mathematics

MDPI

Article

Joint Optimization of Storage Allocation and Picking Efficiency for Fresh Products Using a Particle Swarm-Guided Hybrid Genetic Algorithm

Yixuan Zhou, Yao Xu, Kewen Xie and Jian Li *

College of Information Management, Nanjing Agricultural University, Nanjing 211800, China; 31322223@stu.njau.edu.cn (Y.Z.); 31322326@stu.njau.edu.cn (Y.X.); 31322331@stu.njau.edu.cn (K.X.)
* Correspondence: lijianzh@njau.edu.cn

Abstract

The joint optimization of storage location assignment and order picking efficiency for fresh products has become a vital challenge in intelligent warehousing because of the perishable nature of goods, strict temperature requirements, and the need to balance cost and efficiency. This study proposes a comprehensive mathematical model that integrates five critical cost components: picking path, storage layout deviation, First-In-First-Out (FIFO) penalty, energy consumption, and picker workload balance. To solve this NP-hard combinatorial optimization problem, we develop a Particle Swarm-guided hybrid Genetic-Simulated Annealing (PS-GSA) algorithm that synergistically combines global exploration by Particle Swarm Optimization (PSO), population evolution of Genetic Algorithm (GA), and the local refinement and probabilistic acceptance of Simulated Annealing (SA) enhanced with Variable Neighborhood Search (VNS). Computational experiments based on real enterprise data demonstrate the superiority of PS-GSA over benchmark algorithms (GA, SA, HPSO, and GSA) in terms of solution quality, convergence behavior, and stability, achieving 4.08–9.43% performance improvements in large-scale instances. The proposed method not only offers a robust theoretical contribution to combinatorial optimization but also provides a practical decision-support tool for fresh e-commerce warehousing, enabling managers to flexibly weigh efficiency, cost, and sustainability under different strategic priorities.

Keywords: fresh products; storage allocation; hybrid metaheuristic algorithm; particle swarm optimization; variable neighborhood search

MSC: 90B05

check for updates

Academic Editor: Gyorgy Dosa

Received: 1 September 2025
Revised: 12 October 2025
Accepted: 16 October 2025
Published: 28 October 2025

Citation: Zhou, Y.; Xu, Y.; Xie, K.; Li, J. Joint Optimization of Storage Allocation and Picking Efficiency for Fresh Products Using a Particle Swarm-Guided Hybrid Genetic Algorithm. *Mathematics* **2025**, *13*, 3428. https://doi.org/10.3390/math13213428

Copyright: © 2025 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) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The efficiency of storage allocation and order picking for fresh products is a critical determinant of overall supply chain performance, impacting product preservation, logistics cost control, and end-user delivery timeliness [1,2]. As traditional warehousing paradigms evolve toward intelligent, fine-grained management, the processes for storing and picking fresh products demand greater dynamic adaptability. Modern fresh e-commerce platforms, in the process of handling large volumes of orders and high-frequency deliveries, urgently need a flexible and responsive warehouse scheduling mechanism. In this context, intelligent warehousing systems are gradually being introduced into actual operations, the essence of which is to achieve coordinated control over aspects like slotting, picking paths, and personnel scheduling through algorithmic optimization.

Within intelligent warehousing systems, the order picking process for fresh products is characterized by a high degree of operational flexibility, manifested in several key areas: (1) Multiple Routing Options: A single order can be fulfilled through various picking routes, the optimality of which must be assessed based on the real-time warehouse layout and order prioritization. (2) Flexible Item Sequencing: For certain orders, the sequence of item collection is adjustable, with different sequences significantly impacting overall picking efficiency. (3) Dynamic Task Allocation: Picking tasks can be dynamically distributed among multiple pickers to balance operational workloads. The inherent perishability and short shelf-life of fresh products impose additional constraints; their storage locations must not only satisfy temperature control requirements but also be strategically positioned near exits or packing stations to minimize dwell time and energy consumption. The scope for optimizing the storage allocation of fresh products is substantially greater than for conventional goods, involving a more complex set of decision variables and constraints.

Therefore, the core challenge in fresh produce warehousing lies in the joint optimization of storage location assignment and order picking routing. The primary optimization objectives focus on minimizing picking path costs and optimizing the warehouse storage location layout to meet the aforementioned complex requirements. Simultaneously, considering the actual operational priorities of different warehouses, the relative weights of secondary objectives—such as ongoing energy costs for temperature control, additional operational costs incurred by ensuring FIFO (First-In-First-Out), and order picker task scheduling costs—are dynamically configured and adjusted entirely by warehouse administrators based on actual management needs.

In recent years, the optimization of storage allocation and order picking operations has emerged as a prominent area of research. Hall et al. [3] enhanced picking efficiency by comparing various picking strategies and optimizing warehouse layouts. Petersen et al. [4] investigated the impact of three operational decisions—order picking, storage assignment, and routing—on picker travel distance, concluding that category-based or capacity-based storage strategies yield savings comparable to order batching but are less sensitive to variations in average order size. Early research on fresh product warehousing mainly focused on inventory management. Nahmias [5] reviewed the relevant literature on the problem of determining appropriate ordering policies for perishable inventory with a fixed lifetime and inventory subject to continuous exponential decay. Piramuthu et al. [6] integrated product quality information into management frameworks to enhance retailer profitability, while Bai et al. [7] formulated a single-period inventory and shelf-space allocation model specifically for fresh agricultural produce. They used an improved Generalized Reduced Gradient (GRG) algorithm for solving it. Lin et al. [8] proposed a mathematical model for the coordinated optimization of slotting and AGV (Automated Guided Vehicle) paths, and used an improved genetic algorithm to achieve it. The experimental results verified the effectiveness and stability of the proposed storage allocation optimization algorithm.

With the rise of fresh e-commerce, the complexity of storage allocation and picking has increased significantly, prompting researchers to explore more intelligent optimization algorithms. In the past few decades, more than a dozen different heuristic algorithms have been developed [9], some of which are inspired by biological habits, such as genetic algorithms, ant colony algorithms, bat algorithms, firefly algorithms, and so on. Among them, the genetic algorithm has been used to solve complex problems in real life, such as in energy [10,11], logistics [12], medicine [13], robotics, and other engineering fields [14,15].

