OMCDE: Opposition-based Multi-Stage Matrix-Based Competitive Differential Evolution

Mohammad Sarhangzadeh
Dept. of Computer Engineering
Bilkent University
Ankara, Turkey
m.sarhangzadeh@bilkent.edu.tr

Shahryar Rahnamayan
Department of Engineering
Brock University
St. Catharines, Canada
Srahnamayan@brocku.ca

Doruk Oner
Dept. of Computer Engineering
Bilkent University
Ankara, Turkey
doruk.oner@bilkent.edu.tr

***Abstract*—Differential Evolution has proven to be an effective algorithm for continuous global optimization, but it's modern high-performing variants often introduce additional complexity and computational cost. We propose OMCDE, a matrix-based competitive Differential Evolution algorithm that aims to improve optimization performance while maintaining computational efficiency. The proposed approach integrates competition-driven search dynamics, adaptive behavior between optimization stages, and opposition-based initialization within a parallelizable framework. Empirical results on widely used CEC 2017 functions show that OMCDE achieves strong performance compared to several state-of-the-art DE algorithms, while remaining efficient and scalable. These findings indicate that OMCDE is a promising alternative for solving complex numerical optimization problems.**

***Index Terms*—Differential Evolution, Evolutionary Computation, Global Numerical Optimization, Matrix-Based, Parallel Computing**

I. Introduction

Differential Evolution (DE) [1] is a powerful population-based stochastic optimizer known for its simplicity and effectiveness in solving global numerical optimization problems. A fundamental challenge in its design is balancing exploration and exploitation. Exploration refers to the algorithm's ability to search broad, unknown regions of the solution space to maintain diversity and avoid local optima. Conversely, exploitation focuses on refining the search within promising local neighborhoods to accelerate convergence toward the global optimum.

The development of L-SHADE [2] marked a significant shift in modern DE by introducing success-history-based parameter adaptation and linear population size reduction. Building on this, recent research increasingly focuses on utilizing individual-level evolutionary information rather than relying solely on population-wide settings [3], [4]. This information typically includes the specific fitness of an individual, its historical success in producing better offspring, and its relative standing within the current population. This shift allows algorithms to tailor search strategies to the unique landscape each candidate individual is currently navigating, adjusting parameters like the scaling factor (F) and crossover rate (CR) for every member of the population.

While these individual-level strategies improve search quality, they often come with a trade-off in speed. Many current methods require complex sorting, historical record-keeping, and sequential loops that make them computationally slow and difficult to scale, especially for high-dimensional problems. Furthermore, these methods often rely on fixed logic that may not adapt well as the search environment and population diversity change over time.

To address these challenges, we propose OMCDE (Opposition-based Matrix-Based Competitive Differential Evolution). Our approach adopts the triplet-competition mechanism introduced by TCDE [5], which mirrors the natural process where individuals within a population compete for survival. In this model, candidates are ranked within small groups and assigned mutation roles Best, Medium, or Worst to balance exploration and local refinement based on their competitive standing.

We further extend this by incorporating a stage-dependent mutation strategy, inspired by how biological organisms adapt not just to each other, but to the shifting pressures of their environment. In nature, survival strategies often become more aggressive and focused as resources reach their limit; similarly, OMCDE transitions from broad, diversified exploration to intense, greedy exploitation in the final stages of the search to refine the global optimum. By integrating this competitive logic with opposition-based initialization and a fully matrix-based framework, we enable parallel execution that significantly reduces runtime by eliminating computationally expensive nested loops.

We evaluated OMCDE on the CEC 2017 [6] benchmark suite, where it achieved performance levels near current state-of-the-art variants while remaining exceptionally fast[1]. The main contributions of this study are as follows:

1) A unified matrix-based architecture that reformulates core DE operators into vectorized computations, enabling simultaneous population-wide updates and high-speed execution.
2) A stage-dependent mutation scheme that adapts to the search progress, transitioning from diversified exploration to focused, greedy exploitation.
3) A framework utilizing triplet-based competition managing exploration and exploitation at the individual level.

[1]https://github.com/MSarhangzadeh/OMCDE-Optimization

II. Related Works

While the fundamental operators and standard evolutionary cycle of Differential Evolution (DE) are well-established in the literature [3], [4], this section focuses on the high-level paradigm shifts that define current state-of-the-art variants.

Modern research in DE was advanced by JADE [7], which introduced the DE/current-to-pbest/1 mutation strategy:

$$v_{i,g} = x_{i,g} + F \cdot (x_{pbest,g} - x_{i,g}) + F \cdot (x_{r1,g} - x_{r2,g}) \quad (1)$$

where $x_{pbest,g}$ is randomly selected from the top $p\%$ of the population. This scheme balances exploration and exploitation by using the vector $(x_{pbest,g} - x_{i,g})$ to pull the individual toward the best-known regions, while $(x_{r1,g} - x_{r2,g})$ maintains the stochastic search power needed to discover new areas. This foundation was solidified by L-SHADE [2], which added Linear Population Size Reduction to dynamically shrink the population as the search progresses. By reducing the number of individuals, L-SHADE effectively shifts the computational budget from broad exploration in early stages to intensive local refinement toward the end of the search.

Building on the success of L-SHADE, various descendants emerged to further exploit individual-level evolutionary information. Some, like iLSHADE-RSP [8], introduced stochastic perturbations using Cauchy distributions, leveraging "heavier tails" to enable large steps that help the population escape local optima. Others, most notably LSHADE-RSP [9], refined the selection process with Rank-Based Selective Pressure (RSP). By assigning higher selection probabilities to better-ranked individuals rather than treating them equally, RSP increases the greediness of the mutation operator, significantly accelerating convergence on complex landscapes.

