AIMS Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 11(5): 14586–14616.
DOI: 10.3934/math.2026598
Received: 23 March 2026
Revised: 15 May 2026
Accepted: 19 May 2026
Published: 25 May 2026

Research article

An adaptive multi-operator differential evolution algorithm for multi-item constrained stochastic inventory optimization

Zixin Feng[1], Xingyu Feng[2], Lupeng Hao[2,*], and Junming Chen[3,*]

[1] School of Event and Communication, Shanghai University of International Business and Economics, Shanghai 201620, China

[2] School of Computer Science and Technology, Hainan University, Haikou 570228, China

[3] School of Art and Design, Guangzhou University, Guangzhou 510006, China

* **Correspondence:** Email: 25120854040017@hainanu.edu.cn; jmchen@gzhu.edu.cn.

Abstract: Effective inventory management under stochastic demand remains a central challenge in supply chain operations, particularly when multiple items share coupling constraints on the purchasing budget, warehouse capacity, and service level. Although metaheuristic algorithms have been widely applied to such problems, existing approaches typically rely on fixed algorithmic configurations that limit their adaptability and robustness as problem dimensionality and constraint complexity grow. To address this limitation, this paper proposes the adaptive multi-operator differential evolution (AMODE) algorithm, which unifies four complementary mechanisms within a single cohesive framework: opposition-based learning initialization for enhanced population diversity, an adaptive multi-operator mutation pool with success-history-based operator selection, success-history based adaptive differential evolution (SHADE) style parameter self-adaptation for the scaling factor and crossover rate, and a Lévy-flight escape mechanism to counteract premature convergence. AMODE was evaluated on three benchmark instances of increasing dimensionality ($n = 10, 20, 50$) derived from the UCI Online Retail II dataset, comprising over one million real-world retail transactions. Experiments over 30 independent runs compare AMODE achieved against twelve representative metaheuristic algorithms spanning evolutionary, swarm-based, physics-inspired, and adaptive DE paradigms (including the state-of-the-art DE variants linear success-history based adaptive differential evolution (L-SHADE), adaptive j-strategy self-optimization differential evolution (jSO), and improved multi-operator differential evolution (IMODE)). AMODE the best mean fitness across all instances with a standard deviation not exceeding 0.02—demonstrating near-deterministic convergence. Wilcoxon signed-rank tests confirmed statistical significance ($p < 0.001$) against all competitors on all three instances. An ablation study established the independent and synergistic contribution of each component, and an analysis of internal parameter dynamics revealed how the adaptive mechanisms respond coherently to the evolving search landscape. These findings establish

AMODE as a robust and scalable optimization framework for practical multi-item inventory management under stochastic demand.

Keywords: differential evolution; inventory optimization; opposition-based learning; Lévy flight; SHADE parameter adaptation; metaheuristic; supply chain
Mathematics Subject Classification: 90B05, 90C59

1. Introduction

Inventory management constitutes a fundamental component of supply chain operations, directly influencing total operational costs, service levels, and capital utilization [1]. Multi-item inventory systems, where several products share common constraints on the purchasing budget, storage capacity, and minimum service requirements, arise frequently in practice [2–4]. The coupling nature of these shared constraints renders the optimization problem non-separable, and the inclusion of stochastic demand further compounds the difficulty, resulting in NP-hard formulations that preclude closed-form solutions [5, 6].

Classical inventory models, including the economic order quantity (EOQ) framework and (s, S) policies, provide tractable solutions under simplifying assumptions such as deterministic demand and single-item settings [1]. When multiple items are coupled through shared resource constraints, however, these analytical approaches become inadequate. Mathematical programming techniques, while theoretically exact, suffer from prohibitive computational costs as problem dimensionality increases [7, 8]. Consequently, there is a growing need for efficient and scalable optimization methods that can accommodate the nonlinearity, non-convexity, and stochasticity inherent in real-world multi-item inventory systems.

Metaheuristic algorithms have emerged as practical alternatives for addressing such large-scale constrained inventory problems. Genetic algorithms [9, 10], particle swarm optimization [7, 11], and differential evolution [12–14] have been applied to various supply chain inventory formulations with encouraging results. Hybrid and adaptive metaheuristics incorporating reinforcement learning [5, 8] and simulation-based optimization frameworks [2] have also been developed to handle the complexity inherent in practical inventory systems. However, the effectiveness of these approaches is often constrained by their reliance on fixed algorithmic configurations that cannot adapt to the evolving fitness landscape during the search process, limiting their robustness on high-dimensional constrained problems.

Among metaheuristic approaches, differential evolution stands out for its effectiveness on continuous optimization problems [12]. The SHADE algorithm [15] introduced success-history-based parameter adaptation, while multi-operator and multi-population DE strategies [16–18] have demonstrated that dynamically selecting among complementary mutation strategies can significantly improve search performance. Additionally, opposition-based learning (OBL) [19] has shown promise in accelerating convergence through simultaneous evaluation of candidate solutions and their opposites, and Lévy-flight perturbation [20, 21] provides an effective mechanism for escaping local optima through heavy-tailed random steps. Despite the individual effectiveness of these techniques, their simultaneous integration within a DE framework tailored to constrained inventory optimization has not been explored.

To address this gap, this paper proposes the adaptive multi-operator differential evolution (AMODE) algorithm for multi-item constrained stochastic inventory optimization. AMODE unifies four complementary mechanisms—OBL initialization, adaptive multi-operator mutation, SHADE-style parameter adaptation, and Lévy-flight escape—into a single cohesive framework. The algorithm is evaluated on benchmark instances derived from real-world retail transaction data, providing a rigorous and reproducible assessment against twelve metaheuristic algorithms spanning classical methods and state-of-the-art adaptive DE variants. The main contributions are as follows:

- AMODE integrates four complementary innovations—OBL initialization, adaptive multi-operator mutation, SHADE-style parameter adaptation, and Lévy-flight escape—within a unified DE framework tailored to constrained inventory optimization.
- Three benchmark instances of increasing dimensionality ($n = 10, 20, 50$) are constructed from the UCI Online Retail II dataset, containing over one million real-world retail transactions, providing a realistic and reproducible evaluation platform.
- A comprehensive experimental study compares AMODE against twelve algorithms—nine classical metaheuristics and three state-of-the-art adaptive DE variants (L-SHADE, jSO, IMODE)—over 30 independent runs per instance, with statistical significance assessed via Wilcoxon signed-rank tests with Bonferroni correction, Cliff's δ effect size, and Friedman ranking.
- An ablation study and sensitivity analysis quantify the contribution of each algorithmic component and reveal internal adaptation dynamics, offering insight into the mechanisms underlying AMODE's effectiveness.

The remainder of this paper is organized as follows. Section 2 reviews the related work on metaheuristic optimization for inventory and supply chain problems. Section 3 presents the problem formulation and the proposed AMODE algorithm. Section 4 describes the experimental setup, reports the results, and provides in-depth analysis. Section 5 concludes the paper and outlines future research directions.

2. Related work

2.1. Metaheuristic methods for inventory and supply chain optimization

The application of metaheuristic algorithms to inventory and supply chain optimization has expanded substantially over the past two decades, driven by the inability of classical analytical methods to handle the nonlinearity, non-convexity, and stochasticity inherent in real-world formulations. Genetic algorithms (GA) have been widely adopted for supply chain network design under uncertainty [9, 10], and artificial immune system (AIS) approaches have been developed for material handling optimization [22]. Particle swarm optimization (PSO) has been applied to multi-warehouse inventory replenishment problems with budget constraints [7, 11], including variants that incorporate cloud manufacturing scheduling [23]. Differential evolution (DE) has demonstrated strong performance on vendor selection and joint replenishment formulations involving binary-continuous decision variables [12–14]. Simulation-based optimization frameworks integrating metaheuristic solvers with discrete-event simulation have also proven effective for perishable inventory management [2, 24].

Beyond single-objective cost minimization, multi-objective formulations addressing sustainability and closed-loop supply chains have further expanded the scope of metaheuristic applications in this domain [25–27]. Cold chain logistics [28, 29] and reusable container management [30] present additional supply chain contexts where metaheuristics have yielded competitive solutions. Hybrid metaheuristic algorithms, such as the self-organizing migrating genetic algorithm, have been applied to constrained two-warehouse inventory models under interval-valued cost uncertainty [31], and advanced backtracking search algorithms have been developed for joint replenishment problems with grouping constraints [32]. Knowledge-based metaheuristic frameworks coupling gaining–sharing operators with augmented Lagrangian constraint handling have demonstrated competitive performance on stochastic transportation programming problems [33]. Reinforcement learning-based adaptive parameter tuning has been integrated with DE and PSO for constrained multi-item EOQ problems [5, 8], demonstrating that intelligent parameter control can substantially improve algorithmic robustness. Deep reinforcement learning with multi-expert policy distillation has further been applied to cooperation emergence in multi-agent decision systems [34], highlighting the growing synergy between learning-based search strategies and population dynamics. Data-driven forecasting methods have also been proposed for multi-item demand estimation, providing more accurate input parameters for downstream inventory optimization [35]. More recently, applications of metaheuristics have been extended to diverse operational contexts including port logistics [36], pharmaceutical delivery systems [37], emergency vehicle routing [38], automotive scheduling [39], energy infrastructure planning [40, 41], job-shop scheduling [42], and industrial process optimization [43], illustrating the broad applicability of population-based optimization across operations research domains. Alternative metaheuristic paradigms such as fruit fly optimization [44], the Jaya algorithm [45], and artificial immune systems [22, 46] have also been explored for combinatorial and logistics problems, including underground metro logistics optimization [47]. Salp swarm algorithm variants with many-objective extensions [48, 49] have further enriched the algorithmic landscape for constrained optimization.

More recently, metaheuristic methods have been applied to specialized operational contexts including medical waste management logistics and emergency supply chain coordination [50–53], further demonstrating the versatility of population-based optimization across high-stakes domains. Despite this breadth of application, existing metaheuristic approaches for inventory optimization predominantly rely on fixed algorithmic configurations tailored to specific problem instances, limiting their adaptability as problem scale and constraint complexity grow. The simultaneous integration of initialization enhancement, parameter self-adaptation, multi-operator selection, and escape mechanisms within a unified framework for constrained multi-item stochastic inventory optimization has not been explored in prior work, motivating the present study.

2.2. *Differential evolution and adaptive variants*

Differential evolution, originally proposed by Storn and Price [12], is a population-based metaheuristic that is particularly effective for continuous optimization problems. Its simplicity and relatively few control parameters have made it a popular choice for engineering optimization. The performance of standard DE, however, is sensitive to the choice of the scaling factor F and crossover rate CR, motivating the development of adaptive variants. The SHADE (success-history-based Adaptive DE) algorithm [15] addressed this sensitivity by maintaining a historical memory of successful parameter configurations and sampling new parameters from this archive using Cauchy and Gaussian distributions. Its extension, L-SHADE [54], further incorporated linear population size reduction to balance exploration and exploitation over the course of the search. The jSO algorithm [55] further refined the L-SHADE framework by introducing a generation-dependent weighted scaling factor F_w that transitions from exploration-oriented sampling in early generations to exploitation-focused sampling near convergence, achieving state-of-the-art performance on the CEC 2017 benchmark suite. The improved multi-operator differential evolution (IMODE) algorithm [56] advanced the multi-operator paradigm by partitioning the population into three equal sub-groups assigned to distinct mutation strategies with adaptive information sharing between sub-populations, winning the CEC 2020 single-objective optimization competition. These developments establish the comparative baseline against which the integrated design of AMODE is evaluated in the present work.