Focusing on fresh produce application scenarios, Azadeh et al. [16] formulated a transshipment inventory-routing problem for a single perishable product, which they solved using a genetic algorithm. Similarly, Hiassat et al. [17] addressed a location-inventory-routing model for perishable goods, employing a genetic algorithm enhanced with a novel

chromosome representation and a local search heuristic. Concurrently, hybrid optimization algorithms have offered a new paradigm for tackling these problems. Li et al. [18] designed a hybrid genetic algorithm based on information entropy and game theory to mitigate the tendency of traditional GAs to converge to local optima. Zhang et al. [19] proposed a hybrid Particle Swarm Optimization (HPSO) algorithm for process planning, which replaces the standard particle position and velocity update rules with genetic operators. Addressing the Traveling Salesman Problem (TSP), He et al. [20] developed an Improved Genetic-Simulated Annealing Algorithm (IGSAA) to counteract the premature convergence of GA and the slow convergence of Simulated Annealing (SA). Liu et al. [21] introduced a hybrid GA-PSO algorithm to optimize a five-parameter BRDF (Bidirectional Reflectance Distribution Function) model, demonstrating its superior accuracy and convergence speed through comparative experiments on different materials. Although these methods show potential for solving complex problems, some studies have shown that their algorithmic robustness is susceptible to some strong constraints in fresh product scenarios.

In order to overcome the defects of a single algorithm, existing research has turned to hybrid algorithms that incorporate the advantages of multiple algorithms: Heidari et al. [22] proposed the HHO-VNS framework which innovatively combines the global exploration capability of Harris Hawk Optimization algorithm with Variable Neighborhood Search, but its fixed neighborhood switching mechanism still exists as a bottleneck of searching efficiency in some complex problems; Although the GA-ALNS (Genetic Algorithm-Adaptive Large Neighborhood Search) two-stage architecture developed by Ropke et al. [23] effectively improves the solution space traversal, there is still room for further improvement in the setting of the core parameters of the algorithm. Meanwhile, the above deficiencies are further amplified in fresh storage scenarios, which reveals the core challenge faced by existing studies in fresh scenarios: namely, the lack of modeling level. Some studies have not yet established a joint storage-picking optimization system that integrates temperature control and time constraints, and there is an urgent need to build a modeling system that covers cargo storage allocation, dynamic picking path planning, and human resource scheduling.

In order to effectively fill this research gap, the algorithm design needs to satisfy the above two core challenges of multi-constraints and dynamic equilibrium ability at the same time. In view of this goal, Particle Swarm algorithm (PSO) is often chosen as the mainstream solution algorithm because of its global exploration advantage, especially suitable for solving the high-dimensional coupling characteristics of the solution space triggered by the large-scale dynamic order scheduling, multi-temperature zones coordinated control, and strong time constraints in fresh food. The experimental results verified the effectiveness and stability of the proposed storage allocation collaborative optimization algorithm, where the traditional heuristic algorithms are easily limited by the local search capability. However, the application form of PSO is still limited by inherent defects such as premature convergence risk, exploration–exploitation behavior imbalance and static parameter dependence. Therefore, this study constructs a Particle Swarm-guided hybrid Genetic-Simulated Annealing algorithm (PS-GSA), whose core motivation is to fully use the complementary mechanisms of these algorithms, i.e., the global exploration of PSO, the cross-variance mechanism of GA, and the Metropolis criterion of SA, to match with the scenario of storage allocation for fresh products. We propose the Particle Swarm-guided hybrid Genetic-Simulated Annealing algorithm (PS-GSA), whose core innovativeness is embodied in the design of the three-level co-optimization mechanism.

The main contributions of this study can be summarized in four points:

First, this study pioneers a novel hierarchical and synergistic optimization framework. Unlike conventional hybrid algorithms that merely combine operators, our proposed PS-GSA establishes a distinct hierarchical relationship where Particle Swarm Optimization

(PSO) acts as a high-level global strategist. The PSO algorithm does not directly manipulate the solution but guides the evolutionary trajectory of the entire population within the lower-level Genetic Algorithm (GA). This primary architecture effectively leverages PSO's strengths in rapid global convergence to prevent the GA population from prematurely stagnating in local optima, thus creating a more powerful and purposeful exploration of the solution space.

Second, we introduce a sophisticated dynamic mechanism to explicitly balance exploration and exploitation. This is achieved through the synergistic integration of Simulated Annealing (SA) and Variable Neighborhood Search (VNS). The Metropolis acceptance criterion of SA is employed not as a standalone search algorithm, but as a probabilistic gateway that adaptively controls the trade-off between accepting superior solutions and exploring potentially inferior but promising regions. This process is further enhanced by a VNS-based local search, which is triggered to intensify the search in promising areas. This dual-component mechanism allows the algorithm to dynamically shift its behavior from broad exploration in the early stages to deep exploitation in the later stages, effectively addressing the premature convergence issue common in complex combinatorial optimization problems.

Third, this study contributes a highly comprehensive cost model specifically tailored to the unique operational challenges of fresh e-commerce. Moving beyond the traditional objectives of minimizing travel distance and optimizing storage assignments, our model integrates critical, real-world factors pertinent to the fresh product supply chain. These include the energy consumption costs associated with different temperature zones (ambient, refrigerated, and frozen), the layout optimization costs to ensure logical zoning, and a picker scheduling cost objective designed to balance workload among workers. By formulating this multi-dimensional objective function, our research provides a more holistic and practically relevant decision-making framework for warehouse managers in the fresh e-commerce sector, enabling them to make more informed trade-offs between efficiency, cost, and operational sustainability.

Fourthly, we rigorously evaluate the practical applicability and robustness of our model through systematic trade-off and sensitivity analyses. This investigation not only reveals the quantitative trade-offs among different strategic operational objectives but also confirms the solution's resilience to uncertainties in decision-maker preferences, providing a solid foundation for the model's deployment in complex, real-world environments.

Comparisons with similar studies also reveals the above advantages. For example, Pan et al. [24] used a genetic algorithm combined with space overflow correction mechanism to optimize the storage allocation of cargo for the traditional pick-and-pass warehouse system, which only focuses on the workload balance of multi-pickers and the reduction of SKU (Stock Keeping Unit) out-of-stock, which only applies to small- and medium-sized general warehouses, and does not involve the core characteristics of perishability and temperature control requirements of fresh food. In contrast, this study achieves an all-round breakthrough for fresh storage, not only constructing a three-zone temperature control system of ambient/chilled/frozen, but by clarifying the capacity differentiation of cargo space, introducing the constraints of odor phasing (setting the minimum storage distance according to the intensity of the odor to prevent contamination) and the penalty cost of FIFO (quantifying the priority of storage demand for the products first stored by the time system of the storage system), accurately covering the pain points of fresh quality assurance through the AHP (Analytic Hierarchy Process) to determine the weights, the picking path, cargo layout deviation, energy consumption and carbon emissions, FIFO penalty and personnel scheduling of the five costs into a multi-dimensional model, not only to retain the other side of the concern of the efficiency of the target, but also to add

new fresh products specific to protect product quality and energy-saving goals. Similarly, Zhang et al. [25] focus on non-traditional warehouse layouts such as Flying-V and Fishbone, and adopt the fireworks algorithm (FWA) with adaptive explosion and selection strategy to optimize the picking efficiency and shelf stability as the dual-objective optimization, which can improve the energy efficiency by shortening the picking distance, but it still belongs to the category of general-purpose warehousing, and cannot consider the temperature control dependence of the fresh food (e.g., high energy consumption of the freezer area), perishability-induced storage priority difference and personnel scheduling synergy needs, and did not quantify the matching relationship between multi-objective weights and the actual needs of fresh food. In this paper, the optimization of fresh storage is based on the characteristics of fresh storage, which not only considers the rationality of the warehouse layout but also clarifies the temperature control matching rules of products and cargo space, quantifies the energy consumption cost of different temperature control zones, and solves the problem of high energy consumption of fresh cold chain control that is neglected by the previous one, but also reduces the expenditure of the cold chain and guarantees the product quality of fresh products through energy consumption control.

