Low-Rank Landscape Learning via Tensor Decomposition for Metaheuristic initialization

1st Mohamed Fathallah
School of Systems and Computing
University of New South Wales
Canberra 2600, Australia
m.abouhodaima@unsw.edu.au

2nd Saber M. Elsayed
School of Systems and Computing
University of New South Wales
Canberra 2600, Australia
s.elsayed@unsw.edu.au

3rd Ruhul A. Sarker
School of Systems and Computing
University of New South Wales
Canberra 2600, Australia
r.sarker@unsw.edu.au

***Abstract*—Population-based metaheuristics rely heavily on the quality of their solution initialization; however, traditional strategies scatter solutions uniformly across the search space without exploiting the problem structure, which may limit their search ability. Therefore, in this paper, we propose Tensor-Based Initialization (TBI). This technique approximates high-dimensional fitness landscapes through a low-rank approximation tensor, which divides the input tensor into three components: C (sampled columns/slices), U (a small interlinking core), and R (sampled rows) (CUR). Unlike traditional decomposition methods, such as the Singular Value Decomposition (SVD), CUR preserves interpretability by using actual data elements as basis vectors rather than abstract/eigenvector factors. Therefore, TBI breaks down problems into smaller, easier-to-handle groups of dimensions. It then samples the search space at different levels of detail and combines these samples to build a rough picture of where good solutions are likely to be, without evaluating every possible point. By embedding this approach in optimization algorithms, the proposed framework works in three steps: first, a quick exploration finds diverse candidate regions using the optimizer, second, TBI refines those regions, and third, the optimizer starts from those informed regions instead of random points. Experiments on 29 standard test problems show that TBI helps find better solutions, winning on 16 problems while losing on only 1 when comparing the best results found. TBI adds only a small cost (about 3.4% of the total computation) and works best on problems that have underlying patterns it can exploit.**

***Index Terms*—Metaheuristic initialization, tensor decomposition, CUR decomposition, population-based optimization, landscape approximation**

I. Introduction

Many engineering and scientific challenges require finding the best possible solution within a vast continuous, bound constrained search space [1]–[4]. Consider designing the shape of an aircraft wing to minimize drag while maintaining lift [5], tuning hyperparameters in a deep neural network to minimize prediction error [6], [7], or adjusting process parameters in physical experiments to maximize yield [8]. These problems share a common mathematical structure: they seek to minimize (or maximize) an objective function $f(\mathbf{x})$ where the decision variables $\mathbf{x} \in \mathbb{R}^D$ lie within bounded intervals $[x_{\min}, x_{\max}]^D$. Such problems are called *continuous bound-constrained single-objective optimization problems*.

When the objective function is smooth (differentiable) and convex, classical gradient-based methods can locate the optimum efficiently by following the steepest descent direction [9]. However, many practical problems have an objective function that is neither convex nor smooth [1]. Their fitness landscapes are rugged, filled with multiple peaks and valleys (multimodal), and the gradient either does not exist because the function lacks differentiability or points toward misleading local minima [10], [11]. This fundamental limitation has driven researchers toward a different class of techniques: metaheuristic algorithms.

Metaheuristic algorithms, such as Particle Swarm optimization (PSO) [12] and Differential Evolution (DE) [13], take a fundamentally different approach. Instead of following gradients, they maintain a population of candidate solutions that explore the search space collectively. These candidates share information, compete, and evolve over many iterations, gradually converging toward high-quality regions. Because they require no gradient information, metaheuristics can tackle black-box problems where the objective function's internal structure is unknown [14]. Their flexibility and robustness have made them indispensable tools for complex optimization tasks across science and industry. Yet, metaheuristics have a well-documented weakness: their performance depends critically on where the search begins, known as initialization [3], [15].

Traditional initialization strategies like Latin Hypercube Sampling (LHS) [16] and Opposition-Based Learning (OBL) [17] treat the search space uniformly and attempt to spread the initial population as evenly as possible across the search space [15], [18]. These methods are simple, fast, and require no prior knowledge about the problem structure. However, this very generality is also their weakness. By treating every region of the search space as equally promising, uniform strategies ignore any structure or existing knowledge the problem might have.

Recognizing this limitation, researchers have turned to data-driven initialization methods that attempt to learn where good solutions might lie. Clustering-based approaches [19], [20] first scatter an exploratory sample across the space, then group the results to identify promising regions.

Deep learning methods [21], [22] train neural networks on historical optimization trajectories to predict good starting points for new problems. Another direction is using generative models like Generative Adversarial Networks (GANs), which have been mainly integrated with metaheuristics to generate

new data [23]–[25] either for initialization or to introduce new solutions.

These learning-based methods represent genuine progress, yet they share a fundamental limitation: they learn from solution data rather than from the objective landscape itself, and if the quality of the collected data is bad, they tend to perform badly [21]. Additionally, clustering still relies on an initial random exploration phase whose quality is unpredictable. Deep learning requires large training datasets from similar problems, which may not exist for novel applications. GANs suffer from instability and mode collapse [26]. Crucially, none of these methods constructs an explicit, lightweight approximation of the fitness landscape that could guide initialization directly and efficiently.

What is needed, therefore, is a method that can (1) *learn* the structure of the objective landscape from a small number of function evaluations, (2) *identify* promising regions before the main optimization begins, and (3) *adapt* to arbitrary problem transformations without requiring historical data or prior knowledge of where the optimum lies. If such a method existed, it could provide metaheuristics with an informed starting population-one that is already concentrated in high-quality regions- thereby accelerating convergence and improving final solution quality.

This paper proposes exactly such a method, a TBI framework that approximates high-dimensional fitness landscapes through CUR tensor decomposition. We treat the discretized search space as a tensor (a multi-dimensional grid) whose entries represent objective function values at grid points. TBI partitions the decision space into dimension groups, for example, a 30-dimensional problem might be split into three groups of 10 dimensions each or five groups of 6 dimensions, since handling all 30 dimensions together would be computationally infeasible. TBI applies CUR independently to each group and fuses the results to identify promising regions for initialization.

We integrate TBI with RDEx [14], a state-of-the-art Differential Evolution variant. The combined system, TBI-RDEx, operates in three phases: (1) a scouting phase (a quick exploration to find diverse candidate regions) using RDEx with minimal budget to identify candidate regions, (2) tensor approximation to refine candidates through low-rank reconstruction, and (3) full RDEx optimization from the informed starting population. The tensor phase consumes approximately 3% of the total evaluation budget.

We validate TBI-RDEx on the CEC2025 benchmark suite comprising 29 functions in 30 dimensions across 25 independent runs. TBI-RDEx achieves the best Friedman ranking (2.25) among five algorithms, with statistical significance at $\alpha = 0.01$ ($p = 0.0023$). Against baseline RDEx, TBI-RDEx yields a 14:1 win-loss ratio; against iEACOP, it wins on 25 of 29 functions (86.2%). These results confirm that tensor-based landscape approximation meaningfully enhances metaheuristic initialization.

