ASEAN
Journal of Scientific and Technological Reports
Online ISSN:2773-8752

Research article

Linear Ensemble Algorithm: A Novel Meta-Heuristic Approach for Solving Constrained Engineering Problems

Phatnathee Wongsrisai[1], and Prompong Sugunnasil[2]

[1] Faculty of Engineering, Chiang Mai University, Chiang Mai, 50200, Thailand
[2] Faculty of Engineering, Chiang Mai University, Chiang Mai, 50200, Thailand
* Correspondence: prompong.sugunnasil@cmu.ac.th

Citation:
Wongsrisai, P.; Sugunnasil, P. Linear ensemble algorithm: A novel meta-heuristic approach for solving constrained engineering problems. *ASEAN J. Sci. Tech. Report.* **2025**, *28*(6), e257673. https://doi.org/10.55164/ajstr.v28i6.257673.

Article history:
Received: January 28, 2025
Revised: September 5, 2025
Accepted: September 10, 2025
Available online: October 14, 2025

Abstract: This research This study presents the Linear Ensemble Algorithm (LEAL), which couples evolutionary search with local, linear regression–based surrogates and a neighbor-guided linear combination scheme for constrained engineering problems and standard benchmarks. For single-objective problems, LEAL frequently attains or closely approaches global optima on multimodal functions such as Rastrigin and Griewank, yielding ~55–85% lower mean error than GA, DE, and PSO, and it can occasionally uncover best-known minima in engineering tasks (e.g., Pressure Vessel), indicating an ability to exploit intricate design trade-offs. For multi-objective problems, LEAL generates feasible Pareto fronts but generally trails NSGA-II in convergence and efficiency, exhibiting higher GD$^+$, longer runtimes, and greater memory usage (often by one to two orders of magnitude). These outcomes reflect the computational overhead of maintaining local surrogate ensembles: while LEAL can produce high-quality solutions, its average performance, runtime, and memory footprint are often inferior to lightweight baselines. Comparisons with CMA-ES, Bayesian Optimization, and SSA-NSGA-II confirm the same trade-offs. Overall, LEAL is a robust yet computationally intensive option best suited when ultimate solution quality outweighs runtime; future work will focus on improving efficiency, streamlining ensemble components, and extending applicability to large-scale and dynamic problems.

Keywords: Linear ensemble algorithm; evolution strategy; engineering design; multi-objective problems

1. Introduction

Optimization plays a pivotal role in modern engineering design and decision-making, where solutions must often satisfy multiple objectives and stringent constraints. Constrained optimization problems are particularly challenging due to their nonlinear and non-convex nature, high dimensionality, and the presence of numerous local optima [1-2]. Nonlinear constraints frequently generate complex feasible regions, while high-dimensional decision spaces expand the search domain exponentially. Additionally, the prevalence of multiple local optima increases the risk of premature convergence, making it challenging for algorithms to locate global or near-global solutions consistently. These characteristics are common in real-world applications such as the design of structural components, energy systems, and pressure vessels, where the

quality of solutions directly impacts safety, cost, and performance [3-4]. To address these challenges, meta-heuristic algorithms have become widely adopted because of their flexibility, gradient-free search mechanisms, and ability to approximate near-optimal solutions across diverse problem domains. Among the most established methods are the Genetic Algorithm (GA) [5], Particle Swarm Optimization (PSO) [6], Differential Evolution (DE) [7], and the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [8]. While these algorithms are versatile, they often suffer from limitations such as premature convergence, stagnation in local optima, and difficulties balancing exploration and exploitation. Handling constraints further complicates the search, as infeasible solutions may dominate the population or lead to excessive computational overhead [9].

In recent years, several advanced meta-heuristics and hybrid strategies have been proposed. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [10] is known for robust self-adaptation of search distributions. Recent studies have further extended CMA-ES with improved constraint handling [11, 12], safe optimization variants [13], and surrogate-assisted hybrids [14]. Another direction is Bayesian Optimization (BO) [15, 16], which utilizes surrogate models and acquisition functions to guide exploration. Recent work has successfully applied BO to constrained black-box problems [17], hybridized it with local solvers such as IPOPT [18], and extended it to multi-objective optimization [19,20]. More recently, Surrogate-assisted Evolutionary Algorithms (SAEAs) have gained attention for their ability to reduce computational cost while maintaining solution quality [21]. For example, the Surrogate-assisted NSGA-II (SSA-NSGA-II) has been demonstrated to be effective for high-dimensional and expensive optimization tasks [22,23]. Despite these advances, significant gaps remain. Classical meta-heuristics often lack robustness in handling constraints, while modern methods such as CMA-ES, BO, and surrogate-assisted approaches incur high computational costs or require careful model management. There is therefore a strong need for algorithms that can balance exploration and exploitation, ensure feasibility under nonlinear constraints, and deliver competitive solution quality without prohibitive overhead.

To conclude, this study proposes the Linear Ensemble Algorithm (LEAL), a novel meta-heuristic optimization framework that integrates ensemble learning principles with linear regression modeling. LEAL employs a neighbor-based sampling strategy and linear combination mechanisms to predict promising regions in the search space, aiming to improve both solution diversity and convergence stability. The contributions of this work are fourfold: to design an ensemble-based algorithm capable of effectively handling constrained, nonlinear, and non-convex problems. To systematically compare LEAL against both classical meta-heuristics (GA, PSO, DE, NSGA-II) and modern algorithms (CMA-ES, BO, SSA-NSGA-II) across benchmark functions and engineering design problems. To evaluate LEAL's performance in terms of solution quality, runtime efficiency, memory usage, and constraint-handling ability. To provide insights into the strengths, limitations, and potential applications of ensemble-based meta-heuristics in engineering optimization. Through this comprehensive evaluation, the study seeks to establish LEAL as a competitive addition to the family of meta-heuristic algorithms, bridging the gap between classical heuristics and modern surrogate-assisted approaches.

2. Materials and Methods

2.1 Overview of LEAL

The Linear Ensemble Algorithm (LEAL) is a novel surrogate-assisted meta-heuristic framework designed for solving constrained engineering optimization problems. Unlike classical evolutionary algorithms, which rely solely on stochastic search operators such as crossover and mutation, LEAL integrates local linear surrogate modeling into the search loop. The key hypothesis is that within sufficiently small regions, the objective landscape can be approximated by linear functions. By partitioning the decision space, generating neighbors, and fitting regression planes, LEAL analytically solves systems of linear equations to generate promising candidate solutions. This mechanism is combined with evolutionary variation and survival strategies, creating a balance between deterministic guidance and stochastic diversity. This design aligns with recent advances in surrogate-assisted evolutionary algorithms, which emphasize the importance of local modeling and ensemble learning for computationally expensive optimization [21].

2.2 Algorithm Components

2.2.1 Sampling Strategy

The optimization begins with a domain-splitting sampling strategy. The feasible range of each decision variable is recursively divided into smaller subregions, from which low-discrepancy samples are drawn. This guarantees that all areas of the search space are represented and avoids early convergence toward narrow regions. An adaptive step size, Δx, calculated from the variable ranges, gradually decreases as the search progresses, enabling a natural transition from global exploration to local exploitation.

2.2.2 Neighbor Search

For each incumbent solution x, a neighborhood is generated by perturbing the decision variables with offsets $\{-\Delta x, 0, +\Delta x\}$. Formally, neighbors are defined as

$$x' = x + \delta, \quad \delta \in \{-\Delta x, 0, +\Delta x\}^n \tag{1}$$

where n denotes the dimensionality of the problem. Because $\{-\Delta x, 0, +\Delta x\}^n$ yields up to 3^n candidates, a subset of five neighbors is sampled per incumbent to capture local variation without excessive computational burden. These neighbors enrich the local information surrounding each individual, which forms the basis for surrogate construction.

