

Article

Storage Location Allocation and Crane Scheduling Optimization in Automated Storage and Retrieval Systems with Dual I/O Ports

Zihang Yuan [†] , Wenbin Zhang [†]  and Chunjiang Zhang ^{*}

School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China; u202310397@hust.edu.cn (Z.Y.); wenbinzhang@hust.edu.cn (W.Z.)

^{*} Correspondence: zhangcj@hust.edu.cn

[†] These authors contributed equally to this work.

Abstract

This paper addresses the challenge of integrated optimization for storage space allocation and crane scheduling in Automated Storage and Retrieval Systems (AS/RSs) under the complex constraints of dual I/O ports and Shared Storage strategies. The scheduling of multi-shuttle cranes in such environments constitutes a highly coupled NP-hard combinatorial optimization problem, where task assignment, sequencing, and dynamic storage location allocation must be solved simultaneously. To tackle this, this paper proposes an Elite-Driven Synchronized-Repair Memetic Algorithm (ED-SRMA) based on a sensor-informed total-travel-time objective formulated from horizontal and vertical crane displacements that can be measured by the crane drive sensors. The method combines a dual-layer representation, cross-layer synchronization, a problem-specific feasibility-repair operator, elite preservation, and budgeted local refinement. Numerical experiments were conducted on seven problem scales using 30 paired independent runs and a fixed evaluation budget. Compared with the corresponding dual-I/O operating mode without immediate location reuse, the shared-storage RS mode reduced the mean total travel time by 4.57–14.89%. ED-SRMA reduced the mean objective value by 5.48–39.59% relative to Base-GA and by 3.99–28.82% relative to PSO. For the representative 120×5 instance, the fitness-evaluation-based convergence analysis further shows an earlier objective-value reduction and continued improvement under the common evaluation budget. These results demonstrate the effectiveness, statistical significance, and consistency under the tested conditions of ED-SRMA for the investigated dual-I/O shared-storage scheduling problem.

Keywords: automated storage and retrieval system; scheduling optimization; repair-based constraint handling; memetic algorithm; sensor-informed scheduling



Academic Editor: Raffaele Bruno

Received: 21 May 2026

Revised: 23 June 2026

Accepted: 26 June 2026

Published: 2 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

An Automated Storage and Retrieval System (AS/RS) is the result of the integration of sensor technology [1], automated control technology [2], and Internet of Things (IoT) technology [3]. In this context, sensing and state acquisition provide the foundational layer for closed-loop scheduling, enabling the system to translate physical crane motion into quantifiable operational metrics. A typical AS/RS configuration consists of high-rise storage racks, input/output (I/O) stations, and automated stacker cranes that navigate aisles to perform tasks [4]. As illustrated in Figure 1, the fundamental workflow begins when storage or retrieval requests are generated by the Warehouse Management System (WMS). These tasks are then assigned to specific stacker cranes, which travel along the

aisles to deposit items into a designated storage location or retrieve items for downstream processing [5]. The efficiency of this “Task Assignment-Path Planning-Execution” cycle is the key determinant of the overall throughput of the warehouse system [6].

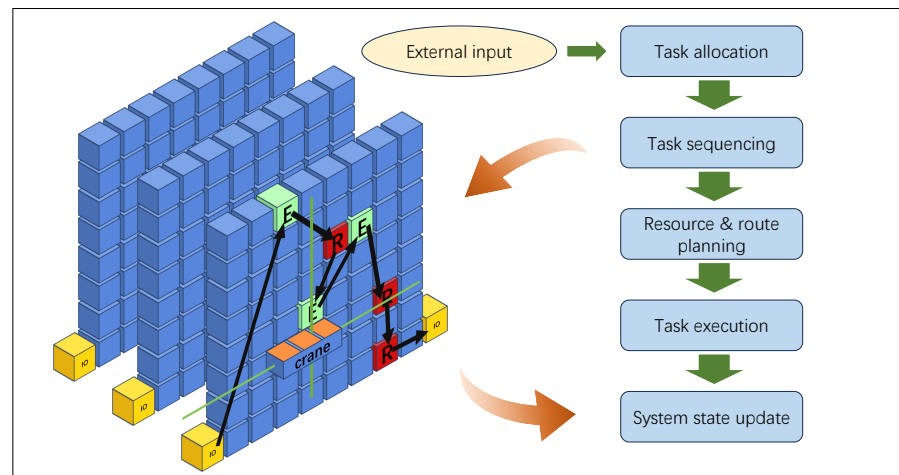


Figure 1. Illustration of the warehouse system workflow. Blue blocks represent storage racks; green and red blocks denote empty and retrieval locations, respectively; the orange block denotes the stacker crane; yellow blocks denote the I/O ports; and the arrows on the right indicate the scheduling and execution workflow.

Under the Industry 4.0 paradigm, the logistics sector is undergoing a significant transformation characterized by high-frequency, fragmented, and time-sensitive order structures [7–9]. Although intelligent warehousing presents a promising prospect for managing massive throughput with minimal human intervention, it also faces substantial operational challenges. The primary difficulty lies in scheduling multi-shuttle stacker cranes, a problem mathematically classified as an NP-hard combinatorial optimization task [10,11]. Unlike single-shuttle systems, multi-shuttle configurations introduce complex constraints related to capacity balancing and task sequencing [12]. Traditional heuristic methods often struggle to escape local optima [13,14] in these high-dimensional search spaces, creating a bottleneck where the physical capabilities of the hardware are limited by inefficient control algorithms [15].

The operating policy and I/O configuration further influence crane travel. Conventional non-shared storage separates storage (S) and retrieval (R) operations, as illustrated in Figure 2A. Under shared storage, a location released by a retrieval task can be immediately reused by a subsequent storage task. This retrieval–storage combination, referred to as an RS operation, eliminates the intermediate movement from the retrieval location to another empty location and can therefore reduce crane travel [16]. In addition, the dual-I/O configuration shown in Figure 2B,D allows the crane to select different entry and exit ports instead of returning to a fixed origin [17,18]. The combination of shared storage and dual-I/O flexibility provides additional opportunities for reducing empty travel, but it also couples storage-location allocation, within-trip operation sequencing, and inter-trip I/O-port selection.

Existing studies generally address only part of this coupled decision process. Storage-allocation studies commonly assume fixed command cycles or predetermined I/O configurations [19], whereas multi-I/O studies mainly optimize request sequencing or port assignment [20]. Research on double-end or multi-crane systems usually focuses on machine coordination and collision avoidance [21]. Consequently, limited attention has been given to the integrated scheduling of a single multi-shuttle stacker crane that performs interleaved storage and retrieval operations while the available empty locations and the starting

I/O port evolve continuously across successive trips. Furthermore, generic evolutionary operators do not inherently preserve task uniqueness, full-load requirements, within-trip prefix feasibility, or consistency between task assignment and operation sequencing.

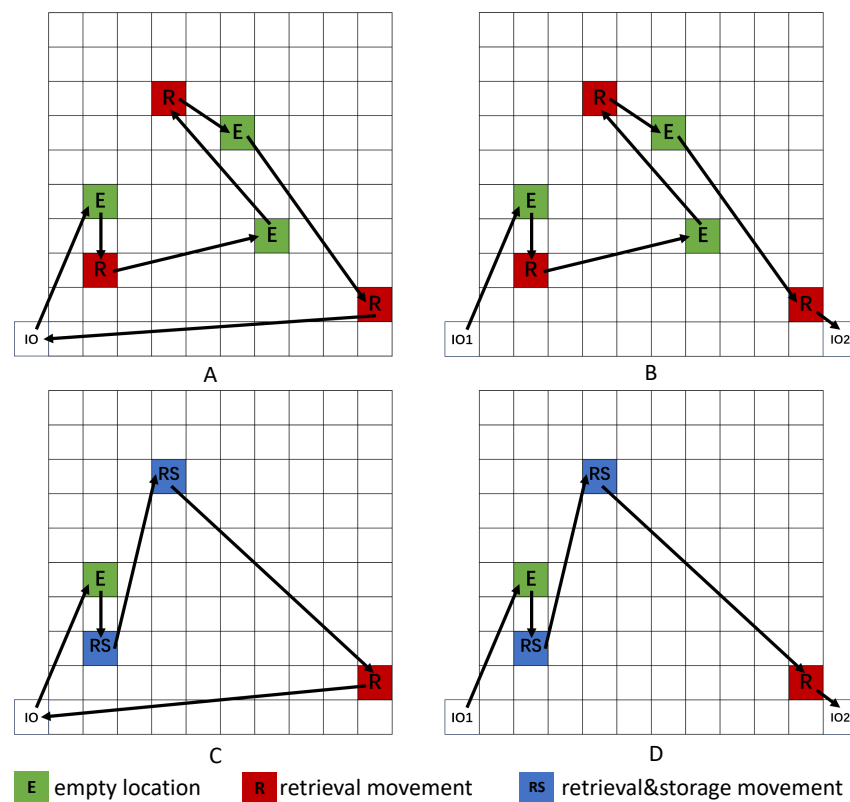


Figure 2. Schematic diagrams of storage (S), retrieval (R), and combined retrieval–storage (RS) operations under different topological layouts: (A) single-I/O non-shared storage; (B) dual-I/O non-shared storage; (C) single-I/O shared storage with RS operations; and (D) dual-I/O shared storage with RS operations. E, R, and RS denote an empty location, a retrieval movement, and a combined retrieval–storage movement, respectively.

To address these limitations, this study formulates an integrated optimization model for a dual-I/O middle-aisle AS/RS with shared storage and a multi-shuttle stacker crane. The model jointly considers task grouping, storage/retrieval sequencing, dynamic empty-location evolution, RS operations, crane-capacity constraints, and inter-trip I/O continuity. Crane travel time is evaluated through a sensor-informed motion model based on horizontal and vertical displacement information derived from the crane drive sensors. This formulation provides a measurable interface between the analytical scheduling objective and the physical motion of the stacker crane.

The scientific significance of this study lies in incorporating dynamic location reuse and I/O-state propagation into a unified state-dependent scheduling model. Unlike models that evaluate crane trips independently, the proposed formulation continuously transfers the empty-location set and the ending I/O state from one trip to the next. From an engineering perspective, the model enables the physical flexibility provided by shared storage, multi-shuttle handling, and dual I/O ports to be converted into feasible reductions in crane travel time.

The main contributions of this study are summarized as follows:

- An integrated dual-I/O shared-storage scheduling model is developed to jointly represent task grouping, storage/retrieval sequencing, dynamic empty-location allocation, RS operations, multi-shuttle capacity, and inter-trip I/O continuity.

- A dual-layer solution representation is introduced to separate retrieval-task assignment from within-trip operation sequencing. A state-dependent decoding procedure continuously updates the crane position and the available empty-location set over all trips.
- An Elite-Driven Synchronized-Repair Memetic Algorithm (ED-SRMA) is proposed. Its synchronization and heuristic repair procedures restore cross-layer consistency and prefix-load feasibility after genetic variation, while elite preservation and budgeted local refinement improve the retention and refinement of promising feasible solutions.

The remainder of this paper is organized as follows: Section 2 reviews the related work on AS/RS scheduling and evolutionary algorithms. Section 3 formulates the integrated scheduling problem and presents an illustrative example. Section 4 describes the ED-SRMA framework, including its solution representation, state-dependent decoding, feasibility repair, cross-layer synchronization, evolutionary operators, and local refinement. Section 5 presents the numerical experiments and statistical analyses. Finally, Section 6 concludes the paper and outlines future research directions.

2. Literature Review

AS/RS research mainly concerns storage-location assignment, storage/retrieval sequencing, crane routing, and travel-time reduction [4,22,23]. These decisions become strongly coupled under shared storage, multiple I/O ports, and multi-shuttle operation because retrieval tasks release reusable locations, storage decisions change subsequent routes, and the ending port of one trip affects the next trip.

2.1. Storage-Location Allocation and Crane Scheduling

Tanaka and Araki studied crane routing under shared storage with separate input and output points [24], while Chen et al. jointly considered storage-location assignment and interleaving decisions [25]. Yang et al. further developed a variable-neighborhood-search method for integrated storage-location allocation and request sequencing in multi-shuttle AS/RS [16]. These studies showed that storage assignment and crane scheduling should be optimized jointly rather than sequentially. Effective encoding is crucial for evolutionary success in high-dimensional problems. Zhang et al. [26] introduced a multi-layer encoding method for integrated storage allocation and crane scheduling. The present study inherits this general integrated-optimization perspective but addresses a distinct dual-I/O shared-storage problem involving RS operations, dynamic empty-location evolution, and inter-trip I/O continuity. Accordingly, ED-SRMA introduces state-dependent decoding, cross-layer synchronization, feasibility repair, elite preservation, and budgeted local refinement.

Recent studies have extended this topic to storage reallocation and I/O-station configuration. Salah et al. proposed a two-step reallocation strategy that moves frequently requested items closer to the I/O station [27], and Alnahhal et al. examined the combined influence of storage reallocation and I/O-station location on throughput [19]. Rizqi et al. jointly considered storage assignment and I/O-point configuration from an energy-efficiency perspective [28]. However, these studies mainly assume offline reallocation, fixed command cycles, or predetermined I/O configurations and do not continuously update the available-location set throughout successive crane trips.

Multiple I/O points and multi-shuttle cranes provide additional routing flexibility but substantially enlarge the scheduling space. Song et al. developed a sorting method for large-scale requests in multi-I/O-depot AS/RSs [20], while Polten and Emde investigated task grouping and sequencing for multi-shuttle cranes [10]. Geng et al. considered routing and collision avoidance in a double-end system with two stacker cranes [18]. Yan et al. analyzed travel time and stacker-path scheduling in double-ended compact storage systems [29,30],

and Hu et al. proposed a modified brain storm optimization algorithm for the same class of systems [31]. Xu et al. studied parallel-crane scheduling under shared storage [21].