The shift toward treating individuals differently based on their standing led to TCDE [5]. Inspired by natural competition, TCDE groups the population into random triplets where individuals are ranked as x_{best}, x_{medium}, and x_{worst}. Leveraging the adaptive parameter logic of L-SHADE, TCDE assigns specific control parameters (F and CR) to each member based on their role. These roles dictate the mutation strategy, ensuring a balance between local refinement and global exploration:

$$v_{best} = x_{best} + F_{best} \cdot (x_{r1} - x_{r2}) \quad (2)$$

$$\begin{aligned} v_{medium} = {} & x_{medium} + F_{med} \cdot (x_{best} - x_{medium}) \\ & + F_{med} \cdot (x_{r1} - x_{r2}) \end{aligned} \quad (3)$$

$$\begin{aligned} v_{worst} = {} & x_{worst} + F_{worst} \cdot (x_{best} - x_{worst}) \\ & + F_{worst} \cdot (x_{med} - x_{worst}) \end{aligned} \quad (4)$$

where $\mathbf{x}_{best}, \mathbf{x}_{med}, \mathbf{x}_{worst}$ are the sorted individuals within the triplet, and $\mathbf{x}_{r1}, \mathbf{x}_{r2}$ are random vectors sampled from the union of the population and the external archive $\mathbf{A}$. The parameter F denotes the adaptive scaling factor associated with each individual.

In this hierarchical structure, the mutation strategies are tailored to the individual's rank. The best individuals focus on exploiting their local neighborhood to refine the solution, while the worst individuals are driven to explore new regions of the search space to maintain diversity.

More recently, the RDE algorithm [10] demonstrated that recombining successful strategies can further push performance limits. RDE integrates two distinct mutation operators: the classic current-to-pbest/1 [7] and the sorted current-to-order-pbest/1 [5]. Unlike static ensembles, RDE employs an adaptive resource allocation mechanism that tracks the cumulative fitness improvement generated by each strategy. It then dynamically adjusts the population ratio assigned to each operator, ensuring that computational resources naturally shift toward the strategy that is currently navigating the landscape most effectively.

While TCDE and RDE effectively utilize competitive relationships, their reliance on iterative sorting and complex history management creates a significant computational bottleneck in high dimensions. OMCDE overcomes these limitations by embedding competitive dynamics into a high-throughput matrix-based framework and introducing a stage-dependent strategy that explicitly shifts the algorithm's structural behavior from exploration to greedy exploitation.

III. Matrix-Based Operations

A. Motivation

Traditional DE [1] implementations rely on iterative loops to process individuals sequentially, a design that inherently fails to exploit the parallel architecture of modern CPUs and GPUs. While the concept of vectorized DE was popularized in early frameworks [11], and recent research has proposed matrix-based structures for vanilla algorithms [12], [13], these approaches have mostly been limited to standard vanilla algorithms. In this work, we extend this matrix-based paradigm to OMCDE by reformulating the entire evolutionary cycle: mutation, crossover, selection, and even dynamic archive management into vectorized linear algebra operations. This unified framework allows for the simultaneous update of the entire population, significantly reducing computational overhead and enabling the use of high-performance computing resources without the need for complex low-level programming.

B. Matrix-Based Operations

Let N denote the population size and D the dimensionality of the search space. Instead of treating individuals as separate objects, we represent the entire population as a matrix $\mathbf{X}$ with dimensions $N \times D$, where each row corresponds to a candidate solution and each column represents a specific dimension. We similarly define the boundary vectors $\mathbf{L}$ and $\mathbf{U}$ with dimensions of $1 \times D$ representing the lower and upper bounds of the search space.

To facilitate vectorized operations, we utilize two auxiliary primitives: a column vector of ones, $\mathbf{1}_{N \times 1}$, used to replicate constraints across the population axis; and a random matrix,

$\mathbf{R}_{N\times D}$, containing uniformly distributed random numbers in $[0, 1]$.

The initialization of the population is performed via a single matrix operation. We use the Hadamard product to broadcast the boundary constraints across the random matrix, projecting values into the feasible space immediately:

$$\begin{aligned}\mathbf{X}_{N\times D} = &(\mathbf{1}_{N\times 1} \times \mathbf{L}_{1\times D}) \\ &+ (\mathbf{1}_{N\times 1} \times (\mathbf{U}_{1\times D} - \mathbf{L}_{1\times D})) \circ \mathbf{R}_{N\times D}\end{aligned} \tag{5}$$

Following initialization and at every subsequent generation, the fitness of the entire population is evaluated in a single parallel step, mapping the objective function across the matrix rows to produce a fitness vector:

$$\mathbf{Fit}_{N\times 1} = f(\mathbf{X}_{N\times D}) \tag{6}$$

In this framework, conditional logic is strictly replaced by Boolean Mask Matrices to avoid branch divergence. For mutation, we row-shuffle the population indices to create permuted matrices $\mathbf{X}_{r1}$, $\mathbf{X}_{r2}$, and $\mathbf{X}_{r3}$. The difference vector calculation is performed as a single block, where the mutation factor $\mathbf{F}$ is defined as a column vector ($N \times 1$) to allow unique adaptive parameters for every individual:

$$\begin{aligned}\mathbf{V}_{N\times D} = &\mathbf{X}_{r1,N\times D} \\ &+ \mathbf{F}_{N\times 1} \circ (\mathbf{X}_{r2,N\times D} - \mathbf{X}_{r3,N\times D})\end{aligned} \tag{7}$$