2.2.3 Linear Regression Modeling

Each incumbent and its neighbors form a training set for building a local linear regression surrogate. Here, the decision variables serve as the predictors, while the objective function values (fitness values) are treated as the targets of the regression model to approximate the local fitness landscape; this approximation is then used to generate new candidate solutions. For constrained problems, we additionally fit a local linear surrogate for the aggregate constraint violation $v(x)$ (e.g., the sum of positive normalized violations) to guide feasibility-first ranking. The regression takes the form

$$\hat{f}(x) = \beta_0 + \sum_{i=1}^{n} \beta_i x_i \tag{2}$$

where coefficients β_0, β_i are estimated using least-squares fitting. These surrogates provide piecewise-linear approximations of the fitness landscape. Rather than fitting a single global model, LEAL simultaneously constructs multiple overlapping local models, forming a micro-ensemble that predicts descent directions in different subregions [21]. To improve numerical stability, inputs and responses are z-score normalized within each neighborhood, and a small L_2 ridge term (e.g., $\lambda \approx 10^{-6}$) is applied if the design matrix is ill-conditioned.

2.2.4 Equation Point Generation

The ensemble of regression equations is then exploited to generate promising candidates. Subsets of equations are selected, and their intersections are obtained by solving.

$$Ax = b, \quad A \in \mathbb{R}^{d \times n},\ b \in \mathbb{R}^d \tag{3}$$

where rows of A represent regression coefficients and entries of b correspond to intercept terms. When $d \neq n$, intersections are computed in the least-squares sense via the Moore–Penrose pseudoinverse. The resulting solutions, referred to as equation points, represent locations where multiple surrogates agree on the fitness landscape. Equation points are clipped within bounds and repaired if they violate feasibility constraints, ensuring that the search remains within the admissible space. Candidates are prioritized using a feasibility-first infill rule that minimizes predicted $\hat{v}(x)$ and, among feasible points, predicted $\hat{f}(x)$ before true evaluation. Final acceptance is always based on true objective/constraint evaluations

2.2.5 Infill Criterion and Mating

In addition to surrogate-driven candidates, evolutionary variation operators are employed to maintain exploration. Parent selection is based on an infill criterion that prioritizes feasibility and minimizes constraint violation. Binary tournament selection is applied, followed by simulated binary crossover (SBX) with probability 0.9 and polynomial mutation (PM) with probability $1/n$. By default, runs terminate after 100

generations. This stochastic mechanism introduces diversity into the search, preventing premature convergence while complementing the deterministic surrogate guidance.

2.2.6 Survival Strategy

The survival stage determines which individuals are passed on to the next generation. A **feasibility-first rule** is applied, followed by non-dominated sorting where appropriate. To preserve diversity, the Least Hypervolume Contribution method is employed, whereby solutions that contribute the least to the overall hypervolume are discarded. This survival mechanism ensures that the evolving population remains both feasible and well-distributed across the objective space. For multi-objective problems, survival and selection adopt Pareto dominance with hypervolume-based pruning, ensuring a well-distributed set of feasible solutions along the Pareto front.

2.3 Algorithm Workflow

The iterative process of LEAL can be described as follows. The algorithm begins with domain-splitting sampling, followed by feasibility repair of initial candidates. In each generation, neighbors are generated, regression models are constructed, and equation points are derived analytically. After repairing infeasible solutions and eliminating duplicates, evolutionary variation operators are applied. Offspring and parents are merged, and survival selection determines the next generation. For multi-objective problems, selection follows Pareto dominance, and survival employs hypervolume-based pruning to preserve a well-distributed Pareto front. This process is repeated until the maximum number of generations or the termination criteria are satisfied. In practice, LEAL is most effective on rugged, multimodal landscapes; for time-critical or resource-limited scenarios, lightweight baselines (DE/GA/PSO; NSGA-II for multi-objective) remain more efficient.

Algorithm 1 Linear Ensemble Algorithm (LEAL)

```
Initialize population P using the sampling strategy
Evaluate objective function and constraints for P
while termination criteria not met do
    for each individual i in P do
        Generate neighbors N_i
        Evaluate objective function for N_i
        Fit linear regression models using i and N_i
    end for
    Generate new candidates by solving linear equations
    if crossover and mutation are enabled then
        Perform selection, crossover, and mutation
        Evaluate offspring
    end if
    Merge P and offspring into combined population C
    Select next generation P by survival strategy
    Adjust search boundaries based on P
    Update sampling strategy with new bounds
end while
return best solution found
```

Figure 1. Linear Ensemble Algorithm (LEAL) Pseudo-code

2.4 Implementation Details

LEAL was implemented in Python 3.10 using the pymoo framework [24] for evolutionary operators and population management. Additional surrogate-assisted modules were tested using pysamoo [25], and Bayesian optimization experiments employed scikit-optimize (skopt) [26] for Gaussian Process, Random Forest, and Extra Trees-based models. Scikit-learn was used to construct regression models, and joblib was used to enable parallel processing. Custom classes included Le_sampling for domain partitioning, Le_Mating for crossover and mutation, and LeastHypervolumeContributionSurvival for survival management. consistent with hypervolume-based selection literature [27]. Benchmark problems included well-known single-objective test functions (Ackley, Rastrigin, Griewank, G-series) and engineering problems such as

Piston Lever Optimization Design (PLOD), Tubular Column Design (TCD), Reinforced Concrete Beam Design (RCBD), and the Pressure Vessel design. To evaluate novelty and competitiveness, additional state-of-the-art algorithms were implemented for comparison: Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [11, 13, 14], Bayesian Optimization (BO) [17–20], and Surrogate-Assisted NSGA-II (SSA-NSGA-II) [22, 23]. CMA-ES adapts the covariance of a Gaussian distribution to guide search directions, with recent work enhancing constraint-handling and safe optimization. BO employs probabilistic surrogates with acquisition functions such as Expected Improvement (EI), with multi-objective extensions via (differentiable/noisy) Expected Hypervolume Improvement [29,30] SSA-NSGA-II integrates surrogate modeling into NSGA-II, alternating between true and predicted evaluations to reduce computational cost while retaining convergence accuracy. Performance indicators such as modified GD/IGD were computed following the EMO literature [28] This design and evaluation protocol are in line with recent surveys on surrogate-assisted evolutionary optimization [31,32].

2.5 Parameter Settings

LEAL was configured with a population size of 100, five neighbors per solution, and five regression combinations for generating equation points. SBX crossover was applied with probability 0.9, and PM mutation with probability $1/n$. Each run was executed for 100 generations, with constraint violations repaired immediately and duplicate individuals eliminated. For the comparative algorithms, CMA-ES was run with a population size of 50 and adaptive covariance updates [11]. BO was configured with Gaussian Process surrogates using Expected Improvement, as well as Random Forest and Extra-Trees variants [18–20]. SSA-NSGA-II utilized a surrogate population size of 100, 50 surrogate generations, and evolutionary operators consistent with LEAL [22,23]. Preliminary sensitivity tests confirmed the robustness of these parameter settings.

2.6 Theoretical Analysis

LEAL rests on three theoretical principles: local convexity, domain decomposition, and surrogate forecasting. By partitioning the search space into fine subregions, the assumption of local convexity ensures that regression planes provide reliable local predictions. Domain decomposition guarantees comprehensive exploration, while overlapping surrogates form an ensemble that reduces bias, consistent with ensemble learning theory [21]. The computational complexity per generation is dominated by neighbor generation at $O(N \cdot n)$, regression fitting at $O(N \cdot n^2)$, and equation solving at $O(k^3)$, where N is population size, n is dimensionality, and k is the number of regression combinations. The overall complexity is therefore

$$O(G \cdot N \cdot (n^2 + k^3)) \tag{4}$$

with G denoting the number of generations.

When compared with modern algorithms, CMA-ES offers robustness through covariance adaptation [11, 13], BO frames optimization is framed as sequential decision-making under uncertainty [17-20], and SSA-NSGA-II leverages surrogate predictions to reduce evaluation costs [22, 23]. LEAL differentiates itself by using equation-point generation from regression ensembles, offering a novel and complementary approach.

2.7 Advantages and Limitations

LEAL's hybrid structure provides deterministic surrogate guidance and stochastic evolutionary diversity, enabling efficient search in multimodal and constrained landscapes. Its hypervolume-based survival preserves both feasibility and diversity. However, regression fitting and equation solving impose computational overhead, particularly in high-dimensional problems where numerical instability may arise. Parameter sensitivity is another limitation; however, preliminary analyses have helped identify robust settings. Comparative studies with CMA-ES, BO, and SSA-NSGA-II show that while LEAL does not dominate universally, it offers a structure-aware and competitive alternative for constrained engineering design.