Exact and network-based approaches have also been developed. Buckow et al. jointly investigated retrieval sequencing and I/O-point assignment and transformed several variants into traveling-salesman problems [32]. Ishikura et al. formulated multi-machine AS/RS scheduling through a time-expanded network [33]. Nevertheless, double-end and parallel-crane systems mainly emphasize machine coordination or collision avoidance, whereas multi-I/O retrieval models often assume retrieval-only operations, fixed storage locations, or single-load handling. They therefore do not directly represent interleaved storage and retrieval operations performed by a multi-shuttle crane under dynamic shared storage and continuous dual-I/O transitions.

2.2. Learning-Based Scheduling Methods

Recent studies have introduced data-driven and learning-based methods into storage-location allocation and crane scheduling. Antomarioni et al. developed a data-driven decision-support system for AS/RS item allocation [34], and Ekren and Arslan applied reinforcement learning to transaction scheduling in shuttle-based storage systems [35]. Rizqi et al. combined ensemble learning with metaheuristic optimization to predict the next request and determine the crane dwell point [36]. Rizqi and Chou subsequently integrated discrete-event simulation, deep reinforcement learning, and NSGA-II for dynamic and energy-aware crane scheduling [37]. Kouloughli et al. applied Q-learning to retrieval-time optimization but also reported reduced effectiveness under dynamic operating conditions [38]. Deep spatiotemporal learning has also been applied to transportation scheduling [39].

Learning-based methods are suitable for uncertain and dynamic environments, but their performance depends on training data, state representation, reward design, and parameter settings. More importantly, they do not inherently guarantee compliance with hard operational constraints, including crane capacity, within-trip load feasibility, dynamic location availability, and continuity between consecutive I/O ports. Consequently, structured combinatorial optimization remains necessary for the deterministic and strongly constrained problem considered in this study.

2.3. Research Gaps and Positioning of This Study

The literature reveals three main gaps. First, storage-location allocation, request sequencing, and I/O-port selection are commonly optimized separately, although these decisions are mutually dependent under shared storage. Second, existing models rarely propagate dynamic empty-location changes and I/O states across all crane trips; therefore, the immediate reuse of a retrieval location as an RS operation is not fully represented. Third, generic metaheuristics and learning methods do not directly preserve task uniqueness, full-load requirements, within-trip prefix feasibility, and cross-trip I/O continuity.

To address these gaps, this study formulates an integrated dual-I/O AS/RS model that simultaneously considers task grouping, storage/retrieval sequencing, dynamic empty-location evolution, RS operations, multi-shuttle capacity, and inter-trip I/O continuity. The proposed ED-SRMA employs a dual-layer representation to describe task assignment and operation sequencing, together with synchronization and heuristic repair operators to restore feasibility after genetic variation. Elite preservation and lightweight local search are incorporated to improve search efficiency, while displacement-sensor information is used to evaluate crane travel time. The proposed method therefore provides a feasible and transparent optimization framework for the integrated scheduling problem that is not directly addressed by existing storage-allocation, multi-I/O, double-end, or learning-based approaches.

3. Problem Description

This study investigates an AS/RS equipped with dual I/O ports and a single stacker crane. The crane operates within a two-dimensional aisle plane, enabling simultaneous movement along both horizontal and vertical axes. The system processes n storage requests and n retrieval requests, with both types of requests potentially assigned to either I/O port; notably, the two ports are not required to process equal numbers of requests.

A crane operation is organized into discrete trips. During trip k , the crane begins at port IO_{a_k} and initially loads m unit loads. It then proceeds to visit a sequence of storage locations denoted by $(c_{k,1}, c_{k,2}, \dots)$ within the rack. Throughout this traversal, storage and retrieval operations may be interleaved: when the crane encounters an empty location, it performs a storage move by depositing one unit load; conversely, when it visits an occupied retrieval location, it executes a retrieval operation, thereby releasing that location. As a result, the set of empty locations evolves dynamically: each retrieval releases a location to the empty set, while each storage move removes one location from it. Upon completing the assigned operations, the crane returns to an ending port IO_{b_k} to unload the retrieved unit loads. Notably, I/O continuity is enforced: the ending port of trip k becomes the starting port of trip $k + 1$.

To make the problem tractable, the following assumptions are adopted.

- Enforced full-load trips: each trip consists of exactly m storage moves and m retrieval operations (i.e., $2m$ tasks per trip), resulting in $K = n/m$ trips; if n is not divisible by m , dummy tasks can be added to balance the workload. Storage and retrieval operations may be interleaved within a trip; however, a prefix feasibility condition is imposed to prevent capacity violations. Specifically, before any operation in a given process, the number of executed retrieval operations cannot exceed the number of executed storage moves.
- Known locations: All retrieval items and their storage locations are known a priori and are initially occupied. All other locations are initially empty, ensuring the availability of a feasible empty location for storage operations.
- Homogeneous handling times: The service times for a storage move (S), a retrieval operation (R), and a possible combined move (R/S) are assumed to be identical and constant. Therefore, they can be incorporated into a constant term and do not affect the comparison of alternative schedules.
- Single-device sequential execution: Only one crane is considered, and trips are executed sequentially; that is, congestion, blocking, and interference effects caused by additional devices are not accounted for.
- Constant speeds with simultaneous motion: The crane moves with constant speeds v_x and v_y along the two axes (in experiments, we set $v_x = v_y$). The travel time between two points is modeled using the Chebyshev time metric, which reflects that the travel duration is dominated by the slower axis-wise component.
- I/O continuity: The ending I/O port of trip k is used as the starting port of trip $k + 1$, ensuring a continuous inter-trip trajectory and consistent time evaluation.

3.1. System Configuration and Sensing Interface

In a practical implementation, displacement sensors can be installed on the horizontal and vertical drives, while position sensors at the two I/O ports can be used to confirm crane arrival and cargo-exchange positions.

Potential for combined retrieval and storage (RS) moves: If a retrieval operation and a storage operation occur sequentially at the same location, the travel time between them is zero.

3.1.1. Task Sets and I/O Assignment

Let $\mathcal{P} = \{1, 2\}$ denote the set of I/O ports. The storage-request set is $S = S^{(1)} \cup S^{(2)}$ and the retrieval-request set is $R = R^{(1)} \cup R^{(2)}$, where

$$\begin{cases} |S^{(1)}| = n_{s1} \\ |S^{(2)}| = n_{s2} \end{cases} \quad n_{s1} + n_{s2} = n \quad (1)$$

$$\begin{cases} |R^{(1)}| = n_{r1} \\ |R^{(2)}| = n_{r2} \end{cases} \quad n_{r1} + n_{r2} = n \quad (2)$$

$$n_{r1}, n_{r2}, n_{s1}, n_{s2} \in \mathbb{Z} \geq 0 \quad (3)$$

3.1.2. Storage Locations and Initial Empties

Let $\mathcal{L}$ be the set of storage locations with cardinality $L = |\mathcal{L}|$. For experimental convenience and to guarantee sufficient initial empty locations, we consider a $q \times q$ grid of locations:

$$q = \lfloor \sqrt{2n} \rfloor + 1, \quad L = q^2 \quad (4)$$

thus $L > 2n$. Let $\mathcal{L}_e(0) \subseteq \mathcal{L}$ be the set of empty locations at time zero. We assume all retrieval locations are known in advance and initially occupied by n items; hence, typically $|\mathcal{L}_e(0)| \geq L - n > n$, which ensures feasibility even in a worst-case sequence.

3.2. Batch Definition and Capacity Constraints

The k -th crane trip is modeled as a batch B_k consisting of a subset of storage tasks and retrieval tasks:

$$B_k = (B_k^S, B_k^R), \quad B_k^S \subseteq S, \quad B_k^R \subseteq R. \quad (5)$$

The capacity constraints are expressed as follows:

$$|B_k^S| \leq m, \quad |B_k^R| \leq m, \quad |B_k^S| + |B_k^R| \leq 2m, \quad \forall k. \quad (6)$$

Under the enforced full-load setting adopted in this study, exactly m storage tasks and m retrieval tasks are completed per trip [40]:

$$|B_k^S| = |B_k^R| = m, \quad p_k = |B_k| = 2m, \quad \forall k. \quad (7)$$

Accordingly, the number of trips is given by $K = n/m \in \mathbb{Z}_{>0}$ (pseudo-tasks can be introduced otherwise). Finally, each task must be assigned to exactly one batch:

$$\bigcup_{k=1}^K B_k = R \cup S, \quad B_k \cap B_{k'} = \emptyset \quad (k \neq k'). \quad (8)$$

3.3. Feasibility: Empty-Location and Load Constraints

3.3.1. Empty-Location Feasibility (Global)

For any prefix of the first k trips, the cumulative number of stored items cannot exceed the initial empty location plus those released by retrievals:

$$\sum_{j=1}^k |B_j^S| \leq |\mathcal{L}_e(0)| + \sum_{j=1}^k |B_j^R|, \quad \forall k. \quad (9)$$

3.3.2. Load Feasibility (Within-Trip Prefix Constraint)

At the beginning of each trip, the crane carries m items. Executing a storage move decreases the onboard load by 1, whereas executing a retrieval operation increases it by 1. To prevent exceeding capacity m , the number of retrieval operations cannot exceed the number of storage moves in any prefix of the trip. Let $\pi_k = (\pi_{k,1}, \dots, \pi_{k,2m})$ be the operation-type sequence with $\pi_{k,j} \in \{S, R\}$; then

$$\sum_{h=1}^j \mathbb{I}[\pi_{k,h} = R] \leq \sum_{h=1}^j \mathbb{I}[\pi_{k,h} = S], \quad \forall k, \forall j = 1, \dots, 2m. \quad (10)$$

In the full-load setting, it is also common to enforce $\pi_{k,2m} = R$ so that the crane returns with m items.

3.4. Dual-I/O Continuity and Time Model

3.4.1. I/O Continuity

Let IO_{a_k} and IO_{b_k} denote the start and end I/O ports of trip k , where $a_k, b_k \in \{1, 2\}$. The “end-port becomes next start-port” rule is captured by

$$a_{k+1} = b_k, \quad k = 1, \dots, K-1, \quad (11)$$

with a given initial port a_1 (e.g., $a_1 = 1$).

3.4.2. Chebyshev Travel Time Measured by Displacement Sensors

This sensor-informed formulation improves the consistency between the analytical travel-time model and the physical motion behavior of the crane, thereby enhancing the practical interpretability of the optimization results. The distance between two points $u = (x_u, y_u)$ and $v = (x_v, y_v)$ on the shelf can be determined using the crane’s displacement sensors. Given the directional speeds of the crane v_x and v_y , the travel time is formulated as follows:

$$t(u, v) = \max\left(\frac{|x_u - x_v|}{v_x}, \frac{|y_u - y_v|}{v_y}\right). \quad (12)$$

High-precision displacement sensors, which are installed on the crane’s motors in the X and Y directions, can also be used to provide feedback and to verify the effectiveness of scheduling in the actual system.

3.4.3. Trip Time and Objective

Let $(c_{k,1}, \dots, c_{k,p_k})$ be the visited location sequence of trip k (each $c_{k,j} \in \mathcal{L}$). The trip duration is

$$T(B_k) = t(IO_{a_k}, c_{k,1}) + \sum_{j=1}^{p_k-1} t(c_{k,j}, c_{k,j+1}) + t(c_{k,p_k}, IO_{b_k}), \quad (13)$$

Since a single crane executes trips sequentially, minimizing the total completion time is equivalent to minimizing the total travel time:

$$\min T = \sum_{k=1}^K T(B_k). \quad (14)$$

3.5. Illustrative Example

A small instance is introduced to illustrate the interaction among shared storage, load feasibility, RS operations, and dual-I/O continuity. Consider a 4×4 rack with dual I/O ports located at $IO_1 = (1, 0)$ and $IO_2 = (4, 0)$. The system contains $n = 6$ storage requests

and $n = 6$ retrieval requests. The crane capacity is $m = 3$; therefore, the requests are completed in $K = n/m = 2$ full-load trips.

The initial retrieval locations are

$$\begin{aligned} R_1 &= (2, 1), & R_2 &= (3, 2), & R_3 &= (4, 1), \\ R_4 &= (1, 2), & R_5 &= (2, 4), & R_6 &= (4, 4), \end{aligned} \quad (15)$$

and all remaining rack locations constitute the initial empty-location set $L_e(0)$. One feasible schedule is

$$\begin{aligned} B_1: & \text{IO}_1 \rightarrow S_1(1, 1) \rightarrow R_1(2, 1) \rightarrow S_2(2, 1) \\ & \rightarrow R_2(3, 2) \rightarrow S_3(3, 2) \rightarrow R_3(4, 1) \rightarrow \text{IO}_2, \\ B_2: & \text{IO}_2 \rightarrow S_4(4, 1) \rightarrow R_6(4, 4) \rightarrow S_5(4, 4) \\ & \rightarrow R_5(2, 4) \rightarrow S_6(2, 4) \rightarrow R_4(1, 2) \rightarrow \text{IO}_1. \end{aligned} \quad (16)$$

For both trips, the operation-type sequence is (S, R, S, R, S, R) . The cumulative numbers of storage and retrieval operations after each step are

$$(1, 0), (1, 1), (2, 1), (2, 2), (3, 2), (3, 3), \quad (17)$$

respectively. Hence, the number of completed retrieval operations never exceeds the number of completed storage operations in any prefix. Starting with three unit loads, the onboard load evolves as

$$3 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 3, \quad (18)$$

which remains within the capacity interval $[0, m]$.

In the first trip, the location released by R_1 is immediately reused by S_2 , and the location released by R_2 is reused by S_3 . These two pairs are therefore RS operations with zero intermediate travel time. After the first trip, the empty-location set becomes

$$L_e(1) = (L_e(0) \setminus \{(1, 1)\}) \cup \{(4, 1)\}. \quad (19)$$

The newly released location $(4, 1)$ is then assigned to S_4 at the beginning of the second trip. This illustrates that storage-location availability must be propagated between consecutive trips rather than evaluated independently.