The structure of this paper is arranged as follows: Section 2 introduces the optimization model and constraint settings for the fresh product storage allocation; Section 3 elaborates on the design of the proposed optimization algorithm; Section 4 presents the simulation process and results evaluation of a practical case study; and Section 5 summarizes the research contributions and looks forward to future work.

2. Mathematical Model

We consider a fresh product warehouse composed of distinct temperature-controlled zones, as depicted in Figure 1. The storage area is partitioned into three specific zones: ambient, refrigerated, and frozen. Each storage location is pre-assigned to a temperature zone, has a width of 0.5 m, and is subject to a fixed capacity C_k. For instance, locations in the frozen zone can hold up to five units of an identical product, whereas locations in other zones have a capacity of one. Order pickers employ a discrete picking (picker-to-parts) strategy, where each picking tour originates and terminates at a single depot (designated as S_0) after visiting all required item locations. To accommodate the perishability, stringent temperature controls, and environmental considerations associated with fresh products, an efficient warehouse management system must simultaneously pursue the following objectives:

(1) Picker Routing Optimization: To minimize the total travel distance and time required to fulfill all orders.
(2) Storage Location Assignment Optimization: To assign products to locations as close as possible to their designated central storage points or the packing station, thereby facilitating First-In, First-Out inventory control and reducing picking times.
(3) Energy and Carbon Footprint Minimization: To assign higher energy cost coefficients to high-consumption zones (e.g., refrigerated and frozen areas) to ensure that the storage strategy minimizes total energy consumption and carbon emissions while satisfying both temperature and picking efficiency constraints.
(4) Workload Balancing: To optimally assign orders among pickers to minimize operational conflicts and route overlaps, balance workloads, and enhance overall collaborative efficiency.

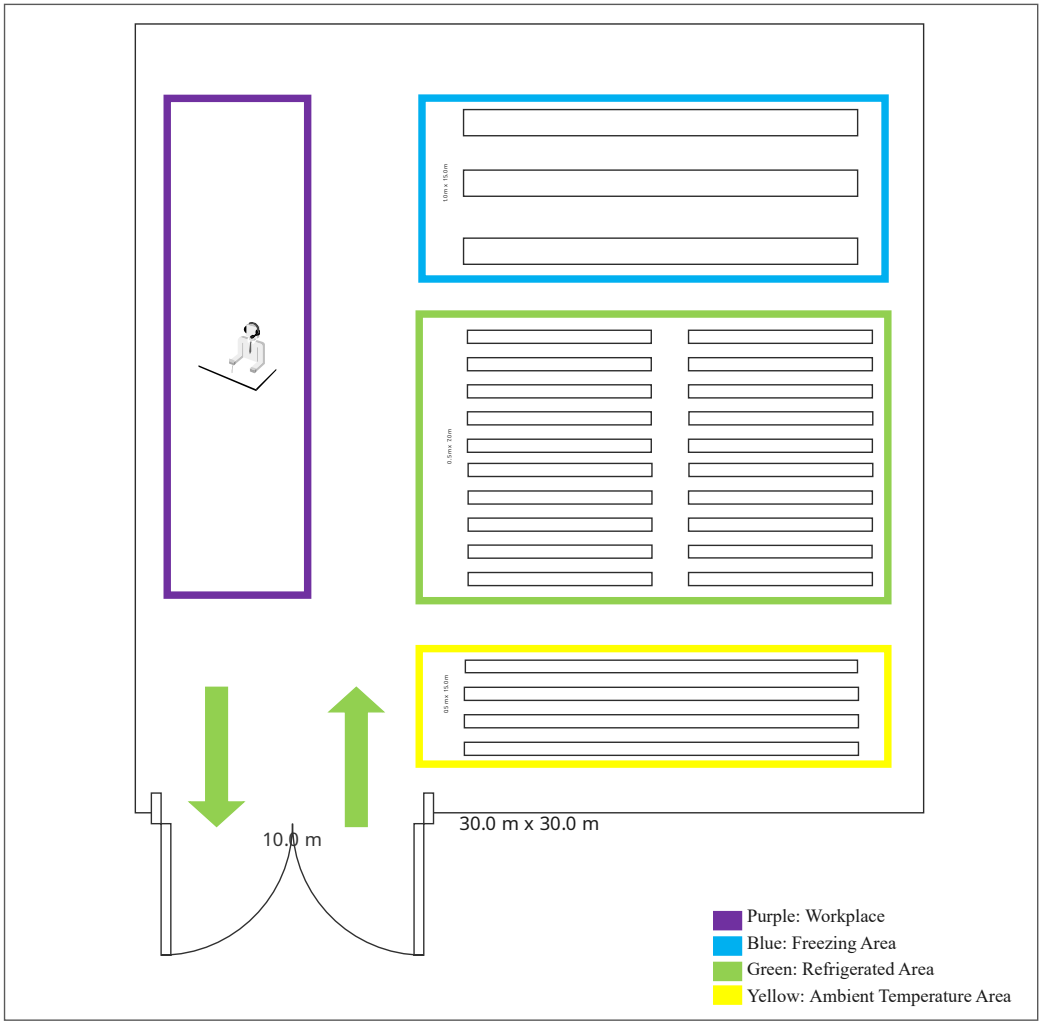


Figure 1. Warehouse layout plan.

The model considers a general fresh product warehouse layout organized as a three-dimensional grid, which can be adapted to various physical configurations. Let N_r, N_c, and N_l denote the total number of rows, columns, and levels in the storage area, respectively. Each discrete storage location k is uniquely identified by a coordinate triplet (r_k, c_k, l_k), where $r_k \in \{1, 2, \ldots, N_r\}$, $c_k \in \{1, 2, \ldots, N_c\}$, and $l_k \in \{1, 2, \ldots, N_l\}$. The set of all storage locations is denoted by K. The set of pickers is denoted by $P = \{1, 2, \ldots, |P|\}$, where $|P|$ is the total number of pickers. Each location k is pre-assigned to a specific temperature zone, represented by a numerical index $Q_k \in Q_T$, where Q_T is the set of all distinct temperature zones. Similarly, each product $i \in I$ has a specific temperature requirement, denoted by an index $Q_i \in Q_T$. A workable storage assignment requires that a product m can only be stored in a location k if their temperature zone indices match (i.e., $Q_i = Q_k$). For the case study in this paper, we consider three distinct zones, thus $Q_T = \{1, 2, 3\}$, corresponding to the ambient, refrigerated, and frozen zones, respectively.