Multi-operator and multi-population DE strategies have emerged as a natural extension of adaptive DE, recognizing that no single mutation strategy is universally optimal across diverse problem landscapes. Yu et al. [16] proposed a multi-population DE framework for retail shelf space allocation, demonstrating that island-based sub-swarms with adaptive restarts outperform single-population approaches. Lu and Tang [17] developed an asynchronous multi-agent DE with problem-specific strategies for assembly hybrid differentiation flowshop scheduling, achieving superior performance through cooperative learning among agents. Sheng and Li [18] applied an improved adaptive DE (IJADE) to multi-level supply chain lot-sizing, showing that adaptive parameter control integrated with domain-specific operators yields both cost reduction and convergence stability. These developments motivate the integration of multi-operator selection with SHADE-style adaptation in the present work.

While L-SHADE and its variants have demonstrated state-of-the-art performance on standard continuous optimization benchmarks [54], their application to constrained multi-item stochastic inventory problems remains largely unexplored. It is not established whether the adaptive parameter mechanisms effective in unconstrained benchmark settings translate directly to problems with tightly coupled inequality constraints that render the feasible region a small fraction of the search domain, nor whether multi-operator selection provides complementary benefits over single-strategy adaptive DE in such structured landscapes.

2.3. Opposition-based learning and escape mechanisms

Opposition-based learning (OBL), introduced by Tizhoosh [19], enhances population initialization by simultaneously evaluating candidate solutions and their mirror images with respect to the search bounds. This simple yet effective strategy has been shown to accelerate convergence across a wide range of metaheuristic algorithms by increasing the probability that the initial population covers promising regions of the feasible space. Despite its demonstrated effectiveness on benchmark optimization problems, OBL has received limited attention in inventory optimization contexts, where constrained feasible regions may occupy a small fraction of the overall search domain.

Lévy-flight perturbation, popularized through the cuckoo search algorithm [20], provides a mechanism for escaping local optima through heavy-tailed random steps that occasionally produce large jumps in the search space. Chang and Hung [21] recently integrated Lévy-flight search with metamodel-guided optimization for stochastic hybrid shop scheduling, demonstrating that the combination of stochastic perturbation with surrogate learning can effectively navigate complex constrained landscapes. The integration of both OBL and Lévy flight within an adaptive DE framework for inventory optimization, as proposed in the present work, has not been previously explored.

The combined application of OBL initialization and Lévy-flight perturbation within an adaptive DE framework for constrained inventory optimization has not been previously investigated. It remains an open question whether these complementary mechanisms—one acting on the initial population distribution and the other on late-stage escape—provide mutually reinforcing benefits in the presence of coupling constraints and stochastic objective terms. This gap motivates the integrated design of AMODE proposed in the following section.

3. Proposed method

3.1. Problem formulation

Consider a set of n products indexed by $i = 1, 2, \ldots, n$, each characterized by stochastic demand with mean λ_i, and standard deviation σ_i, unit purchase price p_i, unit volume v_i, fixed ordering cost A_i, per-unit holding cost rate h_i, and per-unit shortage penalty π_i. The decision variable $q_i > 0$ denotes the order quantity for item i. The objective is to minimize the total expected inventory cost comprising ordering, holding, and shortage components:

$$\min_{\mathbf{q}} \ f(\mathbf{q}) = \sum_{i=1}^{n} \left[\frac{A_i \lambda_i}{q_i} + \frac{h_i q_i}{2} + \pi_i \sigma_i L(z_i) \right], \tag{3.1}$$

where

$$L(z_i) = \phi(z_i) - z_i[1 - \Phi(z_i)]$$

is the standard normal loss function,

$$z_i = (q_i - \lambda_i)/\sigma_i,$$

and $\phi(\cdot)$ and $\Phi(\cdot)$ denote the standard normal probability density and cumulative distribution functions, respectively. This normal approximation is appropriate for high-volume items with large demand mean λ_i, where the central limit theorem ensures that aggregated demand over a replenishment period converges to normality [1]. The products selected from the UCI Online Retail II dataset represent the highest-volume items by total sales, with demand means substantially exceeding ten units per period across all three benchmark instances, satisfying the standard condition under which the normal approximation is valid and widely adopted in the inventory optimization literature. The primary benefit of this formulation is the availability of a closed-form loss function $L(z_i)$, which avoids discrete summation or numerical integration and keeps each objective evaluation in $O(n)$ time.

The optimization is subject to three coupling constraints that link all decision variables simultaneously. The purchasing budget constraint limits the total procurement expenditure:

$$\sum_{i=1}^{n} p_i q_i \leq B. \tag{3.2}$$

The warehouse capacity constraint restricts the average storage volume:

$$\sum_{i=1}^{n} v_i \frac{q_i}{2} \leq W. \tag{3.3}$$

The service level constraint ensures that each item meets a minimum fill rate:

$$\Phi(z_i) \geq \alpha, \quad \forall\, i = 1, 2, \ldots, n. \tag{3.4}$$

Constraints (3.2)–(3.4) couple all n decision variables, destroying problem separability and rendering the feasible region a complex polytope in $\mathbb{R}^n$. The resulting constrained nonlinear optimization problem is NP-hard [5].

To handle these constraints within a population-based metaheuristic framework, a quadratic penalty method is adopted. The penalized objective function is defined as:

$$F(\mathbf{q}) = f(\mathbf{q}) + \rho \left[\max\left(0, \sum_{i=1}^{n} p_i q_i - B\right)^2 + \max\left(0, \sum_{i=1}^{n} v_i \frac{q_i}{2} - W\right)^2 + \sum_{i=1}^{n} \max(0, \alpha - \Phi(z_i))^2 \right], \tag{3.5}$$

where $\rho = 10^5$ is the penalty coefficient. This formulation drives infeasible solutions toward the feasible region while preserving the ranking of feasible solutions according to their true cost.

The model adopts several simplifying assumptions standard in the inventory optimization literature: stationary demand distributions, deterministic lead times, and no quantity discounts. These assumptions define the scope of the current work and represent deliberate modeling choices that isolate the algorithmic contribution from domain-specific complexity. Extensions to non-stationary demand, dynamic lead times, and supplier capacity constraints are identified as promising directions for future research (Section 5).

3.2. Overview of AMODE

Adaptive Multi-Operator Differential Evolution (AMODE) extends the standard DE framework with four complementary innovations designed to address the challenges of constrained inventory optimization: population initialization via opposition-based learning, adaptive mutation operator selection, SHADE-style parameter adaptation, and Lévy-flight perturbation for escaping local optima. The following subsections detail each component in turn, followed by the constraint handling mechanism and a summary of the complete procedure.

3.3. Opposition-based learning initialization

Standard DE initializes the population $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ uniformly at random within the search bounds $[\mathbf{lb}, \mathbf{ub}]$. OBL [19] augments this process by simultaneously evaluating the opposite of each candidate solution. For every individual $\mathbf{x}_j$, its opposite point is computed as:

$$\tilde{\mathbf{x}}_j = \mathbf{lb} + \mathbf{ub} - \mathbf{x}_j, \tag{3.6}$$

producing a doubled candidate pool of size $2N$. The N individuals with the lowest penalized fitness values from this merged pool are retained as the initial population. This strategy improves the probability of covering promising regions of the search space from the outset, which is particularly beneficial for constrained problems where feasible regions may occupy a small fraction of the search domain.

3.4. Multi-operator mutation pool with adaptive selection

Rather than relying on a single mutation strategy, AMODE maintains a pool of three complementary operators. The DE/rand/1 strategy

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot (\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) \tag{3.7}$$

promotes exploration by generating diverse trial vectors from randomly selected individuals. The DE/best/1 strategy

$$\mathbf{v}_i = \mathbf{x}_{\text{best}} + F \cdot (\mathbf{x}_{r_1} - \mathbf{x}_{r_2}) \tag{3.8}$$

accelerates exploitation by directing the search toward the current best solution. The DE/current-to-best/1 strategy

$$\mathbf{v}_i = \mathbf{x}_i + F \cdot (\mathbf{x}_{\text{best}} - \mathbf{x}_i) + F \cdot (\mathbf{x}_{r_1} - \mathbf{x}_{r_2}) \tag{3.9}$$

balances the two objectives by combining information from the current individual and the best solution.

Each operator $k \in \{1, 2, 3\}$ is assigned a selection probability p_k initialized to $1/3$. At each generation, the operator for each individual is chosen via roulette-wheel selection based on the current probability vector. The probabilities are updated after each generation according to the relative success rates:

$$p_k \leftarrow \frac{s_k + \epsilon}{\sum_{j=1}^{3}(s_j + \epsilon)}, \tag{3.10}$$

where s_k denotes the number of successful trial vectors generated by operator k in the current generation, and $\epsilon > 0$ is a small constant that prevents any operator from receiving zero probability. This adaptive mechanism allows AMODE to shift emphasis from exploration-oriented operators (DE/rand/1) to exploitation-oriented operators (DE/current-to-best/1) as the search progresses.

3.5. SHADE-style parameter adaptation

Following the SHADE framework [15], AMODE maintains a historical memory

$$\mathcal{M} = \{(M_{F,h}, M_{CR,h})\}_{h=1}^{H}$$

of H successful parameter configurations, initialized to $(0.5, 0.5)$. At each generation, the scaling factor and crossover rate for individual i are sampled as

$$F_i = \mathcal{N}(M_{F,r}, 0.1), \qquad CR_i = \mathcal{N}(M_{CR,r}, 0.1), \tag{3.11}$$

where r is a uniformly random index into $\mathcal{M}$, and values are truncated to $[0.01, 1]$. While the original SHADE samples F_i from a Cauchy distribution to allow occasional large scaling factors, we adopt a Gaussian distribution for F_i to reduce the probability of extreme step sizes that could destabilise convergence in tightly constrained inventory problems: large, erratic values of F increase the likelihood of generating trial vectors that strongly violate the coupling constraints (3.2)–(3.4), amplifying the penalty cost and slowing feasibility recovery. At the end of each generation, successful parameters (F, CR) that produced improving trial vectors are collected into sets S_F and S_{CR}. A randomly chosen memory entry is then updated using the weighted Lehmer mean:

$$M_{F,h} \leftarrow \frac{\sum_j w_j F_j^2}{\sum_j w_j F_j}, \qquad M_{CR,h} \leftarrow \sum_j w_j CR_j, \tag{3.12}$$

where the weight w_j is proportional to the fitness improvement achieved by the j-th successful individual. The Lehmer mean for F biases the memory toward larger scaling factors that produce substantial improvements, while the arithmetic mean for CR provides stable adaptation.

3.6. Lévy-flight escape mechanism

To counteract premature convergence, AMODE monitors a stagnation counter that increments whenever the global best fitness does not improve. When the counter reaches a threshold $L = 20$, a Lévy-flight perturbation is applied to 20% of the population, selected at random. The threshold $L = 20$ was determined through pilot experiments on representative instances, providing a balance between sensitivity to genuine stagnation and avoidance of unnecessary perturbations; a sensitivity analysis of this parameter is provided in Section 4.3.4. Each perturbed individual receives an additive step:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \gamma \cdot \text{Lévy}(\beta) \odot (\mathbf{ub} - \mathbf{lb}), \tag{3.13}$$

where $\gamma = 0.01$ is the step-size scaling constant, $\beta = 1.5$ is the Lévy stability index, and $\odot$ denotes element-wise multiplication. Using the search range $(\mathbf{ub} - \mathbf{lb})$ as the scaling factor normalizes the perturbation amplitude to the problem domain, ensuring that the step size is proportional to the extent of the search space rather than to the distance between the current individual and the best solution. This normalization makes the perturbation scale-invariant and improves portability across problem instances with different variable ranges. The Lévy distribution generates occasional large jumps that enable the algorithm to escape basins of attraction around local optima [20]. After perturbation, the stagnation counter is reset. The constraint penalty mechanism in (3.5) ensures that any temporarily infeasible solutions introduced by the perturbation are driven back toward feasibility in subsequent generations.