The remainder of this paper is organized as follows. Section II reviews related work on initialization strategies and tensor decomposition methods in evolutionary computation. Section III presents the proposed TBI framework in detail, beginning with mathematical preliminaries and progressing through a step-by-step construction of the algorithm. Section IV describes the experimental setup and presents comprehensive results, including statistical significance tests and performance analysis across function types. Finally, Section V concludes the paper with a summary of findings and directions for future research.

II. Related Work

This section gives a brief review of existing related work.

A. Learning-Based initialization

Data-driven initialization methods improve upon uniform sampling by attempting to identify promising regions before optimization begins. Clustering approaches [19], [20] partition an initial exploratory sample to detect high-quality or diverse regions. Premkumar et al. [22] combine K-means preprocessing with Grey Wolf Optimiser, while Guo et al. [27] apply dimensionality reduction via truncated Single Value Decomposition T-SVD followed by clustering for neural architecture search.

Deep learning methods offer more sophisticated pattern recognition. Oyelade et al. [21] train LSTMs on historical optimization trajectories to generate initial populations, achieving faster convergence on CEC benchmarks. Reinforcement learning variants [28], [29] learn initialization policies through interaction with the search process.

While effective, these approaches share common limitations. Clustering relies on an initial random exploration phase, deep learning requires large training data from similar problems, and RL consumes computational overhead during search. None constructs an explicit approximation of the fitness landscape to guide initialization

B. Matrix and Tensor Decomposition

Mathematical decomposition for initialization remains largely unexplored. Mu et al. [30] applied PCA for dimensionality reduction before PSO, but this transforms the search space rather than initializing within it. Huang et al. [31] used SVD for NMF-specific optimization, not general benchmarks.

Tensor methods have emerged for evolutionary computation, though not for initialization. Wang et al. [32] decompose decision spaces to coordinate sub-populations during search, while their later work [33] uses tensor prediction for dynamic multi-objective problems. These operate on evolving populations, not on approximating the objective landscape itself.

Li et al.'s EVOLER [9] is the closest work, using low-rank representation to identify attention subspaces. However, EVOLER constructs sampling grids centred on predefined regions, failing when optima are shifted from the search space centre. Our TBI framework addresses this by first scouting the landscape via brief metaheuristic exploration, then applying tensor reconstruction around multiple data-driven centres where initial promising solutions were actually found.

III. Methodology

This section presents the TBI framework, a novel population initialization strategy that exploits the structure of high-dimensional search spaces through tensor representations. We first illustrate the core concepts through a 2D numerical example, then present the complete high-dimensional framework.

A. Mathematical Preliminaries

We adopt standard tensor notation from Kolda et al. [34] for definitions of mode-n matricization and TTM products. CUR decomposition approximates the matrix $\mathbf{F} \in \mathbb{R}^{M \times N}$ where M is the number of rows and N is the number of columns, as $\hat{\mathbf{F}} = \mathbf{C}\mathbf{U}^{\dagger}\mathbf{R}$, minimizing Frobenius norm reconstruction error.

B. CUR-Based Landscape Approximation: A 2D Example

To build intuition, we illustrate CUR decomposition on the shifted sphere function $f(x_1, x_2) = (x_1 - 30)^2 + (x_2 - 50)^2$ over the domain $[-100, 100]^2$, discretized into an $L = 8$ uniform grid. This creates landscape matrix $\mathbf{F} \in \mathbb{R}^{8 \times 8}$ as shown in Fig. 1, requiring 64 function evaluations for exhaustive sampling. Values are min-max normalized to $[0, 10]$ via:

$$f_{\text{norm}} = \frac{f - f_{\min}}{f_{\max} - f_{\min}} \times 10 \tag{1}$$

where $f_{\min} = 218$ at $(43, 43)$ and $f_{\max} = 39400$ at $(-100, -100)$. The global minimum is marked with $\star$.

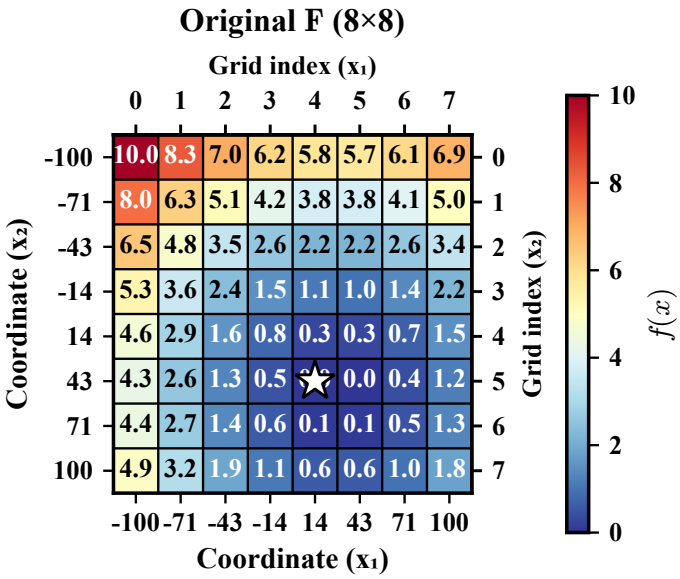

Fig. 1. A 2D objective function landscape discretized on an 8×8 grid (L=8). The colour gradient represents function values, with the global minimum marked by a star. Exhaustive evaluation requires 64 function evaluations.

To avoid exhaustive evaluation, we apply CUR decomposition in four steps, illustrated in Fig. 1 (Input) and Fig. 2 (Reconstruction):

1) **Index Selection:** We select $s = 3$ representative indices $\mathcal{I}_s$ via stratified sampling, which divides the grid into s equal regions and selects one representative from each $\mathcal{I}_s = \{\lfloor (i + 0.5) \cdot L/s \rfloor \mid i = 0, \ldots, s-1\}$. This yields $\mathcal{I}_s^{col}, \mathcal{I}_s^{rows} = \{1, 4, 6\}$, ensuring coverage of boundaries and center, unlike random sampling, which may leave gaps.
2) **Decomposition:** We construct three matrices by evaluating $f(x)$ only at specific intersections:
   - Column matrix $C = F[:, \mathcal{I}_s^{col}] \in \mathbb{R}^{8 \times 3}$
   - Row matrix $R = F[\mathcal{I}_s^{row}, :] \in \mathbb{R}^{3 \times 8}$
   - Core matrix $U = F[\mathcal{I}_s^{row}, \mathcal{I}_s^{col}] \in \mathbb{R}^{3 \times 3}$

   This requires only $M \cdot s + N \cdot s - s^2 = 39$ evaluations, a 39% reduction from the full grid.
3) **Weight Learning:** We compute the Moore-Penrose pseudoinverse [35] $\mathbf{U}^{\dagger}$ using Singular Value Decomposition (SVD): if $\mathbf{U} = \mathbf{V}\mathbf{\Sigma}\mathbf{W}^{\top}$, then $\mathbf{U}^{\dagger} = \mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{V}^{\top}$, where $\mathbf{\Sigma}^{-1}$ inverts the non-zero singular values and $\mathbf{V}$ and $\mathbf{W}$ are the orthogonal matrices of left and right singular vectors. Intuitively, $\mathbf{U}^{\dagger}$ learns the optimal weights to **fill in** missing entries by minimizing reconstruction error.
4) **Reconstruction:** The full landscape is approximated as $\hat{F} \approx C \times U^{\dagger} \times R$ as seen in Fig.2. The minimum is identified from $\hat{F}$ rather than an exhaustive search.