3. Results and Discussion

3.1 Experimental Setup

To evaluate the performance of the Linear Ensemble Algorithm (LEAL), a series of experiments was conducted on both constrained engineering optimization problems and standard single-objective benchmark functions. LEAL's performance was compared against well-established algorithms, including NSGA-II [8], Genetic Algorithm (GA) [5], Differential Evolution (DE) [7], and Particle Swarm Optimization (PSO) [6]. In addition, recent optimization approaches, such as the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [11, 13, 14], Bayesian Optimization (BO) [17-20], and Surrogate-assisted NSGA-II (SSA-NSGA-II) [22, 23], were also considered to provide a broader and more up-to-date comparison.

3.1.1 Evaluation Metrics

The experiments employed several evaluation metrics to assess the algorithm's performance. The mean objective value represented the average objective function values obtained over multiple runs, providing a measure of general performance. The standard deviation was used to quantify the variability of objective values, reflecting the consistency of each algorithm. The minimum value indicated the best (lowest) objective function value achieved across runs, highlighting the algorithm's capacity to find optimal solutions. Finally, the runtime measured the computational time required by the algorithm to complete, offering insights into efficiency. In addition, for multi-objective problems, the Generational Distance Plus (GD^+) indicator [28] was employed to evaluate the closeness of the obtained Pareto front to the true Pareto front, while memory usage was recorded to assess computational resource requirements

3.1.2 Parameter Settings

In all experiments, a population size of 100 was used for LEAL, while the settings for other algorithms were adjusted accordingly. The number of generations was set to 100 for all algorithms, ensuring a fair comparison. LEAL utilized a neighbor population size (neighbour_pop) of 5, and linear combination elements (linear_comb_element) were also set to 5. Crossover and mutation rates for all algorithms were configured using their default settings. For the recent methods added, CMA-ES was run with a default step-size adaptation strategy [11,13], Bayesian Optimization (BO) employed Gaussian Process regression with Expected Improvement (EI) as the acquisition function [17-20,29,30], and SSA-NSGA-II was configured with an initial design of experiments (DOE) size of 50 and 10 surrogate infills per iteration [22,23].

3.1.3 Benchmark Problems

The engineering optimization problems include the Piston Lever Optimization Design (PLOD), Tubular Column Design (TCD), and Pressure Vessel problems [5,19]. These scenarios originate from practical engineering domains where mechanical and structural components must be optimized under rigorous physical and material constraints. The PLOD problem focuses on determining lever dimensions and configurations that minimize stress or deflection in mechanical assemblies. Similarly, TCD involves optimizing the geometry and material distribution of a tubular column to ensure it can support specified loads with minimal weight and deformation. The Pressure Vessel problem aims to reduce costs by optimizing the vessel's thickness and material usage under high-pressure conditions, while adhering to strict safety and fabrication standards. Each of these engineering tasks combines nonlinear constraints, complex feasible regions, and real-world objectives, making them excellent testbeds for assessing an algorithm's ability to handle intricate design landscapes and produce feasible, high-quality solutions. In the realm of single-objective benchmark functions, the Ackley, G12, G2, G4, G6, Griewank, Himmelblau, Rastrigin, and Zakharov problems provide a broad range of computational challenges. Functions like Ackley, Griewank , and Rastrigin are well-known for their multimodality, featuring numerous local minima that test an algorithm's capacity to avoid premature convergence and effectively explore the search space. Himmelblau's function and the G-series constrained problems (G12, G2, G4, G6) introduce additional complexity by imposing nonlinear constraints and known global optima, thereby gauging the precision and constraint-handling abilities of optimization methods. Zakharov and similar smoother landscapes assess how efficiently algorithms can move

toward global optima in less rugged, though still challenging, search spaces. Altogether, these benchmarks challenge exploration, exploitation, constraint management, and fine-tuning of solutions, painting a comprehensive picture of an algorithm's strengths and weaknesses. The multi-objective benchmark problems BNH, CTP7, DF1, DF2, DF3, DF5, DF9, DTLZ1, and SRN each pose unique challenges by requiring simultaneous optimization of multiple, often conflicting objectives. Rather than seeking a single global optimum, algorithms must approximate the true Pareto front with a distribution of solutions that reflect different trade-offs among objectives. Tasks like BNH and the CTP family emphasize intricate constraints and highly nonlinear interactions, testing whether an optimizer can stay within feasible regions while preserving diversity among solutions. Problems such as DF1, DF2, DF3, and DF5 further highlight the interplay of complex constraint handling, whereas DF9, DTLZ1, and SRN integrate additional layers of difficulty—ranging from higher-dimensional search spaces to strict requirements on solution quality. Collectively, these benchmarks measure an algorithm's ability to converge near the Pareto front and maintain an even spread of solutions across it, thereby providing a comprehensive evaluation of multi-objective optimization performance [18, 22].

3.2 Results

3.2.1 Engineering Optimization Problems

In examining the performance of the presented algorithms Differential Evolution (DE), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Bayesian Optimization (BO), and the proposed Linear Ensemble Algorithm (LEAL) across three engineering optimization benchmark problems (Piston Lever Optimization Design, Tubular Column Design, and the Pressure Vessel problem), clear trends emerge in terms of solution quality, computational cost, and resource usage. For the Piston Lever Optimization Design (PLOD) (optimum ≈ 2.25), DE, GA, and PSO achieved mean best objectives of 10^1–10^3 scale with relatively low variance, while CMA-ES and BO performed poorly, reaching means of 3.40×10^5 and 2.03×10^5, respectively. LEAL produced a mean of 1.34×10^5, which, although better than CMA-ES and BO, was still orders of magnitude worse than DE, GA, and PSO. LEAL's runtime (7.81 s) was also far higher than all other algorithms (all <1 s), and its memory usage (4.75) exceeded DE (8.36×10^{-1}) and GA (8.55×10^{-1}), though lower than BO (2.95×10^1). For the Tubular Column Design (TCD) (optimum ≈ 10.5), DE, GA, and PSO consistently converged near 2.66×10^1–2.68×10^1 with small variance, confirming strong stability. CMA-ES and BO delivered weaker results, with mean objectives of 2.92×10^1 and 3.26×10^1, respectively. LEAL again lagged, showing a higher mean (3.13×10^1) and variance (6.90) than DE, GA, and PSO. Its runtime averaged 5.53 seconds, markedly slower than the sub-second runtimes of classical algorithms, and its memory usage (3.92) was larger than DE and GA (<1.0), but below BO (2.93×10^1). For the Pressure Vessel problem (optimum ≈ 300), DE, GA, and PSO performed best, with mean objectives ranging from 7.59×10^3 to 9.36×10^3. CMA-ES and BO showed higher means (1.78×10^4 and 3.06×10^4, respectively). LEAL produced the weakest performance, with a mean of 7.55×10^4 and extremely high variance. Its runtime (8.48 s) was substantially greater than others (all <0.3 s except BO at 58.4 s), and its memory usage (6.08) was again much higher than DE and GA, but notably lower than BO (2.95×10^1). Overall, while DE, GA, and PSO provided the most reliable and efficient performance across all three engineering problems, CMA-ES and BO exhibited inconsistent behavior, accompanied by higher computational costs. LEAL, although occasionally capable of finding competitive minimum values, suffered from large performance variance, high runtime overhead, and greater memory consumption, making it less competitive in these engineering design benchmarks.