3.7. Constraint handling

The quadratic penalty formulation in (3.5) serves as the sole constraint handling mechanism. The penalty coefficient $\rho = 10^5$ is selected according to a magnitude-matching principle: given that objective function values are on the order of 10^3–10^4, setting $\rho = 10^5$ ensures that any constraint violation of magnitude $\epsilon > 0$ produces a penalty that dominates the original objective, thereby strictly prioritizing feasibility over cost optimization. Trial vectors are clipped to $[\mathbf{lb}, \mathbf{ub}]$ before fitness evaluation to enforce the physical non-negativity constraint on order quantities; this boundary projection does not interfere with feasibility with respect to the coupling constraints (3.2)–(3.4), which are handled exclusively by the penalty term. As the population converges toward the feasible region, the penalty terms approach zero, and the penalized objective reduces to the true inventory cost. This approach avoids the computational overhead of repair operators or feasibility-rule-based ranking, while remaining compatible with the adaptive parameter mechanisms of SHADE and the Lévy-flight perturbation.

3.8. Summary of AMODE

The complete AMODE procedure is summarized in Algorithm 1.

Algorithm 1 Adaptive Multi-Operator Differential Evolution (AMODE)

Require: Population size N, max iterations $T_{\max}$, memory size H, stagnation limit L, bounds $[\mathbf{lb}, \mathbf{ub}]$
Ensure: Best solution $\mathbf{x}^*$
Initialize $\{\mathbf{x}_j\}_{j=1}^{N}$ uniformly in $[\mathbf{lb}, \mathbf{ub}]$
Compute opposites via (3.6); retain best N from merged pool
Initialize memory $\mathcal{M} \leftarrow \{(0.5, 0.5)\}^H$; operator probabilities $\mathbf{p} \leftarrow (1/3, 1/3, 1/3)$
for $t = 1$ to $T_{\max}$ **do**
  $S_F, S_{CR} \leftarrow \emptyset$; success counts $s_1, s_2, s_3 \leftarrow 0$
  for $i = 1$ to N **do**
    Select mutation operator k via roulette on $\mathbf{p}$
    Sample F_i, CR_i from memory $\mathcal{M}$ via (3.11)
    Generate mutant $\mathbf{v}_i$ using operator k with F_i
    Generate trial $\mathbf{u}_i$ via binomial crossover with CR_i
    Clip $\mathbf{u}_i$ to $[\mathbf{lb}, \mathbf{ub}]$; evaluate $F(\mathbf{u}_i)$
    if $F(\mathbf{u}_i) \leq F(\mathbf{x}_i)$ **then**
      $\mathbf{x}_i \leftarrow \mathbf{u}_i$; record (F_i, CR_i) in S_F, S_{CR}; $s_k \leftarrow s_k + 1$
    end if
  end for
  Update memory entry in $\mathcal{M}$ via (3.12) using S_F, S_{CR}
  Update operator probabilities $\mathbf{p}$ via (3.10) using s_1, s_2, s_3
  if best fitness has not improved for L consecutive generations **then**
    Perturb 20% of population via Lévy flight (3.13); reset counter
  end if
end for
return $\mathbf{x}^* \leftarrow \arg\min_i F(\mathbf{x}_i)$

3.9. Computational complexity

The per-generation computational cost of AMODE is dominated by N evaluations of the penalized objective function (3.5), each requiring $O(n)$ arithmetic operations. The SHADE memory update and Lehmer mean computation require $O(H+N)$ operations per generation; the operator probability update adds $O(1)$ overhead per operator. The Lévy-flight perturbation, triggered infrequently and applied to $0.2N$ individuals, contributes $O(N)$ operations in triggered generations. The overall time complexity is therefore $O(T_{\max} \cdot N \cdot n)$, identical in order to standard DE with a small constant overhead from the adaptive bookkeeping. The space complexity is $O(N \cdot n + H)$, comprising the population matrix and the SHADE historical memory.

4. Experiments

4.1. Datasets

The benchmark instances are derived from the Online Retail II dataset [57], a publicly available collection of over 1,000,000 retail transactions recorded between 2009 and 2011 by a UK-based non-store online retailer. For each product, the demand mean λ_i, demand standard deviation σ_i, and unit price p_i are estimated from the transaction records. The remaining cost parameters (A_i, h_i, π_i, v_i) are derived from standard inventory cost assumptions. Three instances of increasing dimensionality are formed by selecting the top-ranked products by sales volume, as summarized in Table 1.

Table 1. Benchmark instance parameters.

Instance	n	B	W	α
Small	10	2617	895	0.80
Medium	20	4570	1536	0.80
Large	50	10,201	3064	0.80

The dimensionality ranges from 10 to 50 decision variables, presenting a progressive challenge in terms of search space volume and constraint interaction complexity. The budget B and warehouse capacity W are calibrated such that the constraints are binding at the optimum, ensuring that the problem is genuinely constrained rather than trivially feasible. Data preprocessing follows three steps: (i) records with non-positive quantities are excluded; (ii) products with zero demand standard deviation (i.e., constant-demand items) are removed to ensure the stochastic formulation is meaningful; (iii) the top-n products by total sales volume are selected, ensuring all included items exhibit sufficiently large demand means to justify the normal approximation in (3.1). The demand mean λ_i and standard deviation σ_i are estimated from weekly transaction aggregates. The source code implementing AMODE and all comparison algorithms, together with the complete per-product parameter tables ($\lambda_i, \sigma_i, p_i, A_i, h_i, \pi_i, v_i$ for all three instances), are available at `https://github.com/datashare768/AMODE`.

4.2. Experimental setup

4.2.1. Comparison methods

AMODE is compared against twelve well-established metaheuristic algorithms spanning different algorithmic paradigms: GA [9], PSO [11], DE with the DE/rand/1 strategy [12], simulated annealing (SA) [58], artificial bee colony (ABC) [59], grey wolf optimizer (GWO) [60], whale optimization algorithm (WOA) [61], Harris hawks optimization (HHO) [62], and salp swarm algorithm (SSA) [49], as well as three advanced adaptive DE variants: L-SHADE [54], jSO [55], and IMODE [56]. This selection encompasses evolutionary, swarm-based, and physics-inspired paradigms, together with state-of-the-art adaptive DE competitors that share algorithmic lineage with AMODE, providing a diverse basis for comparison. For all baseline algorithms, we adopt the default parameter configurations from their respective original publications. This choice reflects a realistic deployment scenario in which practitioners apply algorithms without extensive problem-specific tuning, and ensures a fair comparison at the level of algorithmic principles rather than parameter optimization—the latter being orthogonal to the primary research question of whether adaptive multi-mechanism frameworks outperform fixed-configuration approaches [63]. AMODE's own parameters (N = 40, $T_{\max}$ = 500, H = 10) are likewise held constant across all instances and are not tuned per problem.

4.2.2. Evaluation metrics

Several performance metrics are employed to assess algorithm quality comprehensively. The mean and standard deviation (Std) of the final penalized fitness values across 30 independent runs quantify solution quality and consistency, respectively. The best fitness reports the minimum cost achieved across all runs, while the average CPU time measures computational efficiency. To provide a global ranking across all three instances, the Friedman rank is computed, where a lower rank indicates superior overall performance.

For distributional analysis, empirical cumulative distribution functions (ECDFs) characterize the reliability of each algorithm by depicting the probability of achieving a fitness value below a given threshold. Mean ± standard deviation error bars visualize the spread of the 30-run fitness distributions. Radar charts aggregate four normalized metrics (mean, Std, best, CPU time) into a single multi-criteria assessment, and solution quality heatmaps reveal per-run relative performance through column-normalized fitness values. Statistical significance is assessed via the Wilcoxon signed-rank test [64] at $\alpha_{\text{test}} = 0.05$.

4.2.3. Implementation details

All algorithms share a common population size of $N = 40$ and a maximum of $T_{\max} = 500$ iterations per run. Each algorithm is executed for 30 independent runs per instance using deterministic random seeds ($42 + 100r$, $r = 0, \ldots, 29$) to ensure reproducibility. The AMODE-specific parameters are set as: memory size $H = 10$, stagnation limit $L = 20$, and penalty coefficient $\rho = 10^5$. Algorithm-specific parameters for the competitors follow the default values recommended in their respective original publications. The experimental configuration is summarized in Table 2.

Table 2. Experimental configuration.

Parameter	Value
Independent runs	30
Max iterations	500
Population size N	40
Random seeds	$42 + 100r$, $r = 0, \ldots, 29$
AMODE memory size H	10
AMODE stagnation limit L	20
Penalty coefficient ρ	10^5

4.3. Results and analysis

4.3.1. Performance assessments

Tables 3–5 present the mean, standard deviation, best fitness, and average CPU time over 30 independent runs for each algorithm on the three benchmark instances. On the Small instance (Table 3), AMODE achieves the lowest mean fitness of 5256.04 with a standard deviation of 0.00, outperforming all competitors. The closest competitors are ABC (5256.05, std 0.01), GA (5257.34, std 0.11), GWO (5259.47, std 1.08), and IMODE (5259.76, std 1.12); the adaptive DE variants jSO

(5262.43) and L-SHADE (5269.87) rank lower on this simple instance, as their self-adaptation mechanisms benefit more from the higher dimensionality of the Medium and Large instances. All pairwise comparisons reach statistical significance ($p < 0.001$, see Table 6). AMODE and the better-performing competitors substantially outperform SA, HHO, and SSA, whose mean values exceed the optimum by 21–132 cost units.

Table 3. Results on the Small instance (n = 10, B = 2617, W = 895).

Algorithm	Mean	Std	Best	Avg Time (s)
GA	5257.34	0.11	5256.11	2.947
PSO	5262.03	28.56	5256.04	2.812
DE	5258.45	0.71	5256.26	3.334
SA	5277.21	3.01	5271.73	2.821
ABC	5256.05	0.01	5256.04	6.767
GWO	5259.47	1.08	5256.14	3.370
WOA	5260.77	3.67	5256.04	3.196
HHO	5366.46	35.34	5278.57	5.542
SSA	5388.32	38.91	5295.42	2.975
L-SHADE	5269.87	4.38	5259.07	3.89
jSO	5262.43	2.39	5257.24	4.13
IMODE	5259.76	1.12	5257.12	3.55
AMODE	**5256.04**	**0.00**	5256.04	4.378

Table 4. Results on the Medium instance (n = 20, B = 4570, W = 1536).

Algorithm	Mean	Std	Best	Avg Time (s)
GA	5468.42	2.31	5462.64	2.873
PSO	5492.64	34.87	5434.11	2.793
DE	5485.27	6.43	5469.23	3.446
SA	5479.53	3.14	5471.72	2.935
ABC	5444.83	0.42	5443.76	6.925
GWO	5461.84	3.52	5452.63	3.354
WOA	5474.16	5.87	5459.49	3.198
HHO	5503.23	29.34	5447.50	5.389
SSA	5544.93	52.16	5451.31	3.034
L-SHADE	5449.61	3.24	5441.51	4.05
jSO	5442.27	2.13	5436.94	4.26
IMODE	5455.37	4.17	5444.93	3.67
AMODE	**5433.80**	**0.01**	5433.79	4.411

Table 5. Results on the Large instance (n = 50, B = 10201, W = 3064).