Crossover is similarly vectorized by generating a random matrix and comparing it against the crossover rate vector $\mathbf{CR}_{N\times 1}$ to generate a binary mask $\mathbf{M}_{N\times D}$, where 1 indicates a crossover event. The trial population $\mathbf{T}$ is constructed by blending the mutant matrix $\mathbf{V}$ and the target matrix $\mathbf{X}$ using the mask:

$$\begin{aligned}\mathbf{T}_{N\times D} = &\mathbf{M}_{N\times D} \circ \mathbf{V}_{N\times D} \\ &+ (\mathbf{1}_{N\times 1} - \mathbf{M}_{N\times D}) \circ \mathbf{X}_{N\times D}\end{aligned} \tag{8}$$

Selection is performed by evaluating the fitness of the trial matrix $\mathbf{T}$ and comparing it against $\mathbf{X}$ to generate a selection mask $\mathbf{S}_{N\times 1}$. The population update is a masked assignment:

$$\begin{aligned}\mathbf{X}_{new,N\times D} = &\mathbf{S}_{N\times 1} \circ \mathbf{T}_{N\times D} \\ &+ (\mathbf{1}_{N\times 1} - \mathbf{S}_{N\times 1}) \circ \mathbf{X}_{old,N\times D}\end{aligned} \tag{9}$$

Finally, we vectorize the management of historical archives for parameter adaptation. Instead of loop-based appending, we utilize boolean indexing to extract successful parameters in batch using the selection mask $\mathbf{S}$. The archives are updated using vectorized dot products between fitness improvement vectors and successful parameter vectors. This formulation allows the adaptive logic to scale efficiently with population size, avoiding sequential processing bottlenecks.

IV. Proposed OMCDE Algorithm

We propose the Opposition-based Matrix-Based Competitive Differential Evolution (OMCDE). Our main goal is to balance exploration and exploitation while making the algorithm run extremely fast. To do this, OMCDE combines four key mechanisms into a unified matrix framework: Opposition-Based Initialization [14], Triplet Competition [5], Two-Stage Heterogeneous Mutation, and Adaptive Parameter Control with Linear Population Size Reduction [15]. The algorithm is smart; it uses the evolutionary information of each individual to assign specific strategies based on their performance rank. It also uses a stage-dependent mutation strategy, inspired by the environment's effect on evolution, that dynamically adapts the mutation behavior according to the search progress. Finally, because every logic step is converted into vectorized matrix operations, the algorithm is exceptionally fast, maintaining high performance even in high-dimensional search spaces.

A. Opposition-Based Matrix Initialization

Standard uniform initialization relies on stochasticity, often placing the population in all regions. To accelerate convergence, we integrate Opposition-Based Learning. However, unlike traditional methods [14] that require sorting and slicing a combined population of $2N$ individuals which has an overhead of $O(N \log N)$, we employ a high-speed Population-Level Selection mechanism based on total fitness.

First, we generate the opposition matrix $\mathbf{X}_{opp}$ in a single broadcasted operation using the boundary vectors $\mathbf{L}$ and $\mathbf{U}$:

$$\mathbf{X}_{opp,N\times D} = (\mathbf{1}_{N\times 1} \times (\mathbf{L} + \mathbf{U})) - \mathbf{X}_{N\times D} \tag{10}$$

Both the original population $\mathbf{X}$ and the opposition population $\mathbf{X}_{opp}$ are evaluated in parallel to produce fitness vectors $\mathbf{Fit}$ and $\mathbf{Fit}_{opp}$. To maintain maximum computational throughput, we simply compare the total fitness of both matrices:

$$\mathbf{X}^{(0)} = \begin{cases}\mathbf{X}_{opp} & \text{if } \sum \mathbf{Fit}_{opp} < \sum \mathbf{Fit} \\ \mathbf{X} & \text{otherwise}\end{cases} \tag{11}$$

This approach allows us to instantaneously select the statistically superior population distribution without breaking the vectorized workflow, ensuring the initialization phase remains $O(N)$ and exceptionally fast.

B. Triplet Competition Mechanism

OMCDE employs a hierarchical structure where individuals compete within random triplets rather than the entire population, preventing the premature convergence typical of global selection strategies. To avoid the computational overhead of iterative sorting, we utilize tensor reshaping. The population matrix $\mathbf{X}$ ($N \times D$) is instantly reshaped into a tensor of $N/3$ triplets and sorted in parallel along the triplet axis. This operation efficiently stratifies the population into three submatrices $\mathbf{X}_{best}$, $\mathbf{X}_{med}$, and $\mathbf{X}_{worst}$, where the best group contains individuals with the lowest fitness values designated for exploitation, and the worst group consists of those with the highest fitness values targeted for exploration. This enables the simultaneous application of heterogeneous mutation strategies without the need for complex individual ranking.

C. Stage-Dependent Heterogeneous Mutation

To effectively balance global exploration with local exploitation, OMCDE employs a dynamic mutation strategy. The algorithm switches between two distinct behavioral modes based on the FES/FES_{max} ratio.

In the first phase $FES \leq \lambda \cdot FES_{max}$, where λ represents the stage switching parameter that defines the duration of the initial diversity maintenance period, preserving diversity is critical to prevent premature convergence. We employ the standard competitive strategy established in TCDE [5], where the mutation logic is strictly dictated by the individual's rank within its triplet. The superior individuals ($\mathbf{X}_{best}$) exploit their local neighborhood (Eq. 2), the intermediate individuals ($\mathbf{X}_{med}$) employ a balanced strategy (Eq. 3), and the inferior individuals ($\mathbf{X}_{worst}$) are forced to explore by learning from the superior vectors to escape poor regions (Eq. 4). Note that to prevent path duplication, the random difference vectors are selected from the union of the current population and the external Solution Archive $\mathbf{A}$ implemented based on [7]. This archive preserves diversity by storing parent vectors recently replaced by superior offspring, allowing the algorithm to leverage historical genetic information to escape local optima.