The storage capacity C_k depends on the zone, as defined by

$$C_k = \begin{cases} 1, & Q_k \in \{1, 2\} \\ 5, & Q_k = 3 \end{cases} \tag{1}$$

Each product $i \in I$ has a specific temperature requirement $Q_i \in \{1, 2, 3\}$, a cumulative in-warehouse time of τ_i, and a designated central storage coordinate (A_i, B_i, C_i).

The set of orders is denoted by $J = \{1, \ldots, n\}$, where each order $j \in J$ comprises a subset of products $I_j \in I$. The set of pickers is denoted by $|P|$. Each order must be assigned to a single picker (where y_j^p represents this assignment), and workloads should be balanced. All picking tours start from and return to the packing station, S_0.

To calculate the picking path cost, the model first maps the three-dimensional storage location index (r_k, c_k, l_k) to two-dimensional planar coordinates (x_k, y_k) that reflect the physical layout of the warehouse aisles. The planar coordinates are functions of the row index r_k and column index c_k, while the level index l_k does not affect the travel distance in this picker-to-parts model, as horizontal travel along and between aisles makes up the primary component of the total path. The orthogonal grid layout of warehouse aisles renders Euclidean distance an inaccurate measure of a picker's actual travel path. This study employs the Manhattan distance to calculate the travel distance between any two storage locations. For two locations, k and k', with planar coordinates (x_k, y_k) and $(x_{k'}, y_{k'})$, respectively, the Manhattan distance $d(k, k')$ is defined as

$$d(k, k') = |x_k - x_{k'}| + |y_k - y_{k'}| \tag{2}$$

The model assumes a constant travel speed for pickers; therefore, the picking path cost (i.e., picking time) is directly proportional to the total travel distance. Operationally, the model supports order batching, allowing a picker to merge items from multiple orders into a single picking tour, with the total path length minimized by optimizing the pick sequence.

As for energy cost, each location k is associated with an energy cost parameter E_k. The value is highest for the frozen zone, moderate for the refrigerated zone, and zero for the ambient zone.

The symbols and parameters used in the mathematical model are presented in Table 1.

Table 1. Parameters used in the mathematical model.

Notation	Description
P	Set of all available pickers.
K	Set of all storage locations.
$I_j \in I$	Subset of products in order j.
J	Set of all orders.
Y	Decision matrix for the assignment of orders to pickers.
m	Total number of product types.
n	Total number of orders.
U_i	Total demand for product i.
C_k	Capacity of storage location k.
Q_i, Q_k	Temperature requirement of product i and temperature attribute of location k.
N_r, N_c, N_l	Parameters defining the number of rows, columns, and levels in the warehouse grid, respectively.
$\|P\|$	Total number of pickers.
S_0	The packing station, serving as the start and end point for all picking tours.
$p_i = (X_i, Y_i, Z_i)$	Coordinates of the storage location for product i.
$X_i = (A_i, B_i, C_i)$	Coordinates of the designated central location for product i.
α_i	Coefficient for the cumulative in-warehouse time of product i.
E_k	Unit cost of energy and carbon emissions for location k.
R_j	Length of the shortest picking tour for order j, starting and ending at S_0.
c_{path}	Cost coefficient per unit of travel distance for the picking path.
c_{layout}	Cost coefficient for the deviation of a product's storage location from its central location.
c_{FIFO}	Penalty cost coefficient for violating FIFO principles.
$c_{schedule}$	Cost coefficient associated with picker workload imbalance.

The decision variables for the mathematical model are presented in Table 2.

Table 2. Decision variables used in the mathematical model.

Notation	Description
$x_{ik} \in N^*$	The quantity of product i stored in location k (N^* denotes the set of non-negative integers $\{0, 1, 2, \ldots\}$).
$y_{ik} \in \{0, 1\}$	A binary variable that is 1 if product i is stored in location k ($x_{ik} > 0$), and 0 otherwise.
$y_j^p \in \{0, 1\}$	A binary variable that is 1 if order j is assigned to picker p, and 0 otherwise.

The joint optimization model for fresh product storage allocation and picking efficiency proposed seeks to minimize a single, comprehensive objective function, which is formulated by aggregating five key cost components using a weighted sum method. These five cost components are picking path cost (C_{path}), storage layout deviation cost (C_{layout}), FIFO penalty cost (C_{FIFO}), energy and carbon emission cost (C_{energy}), and picker scheduling cost ($C_{schedule}$). The total objective function of the model is formulated as follows:

$$\min F(P, x, y, Y) = C_{path} + C_{layout} + C_{FIFO} + C_{energy} + C_{schedule} \tag{3}$$

Each cost component is detailed below:

(1) Picking Path Cost
For each order $j \in J$, let L_j be the set of storage locations to be visited as follows:

$$L_j = \{(r_k, c_k, l_k) : \exists i \in I_j, x_{ik} > 0\} \tag{4}$$

Let R_j be the length of the shortest tour for picking order j, starting from the depot S_0, visiting all locations in L_j and returning to S_0. The total picking path cost objective is then

$$C_{path} = c_{path} \sum_{j \in J} R_j(P, x, y) \tag{5}$$

C_{path} represents the sum of travel distance costs for all orders.

(2) Storage Layout Deviation Cost
To penalize the deviation of product storage locations from their designated central locations, the following objective function is formulated as follows:

$$C_{layout} = c_{layout} \sum_{i \in I} d(p_i, X_i) \tag{6}$$

C_{layout} represents the total cost incurred from the sum of distances between each product's assigned location and its central location.

(3) FIFO Penalty Cost
Upon receiving products, pickers scan barcodes to log entry information, including arrival time. Given the perishability and short shelf-life of fresh products, we analyze each product's dwell time τ_i and introduce a penalty term to ensure that items with earlier arrival times are stored closer to the depot:

$$C_{FIFO} = c_{FIFO} \sum_{i \in I} \alpha_i d(p_i, S_0) \tag{7}$$

Here, the coefficient a_i is directly proportional to the product's cumulative time in storage (a longer storage time results in a higher a_i), prioritizing it for picking.