Assuming $v_x = v_y = 1$, the travel times of the two trips obtained from Equation (12) are

$$T(B_1) = 1 + 1 + 0 + 1 + 0 + 1 + 1 = 5, \quad (20)$$

$$T(B_2) = 1 + 3 + 0 + 2 + 0 + 2 + 2 = 10. \quad (21)$$

Accordingly, the total travel time of this feasible schedule is

$$T = T(B_1) + T(B_2) = 15. \quad (22)$$

The ending port of B_1 is IO_2 , which is also the starting port of B_2 , thereby satisfying the inter-trip I/O-continuity constraint. This example demonstrates that task sequencing, storage-location allocation, dynamic empty-location evolution, and I/O-port selection must be evaluated jointly.

4. Design and Implementation of ED-SRMA for Integrated Scheduling in Dual-I/O Middle-Aisle Warehouses

This section presents ED-SRMA for the integrated optimization of storage-location allocation and crane-trip scheduling in a dual-I/O middle-aisle AS/RS. The problem

involves two coupled state transitions: inter-trip I/O continuity and the dynamic evolution of empty locations under shared-storage operations.

4.1. Overall Framework and Design Rationale of ED-SRMA

Figure 3 presents the overall framework of ED-SRMA. The algorithm combines population-based search with problem-specific feasibility repair, cross-layer synchronization, elite preservation, adaptive mutation, and budgeted local refinement. The complete procedure is summarized in Algorithm 1, while the principal operators are described in the following subsections.

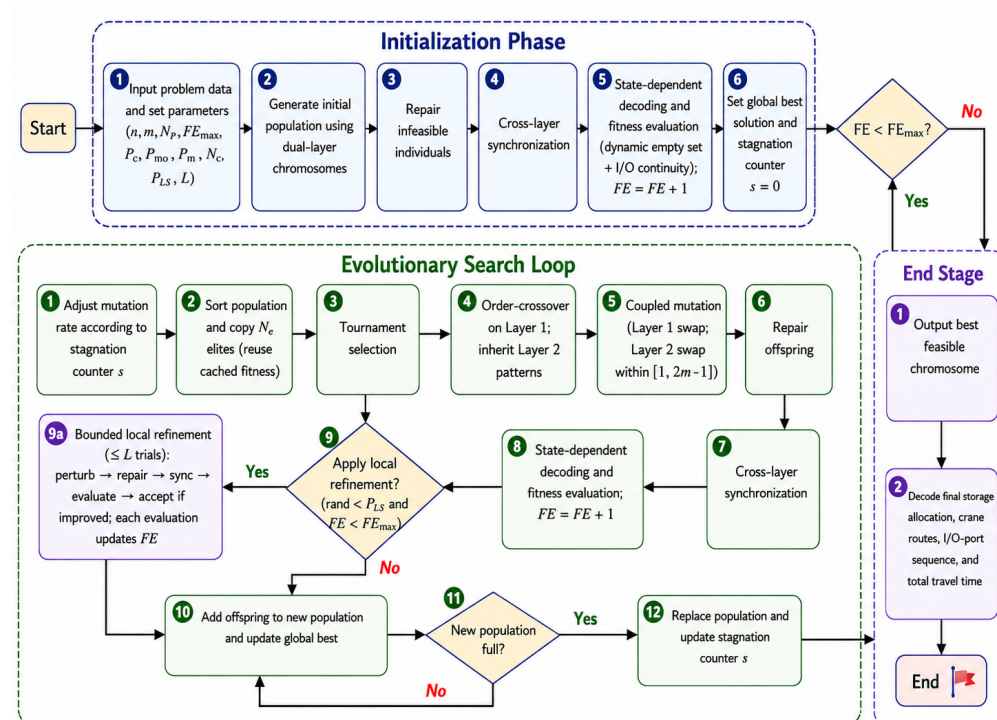


Figure 3. Overall framework of the proposed Elite-Driven Synchronized-Repair Memetic Algorithm.

Algorithm 1 summarizes the complete ED-SRMA procedure. The algorithm initializes and evaluates a repaired dual-layer population, iteratively generates and repairs offspring, applies budgeted local refinement to selected feasible solutions, and updates the elite-driven population until the fitness-evaluation budget is exhausted. The representation, decoding, repair, evolutionary operators, and local-refinement procedure are detailed in the following subsections.

4.2. Dual-Layer Solution Representation and Population Initialization

The integrated optimization problem involves two coupled decisions: assigning retrieval tasks to crane trips and determining the execution order of storage and retrieval operations within each trip. To represent these decisions separately while preserving their relationship, each ED-SRMA individual is defined as a dual-layer chromosome

$$X = \{X^{(1)}, X^{(2)}\}, \quad (23)$$

where $X^{(1)}$ represents the retrieval-task assignment and $X^{(2)}$ represents the within-trip operation sequence.

Algorithm 1: Overall procedure of the proposed ED-SRMA

Input: Problem data $\mathcal{D}$; number of retrieval tasks n ; crane capacity m ; population size N_p ; number of retained elites N_e ; tournament size N_t ; crossover probability p_c ; base mutation probability p_m^0 ; intensified mutation probability p_m^+ ; stagnation threshold τ ; local-refinement probability p_{LS} ; maximum number of local trials L ; maximum fitness-evaluation budget $FE_{\max}$.

Output: Best feasible chromosome X^* and its objective value f^* .

```

1   $K \leftarrow n/m$ 
2   $FE \leftarrow 0$ 
3   $s \leftarrow 0$  // stagnation counter
4  Generate an initial population  $\mathcal{P} = \{X_1, \dots, X_{N_p}\}$  using the dual-layer representation
5  foreach  $X_i \in \mathcal{P}$  do
6     $X_i \leftarrow \text{Repair}(X_i)$ 
7     $X_i \leftarrow \text{Sync}(X_i)$ 
8     $f_i \leftarrow \text{DecodeAndEvaluate}(X_i, \mathcal{D})$ 
9     $FE \leftarrow FE + 1$ 
10 end
11  $(X^*, f^*) \leftarrow \arg \min_{(X_i, f_i) \in \mathcal{P}} f_i$ 
12 while  $FE < FE_{\max}$  do
13    $f^{\text{old}} \leftarrow f^*$ 
14   if  $s > \tau$  then
15      $p_m \leftarrow p_m^+$ 
16   else
17      $p_m \leftarrow p_m^0$ 
18   end
19   Sort  $\mathcal{P}$  in ascending order of fitness
20   Copy the best  $N_e$  chromosome–fitness pairs to  $\mathcal{P}^{\text{new}}$ 
21   // Cached elite fitness values are reused without additional FEs
22   while  $|\mathcal{P}^{\text{new}}| < N_p$  and  $FE < FE_{\max}$  do
23     Select parents  $X_{p_1}$  and  $X_{p_2}$  from  $\mathcal{P}$  by tournaments of size  $N_t$ 
24      $(X_{c_1}, X_{c_2}) \leftarrow \text{Crossover}(X_{p_1}, X_{p_2}, p_c)$ 
25     foreach  $X_c \in \{X_{c_1}, X_{c_2}\}$  do
26       if  $|\mathcal{P}^{\text{new}}| = N_p$  or  $FE = FE_{\max}$  then
27         break
28       end
29        $X_c \leftarrow \text{Mutate}(X_c, p_m)$ 
30        $X_c \leftarrow \text{Repair}(X_c)$ 
31        $X_c \leftarrow \text{Sync}(X_c)$ 
32        $f_c \leftarrow \text{DecodeAndEvaluate}(X_c, \mathcal{D})$ 
33        $FE \leftarrow FE + 1$ 
34       if  $f_c < f^*$  then
35          $(X^*, f^*) \leftarrow (X_c, f_c)$ 
36       end
37       if  $\text{rand}(0, 1) < p_{LS}$  and  $FE < FE_{\max}$  then
38          $(X_c, f_c, FE) \leftarrow \text{LocalRefine}(X_c, f_c, L, FE, FE_{\max}, \mathcal{D})$ 
39         if  $f_c < f^*$  then
40            $(X^*, f^*) \leftarrow (X_c, f_c)$ 
41         end
42       end
43       Add  $(X_c, f_c)$  to  $\mathcal{P}^{\text{new}}$ 
44     end
45   if  $|\mathcal{P}^{\text{new}}| = N_p$  then
46      $\mathcal{P} \leftarrow \mathcal{P}^{\text{new}}$ 
47     if  $f^* < f^{\text{old}}$  then
48        $s \leftarrow 0$ 
49     else
50        $s \leftarrow s + 1$ 
51     end
52   end
53 else
54   break // The FE budget is exhausted
55 end
56 end
57 return  $X^*, f^*$ 

```

4.2.1. Retrieval-Task Assignment Layer

Let n denote the number of retrieval tasks, m the crane capacity, and

$$K = \frac{n}{m} \quad (24)$$

the number of trips under the full-load assumption. The first encoding layer is a $K \times m$ matrix,

$$\mathbf{X}^{(1)} = \left[x_{k,\ell}^{(1)} \right]_{K \times m}, \quad x_{k,\ell}^{(1)} \in \{1, \dots, n\}. \quad (25)$$

The k th row contains the m retrieval tasks assigned to trip k . All entries in $\mathbf{X}^{(1)}$ are distinct and satisfy

$$\left\{ x_{k,\ell}^{(1)} \mid k = 1, \dots, K, \ell = 1, \dots, m \right\} = \{1, \dots, n\}. \quad (26)$$

Therefore, each retrieval task is assigned to exactly one trip, with neither duplication nor omission.

The order of the retrieval identifiers within each row also determines their left-to-right placement in the retrieval positions of the second layer after cross-layer synchronization.

4.2.2. Operation-Sequence Layer

The second encoding layer describes the sequence of storage and retrieval actions within each trip. Let

$$H = \max(K + 1, m + 1) \quad (27)$$

denote the number of symbolic storage genes provided in each row. The second layer is expressed as

$$\mathbf{X}^{(2)} = \left[x_{k,j}^{(2)} \right]_{K \times (m+H)}. \quad (28)$$

For trip k , its initial gene set is

$$\mathcal{G}_k = \left\{ R_{x_{k,1}^{(1)}}, \dots, R_{x_{k,m}^{(1)}}, E_1, \dots, E_H \right\}, \quad (29)$$

where R_i represents retrieval task i , and E_j is a symbolic storage-action gene.

A retrieval gene is associated with a known retrieval location. In contrast, a storage gene does not identify a fixed rack location. Its physical target is selected during state-dependent decoding from the current empty-location set. Thus, the symbolic index j distinguishes storage genes in the chromosome but does not correspond to a physical location.

Only the first $2m$ positions of each row constitute the effective execution window:

$$\mathbf{W}_k = \left(x_{k,1}^{(2)}, x_{k,2}^{(2)}, \dots, x_{k,2m}^{(2)} \right). \quad (30)$$

After feasibility repair, this window contains exactly m retrieval genes and m storage genes. The remaining $H - m$ positions are reserve positions and are not decoded during fitness evaluation.

4.2.3. Population Initialization

For each individual, the first encoding layer is initialized by randomly permuting the n retrieval-task identifiers and reshaping the resulting permutation into a $K \times m$ matrix. For every trip, the corresponding retrieval genes are then combined with the H symbolic storage genes, and the resulting $m + H$ genes are randomly permuted to form one row of $\mathbf{X}^{(2)}$.

The initialization of row k can be expressed as

$$\mathbf{X}_{k,:}^{(2)} = \pi_k \left(R_{x_{k,1}^{(1)}}, \dots, R_{x_{k,m}^{(1)}}, E_1, \dots, E_H \right), \quad (31)$$

where $\pi_k(\cdot)$ denotes an independently generated random permutation.

Because random permutations do not necessarily satisfy the operational constraints, each initialized individual is subsequently processed by the feasibility-repair and cross-layer synchronization procedures:

$$X \leftarrow \text{Sync}(\text{Repair}(X)). \quad (32)$$

The repair operator establishes a feasible storage–retrieval pattern, while the synchronization operator ensures that the retrieval identifiers in the second layer are consistent with the corresponding row of the first layer. Their detailed implementation is presented in Section 4.4.

4.2.4. Illustrative Encoding Example

Consider an instance with $n = 6$ retrieval tasks and a crane capacity of $m = 2$. The number of trips is $K = 3$, and the number of symbolic storage genes is $H = \max(4, 3) = 4$. A possible first-layer chromosome is

$$X^{(1)} = \begin{bmatrix} 4 & 1 \\ 6 & 2 \\ 5 & 3 \end{bmatrix}. \quad (33)$$

Accordingly, retrieval tasks $\{4, 1\}$, $\{6, 2\}$, and $\{5, 3\}$ are assigned to trips 1, 2, and 3, respectively.

After row-wise randomization, feasibility repair, and synchronization, a corresponding second-layer chromosome may be written as

$$X^{(2)} = \begin{bmatrix} E_4 & E_3 & R_4 & R_1 & E_2 & E_1 \\ E_2 & R_6 & E_4 & R_2 & E_1 & E_3 \\ E_3 & E_1 & R_5 & R_3 & E_4 & E_2 \end{bmatrix}. \quad (34)$$

Since $2m = 4$, only the first four positions of each row are executed. For example, the effective sequence of trip 2 is

$$W_2 = (E_2, R_6, E_4, R_2). \quad (35)$$

The last two storage genes in the same row are reserve genes and do not participate in the current trip.

This representation separates retrieval-task assignment from the storage–retrieval operations pattern while retaining their correspondence through synchronization. The conversion of an effective sequence into physical storage locations, retrieval locations, crane movements, and I/O-port decisions is described in the following subsection.

4.3. State-Dependent Decoding and Fitness Evaluation

After feasibility repair and cross-layer synchronization, each chromosome is decoded into a continuous crane schedule. The decoding process is state dependent because the starting I/O port and the available empty locations of a trip depend on the operations completed previously. Accordingly, trips cannot be evaluated independently.

4.3.1. I/O-Linked Trip Continuity

Let $\mathcal{P} = \{IO_1, IO_2\}$ denote the set of I/O ports. The starting and ending ports of trip k are denoted by p_k^{start} and p_k^{end} , respectively. Inter-trip continuity is enforced by

$$p_1^{\text{start}} = p^0, \quad p_k^{\text{start}} = p_{k-1}^{\text{end}}, \quad k = 2, \dots, K, \quad (36)$$

where p^0 is the prescribed initial crane position. In the experiments, $p^0 = IO_1$.