As the search nears completion ($FES > \lambda \cdot FES_{max}$), the algorithm shifts to a greedy convergence mode. In this phase, the priority changes from diversity maintenance to rapid refinement of the best known solution. We first construct a global best matrix $\mathbf{G}_{best}$ by broadcasting the single global best vector $\mathbf{x}_{gbest}$ to match the dimensions of the sub-populations. All three groups then shift their strategies to pull strongly toward this global optimum while retaining their hierarchical learning structure. The Best individuals are pulled toward the Global Best while retaining a difference vector for local fine-tuning:

$$\begin{aligned}\mathbf{V}_{best,\frac{N}{3}\times D} = &\mathbf{X}_{best} + \mathbf{F}_{best} \circ (\mathbf{G}_{best} - \mathbf{X}_{best}) \\ &+ \mathbf{F}_{best} \circ (\mathbf{X}_{r1} - \mathbf{X}_{r2})\end{aligned} \tag{12}$$

The Medium individuals are pulled toward the Global Best, but also continue to learn from the Local Best to bridge the search space:

$$\begin{aligned}\mathbf{V}_{med,\frac{N}{3}\times D} = &\mathbf{X}_{med} + \mathbf{F}_{med} \circ (\mathbf{G}_{best} - \mathbf{X}_{med}) \\ &+ \mathbf{F}_{med} \circ (\mathbf{X}_{best} - \mathbf{X}_{med}) \\ &+ \mathbf{F}_{med} \circ (\mathbf{X}_{r1} - \mathbf{X}_{r2})\end{aligned} \tag{13}$$

Finally, the Worst individuals use the most aggressive strategy, being pulled toward the Global Best, the Local Best, and the Local Medium simultaneously to rapidly close the gap:

$$\begin{aligned}\mathbf{V}_{worst,\frac{N}{3}\times D} = &\mathbf{X}_{worst} + \mathbf{F}_{worst} \circ (\mathbf{G}_{best} - \mathbf{X}_{worst}) \\ &+ \mathbf{F}_{worst} \circ (\mathbf{X}_{best} - \mathbf{X}_{worst}) \\ &+ \mathbf{F}_{worst} \circ (\mathbf{X}_{med} - \mathbf{X}_{worst})\end{aligned} \tag{14}$$

The implementation of this multi-stage mechanism is driven by the observation that optimal search behavior changes over time. By enforcing exploration in the early stages and switching to greedy exploitation in the final quartile, OMCDE mimics the natural adaptation of organisms to shrinking resources, ensuring that the population explores the landscape broadly before focusing efficiently onto the global optimum.

Algorithm 1: Opposition-Controlled Matrix-Based Competitive Differential Evolution (OMCDE)

Input: N_{init}, D, FES_{max}, Bounds $[\mathbf{L}, \mathbf{U}]$
Output: Global Best Solution $\mathbf{x}_{gbest}$

1. Initialization:
Generate random population $\mathbf{X}$ and opposition matrix $\mathbf{X}_{opp}$ by Eq. 10
Select initial $\mathbf{X}^{(0)}$ by comparing total fitness sums by Eq. 11
Initialize history archives $\mathbf{M}_F, \mathbf{M}_{CR}$ with 0.5, set $FES = N_{init}$
while $FES < FES_{max}$ **do**
  2. Linear Reduction & Parameter Generation:
  Calculate new size N_{next} by Eq. 18 and truncate $\mathbf{X}$ if $N_{next} < N$
  Generate $\mathbf{F}, \mathbf{CR}$ vectors utilizing Cauchy/Normal distributions by Eq. 15
  3. Triplet Competition (Matrix Reshaping):
  Reshape $\mathbf{X}$ into tensor $(N/3 \times 3 \times D)$ and sort parallel along axis 1
  Partition into sub-matrices $\mathbf{X}_{best}, \mathbf{X}_{med}, \mathbf{X}_{worst}$
  4. Stage-Dependent Mutation:
  if $FES \leq \lambda \cdot FES_{max}$ **then**
    `// Phase 1: Competitive Exploration`
    Compute $\mathbf{V}_{best}, \mathbf{V}_{med}, \mathbf{V}_{worst}$ utilizing TCDE rules (Eq. 2–4)
  else
    `// Phase 2: Greedy Exploitation`
    Broadcast $\mathbf{x}_{gbest}$ to matrix $\mathbf{G}_{best}$
    Compute $\mathbf{V}_{best}, \mathbf{V}_{med}, \mathbf{V}_{worst}$ utilizing Greedy rules (Eq. 12–14)
  end
  5. Vectorized Crossover & Selection:
  Concatenate sub-matrices to form global mutant $\mathbf{V}$
  Generate trial matrix $\mathbf{T}$ by Eq. 8
  Evaluate fitness $\mathbf{Fit}_{trial}$ and create selection mask $\mathbf{S}$
  Update population $\mathbf{X}$ by Eq. 9
  6. Adaptation:
  Archive rejected solutions into $\mathbf{A}$
  Update $\mathbf{M}_F, \mathbf{M}_{CR}$ utilizing Weighted Lehmer Mean by Eq. 16
  $FES = FES + N$
end

D. Crossover and Selection

Following the heterogeneous mutation stage, the three mutant sub populations are horizontally concatenated to reconstruct the global mutant matrix $\mathbf{V}$ with dimensions $N \times D$. We

then employ a binomial crossover strategy adapted for matrix execution. A random matrix is compared against the crossover rate vector $\mathbf{CR}$ to create a binary mask, while a secondary mask enforces the guaranteed inheritance condition. The final trial population $\mathbf{T}$ is constructed by blending the mutant and target matrices using the masked operation defined in Eq. 8.