(4) Energy and Carbon Emission Cost

To account for the operational energy consumption and carbon emissions of the temperature-controlled zones, an energy cost E_k is assigned to each location k. The total storage energy cost is

$$C_{energy} = \sum_{i \in I} \sum_{k \in K} x_{ik} E_k \tag{8}$$

(5) Picker Scheduling Cost

To quantify and penalize workload imbalance among pickers, the model incorporates a scheduling cost objective based on the variance of individual workloads. For each picker $p \in P$, let J_p denote the subset of orders assigned to them. This set is formally defined based on the decision variable y_j^p as $J_p = \{j \in J \mid y_j^P = 1\}$. The workload for picker p, denoted as W_p, is the sum of the travel distances for all orders in their assigned set: $W_p = \sum_{j \in j^p} R_j(P, x, y)$,where J_p is the set of orders assigned to picker p.

First, the average workload $\bar{W}$ across all pickers is calculated as follows:

$$\bar{W} = \frac{1}{|P|} \sum_{p \in P} W_p = \frac{1}{|P|} \sum_{j \in J} R_j(P, x, y) \tag{9}$$

The scheduling cost objective is the variance of the workloads among pickers, expressed as

$$C_{schedule}(Y) = c_{schedule} * \sum_{p \in P} (W_p - \bar{W})^2 \tag{10}$$

This objective function drives the optimization toward a fair distribution of tasks among all pickers by minimizing the workload variance, achieving a balanced workload.

For in the multi-objective optimization model constructed in this study, the determination of the weight coefficients (C_{path}, C_{layout}, C_{FIFO}, C_{energy}, and $C_{schedule}$) of each cost subcomponent is a key aspect that affects the decision preference of the final optimization scheme. To ensure the interpretability of weight assignment and flexibility in real-world applications, this study used the Analytic Hierarchy Process (AHP) to determine these weights. The reason for choosing AHP as the weight determination method is based on the following considerations: First, decision-making in warehouse management essentially involves trade-offs among multiple conflicting objectives (e.g., efficiency, cost, and quality), which highly depends on the experience and strategic preferences of the decision-makers (e.g., warehouse managers). Unlike purely objective methods, such as the entropy weight method that determines weights based solely on data dispersion, AHP is good at systematically converting such qualitative and subjective judgments into quantitative weights through pairwise comparison, and its structured hierarchical model is a natural fit with the "general objective-sub-objective" structure of this problem. Second, compared with other multi-criteria decision-making (MCDM) methods, AHP provides a mature Consistency Ratio (CR), which can quantify the reliability of the judgment logic, thus guaranteeing the reasonableness and robustness of weight allocation.

The application of AHP hierarchical analysis in this study follows the following steps: first, construct a hierarchical model. We decompose the optimization problem into a Goal Layer, i.e., "minimize the total operating cost", and a Criteria Layer, i.e., five specific cost subcomponents: C_{path}, C_{layout}, C_{FIFO}, C_{energy}, and $C_{schedule}$. Second, the judgment matrix is constructed. We invite domain experts to compare each cost objective in the guideline layer two by two based on Saaty's 1-9 scale to construct a judgment matrix A, so as to quantify the relative importance among the objectives. Third, calculate the weight vector and consistency test. By calculating the maximum eigenvalue ($\lambda_{\max}$) and its corresponding normalized eigenvector of the judgment matrix A, the weight vector of each cost item is obtained $\boldsymbol{w}$. Subsequently, the Consistency Index is calculated by the formula $CI = (\lambda_{\max} - n)/(n-1)$,

and the Consistency Ratio is calculated by the formula $CR = CI/RI$, where n denotes the order of the matrix and RI represents the average random Consistency Index. Only when $CR < 0.1$ is the judgment matrix deemed to possess acceptable consistency, and its corresponding weight vector is valid.

It is worth emphasizing that the target preferences in this model are user-selectable rather than fixed. The specific weights used in the subsequent experiments. are derived from a judgment matrix constructed based on a typical operational scenario in which "order fulfillment timeliness" is the primary goal. Warehouse managers can reconstruct the judgment matrix according to specific business environments and strategic priorities to generate a set of customized weights that meet current needs. For example, the weight of C_{energy} can be significantly increased when energy prices are high, and the weight of C_{path} can be further increased during the response to large-scale promotional activities.

Based on the above, the specific process of weight assignment using hierarchical analysis in this study is as follows: The initial step involves constructing the hierarchical model. The hierarchical model for this study comprises two levels: a Goal Layer and a Criteria Layer. The Goal Layer is defined as the optimization of total operational costs for fresh product storage allocation and picking, while the Criteria Layer consists of the five cost objectives under evaluation:C_{path}, C_{layout}, C_{FIFO}, C_{energy}, and $C_{schedule}$.

The core of the AHP method is the use of pair-wise comparisons to transform qualitative judgments from decision-makers regarding the relative importance of criteria into a quantitative format [26]. This study employs the Saaty 1–9 scale to construct the judgment matrix, with the scale's definitions detailed in Table 3.

Table 3. AHP scale definitions.

Scale	Definitions
1	Equal importance
3	Moderate importance
5	Strong importance
7	Demonstrated importance
9	Extreme importance
2, 4, 6, 8	The median value of the two adjacent judgments mentioned above.
Reciprocals of above	If j is more important than i, take the reciprocal of the corresponding scale.

Based on the high-timeliness requirements and the focus on major operational costs inherent in the fresh e-commerce case study, we established the following prioritization logic based on the data collected from managers:

First, the picking path cost (C_{path}) is identified as the most critical factor, as it directly dictates order fulfillment speed, which is central to ensuring rapid response times for fresh product orders. Second, the storage layout cost (C_{layout}) is a strategic factor whose importance is second only to path cost; it supports rapid picking by optimizing the storage locations of high-turnover items. Subsequently, while product quality is vital, the substantial and continuous energy cost (C_{energy}) associated with refrigerated and frozen zones represents a significant financial expenditure that requires stringent control. Under financial constraints, its priority is ranked higher than the FIFO penalty cost (C_{FIFO}). Finally, the picker scheduling cost ($C_{schedule}$), which aims to balance internal workloads, is considered a secondary optimization objective relative to customer-facing efficiency metrics and major cost-control items.

Based on this analysis, the resulting judgment matrix A is presented in Table 4.

Table 4. Judgment matrix A.

Objectives	C_{path}	C_{layout}	C_{FIFO}	C_{energy}	$C_{schedule}$
C_{path}	1	3	5	7	9
C_{layout}	$\frac{1}{3}$	1	3	4	7
C_{FIFO}	$\frac{1}{5}$	$\frac{1}{3}$	1	3	5
C_{energy}	$\frac{1}{7}$	$\frac{1}{4}$	$\frac{1}{3}$	1	3
$C_{schedule}$	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	1

Subsequently, the weight vector is calculated, and a consistency check is performed. The calculation resulted in a Consistency Ratio (CR) of 0.0919, which is less than the 0.1 threshold, thus passing the consistency test.