Table 1. Performance on Engineering Optimization Benchmark Functions

Test problem	Optimum point	Metrics	DE	GA	PSO	CMA-ES	BO	LEAL
Piston Lever Optimization Design (PLOD)	2.25	Mean	4.46×10^{1}	5.04×10^{3}	1.47×10^{3}	3.40×10^{5}	2.03×10^{5}	1.34×10^{5}
		Std	4.41×10^{1}	1.32×10^{4}	6.28×10^{3}	6.01×10^{5}	4.60×10^{5}	3.43×10^{5}
		Min	1.03×10^{0}	1.52×10^{1}	7.66×10^{-1}	1.35×10^{0}	5.53×10^{1}	2.68×10^{1}
		Runtime mean	2.92×10^{-1}	2.96×10^{-1}	3.30×10^{-1}	2.66×10^{-1}	5.04×10^{1}	7.81×10^{0}
		Runtime std	1.61×10^{-2}	1.60×10^{-2}	1.36×10^{-2}	1.15×10^{-1}	3.50	4.55×10^{0}
		Memory mean	8.36×10^{-1}	8.55×10^{-1}	1.97×10^{0}	2.59×10^{-1}	2.95×10^{1}	4.75×10^{0}
		Memory std	5.55×10^{-4}	2.14×10^{-2}	1.50×10^{-2}	4.51×10^{-2}	5.34×10^{-3}	2.29×10^{0}
		Memory min	8.35×10^{-1}	8.41×10^{-1}	1.94×10^{0}	1.20×10^{-1}	2.95×10^{1}	8.94×10^{-1}
Tubular Column Design (TCD)	10.5	Mean	2.68×10^{1}	2.67×10^{1}	2.66×10^{1}	2.92×10^{1}	3.26×10^{1}	3.13×10^{1}
		Std	1.40×10^{-1}	9.09×10^{-2}	5.25×10^{-2}	2.75×10^{0}	1.91×10^{0}	6.90×10^{0}
		Min	2.66×10^{1}	2.65×10^{1}	2.66×10^{1}	2.69×10^{1}	3.03×10^{1}	2.68×10^{1}
		Runtime mean	2.35×10^{-1}	2.68×10^{-1}	2.48×10^{-1}	1.40×10^{-1}	4.96×10^{1}	5.53×10^{0}
		Runtime std	2.17×10^{-2}	5.12×10^{-2}	1.89×10^{-2}	7.33×10^{-2}	5.35×10^{0}	4.87×10^{0}
		Memory mean	8.39×10^{-1}	8.91×10^{-1}	1.39×10^{0}	2.31×10^{-1}	2.93×10^{1}	3.92×10^{0}
		Memory std	1.76×10^{-2}	7.38×10^{-3}	7.93×10^{-3}	5.44×10^{-2}	3.28×10^{-3}	2.97×10^{0}
		Memory min	8.32×10^{-1}	8.79×10^{-1}	1.37×10^{0}	1.00×10^{-1}	2.93×10^{1}	8.94×10^{-1}
Pressure Vessel	300	Mean	9.36×10^{3}	8.83×10^{3}	7.59×10^{3}	1.78×10^{4}	3.06×10^{4}	7.55×10^{4}
		Std	1.30×10^{3}	1.13×10^{3}	6.83×10^{2}	5.28×10^{3}	1.48×10^{4}	7.57×10^{4}
		Min	6.73×10^{3}	7.11×10^{3}	6.65×10^{3}	8.40×10^{3}	1.44×10^{4}	1.50×10^{3}
		Runtime mean	1.90×10^{-1}	2.01×10^{-1}	2.20×10^{-1}	1.42×10^{-1}	5.84×10^{1}	8.48×10^{0}
		Runtime std	5.18×10^{-3}	1.27×10^{-2}	1.26×10^{-2}	6.49×10^{-2}	1.84×10^{1}	7.01×10^{-1}
		Memory mean	8.32×10^{-1}	8.67×10^{-1}	2.03×10^{0}	2.46×10^{-1}	2.95×10^{1}	6.08×10^{0}
		Memory std	1.23×10^{-3}	9.29×10^{-3}	1.09×10^{-2}	5.40×10^{-2}	5.28×10^{-3}	1.30×10^{-1}
		Memory min	8.29×10^{-1}	8.53×10^{-1}	2.01×10^{0}	1.18×10^{-1}	2.95×10^{1}	5.82×10^{0}

3.2.3 Single-Objective Benchmark Functions

Table 2. Performance on Single-Objective Benchmark Functions

Test problem	Optimum point	Metrics	DE	GA	PSO	CMA-ES	BO	LEAL
Ackley	**0**	**Mean**	7.11×10^{-1}	4.99×10^{-1}	3.79×10^{-1}	3.08×10^{0}	1.72×10^{0}	8.30×10^{-1}
		Std	4.65×10^{-1}	3.60×10^{-1}	2.78×10^{-1}	8.00×10^{-1}	7.90×10^{-1}	1.76×10^{0}
		Min	4.93×10^{-2}	1.05×10^{-1}	2.46×10^{-2}	9.33×10^{-1}	6.63×10^{-1}	4.44×10^{-16}
		Runtime	1.79×10^{-1}	1.99×10^{-1}	2.04×10^{-1}	1.28×10^{-1}	3.25×10^{1}	9.35×10^{0}
G2	**0**	**Mean**	-2.74×10^{-1}	-3.83×10^{-1}	-2.62×10^{-1}	-1.89×10^{-1}	-1.54×10^{-1}	-1.52×10^{-1}
		Std	3.33×10^{-2}	4.14×10^{-2}	4.92×10^{-2}	2.27×10^{-2}	2.38×10^{-2}	2.29×10^{-2}
		Min	-3.42×10^{-1}	-4.44×10^{-1}	-3.83×10^{-1}	-2.35×10^{-1}	-2.01×10^{-1}	-2.08×10^{-1}
		Runtime	2.04×10^{-1}	2.43×10^{-1}	2.86×10^{-1}	1.96×10^{-1}	9.78×10^{1}	3.14×10^{1}
G4	**0**	**Mean**	-3.02×10^{4}	-3.01×10^{4}	-3.04×10^{4}	-2.97×10^{4}	-2.92×10^{4}	-2.96×10^{4}
		Std	2.01×10^{2}	1.70×10^{2}	1.52×10^{2}	4.92×10^{2}	5.89×10^{2}	7.26×10^{2}
		Min	-3.06×10^{4}	-3.03×10^{4}	-3.06×10^{4}	-3.03×10^{4}	-3.02×10^{4}	-3.05×10^{4}
		Runtime	1.87×10^{-1}	1.93×10^{-1}	2.15×10^{-1}	1.50×10^{-1}	5.31×10^{1}	8.85×10^{0}
G6	**0**	**Mean**	-5.90×10^{3}	-4.69×10^{3}	-5.74×10^{3}	-4.21×10^{3}	9.15×10^{5}	-5.85×10^{3}
		Std	6.45×10^{2}	1.17×10^{3}	8.64×10^{2}	1.04×10^{3}	1.29×10^{6}	8.27×10^{2}
		Min	-6.85×10^{3}	-6.73×10^{3}	-6.85×10^{3}	-5.42×10^{3}	-5.78×10^{3}	-6.74×10^{3}
		Runtime	1.77×10^{-1}	1.93×10^{-1}	2.10×10^{-1}	1.37×10^{-1}	4.30×10^{1}	7.74×10^{0}
G12	**0**	**Mean**	-9.97×10^{-1}	-9.98×10^{-1}	-9.99×10^{-1}	-8.20×10^{-1}	-9.09×10^{-1}	-9.93×10^{-1}
		Std	4.31×10^{-3}	3.23×10^{-3}	1.49×10^{-3}	1.37×10^{-1}	7.84×10^{-2}	2.02×10^{-2}
		Min	-9.99×10^{-1}	-1.00×10^{0}	-1.00×10^{0}	-9.89×10^{-1}	-9.84×10^{-1}	-1.00×10^{0}
		Runtime	3.94×10^{-1}	4.12×10^{-1}	4.19×10^{-1}	4.74×10^{-1}	4.90×10^{1}	2.92×10^{1}
Griewank	**0**	**Mean**	7.93×10^{-2}	8.19×10^{-2}	7.53×10^{-2}	4.36×10^{-1}	1.16×10^{-1}	3.17×10^{-2}
		Std	4.44×10^{-2}	5.20×10^{-2}	4.87×10^{-2}	1.95×10^{-1}	7.11×10^{-2}	1.01×10^{-1}
		Min	1.60×10^{-2}	2.74×10^{-3}	3.85×10^{-3}	8.97×10^{-2}	1.56×10^{-2}	0.00×10^{0}
		Runtime	1.79×10^{-1}	1.89×10^{-1}	1.94×10^{-1}	1.17×10^{-1}	5.05×10^{1}	4.08×10^{1}
Himmelblau	**0**	**Mean**	7.49×10^{-2}	1.20×10^{-2}	2.52×10^{-2}	4.54×10^{-1}	9.04×10^{-3}	3.80×10^{0}
		Std	9.03×10^{-2}	1.30×10^{-2}	3.00×10^{-2}	1.08×10^{0}	5.35×10^{-3}	8.75×10^{0}
		Min	1.46×10^{-3}	8.08×10^{-5}	3.19×10^{-4}	1.85×10^{-3}	2.97×10^{-3}	1.94×10^{-4}
		Runtime	1.71×10^{-1}	1.89×10^{-1}	1.91×10^{-1}	1.03×10^{-1}	4.72×10^{1}	7.54×10^{0}
Rastrigin	**0**	**Mean**	1.75×10^{-1}	1.85×10^{-1}	2.23×10^{-1}	4.60×10^{0}	3.94×10^{0}	3.59×10^{-2}
		Std	2.66×10^{-1}	3.46×10^{-1}	2.67×10^{-1}	2.94×10^{0}	4.31×10^{0}	1.61×10^{-1}
		Min	8.59×10^{-3}	5.59×10^{-3}	9.29×10^{-4}	6.23×10^{-1}	2.52×10^{-2}	0.00×10^{0}
		Runtime	2.01×10^{-1}	1.94×10^{-1}	1.96×10^{-1}	1.35×10^{-1}	4.19×10^{1}	3.99×10^{1}
Zakharov	**0**	**Mean**	2.69×10^{-2}	2.30×10^{-3}	1.04×10^{-3}	5.70×10^{-2}	8.16×10^{-2}	4.28×10^{-3}
		Std	3.01×10^{-2}	2.20×10^{-3}	1.11×10^{-3}	9.57×10^{-2}	9.08×10^{-2}	9.42×10^{-3}
		Min	1.49×10^{-3}	9.05×10^{-5}	5.08×10^{-5}	1.40×10^{-4}	5.14×10^{-3}	0.00×10^{0}
		Runtime	1.61×10^{-1}	1.79×10^{-1}	1.78×10^{-1}	1.20×10^{-1}	5.46×10^{1}	7.39×10^{0}