Selection is performed via a parallel greedy comparison between the trial matrix $\mathbf{T}$ and the target matrix $\mathbf{X}$. The fitness of the trial population is evaluated in a single batch step, and a boolean selection mask $\mathbf{S}_{N\times 1}$ is generated to identify trial vectors that outperform their targets. The population update is then executed instantaneously using the masking operation in Eq. 9, discarding inferior solutions while retaining the superior ones for the next generation.

E. Adaptive Parameter Control

The performance of differential evolution is highly sensitive to the control parameters F and CR. To avoid manual tuning, OMCDE utilizes a history based adaptation mechanism [16]. However, unlike standard loop based implementations that generate parameters sequentially, we perform these operations as fully vectorized batch processes.

At the beginning of each generation, we generate parameter vectors $\mathbf{F}$ and $\mathbf{CR}$ for the entire population simultaneously. We maintain two history memory archives, $\mathbf{M}_F$ and $\mathbf{M}_{CR}$, of size H. A vector of random indices $\mathbf{r}$ is generated to sample these archives. The parameters are then constructed by broadcasting the sampled memory values and adding noise derived from Cauchy and Normal distributions, respectively:

$$\begin{aligned} \mathbf{F}_{N\times 1} &= \text{Cauchy}(\mathbf{M}_F[\mathbf{r}], 0.1), \\ \mathbf{CR}_{N\times 1} &= \mathcal{N}(\mathbf{M}_{CR}[\mathbf{r}], 0.1) \end{aligned} \tag{15}$$

After the selection step, the memory archives are updated to reflect the most successful strategies [16]. We first identify the subset of successful parameters $(\mathbf{S}_F, \mathbf{S}_{CR})$ corresponding to trial vectors that improved fitness. We calculate the improvement weights $\Delta\mathbf{f}$ as the absolute difference between the trial and target fitness values. The update value is computed using the Weighted Lehmer Mean. In our matrix framework, this complex summation is reduced to a high-speed vectorized dot product between the weight vector and the parameter powers:

$$mean_{WL} = \frac{\Delta\mathbf{f}_{1\times S}^T \cdot (\mathbf{S}_{F,S\times 1})^{\circ 2}}{\Delta\mathbf{f}_{1\times S}^T \cdot \mathbf{S}_{F,S\times 1}} \tag{16}$$

where $\mathbf{S}_F$ is the vector containing the scaling factors of successful individuals, and $\Delta\mathbf{f}$ represents their corresponding fitness improvements. This computes a weighted mean where parameters resulting in larger fitness gains contribute more significantly to the history update.

Finally, the history memory at position k is updated using a linear combination of the old value and the new Lehmer mean:

$$\mathbf{M}_F[k] = 0.5 \cdot \mathbf{M}_F[k] + 0.5 \cdot mean_{WL} \tag{17}$$

This vectorized formulation allows OMCDE to adapt its search behavior dynamically based on population-wide feedback without incurring the computational penalty of iterative history processing.

To further enhance computational efficiency and accelerate convergence, we dynamically reduce the population size N as the search progresses. This strategy mimics natural resource scarcity, forcing the algorithm to focus its computational power on the most promising individuals during the critical later stages. The population size for the next generation is calculated as:

$$N_{g+1} = \text{round}\left[\left(\frac{N_{min} - N_{init}}{FES_{max}}\right) \cdot FES + N_{init}\right] \tag{18}$$

where N_{init} and N_{min} represent the initial and minimum population sizes, and FES_{max} denotes the maximum number of function evaluations. This mechanism linearly decreases the population size as the search progresses, allocating more computational resources to the best individuals in later stages.

If $N_{g+1} < N_{curr}$, we sort the population matrix $\mathbf{X}$ by fitness and truncate the worst individuals. This reduction creates a naturally increasing selective pressure and ensures the memory footprint of the algorithm scales down linearly.

The complete computational flow of the proposed method is summarized in Algorithm 1. It is crucial to highlight that, unlike traditional implementations that rely on nested loops to process individuals sequentially, Algorithm 1 contains only a single outer loop governing the generations. All internal operations, including mutation, crossover, and selection, are executed as simultaneous matrix instructions, ensuring maximum computational efficiency.

V. EXPERIMENTS

A. Experimental Setup and Parameter Settings

The performance of the proposed OMCDE was evaluated on the CEC 2017 benchmark test suite [6], which consists of 29 test functions ($F_1 - F_{29}$) comprising unimodal, multimodal, hybrid, and composition problems. All experiments were conducted on the Google Colab platform using an Intel Xeon CPU @ 2.20GHz with 12GB of RAM. The algorithm was implemented in Python 3.10 utilizing the NumPy library for vectorized matrix operations.

To ensure a fair comparison, the experimental settings were strictly maintained according to the CEC 2017 guidelines. For each function, 31 independent runs were executed with a dimensionality of $D = 30$. The maximum number of function evaluations (FES_{max}) was set to $10,000 \cdot D$, and the search space bounds were fixed at $[-100, 100]^D$. The error value $f(\mathbf{x}_{best}) - f(\mathbf{x}^*)$ was recorded as the primary performance metric, where $f(\mathbf{x}^*)$ is the global optimum.