C. Scaling CUR to High-Dimensional Problems

1) The Dimensionality Challenge: In Section III-B, we represented a 2D landscape as a matrix $\mathbf{F} \in \mathbb{R}^{L \times L}$, where each axis corresponded to one dimension of the search space. CUR decomposition exploited the matrix structure to reduce evaluations from L^2 to approximately $2 \cdot L \cdot s - s^2$.

For a D-dimensional problem, the natural representation is no longer a matrix but a tensor. With $D = 30$ dimensions and L grid points per dimension, the landscape becomes a 30-way tensor $\mathcal{F} \in \mathbb{R}^{L \times L \times \cdots \times L}$ containing L^{30} entries. For $L = 3$, this requires $3^{30} \approx 2 \times 10^{14}$ function evaluations clearly intractable to calculate.

The fundamental challenge is that this CUR decomposition operates on matrices (2D structures), not tensors (N-dimensional structures). We cannot directly apply the elegant 2D approach from Section III-B to a 30-dimensional tensor. This raises a critical question: how can we leverage the efficiency of CUR decomposition in high-dimensional spaces?

2) Dimensional Grouping: Our solution is to partition the D=30 dimensions into smaller five groups $k = 5$, construct a manageable tensor we call (auxiliary tensor $\mathcal{A}^{(k)}$) for each group, and apply a tensor extension of CUR to each. Rather than building one intractable to calculate L^{30} tensor, we build five tractable L^6 tensors/groups, each focusing on 6 dimensions while keeping others at reduced resolution, which is s in our numerical example. This is elaborated in the following section.

$$\mathcal{P}_k = \{(k-1) \cdot 6 + 1, \ldots, k \cdot 6\}, \quad k = 1, \ldots, 5 \tag{2}$$

Group $\mathcal{P}_1$ contains dimensions 1–6, group $\mathcal{P}_2$ contains dimensions 7–12, and so on. The number 6 is chosen because it produces tensors of size L^6 (for the high-resolution dimensions),

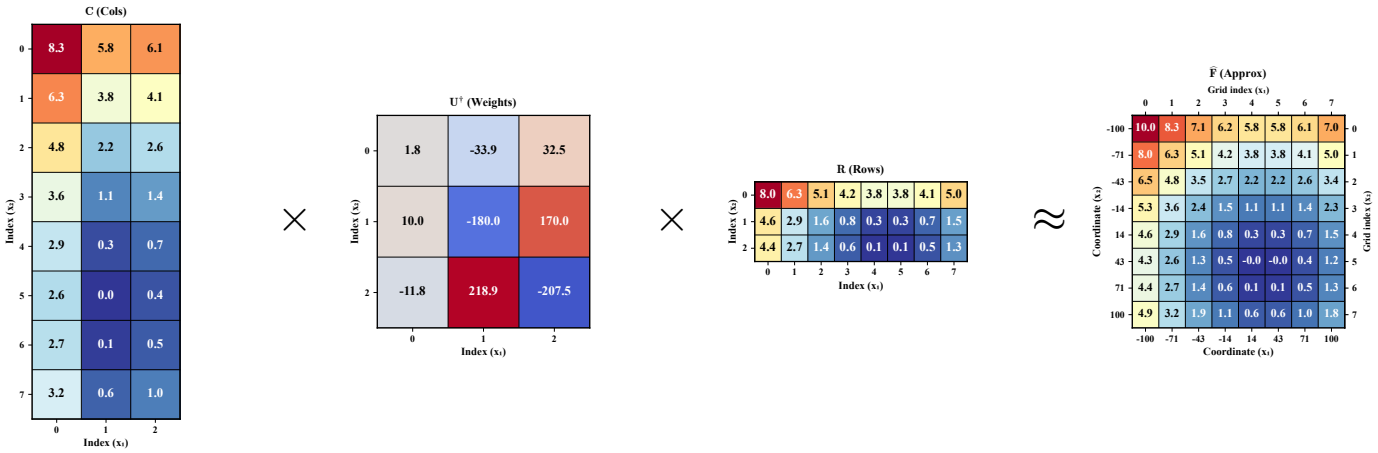

Fig. 2. Complete CUR decomposition formula: $\hat{\mathbf{F}} = \mathbf{C} \times \mathbf{U}^{\dagger} \times \mathbf{R}$. The column matrix $\mathbf{C}$ (8×3), learned weights $\mathbf{U}^{\dagger}$ (3×3), and row matrix $\mathbf{R}$ (3×8) combine to approximate the full 8×8 landscape.

which for $L = 3$ gives $3^6 = 729$ entries (computationally tractable).

3) Auxiliary Tensor Structure: Recall from Section III-B that sampling indices $\mathcal{I}_s$ enabled us to capture the essential structure with fewer evaluations. The same principle applies here; dimensions outside group k are sampled at s representative points rather than full resolution L.

To understand the tensor structure, consider auxiliary tensor $\mathcal{A}^{(1)}$ for the first group (dimensions 1–6). This tensor has 10 dimensions: 6 dimensions at full resolution L (for dimensions 1–6), plus 4 dimensions at sampling resolution s (one representative dimension from each of the other 4 groups). The shape is:

$$\mathcal{A}^{(1)} \in \mathbb{R}^{\underbrace{L \times L \times L \times L \times L \times L}_{\text{6 full-resolution dimensions}} \times \underbrace{s \times s \times s \times s}_{\text{4 sampled dimensions}}} \tag{3}$$

With $L = 3$ and $s = 2$, this tensor has $3^6 \times 2^4 = 729 \times 16 = 11{,}664$ entries. Each entry of $\mathcal{A}^{(1)}$ corresponds to evaluating $f(\mathbf{x})$ which is 30D, at a specific configuration: dimensions 1–6 take values from the full grid, one dimension from each other group takes a sampled value, and remaining dimensions are held fixed at a random point.

$$\vec{x} = [\underbrace{x_1, \ldots, x_6}_{L^6}, \underbrace{x_7, \bar{x}_{8:12}, x_{13}, \bar{x}_{14:18}, x_{19}, \bar{x}_{20:24}, x_{25}, \bar{x}_{26:30}}_{s^4 \text{ sampled + fixed}}]$$

4) Tensor-Based Reconstruction Procedure: Since CUR operates on matrices, we employ mode-n matricization to unfold tensor dimensions into matrix representations. The reconstruction proceeds through four steps, detailed in Algorithm 1.