When examining the results for each benchmark function, a central point of interest is how closely each algorithm's best-found solutions approximate the known global optimum, which, for all the tested single-objective functions, is zero. On the Ackley function, LEAL achieved an exact minimum of 4.44×10^{-16}, which is practically the global optimum. However, its mean (8.30×10^{-1}) was worse than that of PSO (3.79×10^{-1}), GA (4.99×10^{-1}), and DE (7.11×10^{-1}). CMA-ES and BO performed poorly, with higher mean values (3.08×10^{0} and 1.72×10^{0}, respectively). This shows that while LEAL can sometimes locate the global optimum, it lacks the consistency of classical algorithms. For the G-series constrained problems, LEAL achieved near-optimal minima but less stable averages. On G12, it matched the known optimum (-1.00×10^{0}) but had a slightly inferior mean (-9.93×10^{-1}) compared to GA and PSO, both of which consistently converged to the optimum. On G2 and G4, LEAL's minima were close to the optimum, but its mean values trailed those of DE, GA, and PSO. CMA-ES and BO again underperformed, failing to reliably reach optimal values. On G6, LEAL's mean (-5.85×10^{3}) was competitive with DE and PSO, while BO performed extremely poorly (9.15×10^{5}), highlighting instability in surrogate-based optimization for constrained functions. On multimodal landscapes, LEAL's strengths became more evident. For Griewank, LEAL not only achieved the exact optimum (0.00×10^{0}) but also had the lowest mean (3.17×10^{-2}) compared to DE, GA, and PSO (means ~7.5×10^{-2}–8.2×10^{-2}). Both CMA-ES and BO performed worse, with means above 10^{-1} and higher variance. On Rastrigin, another highly multimodal problem, LEAL again reached the global optimum (0.00×10^{0}) and recorded the best mean (3.59×10^{-2}), surpassing DE, GA, and PSO (means ~10^{-1}–10^{-2}) and clearly outperforming CMA-ES (4.60×10^{0}) and BO (3.94×10^{0}). For Zakharov, LEAL found the optimum (0.00×10^{0}), but its mean (4.28×10^{-3}) was slightly worse than PSO (1.04×10^{-3}) and GA (2.30×10^{-3}), although still better than CMA-ES (5.70×10^{-2}) and BO (8.16×10^{-2}). For Himmelblau's function, LEAL performed poorly, with a high mean (3.80×10^{0}) compared to GA (1.20×10^{-2}), PSO (2.52×10^{-2}), and DE (7.49×10^{-2}). BO showed the best result (9.04×10^{-3}) while CMA-ES underperformed (4.54×10^{-1}). This highlights that LEAL struggles with relatively sensitive but straightforward landscapes, where lightweight optimizers or surrogate-based search may excel. Across all functions, runtime patterns were consistent: LEAL required significantly longer execution times (several seconds to tens of seconds) compared to DE, GA, and PSO, which typically solved these problems in under 0.3 s. BO also suffered from extreme runtime costs (tens of seconds to minutes), whereas CMA-ES remained efficient but less reliable in terms of accuracy. In summary, LEAL demonstrated exceptional potential on highly multimodal functions like Griewank and Rastrigin, where it outperformed all other methods and achieved exact global optima. However, its performance on simpler or constrained problems was inconsistent, with higher means and variances than classical algorithms. CMA-ES and BO generally underperformed in both accuracy and stability, with BO additionally incurring very high runtimes. Thus, while LEAL's ensemble approach can provide superior performance on complex landscapes, its computational cost remains a major drawback compared to classical algorithms.