The control parameters are set as: population size $N_{init} = 18D$ linearly reduced to $N_{min} = 4$, archive size $|A| = 1.4N$, memory size $H = 6$ for M_F and M_{CR}, and stage switching at $\lambda = 0.75$. Control parameters were adopted from established historical benchmarks. The stage-switching threshold $\lambda = 0.75$ was selected based on the Friedman mean ranking across all test functions to identify the optimal transition point

TABLE I

COMPARATIVE RESULTS OF OMCDE VS. COMPETITORS ON CEC 2017 ($D = 30$) OVER 31 INDEPENDENT RUNS. SYMBOLS +/=/− INDICATE OMCDE IS SIGNIFICANTLY BETTER THAN / SIMILAR TO / WORSE THAN THE COMPETITOR (WILCOXON RANK-SUM TEST, $\alpha = 0.05$).

2***Func**	**OMCDE**		**TCDE**		**LSHADE-RSP**		**iLSHADE-RSP**		**L-SHADE**		**JADE**	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Unimodal Functions												
F_1	**0.00E+00**	**0.00E+00**	0.00E+00=	0.00E+00	0.00E+00=	0.00E+00	0.00E+00=	0.00E+00	0.00E+00=	0.00E+00	0.00E+00=	0.00E+00
F_3	**0.00E+00**	**0.00E+00**	0.00E+00=	0.00E+00	0.00E+00=	0.00E+00	0.00E+00=	0.00E+00	0.00E+00=	0.00E+00	9.43E+03+	1.54E+04
Simple Multimodal Functions												
F_4	5.86E+01	0.00E+00	5.86E+01=	0.00E+00	5.86E+01=	0.00E+00	2.27E+01−	7.02E-01	5.87E+01=	7.70E-01	4.88E+01−	2.39E+01
F_5	**3.52E+00**	1.62E+00	4.83E+00+	1.31E+00	7.11E+00+	2.09E+00	7.89E+00+	2.38E+00	6.77E+00+	1.60E+00	2.63E+01+	4.00E+00
F_6	0.00E+00	2.53E-08	8.54E-06+	5.44E-06	0.00E+00=	2.71E-08	5.77E-08+	1.34E-07	0.00E+00=	1.92E-08	0.00E+00=	0.00E+00
F_7	**3.60E+01**	1.11E+00	3.64E+01+	9.10E-01	3.95E+01+	2.50E+00	4.08E+01+	3.42E+00	3.77E+01+	1.42E+00	5.46E+01+	4.02E+00
F_8	**5.26E+00**	1.48E+00	6.92E+00+	1.84E+00	7.38E+00+	2.28E+00	7.93E+00+	2.39E+00	7.24E+00+	1.59E+00	2.64E+01+	3.83E+00
F_9	0.00E+00	0.00E+00	0.00E+00=	0.00E+00	0.00E+00=	0.00E+00	0.00E+00=	0.00E+00	0.00E+00=	0.00E+00	3.51E-03=	1.75E-02
F_{10}	**1.46E+03**	3.06E+02	1.98E+03+	1.88E+02	1.92E+03+	3.31E+02	1.90E+03+	3.46E+02	1.49E+03=	1.51E+02	1.92E+03+	2.15E+02
Hybrid Functions												
F_{11}	1.91E+01	2.42E+01	2.01E+01=	2.57E+01	2.62E+00−	2.23E+00	3.41E+00−	5.57E+00	28.8E+01+	2.81E+01	3.68E+01+	2.65E+01
F_{12}	1.84E+02	1.29E+02	2.27E+02=	2.36E+02	9.51E+01−	7.16E+01	1.20E+02=	7.85E+01	1.06E+09+	3.76E+02	1.15E+03+	3.87E+02
F_{13}	**1.09E+01**	9.66E+00	1.58E+01+	9.64E+00	1.73E+01+	5.41E+00	1.80E+01+	4.92E+00	1.72E+01+	4.75E+00	4.79E+01+	5.92E+01
F_{14}	**2.10E+01**	4.13E+00	2.21E+01+	1.27E+00	2.15E+01=	1.23E+00	2.17E+01+	1.06E+00	2.16E+01+	1.24E+00	7.30E+03=	2.70E+04
F_{15}	**1.11E+00**	1.85E-01	1.73E+00+	5.86E-01	1.34E+00=	7.73E-01	1.18E+00=	7.50E-01	3.10E+00+	1.46E+00	7.71E+02+	1.82E+03
F_{16}	2.53E+01	1.06E+01	3.69E+01=	4.12E+01	2.87E+01=	4.35E+01	1.86E+01−	6.87E+00	6.25E+01+	7.43E+01	3.83E+02+	1.45E+02
F_{17}	4.02E+01	9.45E+00	4.48E+01+	1.15E+01	3.78E+01=	7.04E+00	3.89E+01=	7.18E+00	3.31E+01−	6.94E+00	7.70E+01+	3.12E+01
F_{18}	2.10E+01	3.42E-01	2.14E+01+	5.04E-01	2.08E+01−	2.89E-01	2.08E+01−	2.88E-01	2.19E+01+	1.07E+00	1.58E+04=	4.77E+04
F_{19}	**3.46E+00**	1.34E+00	4.18E+00+	1.78E+00	3.49E+00=	1.08E+00	3.49E+00=	6.06E-01	5.38E+00+	1.40E+00	1.27E+03=	3.62E+03
F_{20}	**3.21E+01**	1.00E+01	3.86E+01+	9.20E+00	3.38E+01=	9.50E+00	3.24E+01=	7.02E+00	4.10E+01+	8.81E+00	1.21E+02+	6.10E+01
Composition Functions												
F_{21}	**2.04E+02**	1.02E+00	2.22E+02+	1.87E+00	2.07E+02+	2.19E+00	2.08E+02+	2.36E+00	2.07E+02+	1.49E+00	2.27E+02+	5.36E+00
F_{22}	**1.00E+02**	0.00E+00	1.10E+02+	0.00E+00	1.00E+02=	0.00E+00	1.00E+02=	0.00E+00	1.00E+02=	0.00E+00	1.45E+02=	3.18E+02
F_{23}	3.52E+02	2.86E+00	3.52E+02=	3.07E+00	3.51E+02=	3.47E+00	3.50E+02−	3.29E+00	3.49E+02−	2.70E+00	3.74E+02+	6.00E+00
F_{24}	**4.20E+02**	1.75E+00	4.32E+02+	2.18E+00	4.27E+02+	2.10E+00	4.26E+02+	2.45E+00	4.26E+02+	1.67E+00	4.40E+02+	4.50E+00
F_{25}	3.87E+02	1.73E-02	3.87E+02=	1.54E-02	3.87E+02=	0.00E+00	3.79E+02−	0.00E+00	3.87E+02=	0.00E+00	3.87E+02=	0.00E+00
F_{26}	1.01E+03	4.84E+01	1.03E+03+	5.60E+01	9.38E+02−	3.77E+01	9.33E+02−	3.92E+01	9.28E+02−	3.64E+01	1.18E+03+	1.43E+02
F_{27}	4.90E+02	3.86E+00	5.01E+02+	4.57E+00	4.98E+02+	7.29E+00	4.79E+02−	6.51E+00	5.04E+02+	5.50E+00	5.03E+02+	7.56E+00
F_{28}	3.13E+02	3.46E+01	3.28E+02=	4.57E+01	3.04E+02=	2.21E+01	3.02E+02=	1.60E+01	3.30E+02+	4.86E+01	3.42E+02+	5.91E+01
F_{29}	4.31E+02	2.93E+01	4.42E+02=	3.14E+01	4.46E+02+	1.39E+01	4.14E+02−	2.72E+01	4.34E+02+	6.46E+00	4.82E+02+	3.84E+01
F_{30}	2.02E+03	1.46E+01	2.02E+03=	1.14E+01	1.97E+03−	1.10E+02	1.04E+03−	3.21E+02	1.98E+03−	4.71E+01	2.35E+03=	1.36E+03
W/T/L	-	-	**17/12/0**	-	**9/15/5**	-	**9/10/10**	-	**17/8/4**	-	**19/9/1**	-
Friedman Rank	**2.36**	-	4.02	-	2.98	-	2.64	-	3.53	-	5.47	-