After the $2m$ actions of trip k have been decoded, its ending port is selected according to

$$p_k^{\text{end}} = \arg \min_{p \in \mathcal{P}} t(x_k^{\text{last}}, p), \quad (37)$$

where x_k^{last} is the final visited location and $t(\cdot, \cdot)$ is the Chebyshev travel-time function defined in Section 3.4. When both ports produce the same return time, IO_1 is selected to ensure deterministic evaluation.

4.3.2. Dynamic Empty-Location Set

Let $\mathcal{E}^0$ be the initial set of empty locations. At the beginning of chromosome evaluation,

$$\mathcal{E} \leftarrow \mathcal{E}^0. \quad (38)$$

The set is updated continuously over all trips and is not reinitialized between consecutive trips.

For an ordinary retrieval operation R_i , the crane visits the known retrieval location r_i , after which this location becomes available:

$$\mathcal{E} \leftarrow \mathcal{E} \cup \{r_i\}. \quad (39)$$

For an ordinary storage action, its physical target is selected from the current empty-location set using

$$e^* = \arg \min_{e \in \mathcal{E}} t(x, e), \quad (40)$$

where x is the current crane position. The selected location is then removed:

$$\mathcal{E} \leftarrow \mathcal{E} \setminus \{e^*\}. \quad (41)$$

If several locations have the same minimum travel time, the first location in the fixed empty-set ordering is selected.

An adjacent retrieval–storage pair,

$$(R_i, E_q), \quad (42)$$

is decoded as an RS operation. The storage action immediately reuses the location released by R_i ; therefore, both genes are mapped to r_i and

$$\mathcal{E}_{\text{after}} = \mathcal{E}_{\text{before}}. \quad (43)$$

The movement between the retrieval and storage actions is zero.

4.3.3. Decoding and Fitness Calculation

For trip k , only the effective window

$$W_k = X_{k,1:2m}^{(2)} \quad (44)$$

is decoded. Let $c_{k,1}, \dots, c_{k,2m}$ denote the physical locations obtained from the genes in this window. The travel time of trip k is

$$T_k = t(p_k^{\text{start}}, c_{k,1}) + \sum_{j=1}^{2m-1} t(c_{k,j}, c_{k,j+1}) + t(c_{k,2m}, p_k^{\text{end}}). \quad (45)$$

The chromosome fitness is the total travel time

$$F(\mathbf{X}) = \sum_{k=1}^K T_k. \quad (46)$$

The complete procedure is summarized in Algorithm 2.

Algorithm 2: State-dependent decoding and fitness evaluation

Input: A repaired and synchronized chromosome $\mathbf{X}$; number of trips K ; capacity m ; retrieval locations $\{r_i\}$; initial empty-location set $\mathcal{E}^0$; I/O-port set $\mathcal{P}$; initial port p^0 .

Output: Total travel time $F(\mathbf{X})$ and decoded crane routes.

```

1  $F(\mathbf{X}) \leftarrow 0$ 
2  $\mathcal{E} \leftarrow \mathcal{E}^0$ 
3  $p^{\text{cur}} \leftarrow p^0$ 
4 for  $k \leftarrow 1$  to  $K$  do
5    $W_k \leftarrow X_{k,1:2m}^{(2)}$ 
6    $x \leftarrow p^{\text{cur}}$ 
7    $T_k \leftarrow 0$ 
8    $j \leftarrow 1$ 
9   while  $j \leq 2m$  do
10     $g \leftarrow W_{k,j}$ 
11    if  $g = R_i$  then
12       $y \leftarrow r_i$ 
13       $T_k \leftarrow T_k + t(x, y)$ 
14      if  $j < 2m$  and  $W_{k,j+1}$  is a storage gene then
15        Map both positions  $j$  and  $j + 1$  to  $y$ 
16        // RS operation
17        Leave  $\mathcal{E}$  unchanged
18         $x \leftarrow y$ 
19         $j \leftarrow j + 2$ 
20      else
21        Map position  $j$  to  $y$ 
22         $\mathcal{E} \leftarrow \mathcal{E} \cup \{y\}$ 
23         $x \leftarrow y$ 
24         $j \leftarrow j + 1$ 
25      end
26    else
27       $y \leftarrow \arg \min_{e \in \mathcal{E}} t(x, e)$ 
28      Map the storage gene to  $y$ 
29       $T_k \leftarrow T_k + t(x, y)$ 
30       $\mathcal{E} \leftarrow \mathcal{E} \setminus \{y\}$ 
31       $x \leftarrow y$ 
32       $j \leftarrow j + 1$ 
33    end
34   $p_k^{\text{end}} \leftarrow \arg \min_{p \in \mathcal{P}} t(x, p)$ 
35   $T_k \leftarrow T_k + t(x, p_k^{\text{end}})$ 
36   $F(\mathbf{X}) \leftarrow F(\mathbf{X}) + T_k$ 
37   $p^{\text{cur}} \leftarrow p_k^{\text{end}}$ 
38 end
39 return  $F(\mathbf{X})$  and the decoded routes

```

4.3.4. Illustrative Decoding Example

Consider the effective sequence introduced in Section 4.2:

$$W_2 = (E_2, R_6, E_4, R_2). \quad (47)$$

Let the starting port be $IO_1 = (1, 0)$, the second port be $IO_2 = (5, 0)$, and

$$r_6 = (3, 2), \quad r_2 = (4, 1). \quad (48)$$

Assume that the current empty-location set is

$$\mathcal{E} = \{(2, 1), (1, 3), (4, 4)\}. \quad (49)$$

The sequence is decoded as follows:

1. E_2 selects $(2, 1)$, the nearest empty location to IO_1 . The location is then removed from $\mathcal{E}$.
2. R_6 is immediately followed by E_4 . The pair is therefore decoded as an RS operation at $(3, 2)$, and the empty-location set remains unchanged.
3. R_2 is decoded at $(4, 1)$. Because it is not followed by a storage gene, $(4, 1)$ is added to $\mathcal{E}$.
4. From $(4, 1)$, IO_2 is selected as the ending port. It subsequently becomes the starting port of trip 3.

The corresponding physical visit sequence is

$$[(2, 1), (3, 2), (3, 2), (4, 1)], \quad (50)$$

where the repeated coordinate represents the zero-distance movement within the RS operation. The resulting trip time is

$$\begin{aligned} T_2 &= t((1, 0), (2, 1)) + t((2, 1), (3, 2)) \\ &\quad + t((3, 2), (3, 2)) + t((3, 2), (4, 1)) + t((4, 1), (5, 0)) \\ &= 1 + 1 + 0 + 1 + 1 = 4. \end{aligned} \quad (51)$$

This example shows how the symbolic storage genes are converted into physical storage locations according to the current system state. The feasibility of the operation sequence itself, including the execution-window structure and prefix load constraint, is ensured by the repair and synchronization procedures described in Section 4.4.

4.4. Feasibility Repair and Cross-Layer Synchronization

Genetic variation may disrupt the operation structure of the second encoding layer or make the retrieval identifiers inconsistent with the first layer. Instead of introducing penalty coefficients, ED-SRMA restores each candidate solution through two consecutive procedures:

$$X \leftarrow \text{Sync}(\text{Repair}(X)). \quad (52)$$

The repair operator first establishes a feasible storage–retrieval pattern in each effective execution window. Cross-layer synchronization then assigns the retrieval identifiers represented by the first layer to the repaired retrieval positions in the second layer.

4.4.1. Feasibility Conditions

For trip k , let

$$W_k = (x_{k,1}^{(2)}, \dots, x_{k,2m}^{(2)}) \quad (53)$$

denote the effective execution window. A feasible window must satisfy the following conditions.

First, the window contains exactly m storage genes and m retrieval genes:

$$\begin{aligned}\sum_{j=1}^{2m} \mathbb{I}(x_{k,j}^{(2)} \text{ is E}) &= m, \\ \sum_{j=1}^{2m} \mathbb{I}(x_{k,j}^{(2)} \text{ is R}) &= m.\end{aligned}\quad (54)$$

Second, the first action is a storage action and the terminal action is a retrieval operation:

$$x_{k,1}^{(2)} \in \text{E}, \quad x_{k,2m}^{(2)} \in \text{R}. \quad (55)$$

Third, every prefix of the window satisfies the crane-capacity constraint:

$$\sum_{t=1}^j \mathbb{I}(x_{k,t}^{(2)} \text{ is R}) \leq \sum_{t=1}^j \mathbb{I}(x_{k,t}^{(2)} \text{ is E}), \quad j = 1, \dots, 2m. \quad (56)$$

Since the crane begins each trip with m inbound unit loads, this condition prevents its onboard load from exceeding capacity after a retrieval operation. Retrieval-task completeness and uniqueness are separately guaranteed by the permutation structure of the first encoding layer.

4.4.2. Four-Stage Repair Procedure

The repair operator processes each row of the second layer through four stages.

Stage 1: initial-action correction. If the first position is occupied by a retrieval gene, it is exchanged with the rightmost storage gene in the row. This ensures that the trip begins with a storage action.

Stage 2: execution-window alignment. Any retrieval gene located outside the effective window $[1, 2m]$ is exchanged with a storage gene inside the window. The operation is repeated until all m retrieval genes are contained in the effective window. Consequently, the window contains exactly m storage and m retrieval operations.

Stage 3: terminal-action correction. If position $2m$ contains a storage gene, it is exchanged with the rightmost retrieval gene preceding that position.

Stage 4: prefix-load correction. The repaired window is scanned from left to right. When a retrieval operation at position j would violate Equation (56), the rightmost available storage gene in positions $j, \dots, 2m - 1$ is moved immediately before the violating retrieval operation. The scan is then restarted until all prefixes are feasible. Position $2m$ is excluded from this search so that the terminal retrieval condition is preserved.

This procedure modifies only the operation arrangement. The definitive retrieval-task identifiers are assigned afterward by cross-layer synchronization.

4.4.3. Cross-Layer Synchronization

Let the retrieval positions in the repaired window of trip k be

$$Q_k = (q_{k,1}, q_{k,2}, \dots, q_{k,m}), \quad q_{k,1} < \dots < q_{k,m}. \quad (57)$$

The synchronization operator replaces the retrieval identifiers at these positions according to row k of the first encoding layer:

$$x_{k,q_{k,\ell}}^{(2)} \leftarrow R_{x_{k,\ell}^{(1)}}, \quad \ell = 1, \dots, m. \quad (58)$$

The storage–retrieval pattern remains unchanged; only the retrieval identifiers are updated. Therefore, synchronization preserves the feasibility established by the repair operator while ensuring that both encoding layers represent the same trip-level task assignment.

The complete repair and synchronization procedure is summarized in Algorithm 3.

Algorithm 3: Feasibility repair and cross-layer synchronization

Input: Candidate chromosome $X = \{X^{(1)}, X^{(2)}\}$; number of trips K ; crane capacity m .
Output: Feasible and synchronized chromosome X .

```

1  for  $k \leftarrow 1$  to  $K$  do
    // Stage 1: initial-action correction
2   if  $x_{k,1}^{(2)}$  is a retrieval gene then
3     | Locate the rightmost storage gene in row  $k$ 
4     | Exchange it with  $x_{k,1}^{(2)}$ 
5   end
    // Stage 2: execution-window alignment
6   while a retrieval gene exists outside  $[1, 2m]$  do
7     | Locate the rightmost retrieval gene outside  $[1, 2m]$ 
8     | Locate the rightmost storage gene inside  $[1, 2m]$ 
9     | Exchange the two genes
10  end
    // Stage 3: terminal-action correction
11  if  $x_{k,2m}^{(2)}$  is a storage gene then
12    | Locate the rightmost retrieval gene in  $[1, 2m]$ 
13    | Exchange it with  $x_{k,2m}^{(2)}$ 
14  end
    // Stage 4: prefix-load correction
15   $j \leftarrow 1$ 
16  while  $j \leq 2m$  do
17    | Count the storage and retrieval genes in positions  $1, \dots, j-1$ 
18    | if  $x_{k,j}^{(2)}$  is a storage gene then
19      | |  $j \leftarrow j+1$ 
20    | else if placing the retrieval gene at  $j$  satisfies the prefix constraint then
21      | |  $j \leftarrow j+1$ 
22    | end
23    | else
24      | | Locate the rightmost storage gene in  $[j, 2m-1]$ 
25      | | Move it immediately before the retrieval gene at position  $j$ 
26      | |  $j \leftarrow 1$  // restart the feasibility scan
27    | end
28  end
29 end
30 for  $k \leftarrow 1$  to  $K$  do
31   | Identify the ordered retrieval positions  $q_{k,1} < \dots < q_{k,m}$  in  $X_{k,1:2m}^{(2)}$ 
32   for  $\ell \leftarrow 1$  to  $m$  do
33     |  $x_{k,q_{k,\ell}}^{(2)} \leftarrow R_{x_{k,\ell}^{(1)}}$ 
34   end
35 end
36 return  $X$ 

```

4.4.4. Illustrative Repair and Synchronization Example

Consider the first-layer assignment introduced in Section 4.2:

$$\mathbf{X}_{1,:}^{(1)} = [4, 1], \quad (59)$$

with $m = 2$. Suppose that genetic variation produces the following second-layer row:

$$\tilde{\mathbf{X}}_{1,:}^{(2)} = [R_4, R_1, E_1, E_2, E_3, E_4]. \quad (60)$$

The effective window consists of the first four positions. The row is infeasible because it begins with a retrieval operation and one retrieval gene lies outside the execution window.

Stage 1 exchanges the first gene with the rightmost storage gene:

$$[E_4, R_1, E_1, E_2, E_3, R_4]. \quad (61)$$

Stage 2 moves the retrieval gene at position 6 into the execution window by exchanging it with the rightmost storage gene in that window:

$$[E_4, R_1, E_1, R_4, E_3, E_2]. \quad (62)$$

The terminal position of the effective window is already a retrieval gene, and all prefixes satisfy Equation (56); therefore, no further structural modification is required.

