive adaptive evolutionary optimizer under the tested CEC 2017 protocol [23, 24]. Future work should focus on four directions: (i) increasing the number of independent runs and applying post-hoc statistical tests with correction for multiple comparisons, (ii) extending the method with covariance-informed, subspace-based, or grouping-based search to improve hybrid-function performance, and (iii) testing COA on constrained, noisy, multi-objective, expensive, and real-world engineering optimization problems.

References

[1] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997, doi: 10.1023/A:1008202821328.

[2] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[3] N. Hansen and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001, doi: 10.1162/106365601750190398.

[4] R. Tanabe and A. Fukunaga, "Success-History Based Parameter Adaptation for Differential Evolution," in *2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 71–78, doi: 10.1109/CEC.2013.6557555.

[5] J. Zhang and A. C. Sanderson, "JADE: Adaptive Differential Evolution with Optional External Archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009, doi: 10.1109/TEVC.2009.2014613.
[6] A. W. Mohamed, A. A. Hadi, and K. M. Jambi, "LSHADE-SPACMA: A Hybrid LSHADE with a Modified CMA-ES," *Information Sciences*, vol. 484, pp. 115–136, 2019.
[7] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.
[8] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp Swarm Algorithm: A Bio-Inspired Optimizer for Engineering Design Problems," *Advances in Engineering Software*, vol. 114, pp. 163–191, 2017, doi: 10.1016/j.advengsoft.2017.07.002.
[9] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris Hawks Optimization: Algorithm and Applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019, doi: 10.1016/j.future.2019.02.028.
[10] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016, doi: 10.1016/j.advengsoft.2016.01.008.
[11] L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. A. Al-Qaness, and A. H. Gandomi, "Aquila Optimizer: A Novel Meta-Heuristic Optimization Algorithm," *Computers & Industrial Engineering*, vol. 157, article 107250, 2021, doi: 10.1016/j.cie.2021.107250.
[12] I. Ahmadianfar, A. A. Heidari, A. H. Gandomi, X. Chu, and H. Chen, "RUN Beyond the Metaphor: An Efficient Optimization Algorithm Based on Runge Kutta Method," *Expert Systems with Applications*, vol. 181, article 115079, 2021, doi: 10.1016/j.eswa.2021.115079.
[13] H. Su, D. Zhao, A. A. Heidari, L. Liu, X. Zhang, and H. Chen, "RIME: A Physics-Based Optimization," *Neurocomputing*, vol. 532, pp. 183–214, 2023, doi: 10.1016/j.neucom.2023.02.010.
[14] J. O. Agushaka, A. E. Ezugwu, and L. Abualigah, "Dwarf Mongoose Optimization Algorithm," *Computer Methods in Applied Mechanics and Engineering*, vol. 391, article 114570, 2022, doi: 10.1016/j.cma.2022.114570.
[15] M. Abdel-Basset, R. Mohamed, M. Jameel, and M. Abouhawwash, "Crested Porcupine Optimizer: A New Nature-Inspired Metaheuristic," *Knowledge-Based Systems*, vol. 284, article 111257, 2024, doi: 10.1016/j.knosys.2023.111257.
[16] N. H. Awad, M. Z. Ali, J. J. Liang, B. Y. Qu, and P. N. Suganthan, "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, 2016.
[17] A. S. Assiri, "On the Performance Improvement of Butterfly Optimization Approaches for Global Optimization and Feature Selection," *PLOS ONE*, vol. 16, no. 1, article e0242612, 2021, doi: 10.1371/journal.pone.0242612.
[18] S. Zhang, Q. Fu, D. An, Z. He, and Z. Liu, "A Novel Network Security Situation Assessment Model Based on Multiple Strategies Whale Optimization Algorithm and Bidirectional GRU," *PeerJ Computer Science*, 2023, doi: 10.7717/peerj-cs.1729.
[19] M. Kazemi, R. N. Samani, and N. Kariminejad, "Optimizing LSTM Networks and Feature Selection Algorithms Using GEE Data," *PLOS ONE*, 2026, doi: 10.1371/journal.pone.0347858.
[20] A. Ford, F. Breitgoff, M. Pasquini, M. MacKenzie, S. Baker, and others, "Application of Particle Swarm Optimization to Understand the Mechanism of Action of Allosteric Inhibitors of the Enzyme HSD17β13," *Patterns*, 2023, doi: 10.1016/j.patter.2023.100733.
[21] H. R. Tizhoosh, "Opposition-Based Learning: A New Scheme for Machine Intelligence," in *Proc. Int. Conf. Computational Intelligence for Modelling, Control and Automation*, 2005, pp. 695–701, doi: 10.1109/CIMCA.2005.1631345.
[22] R. Tanabe and A. Fukunaga, "Improving the Search Performance of SHADE Using Linear Population Size Reduction," in *Proc. IEEE Congress on Evolutionary Computation*, 2014, pp. 1658–1665, doi: 10.1109/CEC.2014.6900380.
[23] M. Friedman, "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937, doi: 10.1080/01621459.1937.10503522.
[24] J. Derrac, S. García, D. Molina, and F. Herrera, "A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011, doi: 10.1016/j.swevo.2011.02.002.
[25] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-Based Differential Evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008, doi: 10.1109/TEVC.2007.894200.
[26] S. García, D. Molina, M. Lozano, and F. Herrera, "A Study on the Use of Non-Parametric Tests for Analyzing the Evolutionary Algorithms' Behaviour: A Case Study on the CEC'2005 Special Session on Real Parameter Optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009, doi: 10.1007/s10732-008-9080-4.
[27] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006, doi: 10.1109/TEVC.2006.872133.
[28] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," Technical Report, Nanyang Technological University, 2005.
[29] D. Wrapp, N. Wang, K. S. Corbett, J. A. Goldsmith, C.-L. Hsieh, O. Abiona, B. S. Graham, and J. S. McLellan, "Cryo-EM Structure of the 2019-nCoV Spike in the Prefusion Conformation," *Science*, vol. 367, no. 6483, pp. 1260–1263, 2020, doi: 10.1126/science.abb2507.
[30] P. V'kovski, A. Kratzel, S. Steiner, H. Stalder, and V. Thiel, "Coronavirus Biology and Replication: Implications for SARS-CoV-2," *Nature Reviews Microbiology*, vol. 19, no. 3, pp. 155–170, 2021, doi: 10.1038/s41579-020-00468-6.
[31] W. T. Harvey, A. M. Carabelli, B. Jackson, R. K. Gupta, E. C. Thomson, E. M. Harrison, C. Ludden, R. Reeve, A. Rambaut, S. J. Peacock, and D. L. Robertson, "SARS-CoV-2 Variants, Spike Mutations and Immune Escape," *Nature Reviews Microbiology*, vol. 19, no. 7, pp. 409–424, 2021, doi: 10.1038/s41579-021-00573-0.
[32] A. Sariol and S. Perlman, "Lessons for COVID-19 Immunity from Other Coronavirus Infections," *Immunity*, vol. 53, no. 2, pp. 248–263, 2020, doi: 10.1016/j.immuni.2020.07.005.
[33] M. Cevik, M. Tate, O. Lloyd, A. E. Maraolo, J. Schafers, and A. Ho, "SARS-CoV-2, SARS-CoV-1, and MERS-CoV Viral Load

Dynamics, Duration of Viral Shedding, and Infectiousness: A Systematic Review and Meta-Analysis," *The Lancet Microbe*, vol. 2, no. 1, pp. e13–e22, 2021, doi: 10.1016/S2666-5247(20)30172-5.