The process begins by sampling all 10 dimensions at indices $\mathcal{I}_s$ to obtain a core tensor $\mathcal{R} \in \mathbb{R}^{s \times s \times \cdots \times s}$. This compact representation captures the essential landscape structure at low resolution across all dimensions.

Next, we expand the first 4 dimensions from sampled resolution s to full resolution L. For each dimension n, the tensor is flattened into a 2D matrix where dimension n becomes rows, and all other dimensions are collapsed into columns. From this, we extract column matrix $\mathbf{C}_{(n)} \in \mathbb{R}^{L \times K}$ and core matrix $\mathbf{U}_{(n)} \in \mathbb{R}^{s \times K}$, then compute projection weights $\mathbf{W}_n = \mathbf{C}_{(n)} \cdot \mathbf{U}_{(n)}^{\dagger}$. These weights map from sampled to full resolution and are applied via tensor-times-matrix products: $\mathcal{A}_{\text{partial}} = \mathcal{R} \times_1 \mathbf{W}_1 \times_2 \mathbf{W}_2 \times_3 \mathbf{W}_3 \times_4 \mathbf{W}_4$. After this step, the first 4 dimensions have resolution L while the remaining 6 dimensions stay at resolution s.

The partial tensor is then reshaped into column matrix $\mathbf{C}_{col} \in \mathbb{R}^{L^4 \times s^6}$, and we construct row matrix $\mathbf{R}_{row} \in \mathbb{R}^{s \times (L^2 \cdot s^4)}$ by expanding dimensions 5–6 to full resolution. The complete auxiliary tensor is reconstructed as $\hat{\mathcal{A}}^{(1)} = \mathbf{C}_{col} \times \mathbf{U}_{bridge}^{\dagger} \times \mathbf{R}_{row}$, where $\mathbf{U}_{bridge}$ captures the intersection of column and row spaces. The result has shape $L^6 \times s^4$: full resolution for the current group's 6 dimensions and sampled resolution for the 4 external groups.

Finally, after reconstructing all five auxiliary tensors using the same procedure, we identify each tensor's minimum and extract its 6 full-resolution coordinates. Concatenating these coordinates from all groups yields the final 30D candidate solution $\mathbf{x}^* \in \mathbb{R}^{30}$.

D. Integration with Metaheuristic optimization

The TBI framework provides high-quality seed points for subsequent metaheuristic refinement. However, directly applying TBI to the full search space from the start presents a critical challenge: aggressive early narrowing can trap the optimizer in local optima. In our preliminary experiments, we observed that when TBI immediately narrows the search bounds based on tensor-identified minima, the subsequent optimizer often converges to suboptimal solutions from which it cannot escape. This occurs because the initial tensor reconstruction, while efficient, may identify a promising local region that excludes the global optimum. To address this limitation, we developed a three-phase pipeline that balances informed initialization with exploratory diversity.

1) Phase 1: Initial Scouting (Solution Diversity): Rather than applying TBI directly to the vast 30-dimensional space,

Algorithm 1 Tensor-Based Initialization

Input: Full resolution L, sample resolution s, indices $\mathcal{I}_s$
Output: Solution $\mathbf{x}^* \in \mathbb{R}^{30}$
for each group $k = 1$ to 5 **do**
  Sample all 10 dimensions at $\mathcal{I}_s$ to get core tensor $\mathcal{R}$
  for $n = 1$ to 4 **do**
    Extract $\mathbf{C}_{(n)} \in \mathbb{R}^{L \times K}$ and $\mathbf{U}_{(n)} \in \mathbb{R}^{s \times K}$
    Compute weights: $\mathbf{W}_n = \mathbf{C}_{(n)} \cdot \mathbf{U}_{(n)}^{\dagger}$
  end for
  Expand: $\mathcal{A}_{\text{partial}} = \mathcal{R} \times_1 \mathbf{W}_1 \times_2 \mathbf{W}_2 \times_3 \mathbf{W}_3 \times_4 \mathbf{W}_4$
  Reshape to $\mathbf{C}_{col} \in \mathbb{R}^{L^4 \times s^6}$
  Construct $\mathbf{R}_{row} \in \mathbb{R}^{s \times (L^2 \cdot s^4)}$ by expanding dims 5–6
  Reconstruct: $\hat{\mathcal{A}}^{(k)} = \mathbf{C}_{col} \times \mathbf{U}_{bridge}^{\dagger} \times \mathbf{R}_{row}$
  Find minimum index and extract 6 full-resolution coordinates
end for
Concatenate coordinates from all 5 groups
Return $\mathbf{x}^* \in \mathbb{R}^{30}$

we first perform a brief scouting phase using a fast metaheuristic. The RDEx optimizer runs with a small budget ($B_{\text{scout}} = 3{,}000$ evaluations, approximately 1% of total budget) to broadly explore the search space. This phase serves two purposes: (1) it identifies multiple promising regions rather than committing to a single tensor-identified location, and (2) it provides diverse starting points that reduce the risk of missing the global optimum.

The scouting phase collects a set of candidate solutions $\mathcal{S} = \{(\mathbf{x}_i, f_i)\}_{i=1}^{N_{\text{scout}}}$ along with their fitness values. These solutions are distributed across the search space, providing a coarse map of the fitness landscape.

2) Phase 2: Multi-Center TBI Refinement: From the scouting results, we select K diverse promising centers for TBI refinement. The parameter K controls the trade-off between exploration (more centers) and exploitation (more budget per center). In our experiments, we use $K = 5$ centers, which balances computational cost with robustness to local optima. Center Selection: Centers are selected using a combined quality-diversity criterion. We do not simply choose the K best solutions, as they may cluster in a single region. Instead, we compute:

1) Sort all scouting solutions by fitness (best first).
2) Select the best solution as the first center $\mathbf{c}_1$.
3) For each remaining solution (in fitness order), check if it is far enough from all already-selected centers. Specifically, if

$$\min_j \left\| \frac{\mathbf{x} - \mathbf{c}_j}{\mathbf{x}_{\max} - \mathbf{x}_{\min}} \right\| > \tau \tag{4}$$

add it as the next center, where $\tau = 0.1$ is a distance threshold, $\mathbf{x}$ is a candidate solution, and $\mathbf{x}_{\min}$, $\mathbf{x}_{\max}$ are the global search space bounds.
4) Repeat until K centers are selected.

This ensures that selected centers are both high-quality and spatially separated.