The retrieval positions are 2 and 4. Synchronization then writes the first-layer assignment $[4, 1]$ into these positions:

$$\begin{aligned} &[E_4, R_1, E_1, R_4, E_3, E_2] \\ &\xrightarrow{\text{Sync}} [E_4, R_4, E_1, R_1, E_3, E_2]. \end{aligned} \quad (63)$$

The resulting effective sequence is feasible and consistent with the first-layer task assignment. It can therefore be evaluated using the state-dependent decoding procedure described in Section 4.3.

The same repair–synchronization sequence is applied after crossover, mutation, and every neighborhood perturbation. The evolutionary operators that generate these candidate solutions are described in the following subsection.

4.5. Evolutionary Operators and Elite-Driven Population Update

After initialization and fitness evaluation, ED-SRMA evolves the population through tournament selection, order crossover, coupled mutation, and elite-driven replacement. The evolutionary operators generate candidate solutions, which are subsequently processed by the repair and synchronization procedure described in Section 4.4.

4.5.1. Tournament Selection

To select one parent, N_t individuals are sampled uniformly without replacement from the current population. The individual with the smallest objective value is selected:

$$\mathbf{X}^p = \arg \min_{\mathbf{X} \in \mathcal{T}} F(\mathbf{X}), \quad |\mathcal{T}| = N_t, \quad (64)$$

where $\mathcal{T}$ denotes the tournament set. Two tournaments are conducted independently to obtain a pair of parents. The tournament size used in the experiments is reported in Section 5.1.

4.5.2. Order Crossover

Crossover is applied with probability p_c . Because the first encoding layer is a permutation of all retrieval-task identifiers, it is flattened into a vector before applying order crossover (OX). Let

$$\mathbf{v}^{p_1} = \text{vec}\left(\mathbf{X}^{(1),p_1}\right), \quad \mathbf{v}^{p_2} = \text{vec}\left(\mathbf{X}^{(1),p_2}\right) \quad (65)$$

denote the two parent permutations. Two cut positions $a < b$ are selected randomly. For the first offspring, the segment $\mathbf{v}_{a:b}^{p_1}$ is retained, and the remaining positions are filled in the order in which unused task identifiers appear in $\mathbf{v}^{p_2}$. The second offspring is generated symmetrically:

$$(\mathbf{v}^{c_1}, \mathbf{v}^{c_2}) = \text{OX}(\mathbf{v}^{p_1}, \mathbf{v}^{p_2}; a, b). \quad (66)$$

The offspring vectors are then reshaped into $K \times m$ matrices. This operator preserves the uniqueness and completeness of the retrieval-task identifiers.

The second-layer operation pattern of each offspring is initially inherited from its corresponding parent. After crossover and mutation, cross-layer synchronization updates the retrieval identifiers in the second layer according to the modified first-layer assignment.

4.5.3. Coupled Mutation

Mutation is applied independently to the two encoding layers with probability p_m .

For the first layer, two positions are selected from the flattened retrieval-task permutation and exchanged:

$$(\mathbf{v}_u^c, \mathbf{v}_v^c) \leftarrow (\mathbf{v}_v^c, \mathbf{v}_u^c), \quad u \neq v. \quad (67)$$

This operation changes both trip assignment and retrieval order while preserving the permutation property.

For the second layer, one trip is selected randomly, followed by two positions from

$$\{1, \dots, 2m - 1\}. \quad (68)$$

The two genes are exchanged. Position $2m$ is excluded so that the prescribed terminal retrieval position is not directly disturbed:

$$(\mathbf{x}_{k,u}^{(2)}, \mathbf{x}_{k,v}^{(2)}) \leftarrow (\mathbf{x}_{k,v}^{(2)}, \mathbf{x}_{k,u}^{(2)}), \quad u, v \in \{1, \dots, 2m - 1\}. \quad (69)$$

Although this restriction preserves the terminal position, other feasibility conditions may still be violated. Therefore, every mutated offspring is repaired and synchronized before fitness evaluation.

4.5.4. Adaptive Mutation

A stagnation counter s records the number of consecutive completed generations without improvement in the global best objective value. The mutation probability is adjusted according to

$$p_m = \begin{cases} p_m^0, & s \leq \tau, \\ p_m^+, & s > \tau, \end{cases} \quad p_m^+ > p_m^0, \quad (70)$$

where p_m^0 is the base mutation probability and τ is the stagnation threshold. When an improved global best solution is obtained, the counter is reset to zero. Otherwise, it is increased by one after the generation is completed. This mechanism introduces stronger perturbations only when the search remains stagnant.

4.5.5. Elite-Driven Population Update

At the beginning of each generation, the current population is sorted in ascending order of fitness. The best N_e individuals are copied directly to the next population:

$$\mathcal{P}^{\text{new}} \leftarrow \{X_{(1)}, \dots, X_{(N_e)}\}, \quad (71)$$

where

$$F(X_{(1)}) \leq \dots \leq F(X_{(N_p)}). \quad (72)$$

Their previously calculated fitness values are retained and are not evaluated again. This avoids unnecessary consumption of the fitness-evaluation budget.

The remaining positions in $\mathcal{P}^{\text{new}}$ are filled with repaired and synchronized offspring generated by the preceding operators. Each offspring is evaluated once and may subsequently undergo the budgeted memetic refinement described in Section 4.6. When the new population is complete, it replaces the current population:

$$\mathcal{P} \leftarrow \mathcal{P}^{\text{new}}. \quad (73)$$

The global best solution is updated whenever an offspring or a locally refined solution obtains a smaller objective value. The evolutionary process continues until the common fitness-evaluation budget is exhausted. The common selection, crossover, and adaptive-mutation parameters, together with the controlled ablation settings for elite preservation and local refinement, are reported in Section 5.1.

4.6. Budgeted Memetic Local Refinement

After an offspring has been repaired, synchronized, and evaluated, ED-SRMA may apply a lightweight local-refinement procedure to further improve its objective value. The procedure uses small neighborhood moves, a strict improvement rule, and a limited fitness-evaluation budget. It therefore complements the population-based search without introducing an unrestricted computational overhead.

4.6.1. Triggering Rule and Local Budget

For each evaluated offspring X , local refinement is activated with probability

$$\Pr(\text{LocalRefine}(X)) = p_{\text{LS}}. \quad (74)$$

Once activated, at most L neighboring solutions are evaluated. The procedure also terminates immediately when the remaining global fitness-evaluation budget is exhausted:

$$N_{\text{trial}} \leq \min\{L, \text{FE}_{\text{max}} - \text{FE}\}. \quad (75)$$

The values of p_{LS} and L used in the experiments are reported in Section 5.1.

4.6.2. Neighborhood Structures

Two neighborhoods are defined according to the dual-layer representation.

For the first layer, the task-assignment neighborhood $\mathcal{N}_1(X)$ is generated by flattening $X^{(1)}$, selecting two distinct positions u and v , and exchanging their retrieval-task identifiers:

$$(x_u^{(1)}, x_v^{(1)}) \leftarrow (x_v^{(1)}, x_u^{(1)}). \quad (76)$$

The resulting vector is reshaped into the original $K \times m$ matrix. This move preserves task completeness and uniqueness while modifying the trip assignment or the retrieval order.

For the second layer, the operation-sequence neighborhood $\mathcal{N}_2(X)$ is generated by selecting one trip k and two distinct positions

$$u, v \in \{1, \dots, 2m - 1\}, \quad u \neq v, \quad (77)$$

and exchanging the corresponding genes:

$$(x_{k,u}^{(2)}, x_{k,v}^{(2)}) \leftarrow (x_{k,v}^{(2)}, x_{k,u}^{(2)}). \quad (78)$$

Position $2m$ is excluded to preserve the terminal retrieval position. When $m = 1$, this neighborhood cannot select two distinct positions and only $\mathcal{N}_1$ is used.

At each trial, one of the two neighborhoods is selected with equal probability when both are available. Only one swap operation is applied, which limits the difference between the current solution and its neighbor.

4.6.3. Feasibility Restoration and Acceptance Rule

A neighborhood move may disturb the operation pattern or the correspondence between the two encoding layers. Therefore, every generated neighbor X' is processed according to

$$X' \leftarrow \text{Sync}(\text{Repair}(X')), \quad (79)$$

before its objective value is calculated.

Let X denote the current solution and X' its feasible neighbor. The neighbor is accepted only if

$$F(X') < F(X). \quad (80)$$

Otherwise, the current solution remains unchanged. When a neighbor is accepted, it becomes the reference solution for the subsequent trial. Thus, several successive improvements may be obtained within one invocation of the local procedure.

The complete implementation is given in Algorithm 4.

Algorithm 4: Budgeted memetic local refinement

Input: Evaluated offspring X and its fitness f ; maximum number of trials L ; current fitness-evaluation count FE ; maximum budget $FE_{\max}$; problem data $\mathcal{D}$; crane capacity m .

Output: Locally refined chromosome X ; updated fitness f ; updated fitness-evaluation count FE .

```

1  $t \leftarrow 0$ 
2 while  $t < L$  and  $FE < FE_{\max}$  do
3    $X' \leftarrow X$ 
4   if  $m = 1$  or  $\text{rand}(0, 1) < 0.5$  then
5     Flatten  $X'^{(1)}$  into a retrieval-task vector
6     Select two distinct positions uniformly at random
7     Exchange the two retrieval-task identifiers
8     Reshape the vector into  $X'^{(1)}$ 
9   else
10    Select a trip  $k$  uniformly from  $\{1, \dots, K\}$ 
11    Select two distinct positions  $u$  and  $v$  uniformly from  $\{1, \dots, 2m - 1\}$ 
12    Exchange  $x_{k,u}^{(2)}$  and  $x_{k,v}^{(2)}$ 
13  end
14   $X' \leftarrow \text{Repair}(X')$ 
15   $X' \leftarrow \text{Sync}(X')$ 
16   $f' \leftarrow \text{DecodeAndEvaluate}(X', \mathcal{D})$ 
17   $FE \leftarrow FE + 1$ 
18   $t \leftarrow t + 1$ 
19  if  $f' < f$  then
20     $X \leftarrow X'$ 
21     $f \leftarrow f'$ 
22  end
23 end
24 return  $X$ ,  $f$ , and  $FE$ 

```

Every neighborhood evaluation in Algorithm 4 is counted as one fitness evaluation. Rejected neighbors therefore consume the same evaluation budget as accepted neighbors. The procedure returns the best solution obtained along the accepted sequence and passes it to the population-update process described in Section 4.5. Together with the elite-driven update, this bounded local improvement constitutes the memetic component of ED-SRMA.

5. Numerical Experiments and Performance Analysis

This section evaluates the proposed operating model and ED-SRMA through an RS-mode comparison, a controlled ablation study, an external comparison with PSO, FE-based convergence analysis, and shuttle-capacity sensitivity analysis. All comparisons use paired runs and a common fitness-evaluation budget.

5.1. Experimental Setup and Fair Comparison Protocol

5.1.1. Warehouse Configuration and Instance Generation

The experiments consider a dual-I/O middle-aisle AS/RS with one stacker crane. Let n denote the number of retrieval tasks; an equal number of storage tasks is considered. For each problem instance, the rack is represented by a $q \times q$ square grid, where

$$q = \lfloor \sqrt{2n} \rfloor + 1, \quad L = q^2. \quad (81)$$

Since $L > 2n$, the n retrieval locations are initially occupied and the remaining $L - n$ locations are initially empty. The initial vacancy ratio is therefore greater than 50%, which provides sufficient empty locations for shared-storage operations.

Retrieval locations are sampled without replacement from the rack grid, and their coordinates are assumed to be known before scheduling. The dual I/O ports are located at

$$IO_1 = (1, 0), \quad IO_2 = (q, 0). \quad (82)$$

The horizontal and vertical crane velocities are set to $v_x = v_y = 1.0$. Travel time is calculated using the Chebyshev metric defined in Section 3.4.2. Consequently, the reported objective values represent the total crane travel time under normalized motion speeds.

All algorithms were implemented in MATLAB R2024a and executed on the same computing platform. No parallel computation was used during the formal experiments.

5.1.2. Test Instances and Compared Methods

Seven problem scales were considered:

$$n \times m \in \{12 \times 3, 24 \times 2, 48 \times 6, 60 \times 3, 96 \times 4, 120 \times 5, 144 \times 4\}, \quad (83)$$

where n denotes the number of retrieval requests and m denotes the crane capacity. These instances cover different task sizes and within-trip sequencing complexities.

Three groups of experiments were conducted. First, the Dual-I/O No-RS and Dual-I/O RS operating modes were compared using the same ED-SRMA optimizer to evaluate the effect of immediate location reuse. Second, Base-GA, ED-GA, SR-MA, and ED-SRMA were compared in the ablation study. Their mechanism configurations are summarized in Table 1. Third, particle swarm optimization (PSO) was used as an external metaheuristic benchmark under the same fitness-evaluation budget. The implementation and representation of PSO are described in Section 5.4.

Table 1. Controlled mechanism settings of the four ablation variants.

Algorithm	Elite Preservation	N_e	Local Refinement	p_{LS}	L
Base-GA	No	0	No	0	0
ED-GA	Yes	2	No	0	0
SR-MA	No	0	Yes	0.20	3
ED-SRMA	Yes	2	Yes	0.20	3

For the shuttle-capacity sensitivity experiment, the number of retrieval requests was fixed at $n = 60$, while

$$m \in \{1, 2, 3, 4, 5, 6, 10\}. \quad (84)$$

5.1.3. Algorithm Parameter Settings

To ensure a controlled ablation study, the four genetic variants use the same population initialization, tournament selection, order crossover, coupled mutation, adaptive-mutation rule, feasibility repair, cross-layer synchronization, state-dependent decoding, and termination criterion. Their common parameters are

$$\begin{aligned} N_p &= 30, & N_t &= 3, & p_c &= 0.80, \\ p_m^0 &= 0.05, & p_m^+ &= 0.20, & \tau &= 10. \end{aligned} \quad (85)$$

Here, N_p is the population size, N_t is the tournament size, p_c is the crossover probability, p_m^0 and p_m^+ are the base and intensified mutation probabilities, respectively, and τ is the stagnation threshold. The adaptive-mutation rule is enabled identically in all four variants and is therefore not treated as an ablation factor.