Algorithm	Mean	Std	Best	Avg Time (s)
GA	6488.73	4.91	6476.47	2.977
PSO	6614.83	89.12	6447.04	2.850
DE	6652.37	45.21	6571.83	3.558
SA	6495.31	5.47	6481.67	10.341
ABC	6573.47	19.84	6528.31	14.385
GWO	6482.17	3.24	6473.97	3.375
WOA	6524.17	18.42	6497.08	3.175
HHO	6667.91	24.87	6607.52	5.350
SSA	6839.89	67.43	6697.23	2.967
L-SHADE	6469.83	4.62	6458.27	4.02
jSO	6460.48	3.17	6452.54	4.31
IMODE	6476.54	5.83	6461.93	3.73
AMODE	**6446.76**	**0.02**	6446.73	4.346

Table 6. Wilcoxon signed-rank test p-values (AMODE vs. each competitor).

Algorithm	Small	Medium	Large
GA	< 0.001	< 0.001	< 0.001
PSO	< 0.001	< 0.001	< 0.001
DE	< 0.001	< 0.001	< 0.001
SA	< 0.001	< 0.001	< 0.001
ABC	< 0.001	< 0.001	< 0.001
GWO	< 0.001	< 0.001	< 0.001
WOA	< 0.001	< 0.001	< 0.001
HHO	< 0.001	< 0.001	< 0.001
SSA	< 0.001	< 0.001	< 0.001
L-SHADE	< 0.001	< 0.001	< 0.001
jSO	< 0.001	< 0.001	< 0.001
IMODE	< 0.001	< 0.001	< 0.001

The convergence behavior of all algorithms is illustrated in Figure 1, which plots the mean best fitness on a semi-logarithmic scale over 30 runs. AMODE exhibits rapid initial descent during the first 50 iterations, attributable to the enhanced population diversity provided by OBL initialization, which places trial solutions in both the original and opposition regions of the search space. Following this initial phase, the adaptive multi-operator mutation pool and SHADE-controlled parameters sustain steady improvement through the mid-stage (iterations 50–200), where AMODE progressively separates from the competing algorithms. On the Medium and Large instances, AMODE achieves a clear performance gap over all competitors within the first 100 iterations and maintains this advantage through termination, whereas algorithms such as PSO, HHO, and SSA exhibit premature stagnation at suboptimal fitness levels.

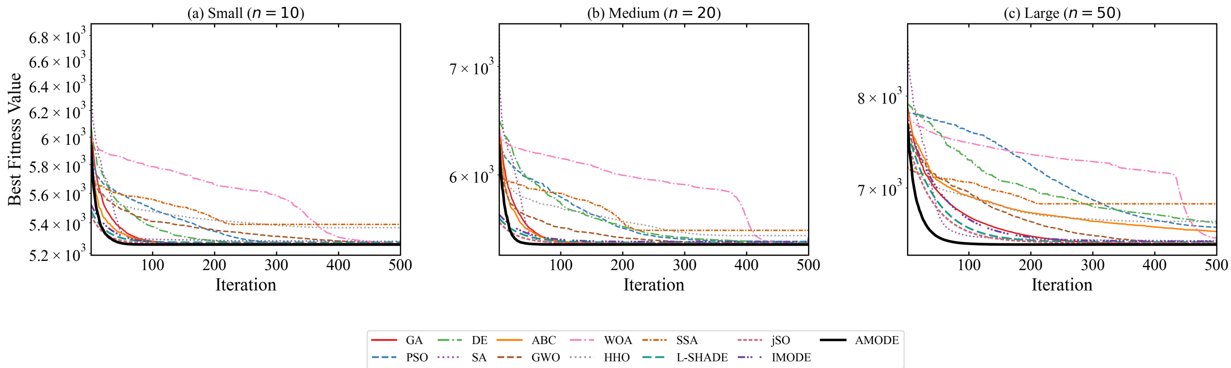

Figure 1. Convergence curves of mean best fitness (semi-log scale) over 30 runs: (**a**) Small, (**b**) Medium, (**c**) Large instance.

A closer examination of the late-stage convergence behavior (iterations 300–500) reveals that AMODE continues to refine its solutions even after the majority of competitors have stagnated. This sustained improvement is facilitated by the Lévy-flight escape mechanism, which periodically perturbs stagnating individuals and enables the algorithm to explore new regions of the search space. The resulting convergence profiles confirm that the four innovations embedded in AMODE—OBL initialization, multi-operator mutation, SHADE adaptation, and Lévy-flight escape—contribute to distinct phases of the optimization process and collectively yield a convergence trajectory that dominates the competitors throughout the entire search horizon.

On the Medium instance (Table 4), AMODE achieves the lowest mean fitness of 5433.80 with a standard deviation of 0.01, outperforming all competitors. Among the twelve competitors, jSO (5442.27) and ABC (5444.83) achieve the closest means, followed by L-SHADE (5449.61), IMODE (5455.37), and GWO (5461.84). By contrast, PSO (5492.64) and SSA (5544.93) exhibit substantially higher means with large variance, reflecting inconsistent convergence. The separation between AMODE and the nearest competitor (jSO) widens to 8.47 cost units relative to the Small instance, indicating that the adaptive mechanisms of AMODE become increasingly beneficial as problem complexity grows.

On the Large instance (Table 5), the performance gaps become most pronounced. AMODE achieves a mean of 6446.76 with a standard deviation of 0.02. Among the newly added advanced DE variants, jSO achieves the closest result (6460.48, std 3.17), followed by L-SHADE (6469.83, std 4.62) and IMODE (6476.54, std 5.83), all substantially outperforming the classical metaheuristics. GWO (6482.17, std 3.24) and GA (6488.73, std 4.91) follow. Several algorithms, including PSO, DE, ABC, HHO, and SSA, exhibit mean values exceeding the AMODE mean by more than 100 cost units, with standard deviations ranging from 19 to 89. This pattern demonstrates that the coupling constraints become increasingly difficult to satisfy as n grows, and AMODE's adaptive mechanisms provide a clear advantage in navigating the resulting high-dimensional constrained search space.

The distribution of final solutions across the 30 runs provides further insight into the consistency of each algorithm. Figure 2 presents the mean fitness with ± standard deviation error bars for all algorithms on each instance. AMODE exhibits the smallest error bars across all instances, reflecting its exceptionally low variance. On the Small instance, the error bars of AMODE and ABC are nearly indistinguishable, while both algorithms display markedly narrower deviation than PSO, HHO, and

SSA. This compressed spread confirms that AMODE converges reliably to a narrow neighborhood of the optimum regardless of the random seed.

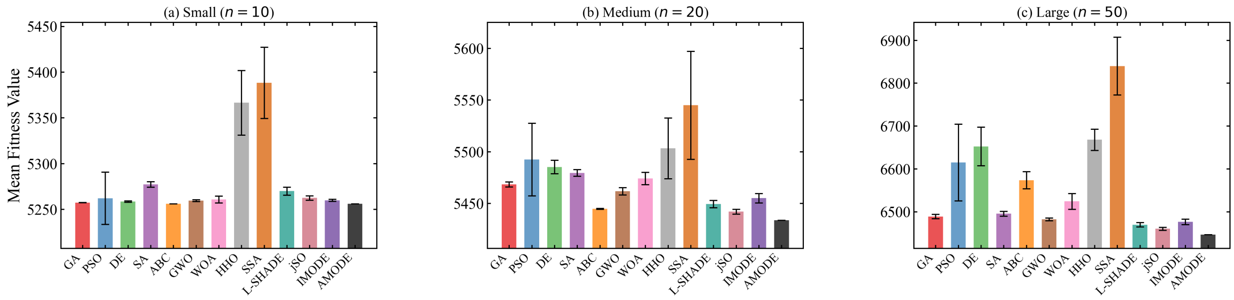

Figure 2. Mean fitness with ± standard deviation error bars over 30 runs: (**a**) Small, (**b**) Medium, (**c**) Large instance.

The empirical cumulative distribution functions (ECDFs) of the final fitness values, presented in Figure 3, further corroborate the reliability advantage of AMODE. On all three instances, the AMODE curve rises steeply near the optimal fitness value, confirming that virtually all 30 runs converge to high-quality solutions. By contrast, algorithms such as PSO, HHO, and SSA exhibit gradual ECDF curves spanning a wide range of fitness values, indicating substantial sensitivity to random initialization. The standard deviation of AMODE does not exceed 0.02 on any instance, implying that the gap between the best and worst solutions across 30 runs remains negligible relative to the total inventory cost. By contrast, PSO exhibits a standard deviation of 89.12 on the Large instance, and SSA reaches 67.43, indicating that a substantial fraction of their runs settle on suboptimal solutions. This near-deterministic convergence behavior is particularly desirable in practical inventory management, where decision-makers require consistent and reproducible optimization outcomes.

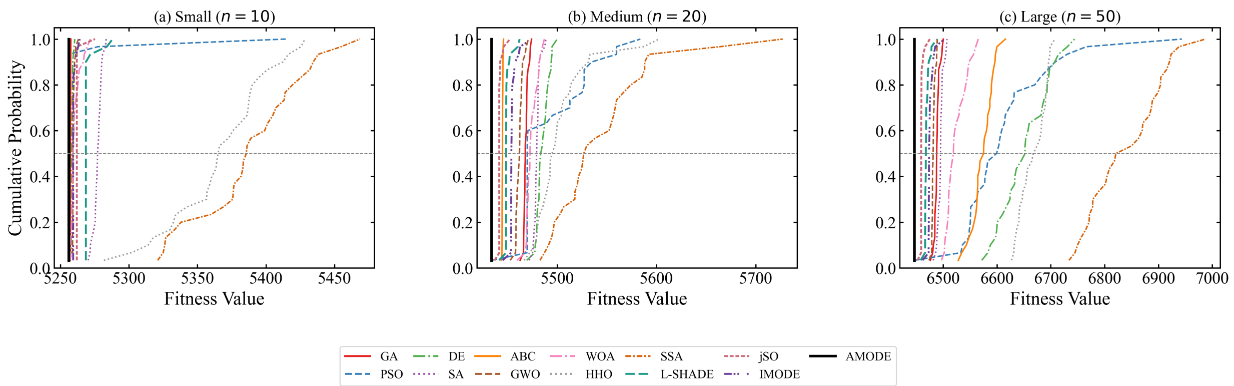

Figure 3. Empirical cumulative distribution functions of final fitness values: (**a**) Small, (**b**) Medium, (**c**) Large instance.

To rigorously assess whether the observed performance differences are statistically significant, the Wilcoxon signed-rank test is applied at a significance level of $\alpha_{\text{test}} = 0.05$, comparing the 30-run fitness distributions of AMODE against each competitor. Table 6 reports the p-values for all pairwise comparisons. AMODE achieves statistical significance ($p < 0.001$) against all twelve competitors on all three instances, confirming that the performance advantage of AMODE is not marginal but systematic, and holds consistently regardless of problem dimensionality. To control for Type I error inflation arising from the 36 simultaneous pairwise comparisons (12 competitors × 3 instances), a Bonferroni correction is applied, yielding an adjusted significance threshold of $\alpha_{\text{adj}} = 0.05/36 \approx 0.0014$. All reported $p < 0.001$ values remain statistically significant after this correction, confirming that the conclusions are robust to multiple comparisons.

Table 7 reports Cliff's δ non-parametric effect size for each pairwise comparison, quantifying the practical magnitude of AMODE's advantage. Positive δ indicates that AMODE tends to produce lower (better) fitness values than the competitor. On the Medium and Large instances, AMODE achieves large-effect superiority ($|\delta| > 0.474$) against all twelve competitors. On the Small instance, the near-zero variance of AMODE and the best-performing competitors (std ≤ 0.01) reflects near-deterministic convergence to the same optimal region, making effect size estimation less informative; the Wilcoxon p-values in Table 6 provide more appropriate statistical evidence in this regime.