Local TBI Application: for each selected center $\mathbf{c}_k$, we define a local search region:

$$\Omega_k = [\mathbf{c}_k - \boldsymbol{\rho},\ \mathbf{c}_k + \boldsymbol{\rho}] \cap [\mathbf{x}_{\min}, \mathbf{x}_{\max}] \tag{5}$$

where $\boldsymbol{\rho}$ defines the region radius (typically 30% of the original search range) and the intersection ensures bounds are respected. Within this local region, TBI constructs auxiliary tensors, applies Tensor-CUR reconstruction, and identifies a refined solution $\mathbf{x}_k^*$.

3) Phase 3: RDEx Global optimization: The refined solutions from Phase 2 form the handoff population:

$$\mathcal{H} = \underbrace{\{\mathbf{x}_1^*, \ldots, \mathbf{x}_K^*\}}_{\text{TBI-refined centres}} \cup \underbrace{\{\mathbf{x}_1^{\text{elite}}, \ldots, \mathbf{x}_{N_e-K}^{\text{elite}}\}}_{\text{remaining elite solutions}} \tag{6}$$

where the TBI-refined centres are the K solutions that underwent tensor reconstruction, and the remaining elite solutions are high-quality candidates from Phase 1 that were not selected as refinement centres. Specifically, if N_{elite} top-performing solutions are retained from scouting and K are selected for TBI refinement, then $N_{\text{elite}} - K$ solutions supplement the refined centres to preserve diversity in the initial population. For instance, with $N_{\text{elite}} = 8$ and $K = 5$, three additional high-quality solutions from scouting are included in the handoff population.

Once the handoff population $\mathcal{H}$ is constructed, RDEx receives it as the initial population but operates within the original full bounds $[\mathbf{x}_{\min}, \mathbf{x}_{\max}]$, not the narrowed TBI regions. This design choice is deliberate: if TBI incorrectly identifies a local optimum, RDEx retains the freedom to explore the entire search space and escape. While the algorithm supports iterative bound contraction to narrow the search space toward a single potential global optimum, this aggressive strategy does not succeed uniformly across all function types, an observation that presents a path for future investigation. By decoupling initialization quality from search space constraints, the TBI-refined solutions provide a warm start that accelerates convergence when the tensor approximation is accurate, without catastrophically constraining the search when it is not.

RDEx consumes the remaining budget (∼96% of total evaluations) for global optimization, benefiting from the high-quality initial population while maintaining full exploratory capability.

IV. Experimental Results

The proposed TBI-RDEx framework was evaluated on the CEC2025 benchmark suite (F1–F29) in 30 dimensions. Each experiment was repeated 25 times independently with a total budget of 300,000 function evaluations. Results are reported as error values $f(\mathbf{x}) - f(\mathbf{x}^*)$. We compare TBI-RDEx against four state-of-the-art algorithms: RDEx (baseline) [14], mLSHADE-RL [36], iEACOP [37], and jSOa [38].

A. Experimental Setup

The TBI framework was configured with the following parameters: grid resolution $L = 5$, sampling size $s = 2$, hierarchical iterations $i_1 = i_2 = 1$, and number of refinement centers

$K = 5$. These values were determined through preliminary experimentation on a subset of benchmark functions.

The iteration parameters i_1 and i_2 control the depth of hierarchical bound contraction. Higher values ($i_1, i_2 > 1$) implement an aggressive narrowing strategy that repeatedly halves the search bounds around the tensor-identified minimum, effectively zooming into a single attention region. While this approach can accelerate convergence when the initial approximation is accurate, it risks premature commitment to local optima. We set $i_1 = i_2 = 1$ to complement RDEx's search behavior. Rather than deep exploitation of a single region, TBI performs breadth-first refinement across K diverse centroids, allowing the subsequent optimizer to select among multiple promising candidates. For the grid resolution L, we observed that values below 5 degraded reconstruction accuracy, while values above 5 provided diminishing returns at increased computational cost. Specifically, the number of tensor evaluations scales with $L > 6$ per dimension group, making $L = 5$ a practical compromise between landscape fidelity and evaluation budget. The number of refinement centres K was evaluated at $K \in 2, 5, 7$. Performance differences across these values were modest; however, computational overhead increases linearly with K since each centre requires independent tensor construction. We selected K = 5 as it provided robust coverage of diverse promising regions without excessive evaluation expenditure.

All experiments were conducted on an Intel Core Ultra 9 275HX processor (2.70 GHz, 24 cores) with 32 GB of RAM, using parallel execution, with all algorithms implemented in Python 3.11.3. Following the CEC2025 competition guidelines, Table I reports the complexity metrics T_1 (average benchmark execution time), T_2 (average algorithm execution time), and $(T_2 - T_1)/T_1 = 0.48$, indicating that the algorithmic overhead is less than half the function evaluation time.

TABLE I
ALGORITHM COMPLEXITY OF TBI-RDEX

Metric	Value
T_1 (seconds)	0.755322
T_2 (seconds)	1.119152
Complexity $(T_2 - T_1)/T_1$	0.481687

B. Performance Comparison

Table II presents the comprehensive comparison of TBI-RDEx with state-of-the-art algorithms across all 29 benchmark functions, with bold values indicating best performance. Table III summarizes the pairwise Win/Tie/Loss (W/T/L) results.

On best-found solutions, TBI-RDEx demonstrates strong competitive performance. Against baseline RDEx, TBI-RDEx wins on 14 functions while losing on only 1 (F29), achieving a 14:1 win-loss ratio. Against mLSHADE-RL, TBI-RDEx wins 16 and loses 9 (64% win rate). The most substantial margin is against iEACOP, with 25 wins out of 29 functions (86.2% win rate). Against jSOa, TBI-RDEx achieves 15 wins versus 11 losses. For mean performance, TBI-RDEx wins 25 functions against iEACOP with only 4 losses, and achieves win rates of 57% and 52% against jSOa and mLSHADE-RL, respectively. The tie with RDEx on mean values (12 wins, 12 losses) indicates that while TBI initialization significantly improves best-found solutions, the average performance benefits are problem-dependent.

Beyond win counts, TBI-RDEx achieves substantial improvements on winning functions: 38.75% average improvement over RDEx, and approximately 76% over mLSHADE-RL, iEACOP, and jSOa. The most notable gains occur on F9 (99.6% improvement over RDEx, mean 0.618 vs 163), F15 (95.7% improvement, mean 0.044 vs 1.01), and F21, where TBI-RDEx escapes the 100-offset trap with 99.99% improvement (mean 0.0138 vs 112). On composition functions, TBI-RDEx achieves near-optimal results on F23 (mean 200) compared to competitors (357–442), and outperforms all algorithms on F26 with over 50% improvement.

C. Statistical Significance Analysis

To validate the statistical significance of our results, we conducted the Friedman test across all five algorithms. Table IV presents the average rankings and test statistics.