The four variants differ only in elite preservation and budgeted local refinement, as summarized in Table 1.

In Table 1, N_e denotes the number of retained elite individuals, p_{LS} is the probability of activating local refinement for an evaluated offspring, and L is the maximum number of neighborhood trials per invocation. Consequently, comparisons among the four variants isolate the individual and combined effects of elite preservation and budgeted local refinement.

5.1.4. FE-Fair Paired Protocol and Statistical Analysis

Algorithms with embedded local refinement may perform more objective-function evaluations within one generation than conventional evolutionary algorithms. Using the same number of generations would therefore provide an unequal computational budget. All algorithms are consequently terminated according to the same maximum number of fitness evaluations:

$$FE_{\max} = 30030. \quad (86)$$

Every call to the objective function is counted as one fitness evaluation, including evaluations during population initialization, offspring assessment, local refinement, and particle assessment. Cached fitness values of retained elite individuals are reused without consuming additional evaluations.

For each problem scale, 30 paired independent runs are conducted. Within each run, all methods use the same warehouse instance and the same instance seed. The four ablation algorithms additionally use the same initial population. Because PSO employs a different representation, it generates its own random-key swarm using the corresponding paired initialization and search seeds. Different seed sets are used across independent runs.

The reported descriptive statistics include the best value, mean value, and standard deviation of the final objective values. Relative improvement is calculated using Base-GA as the reference for the ablation study and PSO as the reference for the external comparison.

For the four-algorithm ablation study, the Friedman test with the Iman–Davenport correction is first applied to examine the overall performance difference. After rejection of the omnibus null hypothesis, paired Wilcoxon signed-rank tests with Holm correction are conducted between ED-SRMA and the three ablation baselines. Win/tie/loss counts and rank-biserial effect sizes are reported for these post hoc comparisons. All statistical tests are two-sided, with the significance level set to

$$\alpha = 0.05. \quad (87)$$

The experiments in this study are numerical simulations based on the sensor-informed travel-time formulation. Physical sensor measurements, digital-twin validation, and deployment on an operational AS/RS platform are outside the scope of the present experiments and are discussed in Section 5.7.

5.2. Effect of the RS Operating Rule

To isolate the operational benefit of the RS mechanism, this subsection compares two operating modes under the same dual-I/O warehouse layout. The first mode, denoted as Dual-I/O No-RS, performs retrieval and storage actions independently. A location released by a retrieval task is added to the dynamic empty-location set, while the subsequent storage task is assigned to another available location according to the decoding rule. The second mode, denoted as Dual-I/O RS, permits an immediately following storage action to reuse the location released by the preceding retrieval operations. In this case, the two actions are completed at the same rack position, and the travel time between them is zero.

The two modes use the same ED-SRMA optimizer, warehouse instances, initial populations, paired random seeds, algorithm parameters, and fitness-evaluation budget. Therefore, the observed difference is attributable to the RS operating rule rather than to the warehouse topology or optimization method.

The relative improvement achieved by the RS mode is calculated as

$$\Delta_{RS} = \frac{\bar{F}_{No-RS} - \bar{F}_{RS}}{\bar{F}_{No-RS}} \times 100\%, \quad (88)$$

where $\bar{F}_{No-RS}$ and $\bar{F}_{RS}$ denote the mean total travel times obtained under the two operating modes, respectively.

As shown in Figure 4, the Dual-I/O RS mode achieves a lower mean total travel time than the Dual-I/O No-RS mode for all seven tested problem scales. The corresponding reductions are 8.22%, 4.72%, 14.89%, 6.25%, 6.12%, 6.26%, and 4.57% for the 12×3 , 24×2 , 48×6 , 60×3 , 96×4 , 120×5 , and 144×4 instances, respectively.

The largest improvement, 14.89%, is obtained for the 48×6 instance, where the relatively large shuttle capacity creates more opportunities to place a storage action immediately after a retrieval operation within the same trip. The 12×3 instance also exhibits a notable reduction of 8.22%. For the remaining instances, the improvement ranges from 4.57% to 6.26%, indicating that the RS mechanism provides a consistent reduction in crane travel time across different task sizes and shuttle capacities.

The benefit of the RS mode arises from the elimination of an intermediate movement between a retrieval location and a separately selected empty location. Under the No-RS mode, a retrieval operation releases its location to the empty-location set, but the subsequent storage action must still travel to an assigned empty location. Under the RS

mode, the released location is reused immediately when an adjacent retrieval–storage pattern occurs. This reduces empty travel without changing the number of tasks, the dual-I/O configuration, or the fitness-evaluation budget.

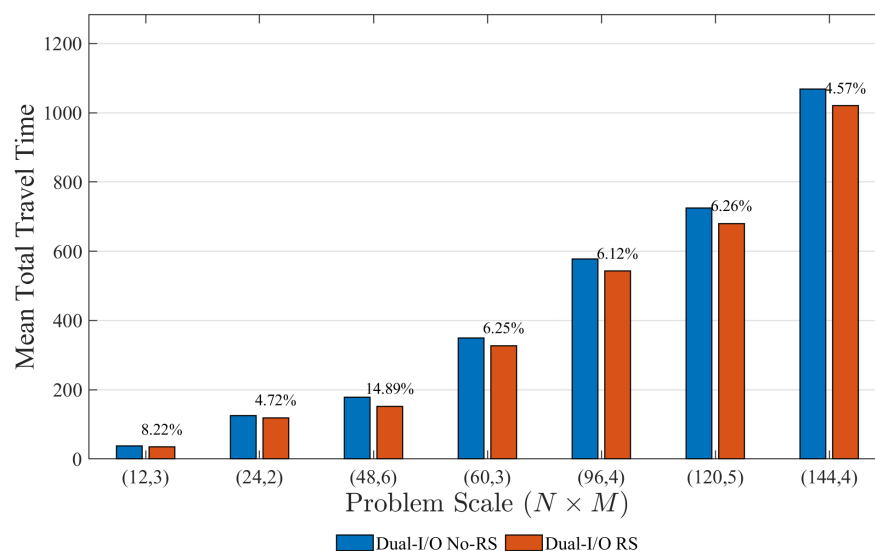


Figure 4. Comparison of the mean total travel times obtained under the Dual-I/O No-RS and Dual-I/O RS operating modes. The percentages above the RS bars indicate the reductions relative to the corresponding No-RS results.

The magnitude of the improvement is not monotonic with the number of requests. It depends jointly on the task sequence, shuttle capacity, spatial distribution of retrieval locations, and the frequency of feasible adjacent retrieval–storage patterns. In particular, a larger capacity can increase the number of potential RS combinations within a trip, but it also enlarges the within-trip sequencing space. Consequently, the operational gain varies across problem scales.

These results confirm that immediate location reuse consistently reduces crane travel under the tested dual-I/O configurations.

5.3. Ablation Study and Statistical Analysis

A controlled two-factor ablation study was conducted to quantify the individual and combined contributions of elite preservation and budgeted local refinement. Base-GA denotes the common genetic backbone with neither mechanism enabled; ED-GA enables only elite preservation; SR-MA enables only local refinement; and ED-SRMA enables both mechanisms.

Apart from these two factors, all variants use identical initialization, tournament selection, crossover, coupled and adaptive mutation, feasibility repair, cross-layer synchronization, state-dependent decoding, objective function, random settings, and fitness-evaluation budget. The mechanism configurations are summarized in Table 2.

Table 2. Mechanism configuration of the ablation algorithms.

Algorithm	Elite-Driven Update	Local Refinement
Base-GA	No	No
ED-GA	Yes	No
SR-MA	No	Yes
ED-SRMA	Yes	Yes

The performance results over 30 paired independent runs are reported in Table 3 and Figure 5. The relative improvement of algorithm a over Base-GA is calculated as

$$\text{Gap}_a = \frac{\bar{F}_{\text{Base-GA}} - \bar{F}_a}{\bar{F}_{\text{Base-GA}}} \times 100\%, \quad (89)$$

where $\bar{F}_a$ denotes the mean final objective value of algorithm a .

Accordingly, any systematic performance difference among the four variants can be attributed to elite preservation, budgeted local refinement, or their interaction under the controlled experimental settings. ED-GA reduces the mean total travel time by 4.38–28.72% relative to Base-GA, showing that elite preservation provides a consistent improvement in solution quality. SR-MA achieves smaller reductions of 0.82–1.94%, indicating that locally improved solutions may be lost when no elite mechanism is used. ED-SRMA obtains the lowest mean objective value for all seven problem scales, with improvements of 5.48–39.59%. The largest reduction occurs for the 120×5 instance, where the mean total travel time decreases from 1125.87 to 680.13 and the best value decreases from 1073.00 to 638.00. Improvements of 39.02%, 35.76%, and 34.81% are also obtained for the 48×6 , 96×4 , and 144×4 instances, respectively.

Table 3. Performance comparison of the four algorithms under an identical budget of 30,030 fitness evaluations. Best, Mean, and Std Dev are calculated over 30 independent runs, and Gap denotes the percentage reduction in the mean total travel time relative to Base-GA.

$n \times m$	Algorithm	Best	Mean	Std Dev	Gap (%)
12×3	Base-GA	31.00	36.50	2.0129	–
	ED-GA	29.00	34.90	2.0569	4.38%
	SR-MA	30.00	35.97	1.9911	1.46%
	ED-SRMA	29.00	34.50	2.0968	5.48%
24×2	Base-GA	124.00	135.80	5.9038	–
	ED-GA	109.00	122.37	7.2848	9.89%
	SR-MA	122.00	134.17	6.2977	1.20%
	ED-SRMA	106.00	119.40	7.1032	12.08%
48×6	Base-GA	238.00	249.33	6.5933	–
	ED-GA	165.00	177.73	8.5980	28.72%
	SR-MA	229.00	244.50	9.0620	1.94%
	ED-SRMA	142.00	152.03	5.5985	39.02%
60×3	Base-GA	421.00	449.83	14.0984	–
	ED-GA	361.00	394.67	18.6276	12.26%
	SR-MA	422.00	444.80	11.8857	1.12%
	ED-SRMA	307.00	327.53	12.9475	27.19%
96×4	Base-GA	809.00	841.37	17.6449	–
	ED-GA	678.00	731.03	28.0977	13.11%
	SR-MA	802.00	831.43	16.6892	1.18%
	ED-SRMA	511.00	540.47	16.7738	35.76%
120×5	Base-GA	1073.00	1125.87	22.0372	–
	ED-GA	905.00	980.47	36.7768	12.91%
	SR-MA	1063.00	1116.63	24.9613	0.82%
	ED-SRMA	638.00	680.13	19.4648	39.59%
144×4	Base-GA	1515.00	1575.67	31.8957	–
	ED-GA	1324.00	1430.00	47.4116	9.24%
	SR-MA	1484.00	1550.47	34.7699	1.60%
	ED-SRMA	975.00	1027.20	23.1180	34.81%

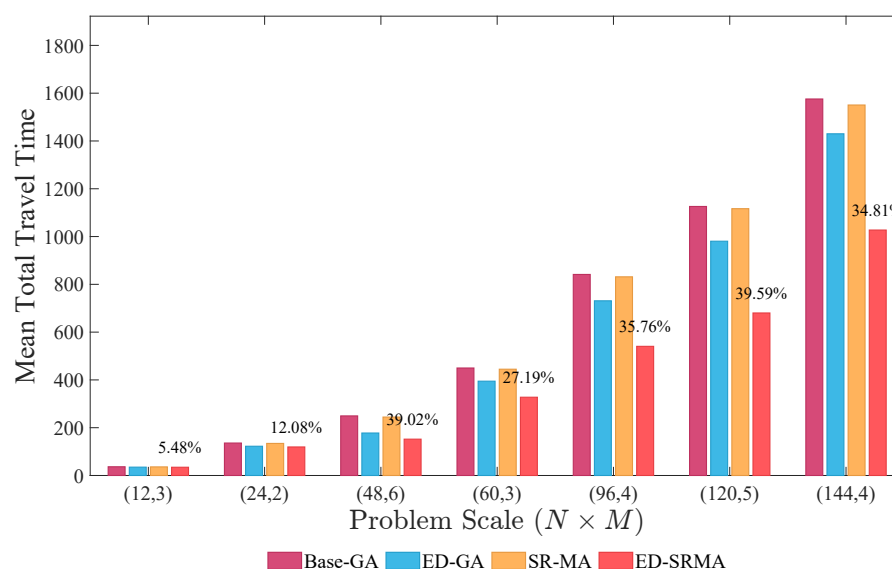


Figure 5. Comparison of the mean total travel times of the four algorithms across all problem scales under an identical fitness-evaluation budget. The labels above the ED-SRMA bars indicate the percentage reduction relative to Base-GA.

These results indicate that elite preservation contributes more consistently than local refinement when the two mechanisms are applied independently. Their combination produces the strongest performance because locally refined solutions can be retained and propagated through subsequent population updates. The increasing performance differences on several medium- and large-scale instances further suggest that the combined mechanism is particularly effective when shuttle capacity enlarges the within-trip sequencing space.

To determine whether the observed differences were attributable to random variation, a Friedman test was applied to the final objective values over the 210 paired scale–run blocks. The null hypothesis of equivalent performance was rejected:

$$\chi_F^2(3) = 552.1433, \quad p = 2.3842 \times 10^{-119}. \quad (90)$$

The corresponding Iman–Davenport statistic was

$$F_{ID}(3, 627) = 1482.1849, \quad p < 0.001. \quad (91)$$

Following the significant omnibus result, paired Wilcoxon signed-rank tests with Holm correction were conducted between ED-SRMA and each ablation baseline. The results are reported once in Table 4.