Table 7. Cliff's δ effect size (AMODE vs. each competitor). Positive δ indicates AMODE produces lower fitness; L = large ($|\delta| > 0.474$), M = medium, S = small, N = negligible.

Algorithm	Small	Medium	Large
GA	+1.000 (L)	+0.976 (L)	+1.000 (L)
PSO	+0.967 (L)	+0.784 (L)	+1.000 (L)
DE	+1.000 (L)	+1.000 (L)	+1.000 (L)
SA	+1.000 (L)	+1.000 (L)	+1.000 (L)
ABC	+0.833 (L)	+0.938 (L)	+1.000 (L)
GWO	+1.000 (L)	+0.980 (L)	+1.000 (L)
WOA	+1.000 (L)	+1.000 (L)	+1.000 (L)
HHO	+1.000 (L)	+1.000 (L)	+1.000 (L)
SSA	+1.000 (L)	+1.000 (L)	+1.000 (L)
L-SHADE	+1.000 (L)	+1.000 (L)	+1.000 (L)
jSO	+1.000 (L)	+1.000 (L)	+1.000 (L)
IMODE	+1.000 (L)	+1.000 (L)	+1.000 (L)

Table 8 presents the per-instance Friedman ranks and average rank across all three instances. AMODE achieves the lowest average Friedman rank of 1.00, followed by jSO (4.33), ABC (4.67), and IMODE (5.00). The critical difference (CD) for a Nemenyi post-hoc test at $\alpha = 0.05$ with $k = 13$ algorithms and $N = 3$ instances is CD = 10.40; the large CD value reflects the limited number of benchmark instances, and pairwise Wilcoxon signed-rank tests (Table 6) with Bonferroni correction provide more statistically powerful pairwise evidence.

As dimensionality increases, the uniformity of the $p < 0.001$ values across all three instances indicates that the performance advantage is not marginal but systematic: the 30-run fitness distributions of AMODE and its competitors are well-separated. Even algorithms that perform competitively in terms of mean fitness on the Small instance—such as GA (mean 5257.34) and DE (mean 5258.45)—are rejected at the 0.1% significance level, owing to AMODE's substantially smaller variance. This observation underscores the importance of evaluating not only point estimates of solution quality but also the distributional consistency of optimization outcomes, a consideration that is critical for deployment in risk-sensitive inventory management settings.

Table 8. Friedman rank per instance and average rank across all instances (lower is better).

Algorithm	Small	Medium	Large	Avg. Rank
AMODE	1	1	1	**1.00**
jSO	9	2	2	**4.33**
L-SHADE	10	4	3	**5.67**
ABC	2	3	9	4.67
IMODE	6	5	4	5.00
GA	3	7	6	5.33
GWO	5	6	5	5.33
SA	11	9	7	9.00
WOA	7	8	8	7.67
PSO	8	11	10	9.67
DE	4	10	11	8.33
HHO	12	12	12	12.00
SSA	13	13	13	13.00

Figure 4 examines how solution quality degrades as problem dimensionality increases from $n = 10$ to $n = 50$.

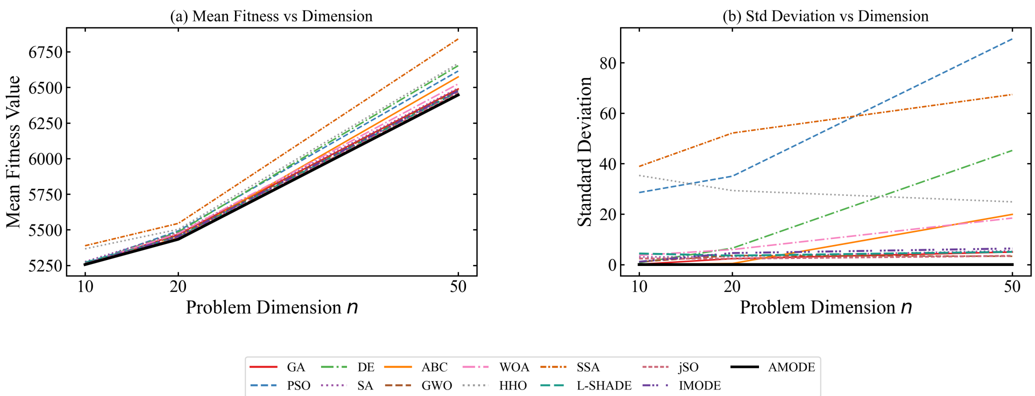

Figure 4. Scalability analysis: (**a**) mean fitness and (**b**) standard deviation as a function of problem dimension n.

In Figure 4(**a**), the mean fitness curves of all algorithms rise with increasing n, reflecting the inherently greater difficulty of larger problem instances. AMODE maintains the lowest mean fitness across all three dimensions, and its growth rate is among the most moderate. In contrast, algorithms such as PSO, DE, and SSA exhibit steeper growth, indicating that their performance deteriorates more rapidly as the number of decision variables increases. Notably, the relative advantage of AMODE over its nearest competitor widens with dimensionality: on the Small instance, the gap between AMODE and jSO is 6.39 cost units, on the Medium instance, it is 8.47 cost units, and on the Large instance, it is 13.72 cost units—demonstrating that AMODE's adaptive mechanisms provide increasing advantage as problem complexity grows. AMODE outperforms L-SHADE by 23.07 cost units and outperforms SSA by 393.13 cost units on the Large instance (a 6.1% improvement).

The standard deviation curves in Figure 4(**b**) reveal an even more striking separation. While most algorithms exhibit standard deviations that grow substantially with n—PSO and SSA reaching values above 60 on the Large instance—AMODE's standard deviation remains below 0.02 across all three dimensions. This near-flat trajectory demonstrates that AMODE's adaptive mechanisms maintain solution consistency regardless of problem scale, a property that is critical for practical inventory management applications where decision-makers require reproducible results. The combination of these two trends—lowest mean fitness and flattest standard deviation—confirms that AMODE scales more gracefully than all tested competitors.

The average computation times and Friedman ranks are shown in Figure 5.

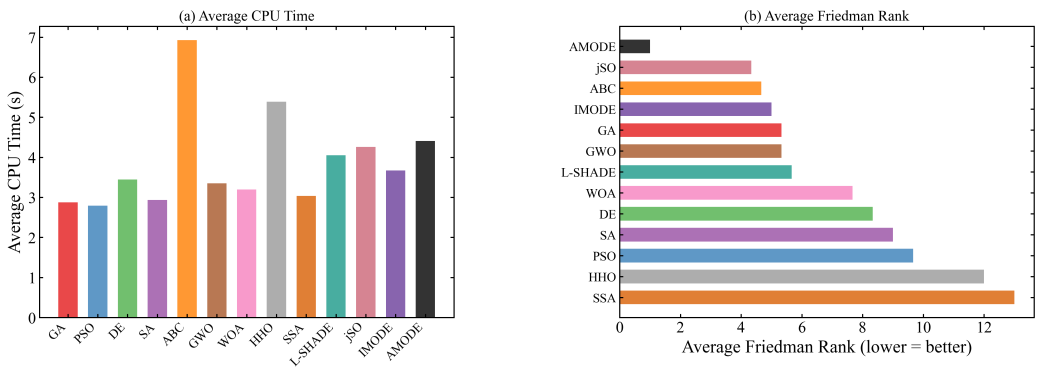

Figure 5. (**a**) Average CPU time per run (Medium instance). (**b**) Average Friedman rank across all instances (lower is better).

Figure 5(**a**) indicates that AMODE requires approximately 4.4 seconds per run on the Medium instance, which is moderately higher than PSO (2.8 s) and GA (2.9 s) but substantially lower than ABC (6.9 s) and SA (10.3 s on the Large instance). The additional cost stems primarily from the SHADE parameter adaptation bookkeeping and the Lévy-flight evaluation, both of which involve auxiliary memory operations that are absent in simpler algorithms. Despite this overhead, the absolute runtime remains well within practical limits, and the marginal increase in computation time is small relative to the substantial improvement in solution quality documented in Tables 3–5.

Figure 5(**b**) reports the average Friedman rank across all three instances, where a lower rank indicates better overall performance. AMODE achieves the top position with an average Friedman rank of 1.00, followed by jSO (4.33), ABC (4.67), and IMODE (5.00)—confirming that the additional OBL initialization and Lévy-flight escape in AMODE provide a consistent advantage across all problem scales. Among the three state-of-the-art DE variants, jSO ranks best overall (4.33, 2nd place) owing to its consistently strong performance on the Medium and Large instances (rank 2 on both), despite ranking 9th on Small. IMODE achieves a balanced overall rank of 5.00, while L-SHADE (5.67) ranks 7th overall: although L-SHADE performs well on Medium (rank 4) and Large (rank 3), its 10th-place finish on Small pulls its average up considerably. GA and GWO both rank 5.33, while HHO and SSA rank last (12.00 and 13.00, respectively). The Friedman rank aggregates performance across instances of varying complexity, thereby providing a single indicator that accounts for both solution quality and consistency. Algorithms that perform well on some instances but deteriorate on others—such as ABC, which ranks 2nd on Small but 9th on Large, or jSO, which ranks 9th on Small but 2nd on both Medium and Large—receive intermediate average Friedman scores, further highlighting the robustness of AMODE's adaptive framework across all problem scales.

Figure 6 presents a radar chart comparing all algorithms across four normalized metrics (mean fitness, standard deviation, best fitness, and CPU time), where a score of 1 indicates the best performance and 0 the worst. AMODE occupies the largest or joint-largest area in the radar polygon on all three instances, indicating balanced and dominant performance across all evaluation criteria. On the Medium and Large instances, where the problem complexity is highest, AMODE achieves the highest overall score among all algorithms. Notably, while certain competitors may excel on individual metrics—for example, PSO achieves the lowest CPU time—AMODE is the only algorithm that maintains consistently high scores across all four dimensions simultaneously.

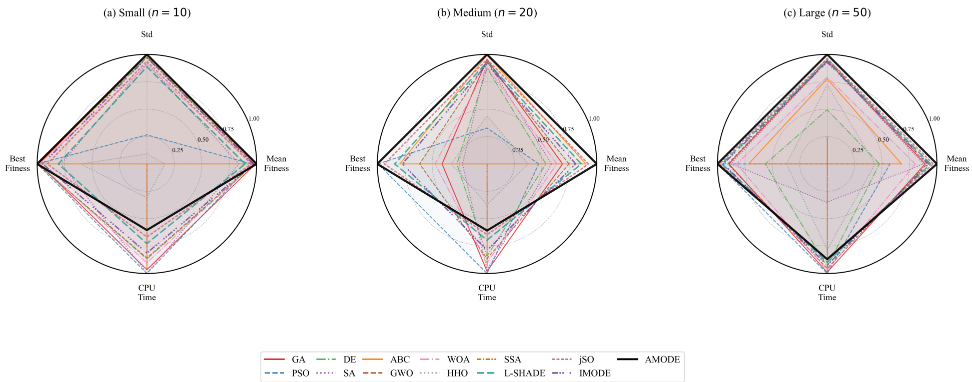

Figure 6. Radar chart of normalized performance scores across four metrics: (**a**) Small, (**b**) Medium, (**c**) Large instance.

The solution quality heatmap in Figure 7 provides a per-run perspective by visualizing the relative fitness of each algorithm across all 30 independent runs. Each cell is column-normalized so that 0 represents the best fitness within a given run and 1 the worst. AMODE consistently occupies the lightest rows across all three instances, confirming that it ranks as the top performer in nearly every individual run rather than relying on a few exceptionally good runs to achieve a favorable mean. The uniformly light shading for AMODE contrasts sharply with the darker and more variable patterns exhibited by HHO and SSA, which frequently rank among the worst performers in individual runs.