The Friedman test reveals statistically significant differences among the algorithms at the $\alpha = 0.01$ significance level. For best-found solutions, TBI-RDEx achieves the lowest (best) average ranking of 2.25, outperforming all competitors, including the baseline RDEx (rank 2.91), jSOa (rank 2.93), mLSHADE-RL (rank 3.00), and iEACOP (rank 3.91). The p-value of 0.0023 confirms that these differences are statistically significant.

For mean performance, TBI-RDEx ranks second (2.50) behind RDEx (2.39), while substantially outperforming the other algorithms. The p-value of 0.0010 indicates highly significant differences among the algorithms.

The difference between TBI-RDEx's ranking on best values (first) versus mean values (second) can be explained by how TBI works. TBI is good at finding regions that contain very good solutions, which is why it often finds the best result in a given run. However, the tensor approximation does not always capture the landscape accurately. When it works well, TBI leads the search to excellent solutions. When it does not, the search may start in a less useful region (sub optima). This inconsistency shows up as higher variation across runs on some functions (e.g., F16, F19), which increases the mean error even when the best values are competitive. RDEx, which uses standard random initialization, performs more consistently but rarely finds the exceptional solutions that TBI can reach. In short, TBI is better suited for problems where finding the best possible solution matters most, while random initialization may be preferred when consistent average performance is the goal.

D. Discussion

The experimental results demonstrate that TBI provides meaningful advantages for population-based optimization. TBI-RDEx excels at finding superior best solutions, ranking

TABLE II
COMPREHENSIVE COMPARISON OF TBI-RDEX WITH STATE-OF-THE-ART ALGORITHMS ON CEC2025 BENCHMARK (D=30, 25 RUNS)

Func	TBI-RDEx			RDEx			mLSHADE-RL			iEACOP			jSOa		
	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std
F1	**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	3.57E-07	2.85E-06	8.17E-07	**0.00E+00**	5.68E-15	7.11E-15
F2	**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	8.75E-09	1.20E-06	1.68E-06	0.00E+00	0.00E+00	0.00E+00
F3	**0.00E+00**	2.80E+01	1.03E+01	3.18E+01	3.18E+01	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00	4.17E-08	1.91E+00	1.99E+00	**0.00E+00**	4.32E-14	2.48E-14
F4	**0.00E+00**	**2.27E+00**	1.88E+00	**0.00E+00**	2.98E+00	2.20E+00	**0.00E+00**	6.93E+00	1.52E+01	1.59E+01	2.99E+01	7.31E+00	5.86E+01	5.86E+01	0.00E+00
F5	**0.00E+00**	1.51E-08	2.79E-08	**0.00E+00**	**7.53E-09**	2.02E-08	2.98E+00	8.08E+00	3.14E+00	2.73E-03	1.32E-01	2.05E-01	5.97E+00	1.08E+01	1.80E+00
F6	3.17E+01	3.56E+01	2.45E+00	3.17E+01	3.60E+01	2.84E+00	**0.00E+00**	2.95E-03	1.05E-02	4.17E+01	5.39E+01	5.75E+00	1.14E-13	**2.74E-09**	9.47E-09
F7	2.20E+01	3.45E+01	8.12E+00	2.20E+01	3.26E+01	7.00E+00	3.59E+01	3.98E+01	3.17E+00	**1.49E+01**	**2.75E+01**	6.33E+00	3.57E+01	4.17E+01	3.00E+00
F8	**0.00E+00**	8.00E-01	6.83E-01	**0.00E+00**	**0.00E+00**	0.00E+00	3.72E+00	7.95E+00	2.66E+00	3.12E-10	2.38E+00	9.73E+00	7.96E+00	1.12E+01	1.99E+00
F9	3.69E-02	6.18E-01	6.17E-01	5.90E+00	1.63E+02	3.41E+02	**0.00E+00**	**0.00E+00**	0.00E+00	1.28E+03	2.64E+03	4.67E+02	**0.00E+00**	9.09E-15	3.15E-14
F10	**1.71E-12**	6.76E-01	1.22E+00	5.77E-07	**1.21E-01**	3.40E-01	1.02E+03	1.47E+03	2.92E+02	1.39E+01	3.16E+01	1.34E+01	9.53E+02	1.44E+03	2.61E+02
F11	1.36E-12	8.13E+00	3.86E+01	**0.00E+00**	**3.47E-02**	3.66E-02	9.95E-01	8.12E+00	1.10E+01	6.94E+02	1.50E+03	4.76E+02	2.12E-05	1.51E+01	2.37E+01
F12	1.31E+01	**1.48E+01**	3.59E+00	1.42E+01	1.57E+01	1.24E+00	4.70E+02	1.18E+03	4.34E+02	3.09E+01	1.58E+02	1.91E+02	**3.93E+00**	2.22E+02	1.80E+02
F13	2.00E+01	2.03E+01	4.76E-01	2.00E+01	2.01E+01	4.23E-02	**2.21E+00**	1.97E+01	8.52E+00	5.29E+01	1.02E+02	2.75E+01	7.58E+00	**1.90E+01**	3.42E+00
F14	2.42E-01	7.10E-01	1.00E+00	4.26E-01	**5.89E-01**	2.19E-01	7.51E+00	2.28E+01	3.42E+00	9.01E+00	7.86E+01	8.31E+01	**6.54E-02**	1.64E+01	9.05E+00
F15	**4.30E-03**	**4.38E-02**	3.60E-02	6.34E-01	1.01E+00	4.41E-01	1.33E+00	1.25E+01	1.32E+01	1.26E+02	3.75E+02	1.49E+02	5.32E-01	3.16E+00	2.16E+00
F16	**5.26E-05**	5.68E+01	1.07E+02	2.06E+01	**2.22E+01**	6.66E-01	3.26E+00	5.57E+01	5.83E+01	4.00E+01	1.08E+02	6.70E+01	8.34E+00	4.40E+01	4.59E+01
F17	**3.23E-01**	**3.67E+00**	7.35E+00	3.54E-01	8.51E+00	1.04E+01	1.40E+01	3.60E+01	9.18E+00	8.38E+01	2.28E+02	5.69E+01	2.26E+01	3.42E+01	6.32E+00
F18	1.44E+00	**4.00E+00**	1.39E+00	2.58E+00	4.33E+00	1.56E+00	2.12E+01	3.05E+01	7.56E+00	1.73E+01	7.72E+01	7.24E+01	**6.20E-01**	1.92E+01	5.36E+00
F19	**1.80E-05**	2.40E+01	9.00E+01	2.04E+01	4.29E+01	6.86E+01	5.11E+00	1.09E+01	4.88E+00	1.02E+02	1.84E+02	3.12E+01	2.39E+00	**5.01E+00**	1.17E+00
F20	1.00E+02	1.37E+02	4.88E+01	1.00E+02	1.11E+02	3.35E+01	1.71E+01	4.15E+01	1.04E+01	2.13E+02	2.24E+02	6.85E+00	**8.99E+00**	**2.82E+01**	7.78E+00
F21	**1.38E-04**	**1.38E-02**	1.50E-02	1.00E+02	1.12E+02	2.43E+01	2.04E+02	2.08E+02	2.02E+00	1.00E+02	1.00E+02	1.56E-08	2.08E+02	2.11E+02	1.70E+00
F22	2.00E+02	2.08E+02	2.76E+01	2.00E+02	2.00E+02	0.00E+00	**1.00E+02**	**1.00E+02**	0.00E+00	3.61E+02	3.77E+02	1.16E+01	**1.00E+02**	**1.00E+02**	0.00E+00
F23	**2.00E+02**	**2.00E+02**	1.24E-13	**2.00E+02**	**2.00E+02**	2.20E-13	3.40E+02	3.57E+02	7.50E+00	4.35E+02	4.42E+02	3.89E+00	3.47E+02	3.54E+02	4.46E+00
F24	4.20E+02	4.20E+02	1.50E-02	4.20E+02	4.20E+02	1.44E-02	4.15E+02	4.25E+02	4.44E+00	**3.83E+02**	**3.86E+02**	1.65E+00	4.24E+02	4.29E+02	2.25E+00
F25	6.21E+02	7.10E+02	3.05E+01	8.28E+02	8.28E+02	1.26E-02	3.79E+02	**3.81E+02**	2.67E+00	**2.00E+02**	1.06E+03	4.73E+02	3.87E+02	3.87E+02	7.78E-03
F26	**4.56E+02**	**4.70E+02**	8.94E+00	4.81E+02	4.91E+02	5.35E+00	8.54E+02	9.91E+02	7.57E+01	4.97E+02	5.12E+02	6.77E+00	8.68E+02	9.47E+02	3.92E+01
F27	4.55E-13	4.55E-13	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00	4.88E+02	5.04E+02	9.06E+00	3.00E+02	3.19E+02	4.34E+01	4.85E+02	4.99E+02	7.06E+00
F28	1.18E+03	1.84E+03	4.11E+02	1.30E+03	1.64E+03	3.02E+02	**3.00E+02**	**3.00E+02**	4.33E-13	3.58E+02	5.19E+02	4.86E+01	**3.00E+02**	3.05E+02	2.28E+01
F29	1.87E+03	1.93E+03	1.32E+02	8.53E+02	8.55E+02	6.79E-01	**3.65E+02**	**4.27E+02**	2.06E+01	2.41E+03	3.22E+03	5.44E+02	3.65E+02	4.28E+02	1.56E+01