3.2.4 Multi-Objective Benchmark Problems

Table 3. Performance on Multi-Objective Benchmark Functions

Test problem	Metrics	SSA-NSGA-II	NSGA-II	LEAL
BNH	**GD+ mean**	1.66×10^{-1}	2.14×10^{-1}	6.90×10^{-1}
	GD+ std	2.52×10^{-3}	1.43×10^{-2}	9.18×10^{-1}
	GD+ min	1.64×10^{-1}	1.93×10^{-1}	5.82×10^{-2}
	Runtime mean (s)	1.84×10^{3}	3.05×10^{-1}	8.57×10^{0}
	Runtime std (s)	1.68×10^{2}	4.59×10^{-2}	7.89×10^{-1}
	Memory mean	2.64×10^{2}	9.05×10^{-1}	4.69×10^{0}
	Memory std	1.58×10^{1}	5.40×10^{-3}	1.55×10^{-1}
CTP7	**GD+ mean**	1.83×10^{-2}	2.27×10^{-2}	8.79×10^{-2}
	GD+ std	2.03×10^{-3}	1.13×10^{-2}	8.03×10^{-2}
	GD+ min	1.62×10^{-2}	6.40×10^{-3}	6.00×10^{-4}
	Runtime mean (s)	1.50×10^{3}	2.46×10^{-1}	7.57×10^{0}
	Runtime std (s)	1.20×10^{2}	8.50×10^{-3}	1.94×10^{-1}
	Memory mean	3.30×10^{2}	8.85×10^{-1}	5.46×10^{0}
	Memory std	1.80×10^{1}	2.30×10^{-3}	1.89×10^{-1}
DF1	**GD+ mean**	4.59×10^{-3}	2.77×10^{-1}	2.10×10^{0}
	GD+ std	1.07×10^{-3}	9.54×10^{-2}	7.71×10^{-1}
	GD+ min	3.36×10^{-3}	9.45×10^{-2}	8.91×10^{-1}
	Runtime mean (s)	6.91×10^{3}	1.79×10^{-1}	1.48×10^{1}
	Runtime std (s)	8.34×10^{2}	7.00×10^{-3}	1.32×10^{-1}
	Memory mean	1.37×10^{3}	8.69×10^{-1}	8.76×10^{0}
	Memory std	9.63×10^{1}	6.80×10^{-3}	7.51×10^{-1}
DF2	**GD+ mean**	5.23×10^{-3}	2.27×10^{-1}	1.43×10^{0}
	GD+ std	1.94×10^{-3}	7.94×10^{-2}	5.45×10^{-1}
	GD+ min	3.96×10^{-3}	2.14×10^{-1}	7.42×10^{-1}
	Runtime mean (s)	5.88×10^{3}	1.76×10^{-1}	1.47×10^{1}
	Runtime std (s)	5.81×10^{2}	6.90×10^{-3}	1.13×10^{-1}
	Memory mean	1.30×10^{3}	8.66×10^{-1}	8.75×10^{0}
	Memory std	5.13×10^{1}	4.70×10^{-3}	5.81×10^{-1}
DF3	**GD+ mean**	1.53×10^{-2}	3.87×10^{-1}	6.13×10^{0}
	GD+ std	7.02×10^{-3}	9.54×10^{-2}	6.27×10^{-1}
	GD+ min	7.38×10^{-3}	2.14×10^{-1}	8.91×10^{-1}
	Runtime mean (s)	3.76×10^{3}	1.83×10^{-1}	1.48×10^{1}
	Runtime std (s)	2.70×10^{2}	1.46×10^{-2}	2.03×10^{-1}
	Memory mean	1.36×10^{3}	8.76×10^{-1}	8.76×10^{0}
	Memory std	1.80×10^{2}	4.70×10^{-3}	5.81×10^{-1}
DF5	**GD+ mean**	9.90×10^{-3}	7.89×10^{-2}	1.02×10^{-1}
	GD+ std	2.57×10^{-3}	2.13×10^{-2}	8.04×10^{-2}
	GD+ min	6.53×10^{-3}	1.53×10^{-2}	2.26×10^{-2}
	Runtime mean (s)	3.50×10^{3}	2.48×10^{-1}	7.65×10^{0}
	Runtime std (s)	1.36×10^{2}	1.14×10^{-2}	2.38×10^{-1}
	Memory mean	9.51×10^{2}	9.02×10^{-1}	5.13×10^{0}
	Memory std	7.87×10^{0}	1.14×10^{-2}	1.37×10^{-1}

Table 3. Performance on Multi-Objective Benchmark Functions

Test problem	Metrics	SSA-NSGA-II	NSGA-II	LEAL
DF9	GD+ mean	3.83×10^{-1}	1.59×10^{0}	6.12×10^{0}
	GD+ std	1.58×10^{-1}	1.31×10^{-1}	3.53×10^{0}
	GD+ min	2.47×10^{-1}	1.33×10^{0}	8.62×10^{-1}
	Runtime mean (s)	2.64×10^{3}	1.80×10^{-1}	1.48×10^{1}
	Runtime std (s)	9.97×10^{1}	5.60×10^{-3}	1.17×10^{-1}
	Memory mean	1.40×10^{3}	8.86×10^{-1}	8.76×10^{0}
	Memory std	1.85×10^{1}	4.70×10^{-3}	5.81×10^{-1}
DTLZ1	**GD+ mean**	1.25×10^{2}	1.59×10^{0}	3.07×10^{0}
	GD+ std	2.77×10^{1}	1.31×10^{-1}	8.60×10^{-1}
	GD+ min	9.16×10^{1}	1.33×10^{0}	2.19×10^{0}
	Runtime mean (s)	2.21×10^{3}	1.79×10^{-1}	3.73×10^{1}
	Runtime std (s)	7.08×10^{1}	8.50×10^{-3}	9.19×10^{-1}
	Memory mean	9.08×10^{2}	9.15×10^{-1}	1.92×10^{1}
	Memory std	9.56×10^{1}	4.80×10^{-3}	8.70×10^{-3}
SRN	**GD+ mean**	6.00×10^{0}	2.11×10^{0}	1.39×10^{1}
	GD+ std	1.44×10^{0}	6.96×10^{-2}	1.49×10^{1}
	GD+ min	4.56×10^{0}	1.52×10^{-2}	1.24×10^{0}
	Runtime mean (s)	7.15×10^{2}	2.40×10^{-1}	7.12×10^{0}
	Runtime std (s)	1.23×10^{1}	1.57×10^{-2}	3.25×10^{-1}
	Memory mean	2.38×10^{2}	8.95×10^{-1}	4.74×10^{0}
	Memory std	1.56×10^{1}	1.07×10^{-2}	2.64×10^{-2}

In evaluating LEAL's performance across nine diverse multi-objective benchmark problems (BNH, CTP7, DF1, DF2, DF3, DF5, DF9, DTLZ1, and SRN), a consistent pattern of underperformance compared to NSGA-II emerges across several critical metrics: Generational Distance Plus (GD+), runtime, and memory usage. The GD+ metric serves as an indicator of how closely an algorithm's solutions approximate the true Pareto front, with lower values signifying better convergence. In nearly all of the evaluated problems, LEAL records higher mean GD+ values than NSGA-II, indicating less effective convergence to optimal solutions. For instance, in the BNH problem, LEAL achieves a mean GD+ of 6.90×10^{-1}, substantially higher than NSGA-II's 2.14×10^{-1}. Similarly, in DF1, LEAL exhibits a GD+ mean of 2.10×10^{0} compared to NSGA-II's 2.77×10^{-1}, underscoring LEAL's challenges in consistently approaching the Pareto optimal set. Delving deeper into runtime performance, LEAL demonstrates significantly longer execution times across all benchmark problems. For example, in the BNH problem, LEAL averages a runtime of 8.57×10^{0} s, which starkly contrasts with NSGA-II's rapid runtime of 3.05×10^{-1} s. This disparity is even more pronounced in the DTLZ1 problem, where LEAL requires an extensive 3.73×10^{1} s per run compared to NSGA-II's swift 1.79×10^{-1} s. Such prolonged runtimes suggest that LEAL's computational processes are considerably more time-consuming, potentially limiting its applicability in scenarios demanding quick decision-making and real-time optimization. Across other problems such as DF1 and SRN, LEAL maintains this trend of longer runtimes, further highlighting its inefficiency relative to NSGA-II. Memory usage presents another area where LEAL lags behind NSGA-II. Across all evaluated problems, LEAL consistently demands more memory resources. In the BNH problem, LEAL utilizes an average of 4.69×10 units of memory, significantly higher than NSGA-II's 9.05×10^{-1} units. This pattern holds for other problems as well: in DF1, LEAL reports a memory usage of 8.76×10^{0} units, compared to NSGA-II's 8.69×10^{-1} units. The consistently higher memory consumption implies that LEAL is less resource-efficient, which could be a substantial drawback in environments with limited memory availability or when scaling to more complex, larger-scale problems. In conclusion, the data from these multi-objective benchmarks strongly suggest that LEAL is less effective and less efficient compared to NSGA-II. LEAL's higher GD+ values across most problems indicate poorer convergence to the Pareto front, while its extended runtimes and elevated memory consumption highlight significant computational inefficiencies.

These combined factors render NSGA-II the more practical and effective choice for achieving high-quality, consistent multi-objective optimization results within the scope of the evaluated benchmark problems.