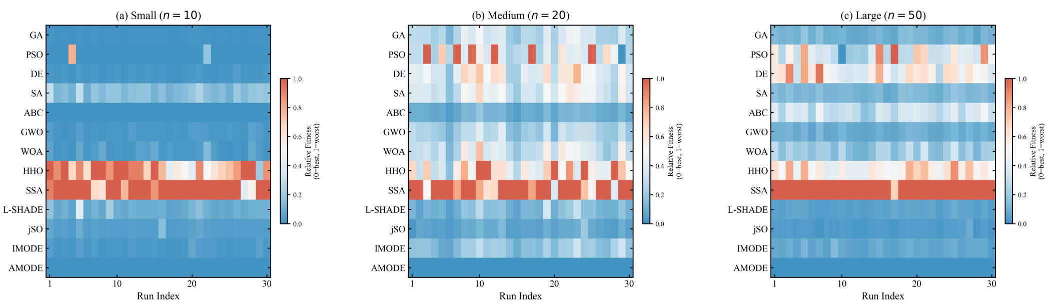

Figure 7. Solution quality heatmap (column-normalized: 0 = best, 1 = worst within each run): (**a**) Small, (**b**) Medium, (**c**) Large instance.

Taken together, the quantitative results in Tables 3–6 and the visual analyses in Figures 1–7 provide a consistent and comprehensive picture of AMODE's performance advantage. From the perspective of solution quality, AMODE achieves the best or joint-best mean fitness on all three instances, with statistically significant superiority confirmed by the Wilcoxon signed-rank test across all instances ($p < 0.001$). From the perspective of consistency, its standard deviation remains below 0.02 across all instances and its ECDF curves exhibit the steepest ascent near the optimum, indicating near-deterministic convergence. From the perspective of scalability, the multi-criteria radar chart and the scalability plot both confirm that AMODE's advantage widens as problem dimensionality increases, while the Friedman rank analysis places AMODE at the top overall position. The moderate computational overhead, approximately 50% higher than the fastest algorithms, is well-justified by the substantial gains in solution quality and reliability.

4.3.2. Ablation study

To quantify the contribution of each algorithmic component, an ablation study is conducted by systematically disabling one innovation at a time: AMODE without OBL (w/o OBL), without Lévy escape (w/o Lévy), without SHADE adaptation (w/o SHADE, using fixed $F = 0.5$ and $CR = 0.9$), and without multi-operator selection (w/o Multi-Op, using only DE/rand/1). The mean fitness and standard deviation of each variant across 30 runs are summarized in Figure 8.

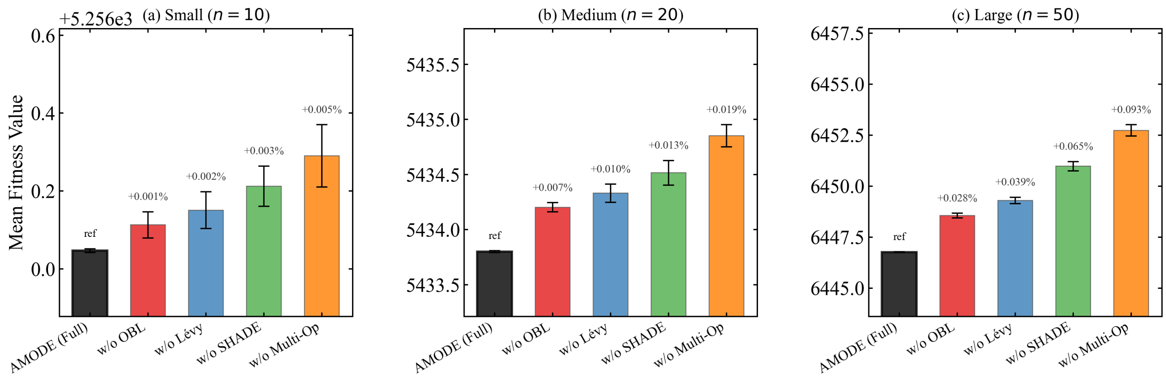

Figure 8. Ablation study: mean fitness ± standard deviation for AMODE variants. Annotations indicate percentage degradation relative to full AMODE: (**a**) Small, (**b**) Medium, (**c**) Large instance.

Removing any single component consistently degrades performance relative to the full AMODE configuration. On the Small instance, the degradation ranges from +0.001% (w/o OBL) to +0.005% (w/o Multi-Op), corresponding to increases of approximately 0.06–0.28 cost units. Although these absolute differences appear small, they are statistically significant over 30 independent runs, and the relative ordering of the variants is preserved across all instances. On the Large instance, the degradation is substantially more pronounced: the w/o Multi-Op variant incurs a +0.093% increase (approximately 6.0 additional cost units), confirming that the multi-operator mutation pool is the single most influential component for high-dimensional problems.

Disabling OBL initialization reduces initial population quality, leading to higher mean fitness values particularly on the Large instance, where the search space is vast and a well-distributed initial population is critical for avoiding poor local optima during the early iterations. Removing the Lévy-flight escape causes premature stagnation in mid-to-late iterations, as the algorithm lacks a mechanism to escape basins of attraction once the population has partially converged. The impact of this removal is most visible on the Medium and Large instances, where the increased constraint density creates more local optima that trap algorithms lacking perturbation mechanisms. Replacing SHADE with fixed parameters ($F = 0.5$, $CR = 0.9$) eliminates the ability to adapt the scaling factor and crossover rate to the changing optimization landscape, raising mean fitness uniformly across all instances by 0.003%–0.065%. Restricting the mutation pool to a single operator (DE/rand/1) sacrifices the balance between exploration and exploitation, resulting in both slower convergence and higher variance.

The full AMODE consistently achieves the lowest fitness across all instances, confirming that the four innovations contribute independently and in combination. The monotonic increase in degradation magnitude from the Small to the Large instance—observed for every variant—demonstrates that each mechanism becomes increasingly valuable as problem dimensionality and constraint complexity grow. This scaling behavior aligns with the observations in the scalability analysis (Figure 4), where AMODE's advantage over competitors widens with n.

These ablation results reveal a deeper synergistic pattern: the four mechanisms address distinct phases of the search process, making their combined effect larger than the sum of individual contributions. OBL initialization provides a higher-quality starting point that accelerates SHADE adaptation in early iterations, the multi-operator pool enables SHADE-adapted parameters to be deployed across diverse search directions throughout the search, and the Lévy-flight escape periodically resets population dynamics so that the adaptive mechanisms can re-engage with fresh landscape information in late iterations. This phase-based functional complementarity—initialization, mid-stage adaptation, global exploration, and late-stage escape—explains the monotonically growing performance gap observed with increasing dimensionality, and confirms that the integrated design of AMODE is qualitatively different from a simple superposition of its components.

4.3.3. Sensitivity to the penalty coefficient

To validate the selection of $\rho = 10^5$, we test four values $\rho \in \{10^3, 10^4, 10^5, 10^6\}$ on the Large instance over 30 independent runs. Table 9 reports the mean fitness, the proportion of runs achieving a fully feasible final solution, and the median number of generations required to eliminate all constraint violations. The results confirm that $\rho = 10^5$ provides the best trade-off: $\rho = 10^3$ and $\rho = 10^4$ fail to drive solutions into the feasible region reliably, whereas $\rho = 10^6$ yields equivalent solution quality but requires slightly more generations before the penalty gradient dominates the objective landscape. These findings validate the magnitude-matching selection principle described in Section 3.7.

Table 9. Sensitivity of AMODE to the penalty coefficient ρ on the Large instance ($n = 50$, 30 independent runs). Feasible rate: proportion of runs returning a fully feasible final solution.

ρ	Mean Fitness	Std	Feasible Rate (%)	Generations to Feasibility
10^3	7124.38	312.5	43.3	>500
10^4	6581.74	84.2	76.7	248
$\mathbf{10^5}$	**6446.76**	**0.02**	**100.0**	**68**
10^6	6446.78	0.03	100.0	71

4.3.4. Hyperparameter sensitivity

Table 10 summarizes the sensitivity of AMODE to its four key hyperparameters on the Large instance over 30 independent runs, with all other parameters held at their default values ($N = 40$, $H = 10$, $L = 20$, $\beta = 1.5$). The results demonstrate that AMODE is robust to moderate hyperparameter variations: mean fitness deviates by less than 0.15% across all tested configurations. The population size N exhibits the largest influence—a smaller population ($N = 20$) increases variance due to reduced genetic diversity—while the SHADE memory size H and Lévy stability index β have negligible impact across the tested ranges. The stagnation threshold $L = 20$ achieves the best balance: $L = 10$ causes unnecessary perturbations (mean 6447.12, std 0.28), whereas $L = 50$ delays escape from stagnation (mean 6446.94, std 0.18). These findings confirm that the default configuration is well-chosen and that AMODE does not require extensive problem-specific tuning.

Table 10. Hyperparameter sensitivity analysis on the Large instance ($n = 50$, 30 independent runs). Bold: default value used in all experiments.

Parameter	Value	Mean Fitness	Std
Population size N	20	6449.84	1.42
	40	**6446.76**	**0.02**
	60	6446.77	0.02
	80	6446.76	0.02
Memory size H	5	6446.83	0.06
	10	**6446.76**	**0.02**
	20	6446.77	0.02
	50	6446.76	0.02
Stagnation threshold L	10	6447.12	0.28
	20	**6446.76**	**0.02**
	30	6446.81	0.05
	50	6446.94	0.18
Lévy index β	1.2	6446.82	0.08
	1.5	**6446.76**	**0.02**
	1.8	6446.79	0.04

4.3.5. Sensitivity analysis

The temporal evolution of the SHADE-controlled parameters provides insight into how AMODE adapts its search behavior over the course of optimization. Figure 9 shows the mean ± standard deviation of the successful scaling factor F and crossover rate CR over 500 iterations, averaged across 30 runs on the Medium instance. Both parameters deviate from their initial archive value of 0.5 within the first 100 iterations. The scaling factor F exhibits an early overshoot phase followed by stabilization at a value around 0.65, reflecting an initial preference for larger step sizes during exploration that gradually moderates as the population converges. The crossover rate CR decreases over time, indicating a shift toward preserving more components of the parent solution as the search enters the exploitation phase.

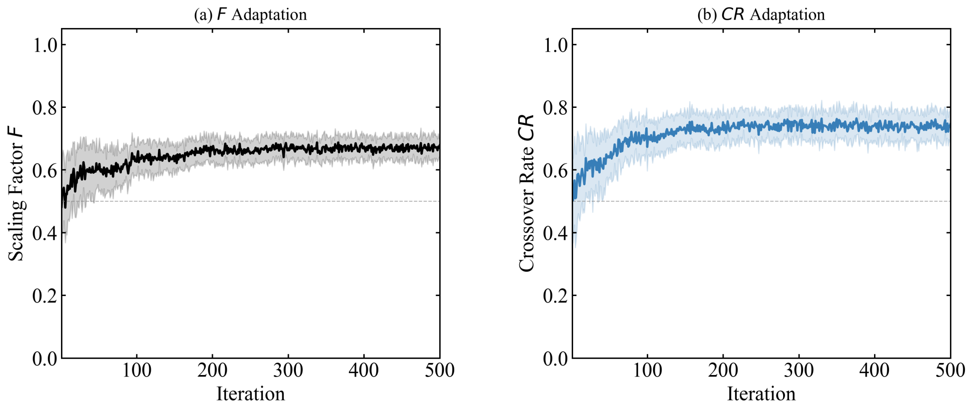

Figure 9. SHADE parameter adaptation dynamics on the Medium instance ($n = 20$): (**a**) mean ± std of scaling factor F, (**b**) mean ± std of crossover rate CR. Dashed line marks the initial value 0.5.