TABLE III
WIN/TIE/LOSS SUMMARY: TBI-RDEX VS OTHER ALGORITHMS

Algorithm	Best			Mean		
	W	T	L	W	T	L
RDEx [14]	14	14	1	12	5	12
mLSHADE-RL [36]	16	4	9	15	2	12
iEACOP [37]	25	0	4	25	0	4
jSOa [38]	15	2	11	16	1	11

TABLE IV
FRIEDMAN TEST RESULTS AND AVERAGE RANKINGS

Algorithm	Rank (Best)	Rank (Mean)
TBI-RDEx (our model)	**2.25**	2.50
RDEx [14]	2.91	**2.39**
jSOa [38]	2.93	3.09
mLSHADE-RL [36]	3.00	3.02
iEACOP [37]	3.91	4.00
Friedman χ^2 (Best): 16.66, p-value: 0.0023		
Friedman χ^2 (Mean): 18.45, p-value: 0.0010		

first among all algorithms (Friedman rank: 2.25), which validates the core premise that CUR decomposition can effectively identify promising regions in the search landscape. In terms of function-type performance, TBI-RDEx shows particular strength on composition functions (F21–F29), where the tensor reconstruction successfully guides the search toward global optima; on unimodal functions (F1–F2), all algorithms achieve optimal solutions, while on simple multimodal functions, performance is competitive with mLSHADE-RL and jSOa. On hybrid functions (F11–F20), TBI-RDEx demonstrates consistent improvement over iEACOP (86.2% win rate) and competitive performance against other algorithms, with notable wins on F12, F15, F17, and F18.

The framework has limitations. On F28–F29, mLSHADE-RL and jSOa achieve substantially better results; these composition functions may not exhibit the low-rank structure that TBI exploits, suggesting that the tensor approximation quality is problem-dependent. Regarding computational overhead, the TBI phase consumes only 3.4% of the total evaluation budget while providing significant improvements on 16–25 functions (depending on the competing algorithm), demonstrating an excellent trade-off between overhead and performance gain.

V. CONCLUSION

This paper introduced TBI, a TBI framework that exploits low-rank structure in optimization landscapes to guide population-based metaheuristics toward promising regions. By partitioning high-dimensional spaces into dimension groups and applying CUR decomposition to auxiliary tensors, TBI approximates the fitness landscape from sparse samples without exhaustive evaluation. The experimental results on CEC2025 benchmarks confirm that informed initialization translates to better optimization outcomes. TBI-RDEx found superior best solutions on 16 functions while losing on only one, a statistically significant improvement over baseline RDEx. The framework proved its particular effectiveness. The approach has limitations. Mean performance showed mixed results, indicating that tensor reconstruction can occasionally mislead the search toward local optima on highly irregular landscapes. Composition functions F28–F29 proved challenging, suggesting that not all problem classes exhibit the low-rank properties TBI exploits. Future work will explore adaptive mechanisms to

detect when tensor guidance is beneficial, direct tensor decompositions such as Tensor Cross Interpolation (TCI) that operate on tensor fibers without requiring matricization, integration with other metaheuristics, and extension to dynamic and multi-objective optimization.

ACKNOWLEDGEMENTS

This research was supported by the Commonwealth through an Australian Government Research Training Program Scholarship [DOI: https://doi.org/10.82133/C42F-K220].

REFERENCES

[1] H. Eskandar, A. Sadollah, A. Bahreininejad, and M. Hamdi, "Water cycle algorithm—a novel metaheuristic optimization method for solving constrained engineering optimization problems," *Computers & Structures*, vol. 110–111, pp. 151–166, 2012.

[2] X.-S. Yang, S. Deb, Y.-X. Zhao, S. Fong, and X. He, "Swarm intelligence: past, present and future," *Soft Comput*, vol. 22, no. 18, pp. 5923–5933, 2018.

[3] Q. Li, S.-Y. Liu, and X.-S. Yang, "Influence of initialization on the performance of metaheuristic optimizers," *Applied Soft Computing*, vol. 91, p. 106193, 2020.

[4] I. O. Essiet, Y. Sun, and Z. Wang, "Optimized energy consumption model for smart home using improved differential evolution algorithm," *Energy*, vol. 172, pp. 354–365.