3.3 Analysis of Results

The experimental findings across engineering, single-objective, and multi-objective benchmarks (Tables 1–3) provide a comprehensive assessment of the Linear Ensemble Algorithm (LEAL) in comparison with both classical and recent meta-heuristics. Engineering problems, such as PLOD, TCD, and Pressure Vessel, LEAL, occasionally identify competitive minima. In the Pressure Vessel problem, the best solution (~1.5×10^3) was more than 75% lower than the best solutions of GA and PSO (~$7.6–8.8\times10^3$), highlighting its capacity to exploit nonlinear constraints. However, LEAL's mean objective values (7.55×10^4) were substantially worse than classical algorithms, and its runtimes (5–8 s) were an order of magnitude higher. This indicates that while LEAL can uncover exceptional minima, it struggles to maintain consistent average performance. Single-objective benchmarks on multimodal landscapes such as Rastrigin and Griewank, LEAL excelled. On Rastrigin, it achieved a mean error of 3.59×10^{-2}, representing an improvement of ~80–85% over GA, DE, and PSO. On Griewank, LEAL's mean of 3.17×10^{-2} was ~55–65% better than the classical methods. Conversely, on constrained functions such as G2 and G4, as well as simpler problems like Himmelblau, LEAL performed poorly, with its mean error (3.80) being more than two orders of magnitude worse than GA or PSO. Runtime patterns were consistent: LEAL required several seconds to tens of seconds, compared with <0.3 s for DE, GA, and PSO, and ~30–90 s for BO. These findings confirm LEAL's strength in tackling rugged, multimodal problems, but also highlight its inefficiency and inconsistency on constrained or smoother landscapes. Multi-objective benchmarks, LEAL generally underperformed relative to NSGA-II [8]. Its GD^+ values [28] were consistently higher; for example, in DF1, LEAL's GD^+ mean was 2.10 compared to NSGA-II's 0.277. Runtime differences were extreme: LEAL required ~15 s on DF1 (~80× slower than NSGA-II) and ~37 s on DTLZ1 (~200× slower). Memory usage also reflected inefficiency, with LEAL consuming 10–20× more than NSGA-II across most problems. SSA-NSGA-II [22,23] achieved competitive GD^+ convergence, but at a prohibitive cost, requiring runtimes in the 10^3-second range and memory exceeding 10^3 units. Comparisons with recent methods reveal similar trade-offs: CMA-ES [11,13,14] sometimes achieves stable results but with high variance, while BO [17–20] produces unstable solutions with runtimes far longer than those of classical methods. Overall synthesis. The results highlight LEAL's dual nature. It demonstrates strong performance on multimodal single-objective problems and has the potential to identify breakthrough minima in engineering design; however, it suffers from high runtime, extensive memory usage, and inconsistent averages. Compared with GA, DE, and PSO, LEAL trades efficiency for occasional superior accuracy. In comparison to newer methods such as CMA-ES, BO, and SSA-NSGA-II, it remains competitive in some cases but is generally less efficient. NSGA-II continues to provide the most balanced performance for multi-objective problems, while LEAL is best suited for tasks that prioritize ultimate solution quality over computational efficiency. In practice, LEAL is most suitable for rugged, multimodal landscapes or complex engineering designs where ultimate solution quality outweighs runtime and memory overhead. In time-critical or resource-constrained scenarios, lightweight baselines such as DE or GA/PSO are generally more efficient; for multi-objective problems, NSGA-II typically remains the preferred default due to better GD^+ and lower runtime/memory.

3.4 Discussion

In examining LEAL's performance across engineering, single-objective, and multi-objective problems, several key patterns emerge that highlight both its promise and limitations. The integration of evolutionary search with linear regression–based ensemble modeling offers a distinct advantage for navigating rugged and multimodal landscapes. This was particularly evident in functions such as Griewank and Rastrigin, where LEAL not only achieved the exact global optimum but also delivered mean values more than 80% lower than those of GA [5], DE [7], and PSO [6]. Such results underscore the capability of LEAL's ensemble strategy to balance exploration and exploitation in high-dimensional, multimodal spaces. In engineering problems such as Pressure Vessel Design, LEAL occasionally produced best-known minima far surpassing those of classical algorithms, confirming its capacity to exploit nonlinear constraints and structural interactions. Its best minima (~1,500) were over 75% better than the 6,600–7,100 range reported by GA and PSO, demonstrating its ability

to deliver breakthrough solutions. However, mean values and variance were often inferior to those of DE, GA, and PSO, while runtimes averaged several seconds, which was an order of magnitude higher than those of classical methods. This inconsistency suggests that LEAL is powerful in isolated cases but lacks robustness across repeated runs. For multi-objective optimization, the evidence indicates clear underperformance relative to NSGA-II [8]. LEAL generally produced higher GD^+ values [28], longer runtimes, and greater memory demands. On DF1 and DTLZ1, LEAL's runtime was 30–200× higher than NSGA-II, and its memory usage was 10–20× greater. SSA-NSGA-II [22,23] achieved competitive convergence, but at the expense of extreme computational costs (runtime in the 10^3-second range, with memory exceeding 10^3 units), highlighting scalability issues for surrogate-assisted ensembles. Comparisons with more recent methods further emphasize the trade-offs: CMA-ES [11,13,14] has shown competitive results on some constrained benchmarks but exhibits high variance, while BO [17–20] often requires excessive runtimes with unstable accuracy. These findings place LEAL within a broader context, showing that even advanced methods suffer from computational overheads and variance, although LEAL's costs remain among the most pronounced. Theoretically, LEAL's ensemble structure increases computational complexity due to repeated regression fitting and neighbor evaluations, which scale with population size and problem dimension. This complexity helps explain the observed runtime overhead and memory demands. While current experiments provide empirical validation, formal convergence proofs and computational complexity analyses remain essential to establish stronger theoretical guarantees. Overall, LEAL demonstrates reliability in consistently producing feasible solutions and occasionally uncovering exceptional minima. However, the trade-off is clear: substantial computational demands and memory requirements. For applications where solution quality outweighs runtime, such as high-stakes engineering design or safety-critical optimization, LEAL's approach may prove invaluable. Future refinements should aim to reduce overhead through parallelization, surrogate simplification, or adaptive ensemble sizing, making the algorithm more competitive in broader contexts.

4. Conclusions

The Linear Ensemble Algorithm (LEAL) demonstrates considerable promise as an optimization method by combining ensemble regression with evolutionary search operators. This methodological integration allows LEAL to approximate promising regions of the search space while maintaining global exploration, thereby strengthening its convergence capacity on rugged landscapes. In addition, the use of a neighbor-based linear combination mechanism provides a more adaptive search process, enabling LEAL to balance exploration and exploitation more effectively than conventional evolutionary algorithms. On single-objective benchmark functions, LEAL frequently attained or closely approached the global optimum. For multimodal problems such as Rastrigin and Griewank, it achieved improvements of 55–85% in mean performance compared to GA [5], DE [7], and PSO [6]. These results confirm that the ensemble regression and neighbor-based combination strategies significantly enhance solution accuracy. In engineering optimization tasks, LEAL occasionally uncovered breakthrough minima. For instance, in the Pressure Vessel Design problem, the best solution (~1.5×10^3) was more than 75% better than those reported by GA and PSO (~$7.6–8.8\times10^3$), validating the algorithm's ability to exploit nonlinear constraints. However, higher mean values, longer runtimes, and greater memory usage reflect the computational cost of maintaining ensemble structures and neighbor-based modeling. In multi-objective problems, LEAL consistently underperformed relative to NSGA-II [8], showing higher GD^+ [28], runtimes one to two orders of magnitude longer, and memory usage 10–20× higher. Comparisons with CMA-ES [11,13,14], BO [17–20], and SSA-NSGA-II [22,23] further illustrate the trade-offs among recent methods. CMA-ES demonstrated competitive results but with high variance, while BO often suffered from unstable accuracy and long runtimes. In contrast, SSA-NSGA-II provided strong convergence but at a prohibitive computational cost. Against this backdrop, LEAL remains competitive in producing feasible solutions but is less efficient overall. In conclusion, LEAL's key methodological contributions, its ensemble regression with evolutionary operators, neighbor-based linear combination, and systematic benchmarking against both classical and recent algorithms, underline its novelty and potential. LEAL is most suitable for single-objective and constrained engineering optimization where solution quality is more important than runtime. To enhance its competitiveness in broader domains, future research should

focus on: (i) reducing computational complexity through parallelization and lightweight surrogate modeling; (ii) improving efficiency via adaptive ensemble sizing; (iii) providing formal convergence and complexity guarantees; and (iv) extending applications to large-scale, dynamic, and real-world optimization tasks. With these refinements, LEAL could evolve into a more broadly applicable and impactful optimization framework.