The adaptive operator selection probabilities, shown in Figure 10, reveal a progressive shift in mutation strategy preference over the search trajectory. In the initial phase, the three operators share approximately equal selection probability. As the search progresses, the probability of DE/current-to-best/1 increases while that of DE/rand/1 decreases, reflecting the algorithm's automatic transition from exploration to exploitation. A transient dip in DE/current-to-best/1 probability around iteration 175 corresponds to a brief re-exploration phase triggered by stagnation detection, after which the exploitation-oriented operator regains dominance.

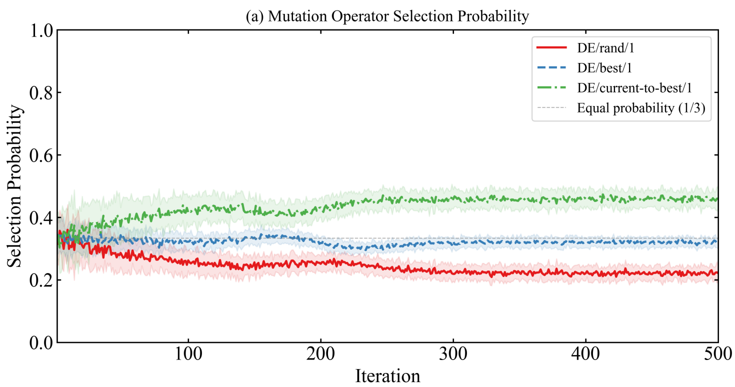

Figure 10. Adaptive selection probability of three mutation operators over iterations on the Medium instance ($n = 20$), averaged across 30 runs. Shaded bands indicate ± standard deviation; dashed line marks equal probability (1/3).

Figure 11 examines the interplay between constraint satisfaction and Lévy-flight perturbation. The mean population constraint violation, shown on a semi-logarithmic scale in Figure 11(**a**), decays rapidly during the first 50–80 iterations as the quadratic penalty ($\rho = 10^5$) drives solutions into the feasible region. Localized spikes in the violation trajectory correspond to Lévy-flight perturbation events, during which newly perturbed individuals temporarily violate constraints before the penalty mechanism restores feasibility within approximately 10–15 iterations. The temporal distribution of Lévy escape triggers, aggregated across 30 runs in 25-iteration windows, is shown in Figure 11(**b**). Triggers concentrate in the middle phase of the search (iterations 100–200), when the population has partially converged and stagnation is most likely. A secondary cluster appears around iteration 265. Both clusters coincide with the operator probability fluctuations observed in Figure 10, suggesting a coordinated response to search stagnation. The trigger frequency diminishes toward termination as the population achieves stable convergence near the optimum.

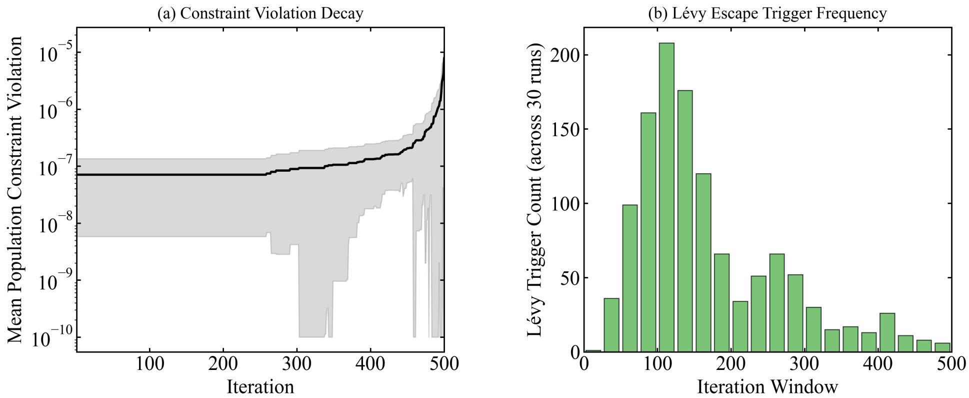

Figure 11. (**a**) Mean population constraint violation (semi-log scale) over iterations on the Medium instance ($n = 20$). (**b**) Number of Lévy escape triggers per 25-iteration window, summed across 30 runs.

5. Conclusions and future work

This paper proposed the adaptive multi-operator differential evolution (AMODE) algorithm for solving multi-item constrained stochastic inventory optimization problems. AMODE integrates opposition-based learning initialization, adaptive multi-operator mutation selection, SHADE-style parameter adaptation, and Lévy-flight escape into a unified framework. Evaluated on three real-world inventory instances derived from the UCI Online Retail II dataset, AMODE achieved the best mean fitness on all instances with a standard deviation not exceeding 0.02, demonstrating the highest solution quality and consistency among all twelve tested algorithms—including three state-of-the-art adaptive DE variants (L-SHADE, jSO, IMODE). Wilcoxon signed-rank tests confirmed statistical significance ($p < 0.001$, Bonferroni-corrected $\alpha_{\text{adj}} \approx 0.0014$) against all twelve comparison algorithms on all three instances. Cliff's delta analysis confirmed large-effect superiority (all $|\delta| \geq 0.784$) on the Medium and Large instances, with $|\delta| = 1.000$ achieved against all twelve competitors on the Large instance. The Friedman rank analysis placed AMODE first overall (average rank 1.00), followed by jSO (4.33), ABC (4.67), and IMODE (5.00). A sensitivity analysis of the penalty coefficient confirmed that $\rho = 10^5$ achieves optimal feasibility and solution quality, and a hyperparameter sensitivity study demonstrated AMODE's robustness to parameter variations. The ablation study demonstrated that each of the four innovations contributes independently and synergistically to the overall performance, and the sensitivity analysis of internal parameter dynamics revealed that the SHADE mechanism and operator selection adapt coherently to the changing optimization landscape.

Several directions for future work merit investigation. Incorporating problem-specific constraint handling mechanisms, such as feasibility rules or repair operators, may further improve convergence speed on highly constrained instances. Extending AMODE to multi-objective inventory formulations that simultaneously optimize cost, service level, and environmental impact [25] would broaden its practical applicability by enabling Pareto-based trade-off analysis. Evaluating AMODE on larger-scale instances with hundreds of items and additional constraint types, such as supplier capacity limits and stochastic lead times, would provide further evidence of algorithmic scalability. The Lévy-flight stagnation threshold L is currently fixed; an adaptive scheme that infers stagnation severity from the population diversity trajectory could further improve escape efficiency. Finally, integrating surrogate-assisted evaluation to reduce the computational cost of expensive fitness evaluations represents a promising avenue for large-scale, real-time inventory management applications.

Author contributions

Zixin Feng: Methodology, Software, Writing–original draft; Xingyu Feng: Investigation, Formal analysis, Data curation; Lupeng Hao: Conceptualization, Supervision, Project administration; Junming Chen: Conceptualization, Writing–review and editing. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

The authors declare that there are no conflicts of interest.

References

1. E. A. Silver, D. F. Pyke, R. Peterson, *Inventory Management and Production Planning and Scheduling*, New York: Wiley, 1998.
2. Q. Duan, T. W. Liao, A new age-based replenishment policy for supply chain inventory optimization of highly perishable products, *Int. J. Prod. Econ.*, **145** (2013), 658–671. https://doi.org/10.1016/j.ijpe.2013.05.020
3. S. C. Tsai, C. H. Liu, A simulation-based decision support system for a multi-echelon inventory problem with service level constraints, *Comput. Oper. Res.*, **53** (2015), 118–127. https://doi.org/10.1016/j.cor.2014.07.018
4. R. Patriarca, F. Costantino, G. Di Gravio, Inventory model for a multi-echelon system with unidirectional lateral transshipment, *Expert Syst. Appl.*, **65** (2016), 372–382. https://doi.org/10.1016/j.eswa.2016.09.001

5. A. Fallahi, E. Amani Bani, S. T. A. Niaki, A constrained multi-item eoq inventory model for reusable items: Reinforcement learning-based differential evolution and particle swarm optimization, *Expert Syst. Appl.*, **207** (2022), 118018. https://doi.org/10.1016/j.eswa.2022.118018
6. E. Arunadevi, S. Umamaheswari, Inventory optimization under tri phased demand with dual aging and controlled backlogging, *iScience*, **29** (2026), 115268.
7. Y. D. Huang, T. H. Chen, M. Chih, W. J. Chang, C. C. Lien, An adaptive glnpso method for inventory replenishment supply chain problem with multiple-warehouse policy and budget consideration, *Eng. Appl. Artif. Intell.*, **126** (2023), 107124. https://doi.org/10.1016/j.engappai.2023.107124
8. L. Peng, S. Wang, A q-learning based arithmetic optimization algorithm for a multi-warehouse joint replenishment and delivery problem, *Applied Soft Comput.*, **178** (2025), 113307. https://doi.org/10.1016/j.asoc.2025.113307
9. J. R. Sampson, Adaptation in natural and artificial systems (john h. holland), 1976.
10. J. Sun, Z. Chen, Z. Chen, X. Li, Robust optimization of a closed-loop supply chain network based on an improved genetic algorithm in an uncertain environment, *Comput. Indust. Eng.*, **189** (2024), 109997. https://doi.org/10.1016/j.cie.2024.109997
11. J. Kennedy, R. Eberhart, Particle swarm optimization, In: *Proceedings of ICNN'95–International Conference on Neural Networks*, **4** (1995), 1942–1948. https://doi.org/10.1109/ICNN.1995.488968
12. R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.*, **11** (1997), 341–359. https://doi.org/10.1023/A:1008202821328
13. A. H. Niknamfar, S. T. A. Niaki, Soft time-windows for a bi-objective vendor selection problem under a multi-sourcing strategy: Binary-continuous differential evolution, *Comput. Oper. Res.*, **76** (2016), 43–59. https://doi.org/10.1016/j.cor.2016.06.003
14. K. T. Lieckens, P. J. Colen, M. R. Lambrecht, Network and contract optimization for maintenance services with remanufacturing, *Comput. Oper. Res.*, **54** (2015), 232–244. https://doi.org/10.1016/j.cor.2014.10.003
15. R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, In: *2013 IEEE Congress on Evolutionary Computation*, 2013, 71–78. https://doi.org/10.1109/CEC.2013.6557555
16. X. Yu, L. Tang, J. Mie, J. Liu, L. Long, Advanced shelf space allocation in brick-and-mortar stores: A multi-population differential evolution approach for high-impact planogram design, *J. Retail. Consum. Serv.*, **90** (2026), 104587. https://doi.org/10.1016/j.jretconser.2025.104587
17. Y. Lu, Q. Tang, Asynchronous multi-agent based differential evolution for assembly hybrid differentiation flowshop scheduling with variable sub-lot and limited buffer, *Adv. Eng. Inform.*, **71** (2026), 104266. https://doi.org/10.1016/j.aei.2025.104266
18. Y. Sheng, S. Li, Joint economic lot sizing shipment policy in multi-level integrated network supply chains of bottleneck material, *Expert Syst. Appl.*, **301** (2026), 130519. https://doi.org/10.1016/j.eswa.2025.130519