Almost all Holm-adjusted p -values are below 0.001, and the rank-biserial effect sizes range from 0.9769 to 1.0000, indicating statistically significant and large effects in favor of ED-SRMA. The median improvements over Base-GA, ED-GA, and SR-MA are 34.04%, 18.19%, and 32.73%, respectively. For the smallest 12×3 instance, however, the difference between ED-SRMA and ED-GA is not significant after Holm correction ($p_{\text{Holm}} = 0.0911$), because the restricted search space allows both methods to obtain similar solutions. The advantage of the complete framework becomes clearer as the scheduling and within-trip sequencing complexity increase.

Table 4. Statistical analysis of the ablation study using paired Wilcoxon signed-rank tests with Holm correction.

Comparison	W/T/L	Holm-Adjusted p	Rank-Biserial Effect Size	Median Improvement (%)
ED-SRMA vs. Base-GA	207/3/0	<0.001	1.0000	34.04
ED-SRMA vs. ED-GA	186/13/11	<0.001	0.9769	18.19
ED-SRMA vs. SR-MA	206/3/1	<0.001	0.9995	32.73

5.4. External Comparison with Particle Swarm Optimization

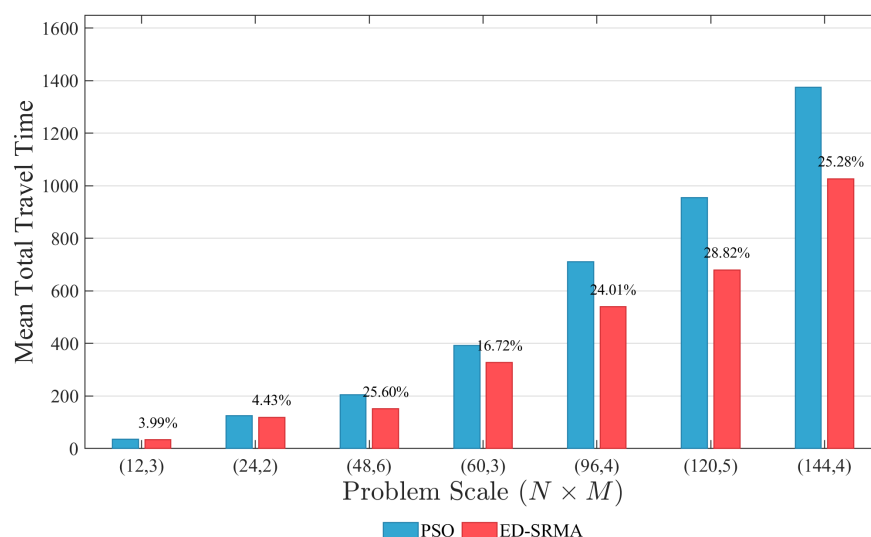
Particle swarm optimization (PSO) was selected as an external benchmark to contrast a continuous random-key metaheuristic against the problem-specific discrete representation of ED-SRMA. PSO represents candidate schedules using continuous random keys, which are deterministically decoded into retrieval-task assignments and within-trip operation sequences. Both algorithms use the same warehouse instances, population size, objective function, and fitness-evaluation budget defined in Equation (86) over 30 paired independent runs. All evaluations generated by the local-refinement procedure of ED-SRMA are included in this budget.

The relative improvement of ED-SRMA over PSO is defined as

$$\Delta_{\text{PSO}} = \frac{\bar{F}_{\text{PSO}} - \bar{F}_{\text{ED-SRMA}}}{\bar{F}_{\text{PSO}}} \times 100\%, \quad (92)$$

where $\bar{F}_{\text{PSO}}$ and $\bar{F}_{\text{ED-SRMA}}$ denote the corresponding mean final total travel times.

As shown in Figure 6, ED-SRMA obtains lower mean total travel times than PSO for all seven tested instances. The reductions are 3.99%, 4.43%, 25.60%, 16.72%, 24.01%, 28.82%, and 25.28% for the 12×3 , 24×2 , 48×6 , 60×3 , 96×4 , 120×5 , and 144×4 instances, respectively. The differences are small for the two smallest instances but become more pronounced for the medium- and large-scale problems. The largest improvement, 28.82%, is obtained for the 120×5 instance.

**Figure 6.** Comparison of the mean total travel times obtained by PSO and ED-SRMA across seven problem scales under the same fitness-evaluation budget. The percentages above the ED-SRMA bars denote the reductions relative to PSO.

The stronger performance on the more complex instances can be attributed to the problem-specific representation and constraint-handling procedures of ED-SRMA. Its dual-layer structure separates task assignment from operation sequencing, while repair and

synchronization ensure that only feasible and internally consistent schedules are evaluated. Elite preservation and local refinement further retain and improve promising solutions. By contrast, PSO searches in a continuous random-key space and depends on decoding to construct discrete schedules.

5.5. FE-Based Convergence Analysis

To examine the convergence behavior of the four ablation algorithms under a fair computational budget, the 120×5 instance was selected as a representative case. This instance contains a relatively large number of tasks and a complex within-trip sequencing structure. Figure 7 shows the median best-so-far objective values obtained by Base-GA, ED-GA, SR-MA, and ED-SRMA over 30 paired independent runs.

Within each paired run, the four algorithms use the same problem instance and initial population. The horizontal axis represents the cumulative number of fitness evaluations rather than generations. All objective-function calls, including those performed during population initialization, offspring evaluation, and local refinement, are counted toward the common budget of 30,030 fitness evaluations.

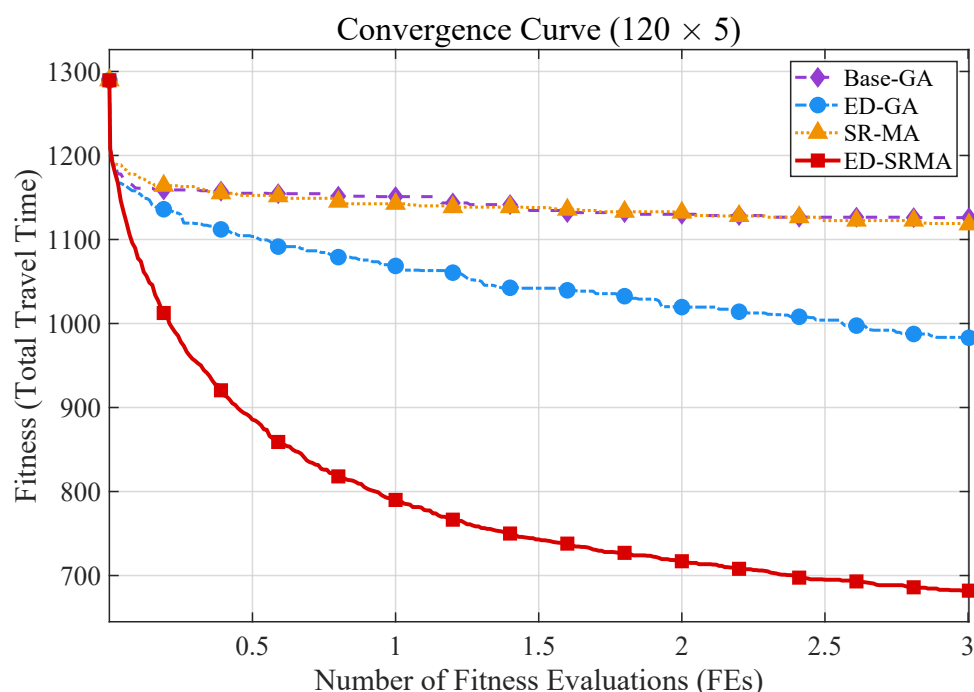


Figure 7. FE-based convergence comparison of the four ablation algorithms for the 120×5 instance. Each curve represents the median best-so-far objective value over 30 paired independent runs under a common budget of 30,030 fitness evaluations.

As shown in Figure 7, the four algorithms begin from similar objective-value ranges because they share the same initial population. ED-SRMA produces a more rapid reduction during the early stage, decreasing below approximately 900 within the first 5000 fitness evaluations. Its best-so-far objective value continues to decrease over the remaining budget and reaches approximately 680 at termination.

ED-GA also improves progressively but stabilizes at a higher objective value of approximately 980. Base-GA and SR-MA obtain most of their improvements during the early stage and subsequently remain near 1125 and 1120, respectively. These convergence patterns are consistent with the final solution-quality results reported in Section 5.3.

The comparison further illustrates the complementary contributions of the mechanisms incorporated into ED-SRMA. Elite preservation prevents promising solutions from being discarded during population replacement, whereas the budgeted local-refinement

procedure provides additional opportunities to improve selected feasible offspring. ED-GA benefits from elite preservation but does not include neighborhood refinement. In contrast, SR-MA applies local refinement without explicitly retaining elite individuals, so improvements obtained by individual offspring may not be preserved during subsequent population updates.

Under the common fitness-evaluation budget, ED-SRMA therefore achieves both an earlier reduction in the objective value and a lower final total travel time for the tested instance. The result indicates that the combined framework uses the available objective evaluations more effectively than the three ablation variants, although the conclusion is restricted to the examined instance and experimental settings.

5.6. Sensitivity Analysis of Shuttle Capacity

The shuttle capacity m , which denotes the number of unit loads a crane can carry simultaneously, is a critical hardware parameter that significantly affects the operational efficiency of the AS/RS. In this section, we examine the impact of m on the total travel time by evaluating seven capacity settings ranging from 1 to 10 while keeping the total request size constant at $n = 60$. The experimental results are presented in Figure 8.

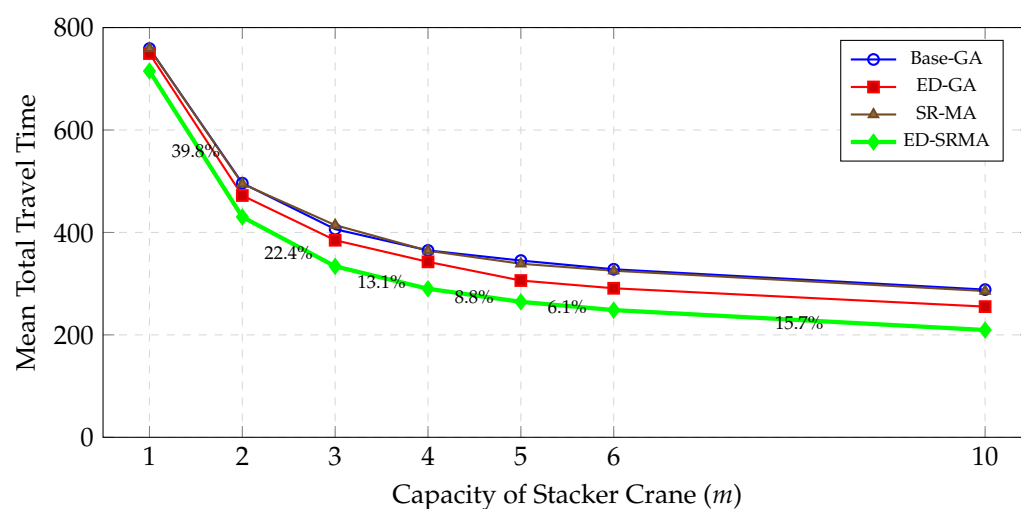


Figure 8. Capacity sensitivity analysis: the decrease rate between adjacent shuttle capacities is annotated for ED-SRMA, illustrating diminishing marginal returns as the shuttle capacity m increases ($n = 60$).

As shown in Figure 8, there is a clear negative correlation between the shuttle capacity m and the total travel time. Increasing m allows the crane to consolidate more storage and retrieval tasks into a single trip, thereby reducing the frequency of empty return trips to the I/O ports and enhancing the utilization of RS operations. However, a phenomenon of diminishing marginal returns is observed: the most significant reductions in travel time occur when m increases from 1 to 4. When m exceeds 6, the performance curve begins to plateau, and the marginal benefit of further increasing capacity becomes negligible. This suggests that, for the tested warehouse scale, a capacity of $m \in [4, 6]$ represents an optimal trade-off between hardware cost and operational throughput.

A noteworthy observation is that the performance gap between the proposed ED-SRMA and the Base-GA widens significantly as m increases. When the shuttle capacity is large, the number of possible task sequences and interleaved storage/retrieval patterns within each trip grows exponentially. Specifically, for a trip with capacity m , the theoretical number of operation sequences is related to $(2m)!$, leading to a combinatorial explosion in the solution space. Under these conditions, the Base-GA often fails to identify high-quality sequences due to the lack of specialized search pressure, resulting in stagnant

performance at high m values. In contrast, the ED-SRMA leverages its memetic local search and synchronized repair operators to effectively prune the search space and refine complex operation sequences. This capability allows ED-SRMA to maintain a steep efficiency improvement curve even under high loads, demonstrating its robust optimization capability for high-dimensional combinatorial problems.

5.7. Discussion and Limitations of Numerical Validation

5.7.1. Discussion of the Experimental Findings

Taken together, the experiments provide complementary evidence regarding the operating rule and the proposed solution method. The comparison under the same dual-I/O configuration confirms that immediate reuse of a released retrieval location can reduce unnecessary crane travel. The ablation study further shows that elite preservation and budgeted local refinement play complementary roles: local refinement improves selected feasible solutions, while elite preservation enables these improvements to be retained during subsequent population updates. The comparison with PSO indicates that the observed advantage is not limited to closely related genetic variants, and the FE-based convergence analysis for the representative 120×5 instance shows that ED-SRMA uses the common evaluation budget more effectively.

The performance advantage of ED-SRMA generally becomes more evident when larger task sets or shuttle capacities enlarge the feasible sequencing space. At the same time, increasing shuttle capacity reduces the number of crane trips but provides diminishing marginal reductions in travel time. These findings indicate that warehouse performance depends jointly on equipment capacity, operating rules, and scheduling quality.

5.7.2. Limitations and Scope of Validation

The conclusions above should be interpreted within the scope of the numerical experiments. First, the experiments use synthetically generated warehouse instances and a sensor-informed travel-time model. No measurements from an operational AS/RS, digital-twin platform, or physical prototype are used to validate the predicted travel times. The displacement and I/O-position sensors described in the model therefore represent the intended data interface for future deployment rather than a source of experimental measurements in the present study.