5. Acknowledgements

Author Contributions: Conceptualization, Phatnathee Wongsrisai and Prompong Sugunnasil; methodology, Phatnathee Wongsrisai; software, Phatnathee Wongsrisai; validation, Phatnathee Wongsrisai and Prompong Sugunnasil; formal analysis, Phatnathee Wongsrisai; investigation, Phatnathee Wongsrisai; resources, Phatnathee Wongsrisai; data curation, Phatnathee Wongsrisai; writing—original draft preparation, Phatnathee Wongsrisai; writing—review and editing, Phatnathee Wongsrisai and Prompong Sugunnasil; visualization, Phatnathee Wongsrisai; supervision, Prompong Sugunnasil; Prompong Sugunnasil is the corresponding author. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Faculty of Engineering, Chiang Mai University, under the Research Assistant program (Grant No. RA/009/2563). In addition, this work was supported by the Erawan HPC Project, Information Technology Service Center (ITSC), Chiang Mai University, Chiang Mai, Thailand.

Conflicts of Interest: The authors declare no conflict of interest

References

[1] Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley: Reading, MA, USA, **1989**.

[2] Kennedy, J.; Eberhart, R. Particle Swarm Optimization. *Proc. IEEE Int. Conf. Neural Networks* **1995**, *4*, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968.

[3] Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*(2), 182–197. https://doi.org/10.1109/4235.996017.

[4] Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Global Optim.* **1997**, *11*(4), 341–359. https://doi.org/10.1023/A:1008202821328.

[5] Hansen, N.; Ostermeier, A. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evol. Comput.* **2001**, *9*(2), 159–195. https://doi.org/10.1162/106365601750190398.

[6] Glover, F.; Kochenberger, G. A., Eds. *Handbook of Metaheuristics*; Springer: Boston, MA, USA, **2003**. https://doi.org/10.1007/b101874.

[7] Yang, X. S.; Deb, S.; Fong, S. Metaheuristic Algorithms: Optimal Balance of Intensification and Diversification. *Appl. Math. Inf. Sci.* **2011**, *5*(3), 236–254. Available online: https://www.naturalspublishing.com/Article.asp?ArtcID=1637 (accessed 4 Sep 2025).

[8] Blum, C.; Roli, A. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Comput. Surv.* **2003**, *35*(3), 268–308. https://doi.org/10.1145/937503.937505.

[9] Coello-Coello, C. A. Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art. *Comput. Methods Appl. Mech. Eng.* **2002**, *191*(11–12), 1245–1287. https://doi.org/10.1016/S0045-7825(01)00323-1.

[10] Hansen, N. The CMA Evolution Strategy: A Tutorial. *arXiv* **2016**, arXiv:1604.00772. Available online: https://arxiv.org/abs/1604.00772 (accessed 4 Sep 2025).

[11] Arnold, D. V.; Hansen, N. A (1+1)-CMA-ES for Constrained Optimisation. *Proc. Genetic and Evolutionary Computation Conference (GECCO)* **2012**, 297–304. https://doi.org/10.1145/2330163.2330207.

[12] Sakamoto, N.; Akimoto, Y. Adaptive Ranking Based Constraint Handling for Explicitly Constrained Black-Box Optimization. In *Parallel Problem Solving from Nature – PPSN XV*; Auger, A., Fonseca, C.,

Lourenço, N., Eds.; Springer: Cham, Switzerland, **2018**; pp. 72–83. https://doi.org/10.1007/978-3-319-99259-4_6.

[13] Morinaga, S.; Akimoto, Y. Safe Control with CMA-ES by Penalizing Unsafe Search Directions. *Proc. Genetic and Evolutionary Computation Conference (GECCO)* **2024**, 344–352. https://doi.org/10.1145/3638529.3654047.

[14] He, X.; He, Y.; He, Z.; Yang, S. KbP-LaF-CMAES: Knowledge-based Principal Landscape-aware Feasible CMA-ES for Constrained Optimization. *Inf. Sci.* **2024**, *670*, 119–141. https://doi.org/10.1016/j.ins.2024.01.088.

[15] Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R. P.; de Freitas, N. Taking the Human out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* **2016**, *104*(1), 148–175. https://doi.org/10.1109/JPROC.2015.2494218.

[16] Snoek, J.; Larochelle, H.; Adams, R. P. Practical Bayesian Optimization of Machine Learning Algorithms. *Adv. Neural Inf. Process. Syst. (NeurIPS)* **2012**, *25*, 2951–2959. https://dl.acm.org/doi/10.5555/2999325.2999464.

[17] Eriksson, D.; Pearce, M.; Gardner, J.; Turner, R.; Poloczek, M. Scalable Constrained Bayesian Optimization. *Proc. Int. Conf. Machine Learning (ICML)* **2021**, *139*, 2949–2958. https://proceedings.mlr.press/v139/eriksson21a.html.

[18] Regli, J. B.; Shoemaker, C. A. Hybrid Bayesian Optimization and IPOPT for Constrained Black-Box Optimization. *Struct. Multidiscip. Optim.* **2025**, *66*(2), 45–61. https://doi.org/10.1007/s00158-025-03729-9.

[19] Daulton, S.; Balandat, M.; Bakshy, E. Parallel Bayesian Optimization of Multiple Noisy Objectives with Expected Hypervolume Improvement. *Adv. Neural Inf. Process. Syst. (NeurIPS)* **2020**, *33*, 752–764. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

[20] Deb, K. An Efficient Constraint Handling Method for Genetic Algorithms. *Comput. Methods Appl. Mech. Eng.* **2000**, *186*(2–4), 311–338. https://doi.org/10.1016/S0045-7825(99)00389-8.

[21] Lim, D.; Ong, Y. S.; Jin, Y.; Sendhoff, B. A Study on Metamodeling Techniques, Ensembles, and Multi-Surrogates in Evolutionary Computation. *Proc. Genetic and Evolutionary Computation Conference (GECCO)* **2007**, 1288–1295. https://doi.org/10.1145/1276958.1277180.

[22] Ong, Y. S.; Nair, P. B.; Keane, A. J. Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling. *AIAA J.* **2003**, *41*(4), 687–696. https://doi.org/10.2514/2.2010.

[23] Jin, Y. A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. *Soft Comput.* **2005**, *9*(1), 3–12. https://doi.org/10.1007/s00500-003-0328-6.

[24] Blank, J.; Deb, K. Pymoo: Multi-Objective Optimization in Python. *IEEE Access* **2020**, *8*, 89497–89509. https://doi.org/10.1109/ACCESS.2020.2990567.

[25] Brockhoff, D.; Trautmann, H. Pysamoo: Surrogate-Assisted Multi-Objective Optimization in Python. *Proc. Genetic and Evolutionary Computation Conference (GECCO Companion)* **2021**, 1840–1847. https://doi.org/10.1145/3449726.3463204.

[26] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, É. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

[27] Beume, N.; Naujoks, B.; Emmerich, M. SMS-EMOA: Multiobjective Selection Based on Dominated Hypervolume. *Eur. J. Oper. Res.* **2007**, *181*(3), 1653–1669. https://doi.org/10.1016/j.ejor.2006.08.008. *(Least HV contribution / HV-based survival)*

[28] Ishibuchi, H.; Masuda, H.; Tanigaki, Y.; Nojima, Y. Modified Distance Calculation in Generational Distance and Inverted Generational Distance. In *Evolutionary Multi-Criterion Optimization (EMO 2015)*; Lecture Notes in Computer Science, Vol. 9019; Springer: Cham, **2015**; pp 110–125.

[29] Daulton, S.; Balandat, M.; Bakshy, E. Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization. *NeurIPS* **2020**, *33*, 9851–9864.

[30] Daulton, S.; Balandat, M.; Bakshy, E. Noisy Expected Hypervolume Improvement for Multi-Objective Bayesian Optimization. *NeurIPS* **2021**, *34*, 24368–24381.

[31] Liu, S.; Gong, M.; Zhang, Q.; Jin, Y. Surrogate-Assisted Evolutionary Algorithms for Expensive Optimization: A Recent Survey. *IEEE Comput. Intell. Mag.* **2024**, *19*(2), 45–73.

[32] Yu, L.; Wang, H.; Jin, Y. Surrogate-Assisted Differential Evolution: A Survey. *Swarm Evol. Comput.* **2025**, *85*, 101507.