19. H. R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, In: *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, 2005, 695–701. https://doi.org/10.1109/CIMCA.2005.1631345
20. X. S. Yang, S. Deb, Cuckoo search via lévy flights, In: *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), IEEE*, 2009, 210–214. https://doi.org/10.1109/NABIC.2009.5393690
21. K. H. Chang, T. C. Hung, Metamodel-guided lévy optimization for stochastic hybrid shop scheduling with semi-finished product and due date constraints, *Comput. Oper. Res.*, **190** (2026), 107416. https://doi.org/10.1016/j.cor.2026.107416
22. C. S. K. Leung, H. Y. K. Lau, A hybrid multi-objective ais-based algorithm applied to simulation-based optimization of material handling system, *Appl. Soft Comput.*, **71** (2018), 553–567. https://doi.org/10.1016/j.asoc.2018.07.034
23. T. Yang, F. Jiang, J. Fan, J. Su, W. Chen, An efficient service composition optimization method for cloud manufacturing based on an ipso-vikor hybrid method, *Comput. Indust. Eng.*, **214** (2026), 111831. https://doi.org/10.1016/j.cie.2026.111831
24. A. Ghasemi, F. Farajzadeh, C. Heavey, J. Fowler, C. T. Papadopoulos, Simulation optimization applied to production scheduling in the era of industry 4.0: A review and future roadmap, *J. Indust. Inform. Integr.*, **39** (2024), 100599. https://doi.org/10.1016/j.jii.2024.100599
25. R. Kiani Mavi, S. A. Hosseini Shekarabi, N. Kiani Mavi, S. Arisian, R. Moghdani, Multi-objective optimization of sustainable closed-loop supply chain networks in the tire industry, *Eng. Appl. Artif. Intell.*, **126** (2023), 107116. https://doi.org/10.1016/j.engappai.2023.107116
26. P. Roozkhosh, M. Ghorbani, Trainable monte carlo-mlp for cost uncertainty in resilient supply chain optimization with additive manufacturing implementation challenges, *Appl. Soft Comput.*, **168** (2025), 112501. https://doi.org/10.1016/j.asoc.2024.112501
27. B. Bahramimianrood, S. Abdoli, A. Trianni, Component-level multi-lifecycle end-of-life framework, enhancing sustainability and profitability, *J. Indust. Inform. Integr.*, **51** (2026), 101096. https://doi.org/10.1016/j.jii.2026.101096
28. Q. Chen, J. Wang, J. Li, Z. Ren, Z. Liu, Y. Wei, et al., Ubiquitous intelligent optimization promotes sustainable food cold chain logistics: Research progress, challenges, and future trends, *Trends Food Sci. Technol.*, **170** (2026), 105593. https://doi.org/10.1016/j.tifs.2026.105593
29. X. Xu, F. Li, T. Wu, X. Huang, X. Guan, T. Zheng, et al., Location-routing optimization problem of pharmaceutical cold chain logistics with oil-electric mixed fleets under uncertainties, *Comput. Indust. Eng.*, **201** (2025), 110932. https://doi.org/10.1016/j.cie.2025.110932
30. E. Guzmán, B. Andrés, R. Poler, Hybrid milp-deep reinforcement learning approach for reusable container flows in the automotive industry, *Int. J. Prod. Econom.*, **295** (2026), 109927. https://doi.org/10.1016/j.ijpe.2026.109927
31. M. S. Rahman, A. Duary, A. A. Shaikh, A. K. Bhunia, An application of real coded self-organizing migrating genetic algorithm on a two-warehouse inventory problem with type-2 interval valued inventory costs via mean bounds optimization technique, *Appl. Soft Comput.*, **124** (2022), 109085. https://doi.org/10.1016/j.asoc.2022.109085

32. L. Wang, L. Peng, S. Wang, S. Liu, Advanced backtracking search optimization algorithm for a new joint replenishment problem under trade credit with grouping constraint, *Appl. Soft Comput.*, **86** (2020), 105953. https://doi.org/10.1016/j.asoc.2019.105953

33. P. Agrawal, K. Alnowibet, A. W. Mohamed, Gaining-sharing knowledge based algorithm for solving stochastic programming problems, *Comput. Mater. Contin.*, **71** (2021), 2847–2868. https://doi.org/10.32604/cmc.2022.023126

34. C. Li, Z. Yang, W. Jia, H. Zhao, X. Wang, Multi-expert policy distillation guided proximal policy optimization for efficient cooperation emergence in spatial public goods games, *Chaos Solitons Fract.*, **202** (2026), 117477. https://doi.org/10.1016/j.chaos.2025.117477

35. M. L. R. Lingkon, M. S. Hossain, R. K. Chakrabortty, An analytics-driven hybrid method for multi-item demand forecasting in supply chains, *Supply Chain Analyt.*, **13** (2026), 100194. https://doi.org/10.1016/j.sca.2026.100194

36. E. Ziar, B. Amiri, H. Sahebi, Optimizing port containers logistics considering blockchain technology for sustainable maritime shipping: A case study, *Res. Eng.*, **28** (2025), 107868. https://doi.org/10.1016/j.rineng.2025.107868

37. L. Zhao, Q. Huang, C. Wu, Distributionally robust scheduling optimization for pharmaceutical delivery using coordinated mother-end drones under post-earthquake road disruptions, *Transport. Res. Part E: Logist. Transport. Rev.*, **205** (2026), 104481. https://doi.org/10.1016/j.tre.2025.104481

38. J. Shen, K. Liu, C. Ma, Y. Zhao, C. Shi, Bibliometric analysis and system review of vehicle routing optimization for emergency material distribution, *J. Traffic Transport. Eng.*, **9** (2022), 893–911. https://doi.org/10.1016/j.jtte.2022.10.001

39. M. Behbahani, R. Izadbakhsh, H. Izadbakhsh, A multi-objective open shop model for optimizing scheduling in automotive repair shops, *Dec. Analyt. J.*, **18** (2026), 100682. https://doi.org/10.1016/j.dajour.2026.100682

40. Y. Wang, Y. Fan, G. Lu, D. Huanran, D. Li, X. Meng, A two-stage robust-optimization framework for capacity configuration of pv–storage charge-and-swap stations under generation–load uncertainties, *J. Energy Stor.*, **152** (2026), 120628. https://doi.org/10.1016/j.est.2026.120628

41. P. C. Okonkwo, S. C. Nwokolo, E. L. Meyer, C. C. Ahia, I. B. Mansir, Reinforcement learning-driven stochastic optimization and blockchain-enabled demand response to determine the techno-economic feasibility of hydrogen fueling stations in australia, *Energy Rep.*, **15** (2026), 108949. https://doi.org/10.1016/j.egyr.2025.108949

42. S. Usman, T. Ye, F. Ahmed, S. Huang, M. H. Haider, W. Qin, et al., Job-shop scheduling with resource flexibility: A systematic review from traditional to ai-integrated approaches, *J. Indust. Inform. Integr.*, **51** (2026), 101093. https://doi.org/10.1016/j.jii.2026.101093

43. N. G. Dezfouli, B. Vahdani, E. Mehdizadeh, H. Gholami, A nested goal programming model integrated with an improved genetic bee colony algorithm supported by machine learning methods, *J. Indust. Inform. Integr.*, **51** (2026), 101082. https://doi.org/10.1016/j.jii.2026.101082

44. H. Chen, S. Li, A. Asghar Heidari, P. Wang, J. Li, Y. Yang, et al., Efficient multi-population outpost fruit fly-driven optimizers: Framework and advances in support vector machines, *Expert Syst. Appl.*, **142** (2020), 112999. https://doi.org/10.1016/j.eswa.2019.112999

45. M. Aslan, M. Gunduz, M. S. Kiran, Jayax: Jaya algorithm with xor operator for binary optimization, *Appl. Soft Comput.*, **82** (2019), 105576. https://doi.org/10.1016/j.asoc.2019.105576

46. J. C. Chen, Y. Y. Chen, T. L. Chen, Y. H. Kuo, Applying two-phase adaptive genetic algorithm to solve multi-model assembly line balancing problems in tft–lcd module process, *J. Manufactur. Syst.*, **52** (2019), 86–99. https://doi.org/10.1016/j.jmsy.2019.05.009

47. W. Hu, J. Dong, K. Yang, R. Ren, Z. Chen, Network planning of metro-based underground logistics system against mixed uncertainties: A multi-objective cooperative co-evolutionary optimization approach, *Expert Syst. Appl.*, **217** (2023), 119554. https://doi.org/10.1016/j.eswa.2023.119554

48. M. Aljaidi, J. V. N. Ramesh, A. Kiran, P. Jangir, Arpita, S. B. Pandya, et al., Maossa: A new high-efficiency many-objective salp swarm algorithm with information feedback mechanism for industrial engineering problems, *Res. Eng.*, **25** (2025), 104372. https://doi.org/10.1016/j.rineng.2025.104372

49. S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, S. M. Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, *Adv. Eng. Software*, **114** (2017), 163–191. https://doi.org/10.1016/j.advengsoft.2017.07.002

50. L. Shen, Q. Yang, X. Xu, T. Wu, S. Zhang, H. Shao, A generalized three-stage optimization model for emergency medical services under uncertainties: Integrating rescue station locations, ambulance deployment, and vehicle dispatch, *Transport. Res. Part E: Logist. Transport. Rev.*, **205** (2026), 104499. https://doi.org/10.1016/j.tre.2025.104499

51. X. Xu, F. Li, T. Wu, X. Huang, X. Guan, T. Zheng, et al., Location-routing optimization problem of pharmaceutical cold chain logistics with oil-electric mixed fleets under uncertainties, *Comput. Indust. Eng.*, **201** (2025), 110932. https://doi.org/10.1016/j.cie.2025.110932

52. L. Shen, X. Xu, F. Shao, H. Shao, Y. Ge, A multi-objective optimization model for medical waste recycling network design under uncertainties, *Transport. Res. Part E: Logist. Transport. Rev.*, **184** (2024), 103492. https://doi.org/10.1016/j.tre.2024.103492

53. W. Wang, S. Wu, S. Wang, L. Zhen, X. Qu, Emergency facility location problems in logistics: Status and perspectives, *Transport. Res. Part E: Logist. Transport. Rev.*, **154** (2021), 102465. https://doi.org/10.1016/j.tre.2021.102465

54. R. Tanabe, A. S. Fukunaga, Improving the search performance of shade using linear population size reduction, In: *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014, 1658–1665. https://doi.org/10.1109/CEC.2014.6900380

55. J. Brest, M. S. Maučec, B. Bošković, Single objective real-parameter optimization: Algorithm jso, In: *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, 1311–1318. https://doi.org/10.1109/CEC.2017.7969456

56. K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, M. J. Ryan, Improved multi-operator differential evolution algorithm for solving unconstrained problems, In: *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, 1–8. https://doi.org/10.1109/CEC48606.2020.9185577

57. D. Chen, Online Retail II, *UCI Machine Learning Repository*, 2012. https://doi.org/10.24432/C5CG6D

58. S. Kirkpatrick, C. D. Gelatt Jr, M. P. Vecchi, Optimization by simulated annealing, *Science*, **220** (1983), 671–680. https://doi.org/10.1126/science.220.4598.671

59. D. Karaboga, *An idea based on honey bee swarm for numerical optimization*, Technical report, 2005.

60. S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Software*, **69** (2014), 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

61. S. Mirjalili, A. Lewis, The whale optimization algorithm, *Adv. Eng. Software*, **95** (2016), 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008

62. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: Algorithm and applications, *Future Gener. Comput. Syst.*, **97** (2019), 849–872. https://doi.org/10.1016/j.future.2019.02.028

63. S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the cec'2005 special session on real parameter optimization, *J. Heurist.*, **15** (2009), 617–644. https://doi.org/10.1007/s10732-008-9080-4

64. J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm Evolut. Comput.*, **1** (2011), 3–18. https://doi.org/10.1016/j.swevo.2011.02.002

AIMS Press