Therefore, the AHP yields the following weights for the cost components: (1) picking path cost ($C_{path} = 0.51062$); (2) storage layout deviation cost ($C_{layout} = 0.23549$); (3) energy and carbon emission cost ($C_{energy} = 0.12017$); (4) FIFO penalty cost ($C_{FIFO} = 0.09835$); and (5) picker scheduling cost ($C_{schedule} = 0.03537$).

To investigate the impact of different managerial strategies on the solution, this study conducted a trade-off analysis. We designed three realistic managerial scenarios: (1) Balanced Approach, representing the operational model with a comprehensive consideration of all costs as previously described; (2) Cost-Centric Strategy, which prioritizes the control of key expenditures such as storage layout and energy consumption; and (3) Efficiency-Driven Strategy, which designates order fulfillment timeliness as the paramount optimization goal.

For each scenario, we constructed a corresponding AHP judgment matrix and recalculated the weight vector. Subsequently, the proposed PS-GSA algorithm was used for solving. Table 5 details the final values of each cost component and the changes in the total weighted cost under different weight configurations.

Analysis of the results in Table 5 reveals that under the "Cost-Centric Strategy," although the energy cost (C_{energy}) and layout deviation cost (C_{layout}) were reduced by 14.4% and 4.4%, respectively, this led to a 32.8% increase in the picking path cost (C_{path}). This exposes a phenomenon: to concentrate products in low-energy-consumption zones or ideal storage locations, pickers are compelled to travel longer distances to fulfill orders. Conversely, under the "Efficiency-Driven Strategy," while the FIFO penalty cost (C_{FIFO}) was reduced by 18.3%, the energy cost (C_{energy}) and layout deviation cost (C_{layout}) increased by 22.4% and 30.1%, respectively. This shows that to achieve the shortest picking paths, the algorithm stores high-frequency items in locations that are close to the depot but may belong to high-energy-consumption temperature zones, causing an increase in other costs.

Table 5. Results of the trade-off analysis under different managerial scenarios.

Scenario	Objective	Weight Vector	Cost Component Value	Total Weighted Cost
Balanced Approach (Baseline)	C_{path}	0.5106	652.0	649.67
	C_{layout}	0.2355	666.0	
	C_{FIFO}	0.0984	628.4	
	C_{energy}	0.1202	700.0	
	$C_{schedule}$	0.0353	394.8	
Cost-Centric Strategy	C_{path}	0.1144	866.0 (+32.8%)	636.01
	C_{layout}	0.2633	637.0 (−4.4%)	
	C_{FIFO}	0.0840	644.3 (+2.5%)	
	C_{energy}	0.5029	599.4 (−14.4%)	
	$C_{schedule}$	0.0354	385.97 (−2.2%)	
Efficiency-Driven Strategy	C_{path}	0.4535	807.0 (+23.8%)	663.08
	C_{layout}	0.0678	866.5 (+30.1%)	
	C_{FIFO}	0.2981	513.5 (−18.3%)	
	C_{energy}	0.0359	856.7 (+22.4%)	
	$C_{schedule}$	0.1447	376.8 (−4.6%)	

Note: The percentage next to the cost component values shows the rate of change compared to the baseline scenario. The total weighted cost is calculated using the respective weight vector of each scenario.

To test the robustness of the baseline solution, this study conducted a sensitivity analysis. We employed the one-at-a-time perturbation method, sequentially perturbing the weights of the five cost objectives by ±5% and ±10%. Concurrently, the other weights were proportionally adjusted to ensure their sum remained 1, and the model's total cost was then recalculated. The analysis results are summarized in Table 6.

Table 6. Results of the sensitivity analysis on objective weights.

Perturbed Objective	Baseline Weight	Perturbation	New Total Cost	% Change from Baseline
C_{path}	0.5106	−10%	656.53	+0.69%
		−5%	654.81	+0.43%
		+5%	649.42	−0.40%
		+10%	646.91	−0.78%
C_{layout}	0.2355	−10%	668.82	+0.42%
		−5%	667.44	+0.22%
		+5%	664.70	−0.20%
		+10%	663.38	−0.39%
C_{FIFO}	0.0984	−10%	632.12	+0.59%
		−5%	630.59	+0.35%
		+5%	626.52	−0.30%
		+10%	624.99	−0.54%
C_{energy}	0.1202	−10%	704.51	+0.64%
		−5%	702.78	+0.40%
		+5%	697.34	−0.38%
		+10%	695.65	−0.62%
$C_{schedule}$	0.0353	−10%	398.40	+0.91%
		−5%	396.52	+0.44%
		+5%	392.88	−0.49%
		+10%	390.98	−0.97%

The results in Table 6 strongly show the robustness of the baseline solution. The fluctuation in each weight cost consistently remained within a minimal range of ±1%. This shows that the model's optimization results are not sensitive to minor deviations that may exist in the AHP judgments, rendering the decision reliable.

The following is a description of the mathematical model. The model is subject to the following constraints:

(1) Storage and Capacity Constraints

The total quantity of a product i stored across all locations must equal its total demand U_i. This is expressed as

$$\sum_{k \in K} x_{ik} = U_i, \quad \forall i \in I \tag{11}$$

The quantity of product i in location k cannot exceed its capacity. Except for the frozen zone, each location can hold only one product package. A single location cannot be used to store different product types:

$$x_{ik} \leq C_k y_{ik}, \quad \forall i \in I, \forall k \in K \tag{12}$$

$$\sum_{i=1}^{N} y_{ik} \leq 1, \quad \forall k \in K \tag{13}$$

Finally, a temperature matching constraint ensures that products are stored only in locations that meet their temperature requirements. If $Q_i \neq Q_k$, then

$$x_{ik} = 0, \quad \forall i \in I, \forall k \in K \tag{14}$$

(2) Routing and Scheduling Constraints

The picking path for each order, $R_j(P, x, y)$, is determined by the coordinates of the locations of its constituent products and must form a tour starting and ending at the depot S_0. Each order must be assigned to exactly one picker:

$$\sum_{p=1}^{P} y_j^p = 1, \quad \forall j \in J \tag{15}$$

(3) Product Odor Compatibility Constraint

In fresh product storage allocation, the olfactory properties of different items can significantly affect product quality and the picking environment. Products with strong odors in particular may adversely affect adjacent odor-sensitive or easily contaminated items, leading to quality degradation and a negative customer experience. To address this factor during layout optimization, we introduce a constraint based on product odor compatibility. We assign an odor index O_i ($0 \leq O_i \leq 10$) to each product i, where higher values signify stronger odors and a greater risk of cross-contamination. To prevent contamination, we define a minimum required separation distance $D_{\min}$ as a function of the odor indices of two products, i and j:

$$D_{\min}(O_i, O_j) = \gamma(O_i, O_j) + \delta \tag{16}$$

The coefficients γ and δ are empirical values determined through multiple rounds of Delphi method interviews with warehouse management experts. For any pair of products i and j ($i \neq j$) whose combined odor indices exceed a predefined thresh-

old $T_O = 5$, their assigned storage locations must be separated by at least this minimum distance:

$$d(p_i, p_j) \geq D_{\min}(O_i, O_j) \tag{17}$$

(4) Temperature Zone Matching Constraint

To ensure product integrity, it is imperative that each product is assigned only to storage locations that match its specific temperature requirements. For example, ambient products must not be placed in refrigerated or frozen zones. This is enforced by the following constraint. The formulation stipulates that for any product i, the sum of its assignments to all locations k whose temperature zone Q_k does not match the product's required zone Q_i must be zero. This effectively forces the decision variable x_{ik} to be zero for any mismatched pair, prohibiting such assignments.