B. Statistical Results and Analysis

To evaluate the robustness of OMCDE on widely accepted historical benchmarks, we compared its performance on the CEC 2017 [6] test suite ($D = 30$) against five established algorithms: TCDE [5], LSHADE-RSP [9], iLSHADE-RSP [8], L-SHADE [2], and JADE [7]. These competitors were selected to benchmark OMCDE against both its direct predecessor TCDE and the high-performing LSHADE family. Note that function F_2 is excluded from the analysis as it was officially removed from the CEC 2017 competition due to instability.

To rigorously validate these findings, we applied Friedman mean ranking test to provide a global performance assessment. As shown at the bottom of Table I, OMCDE achieves the best overall rank of 2.36, followed by iLSHADE-RSP (2.64) and LSHADE-RSP (2.98), confirming its superior robustness across the entire benchmark suite. Furthermore, Wilcoxon rank-sum test ($\alpha = 0.05$) was conducted, where symbols $-$, $+$, and $=$ indicate if a competitor is significantly better, worse, or equivalent to OMCDE. The analysis reveals a clear performance trajectory across landscape types. On Unimodal functions, OMCDE matches the perfect convergence of state-of-the-art methods, confirming its exploitation efficiency. In Simple Multimodal tasks, the Triplet Competition mechanism ensures sufficient diversity to escape local optima, consistently outperforming baselines like TCDE. Most importantly, OMCDE exhibits exceptional robustness on the challenging Hybrid and Composition functions ($F_{11} - F_{30}$). By effectively balancing early-stage exploration with late-stage greedy exploitation, the algorithm navigates these deceptive, non-separable landscapes more effectively than L-SHADE variants, securing significant wins on complex problems where traditional adaptive mechanisms often stagnate.

Table I summarizes the comparative results over 31 independent runs. The results indicate that OMCDE exhibits superior performance across the test suite, particularly against its predecessor TCDE (17/12/0) and JADE (19/9/1), where it was only outperformed on a single function. It also maintains a significant lead over L-SHADE (17/8/4) and remains highly competitive against state-of-the-art variants LSHADE-RSP (9/15/5) and iLSHADE-RSP (9/10/10). While parity is maintained on unimodal landscapes, the algorithm demonstrates exceptional robustness on complex Hybrid ($F_{11} - F_{20}$) and Composition ($F_{21} - F_{30}$) functions. This success is primarily driven by the stage-dependent mutation, which enforces a more greedy scheme in final iterations to encourage superior exploitation around the global optimum. Complementing this, the Triplet Competition mechanism provides the necessary diversity to prevent the local optima stagnation common in traditional adaptive variants, securing a best-in-class Friedman rank of 2.36.