[5] Z. Lyu, G. K. W. Kenway, and J. R. R. A. Martins, "Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark," *AIAA Journal*, vol. 53, no. 4, pp. 968–985, Apr. 2015.

[6] M. Feurer and F. Hutter, *Hyperparameter Optimization*. Cham: Springer International Publishing, 2019, pp. 3–33.

[7] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020.

[8] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jan. 2016.

[9] B. Li, Z. Wei, J. Wu, S. Yu, T. Zhang, C. Zhu, D. Zheng, W. Guo, C. Zhao, and J. Zhang, "Machine learning-enabled globally guaranteed evolutionary computation," *Nat Mach Intell*, vol. 5, no. 4, pp. 457–467, 2023.

[10] X. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*, 1st ed. Wiley, Jun. 2010.

[11] S. A. Kauffman, *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University PressNew York, NY, Jun. 1993.

[12] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[13] 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.

[14] S. Tao, "Sop technical report: Rdex algorithm for ieee cec 2025 competition," GitHub Repository, Tech. Rep., 2025, technical report for the IEEE Congress on Evolutionary Computation (CEC) 2025 Competition. [Online]. Available: https://github.com/SichenTao/IEEE-CEC-2025-Competition-RDEx/blob/main/SOP_Technical_Report.pdf

[15] M. Sarhani, S. Voß, and R. Jovanovic, "Initialization of metaheuristics: comprehensive review, critical analysis, and research directions," *Int Trans Operational Res*, vol. 30, no. 6, pp. 3361–3397, 2023.

[16] A. Olsson, G. Sandberg, and O. Dahlblom, "On latin hypercube sampling for structural reliability analysis," *Structural Safety*, vol. 25, no. 1, pp. 47–68, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167473002000395

[17] 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.

[18] S. Elsayed, R. Sarker, and C. A. Coello Coello, "Sequence-based deterministic initialization for evolutionary algorithms," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2911–2923, 2017.

[19] I. Poikolainen, F. Neri, and F. Caraffini, "Cluster-based population initialization for differential evolution frameworks," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, ser. Communications in Computer and Information Science, vol. 444. Springer, 2015, pp. 605–614.

[20] L. Bajer, Z. Pitra, and M. Holeňa, "A population initialization method for evolutionary algorithms based on clustering and cauchy deviates," *Expert Systems with Applications*, vol. 60, pp. 294–310, 2016.

[21] O. N. Oyelade, A. E. Ezugwu, A. K. Saha, N. V. Thieu, and A. H. Gandomi, "Deep learning at the service of metaheuristics for solving numerical optimization problems," *Neural Comput & Applic*, vol. 37, no. 27, pp. 22 493–22 528, 2025.

[22] M. Premkumar, G. Sinha, M. D. Ramasamy, S. Sahu, C. B. Subramanyam, R. Sowmya, L. Abualigah, and B. Derebew, "Augmented weighted k-means grey wolf optimizer: An enhanced metaheuristic algorithm for data clustering problems," *Sci Rep*, vol. 14, no. 1, p. 5434, 2024.

[23] C. He, S. Huang, R. Cheng, K. C. Tan, and Y. Jin, "Evolutionary multiobjective optimization driven by generative adversarial networks (GANs)," *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3129–3142, 2021, arXiv preprint: 1910.04966 (October 2019).

[24] H. Cheng, G.-G. Wang, L. Chen, and R. Wang, "A dual-population multi-objective evolutionary algorithm driven by generative adversarial networks for benchmarking and protein-peptide docking," *Computers in Biology and Medicine*, vol. 168, p. 107727, January 2024.

[25] Y. Guo, L. Zhao, R. Zhang, H. Han, and X. Yin, "Multi-objective combinatorial generative adversarial network: Mo-cgan for combinatorial optimization problems," *Algorithms*, vol. 13, no. 9, p. 221, 2020.

[26] M. Fathallah, M. Sakr, and S. Eletriby, "Stabilizing and improving training of generative adversarial networks through identity blocks and modified loss function," *IEEE Access*, vol. 11, pp. 43 276–43 285, 2023.

[27] X. Guo, J. Hu, H. Yu, M. Wang, and B. Yang, "A new population initialization of metaheuristic algorithms based on hybrid fuzzy rough set for high-dimensional gene data feature selection," *Computers in Biology and Medicine*, vol. 166, p. 107538, 2023.

[28] W. Huang, Y. Liu, and X. Zhang, "Hybrid particle swarm optimization algorithm based on the theory of reinforcement learning in psychology," *Systems*, vol. 11, no. 2, p. 83, 2023.

[29] R. Pitakaso, T. Srichok, S. Khonjun, N. Nanthasamroeng, A. Sawettham, P. Khampukka, S. Dinkoksung, K. Jungvimut, G. Jirasirilerd, C. Supasarn, P. Mongkhonngam, and Y. Boonarree, "A hybrid deep reinforcement learning and metaheuristic framework for heritage tourism route optimization in warin chamrap's old town," *Heritage*, vol. 8, no. 8, p. 301, 2025.

[30] B. Mu, X. Qin, S. Yuan, and S. Wen, "Ppso: Pca based particle swarm optimization for solving conditional nonlinear optimal perturbation," *Computers & Geosciences*, vol. 83, pp. 65–71, 2015.

[31] K. Huang, N. D. Sidiropoulos, and A. Swami, "Nonnegative matrix factorization by optimization on the stiefel manifold with svd initialization," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2016, pp. 1138–1144.

[32] Q. Wang, L. Zhang, S. Wei, B. Li, and Y. Xi, "Tensor factorization-based particle swarm optimization for large-scale many-objective problems," *Swarm and Evolutionary Computation*, vol. 69, p. 100995, 2022.

[33] X. Wang, Y. Zhao, L. Tang, and X. Yao, "MOEA/d with spatial–temporal topological tensor prediction for evolutionary dynamic multiobjective optimization," *IEEE Trans. Evol. Computat.*, vol. 29, no. 3, pp. 764–778, 2025.

[34] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[35] R. Penrose, "A generalized inverse for matrices," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 51, no. 3, p. 406–413, 1955.

[36] D. Chauhan, A. Trivedi, and Shivani, "A multi-operator ensemble lshade with restart and local search mechanisms for single-objective optimization," 2024.

[37] A. Tangherloni, V. Coelho, F. M. Buffa, and P. Cazzaniga, "A Modified EACOP Implementation for Real-Parameter Single Objective Optimization Problems," in *2024 IEEE Congress on Evolutionary Computation (CEC)*. Yokohama, Japan: IEEE, jun 2024, pp. 1–8.

[38] P. Bujok, "Progressive Archive in Adaptive jSO Algorithm," *Mathematics*, vol. 12, no. 16, p. 2534, aug 2024.