$$\sum_{k \in K, Q_i \neq Q_k} x_{ik} = 0, \quad \forall i \in I \tag{18}$$

In summary, the mathematical model for the joint optimization of fresh product storage allocation and picking efficiency proposed in this paper is formulated as follows:

$$\begin{aligned} \min F(P, x, y, Y) = {} & c_{path} \sum_{j \in J} R_j(P, x, y) + c_{layout} \sum_{i \in I} d(p_i, X_i) \\ & + c_{FIFO} \sum_{i \in I} \alpha_i d(p_i, S_0) + \sum_{i \in I} \sum_{k \in K} x_{ik} E_k + c_{schedule} \sum_{p \in P} (W_p - \bar{W})^2 \end{aligned} \tag{19}$$

s.t.

$$\sum_{k \in K} x_{ik} = U_i, \qquad \forall i \in I$$

$$x_{ik} \leq C y_{ik}, \qquad \forall i \in I, \forall k \in K$$

$$\sum_{i \in I} y_{ik} \leq 1, \qquad \forall k \in K$$

$$\sum_{p \in P} y_j^p = 1, \qquad \forall j \in J$$

$$d(p_i, p_j) \geq D_{\min}(O_i, O_j)$$

$$\sum_{k \in K, Q_k \neq Q_i} x_{ik} = 0, \qquad \forall i \in I$$

$$x_{ik} \in N^*, y_{ik} \in \{0, 1\}, y_j^p \in \{0, 1\}$$

Although the weighted sum method is straightforward to implement, it is well known for its potential inability to find solutions in non-convex regions of the Pareto front [27]. As this problem requires a single, actionable solution and the weights can reflect managerial preferences, the weighted sum approach is deemed appropriate for this context.

3. Solution Methodology

The problem addressed in this study is the integrated Storage Location Assignment and Picker Routing Problem for fresh products, which is a well-established NP-hard combinatorial optimization problem. Its complexity arises from the tight coupling of two interdependent subproblems: the assignment of products to storage locations and the determination of the shortest picking routes for fulfilling customer orders. The latter sub-

problem is analogous to the computationally intractable Traveling Salesperson Problem (TSP), rendering the overall integrated problem NP-hard. To validate the efficacy of the proposed algorithm, we selected several established algorithms as performance benchmarks: the Genetic Algorithm (GA), Simulated Annealing (SA), discrete Particle Swarm Optimization (PSO), and a hybrid Genetic-Simulated Annealing (GSA) algorithm. These conventional algorithms frequently encounter challenges such as premature convergence to local optima and inadequate convergence rates when applied to high-dimensional, complex problems. To overcome these limitations, we designed a Particle Swarm-guided hybrid Genetic-Simulated Annealing (PS-GSA) algorithm, which achieves a dynamic balance between global exploration and local exploitation [28].

The proposed PS-GSA differs from conventional hybrids through a deep procedural integration of search philosophies into a unified, discrete framework, rather than a superficial combination of operators. Unlike continuous PSO engines requiring a separate decoding step, PS-GSA's core is natively discrete, redefining a particle's "velocity" as a sequence of swaps (SOS)—a list of concrete permutation operations that navigate the solution space directly. This is embedded within a synergistic, ordered pipeline where a global move guided by the SOS-based PSO is intensively refined by a constraint-aware Variable Neighborhood Search (VNS). The resulting solution is then evaluated by the Metropolis acceptance criterion of Simulated Annealing (SA) to probabilistically escape local optima.

To ensure the proposed PS-GSA algorithm exhibits optimal performance, a series of comprehensive sensitivity analyses were conducted on its key parameters. Through computational experiments across multiple test instances, we evaluated the impact of different parameter configurations on both solution quality and convergence efficiency. The parameter values presented in Table 7 result from this rigorous calibration process, representing the configuration that yields the best performance for the problem context.

Table 7. Algorithm parameters.

Category	Parameter	Notation	GA	SA	HPSO	GSA	PS-GSA
General Settings	Population Size	N_{pop}	100	-	100	100	100
	Max Iterations	M	200	200	200	200	200
Genetic Operators	Crossover Probability	P_c	0.9	-	0.9	0.9	-
	Mutation Probability	P_m	0.05	-	0.05	0.05	Adaptive
	Tournament Size	-	-	-	-	-	2
	Elite Retention	ρ_{elite}	-	-	-	20%	20%
Annealing Strategy	Initial Temperature	T_0	-	1000	-	1000	1000
	Cooling Rate	α	-	0.98	-	0.98	0.98
	Final Temperature	T_1	-	1.0×10^{-20}	-	1.0×10^{-20}	1.0×10^{-20}
	Markov Chain Length	L	-	10	-	-	-
PSO Dynamics	Inertia Weight	w	-	-	0.9→0.4	-	0.9→0.2 (Dynamic)
	Cognitive Factor	c_1	-	-	2	-	2.5→1.875 (Dynamic)
	Social Factor	c_2	-	-	2	-	1.25→2.5 (Dynamic)
PS-GSA Hybrid Strategy	Local Search Probability	P_{ls}	-	-	-	-	0.55 (Adaptive)
	Stagnation Threshold	S_{thresh}	-	-	-	-	25 (Iterations)
	Elite Enhancement Freq.	f_{alite}	-	-	-	-	5 (Iterations)

To ensure the robustness and optimal performance of the proposed PS-GSA algorithm, a comprehensive sensitivity analysis was conducted on its key parameters that critically influence the search behavior. Through extensive preliminary computational experiments, we systematically evaluated the impact of various parameter combinations on the solution quality and convergence speed. The analysis revealed that the algorithm achieves its best overall performance on the test instances when the crossover probability is set to 0.9, the mutation probability is set to 0.05, the elite retention proportion is configured to 0.25, and the local search probability is determined to be 0.55. For PSO Dynamics, this strategy is designed to foster a dynamic transition from global exploration to local exploitation [29,30] .

In the early stages, a larger inertia weight $w(t)$ and cognitive factor $c_1(t)$ encourage diverse searching [31]. In contrast, the social factor $c_2(t)$ is intentionally increased over iterations to strengthen social learning, guiding the swarm toward the global optimum in the later stages and promoting convergence.