Second, the model assumes a single crane, known task locations, deterministic task availability, constant axis velocities, homogeneous handling times, and full-load trips. Acceleration, deceleration, mechanical delays, energy consumption, unexpected equipment states, dynamic order arrivals, and interference among multiple cranes are not considered. These assumptions make it possible to isolate the scheduling mechanisms but limit direct generalization to more complex industrial environments.

Third, the algorithmic results are based on the selected problem scales, parameter settings, and fitness-evaluation budget. Although paired runs and nonparametric tests reduce the influence of random variation, they do not establish universal superiority over all possible instances or optimization methods. Runtime values are also dependent on the computing platform and implementation details. In addition, PSO is used as a representative external metaheuristic rather than as an exhaustive comparison with all available optimization approaches.

6. Conclusions

This study shows that jointly considering storage-location allocation, storage/retrieval sequencing, dynamic location reuse, and inter-trip I/O continuity can substantially reduce crane travel in a dual-I/O shared-storage AS/RS. The proposed ED-SRMA was used to solve this integrated scheduling problem under a common fitness-evaluation budget.

The numerical results lead to three main findings. First, immediate reuse of retrieval locations through the RS operating rule reduced the mean total travel time by 4.57–14.89% relative to the corresponding dual-I/O No-RS mode, confirming that dynamic location reuse provides a measurable operational benefit. Second, ED-SRMA achieved the lowest mean objective value at all seven tested scales, with reductions of 5.48–39.59% relative to Base-GA and 3.99–28.82% relative to PSO. Third, the ablation results indicate that elite preservation and budgeted local refinement are complementary: local refinement improves selected feasible solutions, whereas elite preservation helps retain these improvements during subsequent population updates. The Friedman and paired Wilcoxon tests with Holm correction confirmed significant differences in the ablation comparisons, with adjusted p -values below 0.001 and rank-biserial effect sizes of 0.9769–1.0000. For the representative 120×5 instance, ED-SRMA also exhibited an earlier reduction in the objective value and continued improvement under the common fitness-evaluation budget.

These findings suggest that the performance of dual-I/O shared-storage systems depends not only on equipment capacity, but also on whether dynamic storage-location changes and I/O-state transitions are explicitly incorporated into scheduling decisions. The increasing advantage observed for several medium- and large-scale instances further indicates that problem-specific feasibility restoration and solution retention become more important as the within-trip sequencing space expands. The sensor-informed travel-time formulation also provides a direct interface through which analytical schedules may subsequently be evaluated using measured crane-motion data.

The conclusions remain limited to numerical experiments conducted under simplified and deterministic operating assumptions. Their generalizability to industrial systems with realistic motion dynamics, uncertain order arrivals, multiple interacting cranes, and heterogeneous handling conditions has not yet been established. Future research will therefore focus on validation using measured data or a digital-twin platform, extensions to dynamic and multi-crane environments, and comparisons with exact and additional metaheuristic methods on broader industrial benchmark sets. In addition, a trajectory-level analysis will be conducted by recording the accepted and rejected evolutionary and local-refinement moves and mapping them to transitions among attraction basins. Local-optima-network and search-trajectory-network analyses [41,42] could then be used to distinguish successful, unsuccessful, deceptive, and rejected search behaviors, thereby providing a more precise explanation of why particular operators succeed or fail at different stages of the search.

Author Contributions: Conceptualization, Z.Y. and W.Z.; methodology, Z.Y. and W.Z.; software, Z.Y.; validation, Z.Y. and W.Z.; formal analysis, Z.Y.; investigation, Z.Y.; resources, W.Z. and C.Z.; data curation, Z.Y.; writing—original draft preparation, Z.Y.; writing—review and editing, W.Z. and C.Z.; visualization, Z.Y.; supervision, W.Z. and C.Z.; project administration, C.Z.; funding acquisition, C.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the 2025 Interdisciplinary Research Support Program of Huazhong University of Science and Technology, Grant No. 2025JCYJ021.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

- Chen, C.; Liu, J.; Yin, H.; Huang, B. A Vision-Based Method for Detecting the Position of Stacked Goods in Automated Storage and Retrieval Systems. *Sensors* **2025**, *25*, 2623.
- Wang, Y.; Qin, J.; Mou, S.; Huang, K.; Zhao, X. DSS approach for sustainable system design of shuttle-based storage and retrieval systems. *Flex. Serv. Manuf. J.* **2023**, *35*, 698–726.
- Kostrzewski, M.; Filina-Dawidowicz, L.; Walusiak, S. Modern technologies development in logistics centers: The case study of Poland. *Transp. Res. Procedia* **2021**, *55*, 268–275.
- Gagliardi, J.P.; Renaud, J.; Ruiz, A. Models for automated storage and retrieval systems: A literature review. *Int. J. Prod. Res.* **2012**, *50*, 7110–7125.
- Zaerpour, N.; Yu, Y.; deKoster, R.B. Storing fresh produce for fast retrieval in an automated compact cross-dock system. *Prod. Oper. Manag.* **2015**, *24*, 1266–1284.
- Wang, Y.Z.; Tian, X.D.; Du, Y.Y.; Wu, Z.L.; Hu, Z.H. Synchronized scheduling for twin stacking cranes and handshake bay optimization considering loading-first at automated container terminals. *Flex. Serv. Manuf. J.* **2026**, 1–43. <https://doi.org/10.1007/s10696-026-09656-7>.
- Holubčík, M.; Koman, G.; Soviar, J. Industry 4.0 in logistics operations. *Transp. Res. Procedia* **2021**, *53*, 282–288.
- Barreto, L.; Amaral, A.; Pereira, T. Industry 4.0 implications in logistics: An overview. *Procedia Manuf.* **2017**, *13*, 1245–1252.
- Tubis, A.A.; Rohman, J. Intelligent warehouse in industry 4.0—Systematic literature review. *Sensors* **2023**, *23*, 4105.
- Polten, L.; Emde, S. Multi-shuttle crane scheduling in automated storage and retrieval systems. *Eur. J. Oper. Res.* **2022**, *302*, 892–908.
- Lin, Y.; Xu, Y.; Zhu, J.; Wang, X.; Wang, L.; Hu, G. MLATSO: A method for task scheduling optimization in multi-load AGVs-based systems. *Robot. Comput.-Integr. Manuf.* **2023**, *79*, 102397.
- Fan, H.; Ouyang, B.; Fang, Z.; Yan, Z.; He, J.; Zhang, Z.; Wang, Y. A Lifelong Multi-Shuttle Scheduling Framework for the AS/RS System. *IEEE Trans. Autom. Sci. Eng.* **2025**, *22*, 11577–11588.
- Satori, K.; Yoshikawa, N. Quantum optimization for location assignment problem in ASSR. In Proceedings of the PHM Society Asia-Pacific Conference, Tokyo, Japan, 11–14 September 2023; Volume 4.
- Hu, Y.; Liu, X.; An, K.; Ji, Y.; Qian, X.; Peng, Y.; Chen, G.; Cao, Y.; Wu, F.; Cheng, Y. Optimizing Path Planning and Conflict Resolution in High-Density AS/RS with a Multi-Shuttle Cooperative Scheduling Framework. *IEEE Trans. Instrum. Meas.* **2025**, *74*, 3547423.
- Lyu, Z.; Lin, P.; Guo, D.; Huang, G.Q. Towards zero-warehousing smart manufacturing from zero-inventory just-in-time production. *Robot. Comput.-Integr. Manuf.* **2020**, *64*, 101932.
- Yang, P.; Miao, L.; Xue, Z.; Ye, B. Variable neighborhood search heuristic for storage location assignment and storage/retrieval scheduling under shared storage in multi-shuttle automated storage/retrieval systems. *Transp. Res. Part E Logist. Transp. Rev.* **2015**, *79*, 164–177.
- Cai, A. Stacker crane scheduling strategy in the same-track dual-car operation mode. *Comput. Integr. Manuf. Syst.* **2018**, *24*, 1483–1493.
- Geng, S.; Wang, L.; Li, D.; Jiang, B.; Su, X. Research on scheduling strategy for automated storage and retrieval system. *CAAI Trans. Intell. Technol.* **2022**, *7*, 522–536.
- Alnahhal, M.; Salah, B.; Ahmad, R. Increasing throughput in warehouses: The effect of storage reallocation and the location of input/output station. *Sustainability* **2022**, *14*, 4611.
- Song, Y.B.; Mu, H.B. Large-Scale Storage/Retrieval Requests Sorting Algorithm for Multi-I/O Depots Automated Storage/Retrieval Systems. *Discret. Dyn. Nat. Soc.* **2021**, *2021*, 6646180.
- Xu, R.; Tuo, Y.; Chen, H.; Xu, J. Scheduling parallel cranes for unit-load automated storage and retrieval systems with shared storage. *Systems* **2023**, *12*, 3.
- Roodbergen, K.J.; Vis, I.F. A survey of literature on automated storage and retrieval systems. *Eur. J. Oper. Res.* **2009**, *194*, 343–362.
- Boysen, N.; Stephan, K. A survey on single crane scheduling in automated storage/retrieval systems. *Eur. J. Oper. Res.* **2016**, *254*, 691–704.
- Tanaka, S.; Araki, M. Routing problem under the shared storage policy for unit-load automated storage and retrieval systems with separate input and output points. *Int. J. Prod. Res.* **2009**, *47*, 2391–2408.
- Chen, L.; Langevin, A.; Riopel, D. The storage location assignment and interleaving problem in an automated storage/retrieval system with shared storage. *Int. J. Prod. Res.* **2010**, *48*, 991–1011.
- Zhang, W.; Deng, Z.; Zhang, C.; Shen, W. Integrated optimization of storage space allocation and crane scheduling in automated storage and retrieval systems. *Robot. Comput.-Integr. Manuf.* **2025**, *93*, 102918.

27. Salah, B.; Alnahhal, M.; Ahmad, R. Automated stacker cranes: A two-step storage reallocation process for enhanced service efficiency. *Processes* **2021**, *10*, 2.
28. Rizqi, Z.U.; Chou, S.Y.; Khairunisa, A. Energy harvesting for automated storage and retrieval system with sustainable configuration of storage assignment and input/output point. *Transp. Res. Part E Logist. Transp. Rev.* **2024**, *192*, 103781.
29. Yan, Q.; Lu, J.; Tang, H.; Zhan, Y.; Zhang, X.; Li, Y. Travel time analysis and dimension optimisation design of double-ended compact storage system. *Int. J. Prod. Res.* **2023**, *61*, 6718–6745.
30. Yan, Q.; Lu, J.; Shao, Y.; Xu, L.; Ren, C. A scheduling optimization method for stacker path in double-ended compact storage system. *Adv. Mech. Eng.* **2023**, *15*, 16878132231154173.
31. Hu, L.; Geng, S.; Zhang, W.; Yan, C.; Hu, Z.; Cai, Y. A Modified Brain Storm Optimization Algorithm for Solving Scheduling of Double-End Automated Storage and Retrieval Systems. *Symmetry* **2024**, *16*, 1068.
32. Buckow, J.N.; Goerigk, M.; Knust, S. Retrieval optimization in a warehouse with multiple input/output-points. *OR Spectr.* **2025**, *47*, 1–34.
33. Ishikura, H.; Tateiwa, N.; Egi, S.; Oe, I.; Hata, N.; Mitsutake, T.; Yamamura, K.; Fujii, M.; Fujisawa, K. Scheduling system for automated storage and retrieval system with multiple machines using a time-expanded network. *Jpn. J. Ind. Appl. Math.* **2024**, *41*, 447–474.
34. Antomarioni, S.; Lucantoni, L.; Ciarapica, F.E.; Bevilacqua, M. Data-driven decision support system for managing item allocation in an ASRS: A framework development and a case study. *Expert Syst. Appl.* **2021**, *185*, 115622.
35. Ekren, B.Y.; Arslan, B. A reinforcement learning approach for transaction scheduling in a shuttle-based storage and retrieval system. *Int. Trans. Oper. Res.* **2024**, *31*, 274–295.
36. Rizqi, Z.U.; Chou, S.Y.; Yu, T.H.K. Data-driven approach for dwell point positioning in automated storage and retrieval system: A metaheuristic-optimized ensemble learning. *Ann. Oper. Res.* **2024**, 1–24. <https://doi.org/10.1007/s10479-024-06342-5>.
37. Rizqi, Z.U.; Chou, S.Y. Dynamic crane scheduling for green automated warehousing: learning-based simulation-optimization approach. *Flex. Serv. Manuf. J.* **2025**, 1–28. <https://doi.org/10.1007/s10696-025-09626-5>.
38. Kouloughli, I.; Feciane, M.K.; Fandi, W. A Q-learning approach for optimizing the retrieval time in an AS/RS. *Int. J. Adv. Manuf. Technol.* **2025**, *140*, 5515–5534.
39. Sun, B.; Xu, Z.; Ranjitkar, P.; Wei, M. DS-MGCSTNet: Multi-Graph convolution spatial-temporal network with dynamic and static graph fusion for flight delay prediction. *J. Air Transp. Manag.* **2026**, *136*, 103047. <https://doi.org/10.1016/j.jairtraman.2026.103047>.
40. Yang, P.; Peng, Y.; Ye, B.; Miao, L. Integrated optimization of location assignment and sequencing in multi-shuttle automated storage and retrieval systems under modified 2 n-command cycle pattern. *Eng. Optim.* **2017**, *49*, 1604–1620.
41. Ochoa, G.; Tomassini, M.; Vérel, S.; Darabos, C. A study of NK landscapes' basins and local optima networks. In *Proceedings of the GECCO '08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*; Association for Computing Machinery: New York, NY, USA, 2008; pp. 555–562. <https://doi.org/10.1145/1389095.1389204>.
42. Ochoa, G.; Malan, K.M.; Blum, C. Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics. *Appl. Soft Comput.* **2021**, *109*, 107492. <https://doi.org/10.1016/j.asoc.2021.107492>.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.