C. Empirical Runtime Analysis

To quantify the practical efficiency gains yielded by the matrix-based formulation, we conducted a runtime scalability experiment comparing the execution time of OMCDE

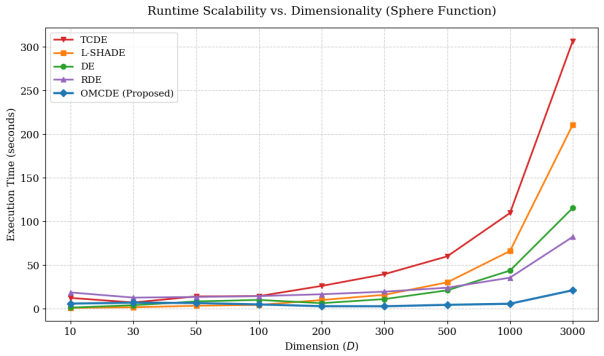

Fig. 1. Runtime scalability analysis comparing the vectorized OMCDE against a other variants of DE on the Sphere function ($N = 100, FES = 10^5$). The results highlight the superior efficiency of the matrix-based framework, which drastically reduces the computational overhead as dimensionality increases from $D = 30$ to $D = 3000$, unlike steep linear growth of the standard loop-based implementation.

against several standard Differential Evolution implementations. To isolate the algorithmic overhead from the objective function complexity, the computationally inexpensive Sphere function was selected as the test bed. All experiments were run with a population size of $N = 100$ for $100,000$ Function Evaluations on varying dimensions $D \in \{10, 30, 50, 100, 200, 300, 500, 1000, 3000\}$ using the same computing environment.

Figure 1 illustrates the execution time as a function of dimensionality. As observed, while the runtime for all algorithms increases with dimension, the slope for OMCDE is significantly flatter than that of the traditional loop-based approaches. This demonstrates that as the dimensionality and problem scale increase, the vectorized matrix operations of OMCDE effectively reduce the computational overhead, resulting in a much faster and more scalable algorithm compared to its iterative counterparts.

VI. Conclusion

In this study, we proposed the Opposition-based Matrix-Based Competitive Differential Evolution (OMCDE), a unified framework that integrates Opposition-Based Learning, Triplet Competition, and Stage-Dependent Mutation into fully vectorized matrix operations. By replacing traditional iterative logic with parallel linear algebra computations, OMCDE significantly enhances computational throughput while maintaining robust search capabilities. Our experiments on the CEC 2017 benchmark test suite demonstrate that this matrix-based formulation effectively balances exploration and exploitation, offering a competitive and highly efficient alternative to standard loop-based evolutionary algorithms. The success of OMCDE suggests that future differential evolution research can achieve significant efficiency gains by prioritizing vectorized implementations over complex control structures. Future work will focus on applying OMCDE to high-dimensional real-world optimization challenges and large-scale engineering design problems. Additionally, the potential of stage-dependent mutation strategies will be explored further. While the current mechanism imposes a global switch, future research should investigate adaptive frameworks where the transition between exploration and exploitation is determined at the individual level, driven by the unique evolutionary history and local landscape feedback of each candidate solution.

Acknowledgment

The authors acknowledge the use of a Gemini LLMS for refining the grammatical structure and stylistic presentation of the text. All algorithmic concepts, mathematical formulations, experimental designs, and scientific conclusions presented in this manuscript are the original work of the authors and were not generated by any LLMS.

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.

[2] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *2014 IEEE congress on evolutionary computation (CEC)*. IEEE, 2014, pp. 1658–1665.

[3] M. F. Ahmad, N. A. M. Isa, W. H. Lim, and K. M. Ang, "Differential evolution: A recent review based on state-of-the-art works," *Alexandria Engineering Journal*, vol. 61, no. 5, pp. 3831–3872, 2022.

[4] M. Pant, H. Zaheer, L. Garcia-Hernandez, A. Abraham *et al.*, "Differential evolution: A review of more than two decades of research," *Engineering Applications of Artificial Intelligence*, vol. 90, p. 103479, 2020.

[5] Q. Yang, Z.-Y. Qiao, P. Xu, X. Lin, X.-D. Gao, Z.-J. Wang, Z.-Y. Lu, S.-W. Jeon, and J. Zhang, "Triple competitive differential evolution for global numerical optimization," *Swarm and Evolutionary Computation*, vol. 84, p. 101450, 2024.

[6] G. Wu, R. Mallipeddi, and P. Suganthan, "Problem definitions and evaluation criteria for the cec 2017 competition and special session on constrained single objective real-parameter optimization," *Nanyang Technol. Univ., Singapore, Tech. Rep*, pp. 1–18, 2016.

[7] 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.

[8] T. J. Choi and C. W. Ahn, "An improved lshade-rsp algorithm with the cauchy perturbation: ilshade-rsp," *Knowledge-Based Systems*, vol. 215, p. 106628, 2021.

[9] V. Stanovov, S. Akhmedova, and E. Semenkin, "Lshade algorithm with rank-based selective pressure strategy for solving cec 2017 benchmark problems," in *2018 IEEE congress on evolutionary computation (CEC)*. IEEE, 2018, pp. 1–8.

[10] S. Tao, R. Zhao, K. Wang, and S. Gao, "An efficient reconstructed differential evolution variant by some of the current state-of-the-art strategies for solving single objective bound constrained problems," *arXiv preprint arXiv:2404.16280*, 2024.

[11] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer, 2005.

[12] X. Li, M. Cao, J.-Y. Li, J. Xu, J. Zhang, and Z.-H. Zhan, "Matrix-based ant colony optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2024, pp. 747–750.

[13] Z.-H. Zhan, J. Zhang, Y. Lin, J.-Y. Li, T. Huang, X.-Q. Guo, F.-F. Wei, S. Kwong, X.-Y. Zhang, and R. You, "Matrix-based evolutionary computation," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 2, pp. 315–328, 2021.

[14] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary computation*, vol. 12, no. 1, pp. 64–79, 2008.

[15] J. Brest, M. S. Maučec, and B. Bošković, "il-shade: Improved l-shade algorithm for single objective real-parameter optimization," in *2016 IEEE congress on evolutionary computation (CEC)*. IEEE, 2016, pp. 1188–1195.

[16] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *2013 IEEE congress on evolutionary computation*. IEEE, 2013, pp. 71–78.