The core framework of the PS-GSA algorithm is founded on population-based iterative optimization, yet it integrates various optimization mechanisms, including Simulated Annealing and Particle Swarm strategies, at critical junctures. Figure 2 reveals the process of the algorithm.

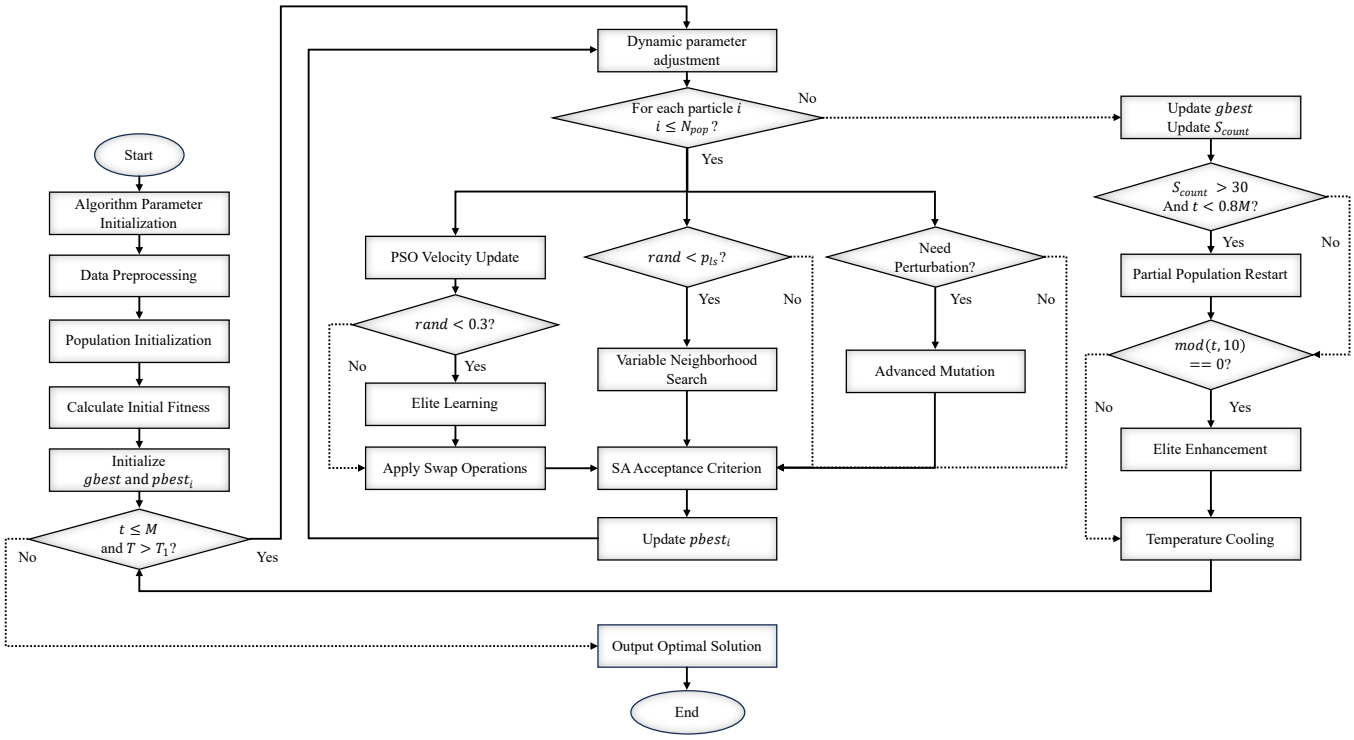


Figure 2. PS-GSA flowchart.

In order to clearly illustrate the implementation details of the Particle Swarm-guided hybrid Genetic–Simulated Annealing algorithm (PS-GSA) proposed in this study, its detailed algorithmic step-by-step description with pseudo-code is provided below. Unlike traditional meta-heuristic algorithms, PS-GSA organically integrates the global guidance ability of discrete Particle Swarm Optimization (PSO), the local mining ability of Variable Neighborhood Search (VNS), and the probabilistic jump-out mechanism of Simulated Annealing (SA).

Step 1. Population initialization. A hybrid strategy is used to generate an initial population P_0 of size N_{pop}. A total of 20% of the individuals are generated by a greedy heuristic rule to ensure the quality of the initial solution; the remaining 80% of the individuals are generated completely randomly to ensure the diversity of the population.

Step 2. Calculation of fitness. Based on the integrated cost objective function constructed in Section 2, the fitness value of each individual in the population was calculated $f(X_i)$.

Step 3. Initialize individual and global optima. For each particle i, set its initial position to the individual historical optimal position X_{pbest}, i, and its fitness to the individual optimal fitness f_{pbest}, i. Find the individual with the optimal fitness from the whole population, and record its position and fitness as the global optimal

X_{gbest} and f_{gbest}. Initialize the current temperature of the Simulated Annealing, T, and the stagnation counter, S_{count}.

Step 4. Main loop. When the number of iterations *iter* does not reach the maximum number of iterations M, repeat steps 5 through 10.

Step 5. Dynamic parameter adjustment. According to the current iteration number t and the maximum iteration number M, the inertia weight $w(t)$, the cognitive factor $c_1(t)$, and the social factor $c_2(t)$ of the PSO are dynamically updated based on Equation (19).

Step 6. Particle Updates. An enhanced particle update process is performed for each particle X_i in the population (see Algorithm 2 for details):

(1) Velocity update: Generate a discrete "exchange sequence" of particle velocities by comparing the difference between the current solution and the individual optimal, global optimal, and elite solutions.
(2) Position update: Apply the exchange sequence to the current solution to generate a candidate solution X_i'.
(3) Local enhancement: Perform Variable Neighborhood Search (VNS) on the candidate solution X_i' with probability P_{ls} to get the enhanced solution X_i''.
(4) Acceptance decision: Based on the Metropolis acceptance criterion, decide with a certain probability whether to accept the augmented solution x_i'' as the new position of the particle i or not.
(5) Individual optimal update: If the fitness of the new position is better than the individual historical optimal fitness of the particle $(f_{pbest},\ i)$, then update $(X_{pbest},\ i)$ and $(f_{pbest},\ i)$.

Step 7. Global Optimization Update. After all particles have completed the update, re-evaluate the entire population and if a new global optimum solution is found, update X_{gbest} and f_{gbest} and zero out the stagnation counter S_{count}; otherwise, S_{count} is added by one.

Step 8. Stagnation restarts with elite enhancement. If S_{count} exceeds the threshold S_{thresh}, a partial population restart mechanism is triggered as follows: the top 20% of elite individuals are kept and the bottom 80% are re-initialized randomly to jump out of the local optimum. Meanwhile, every f_{elite} generations (e.g., 10 generations), a deep local search is executed on the current global optimal solution X_{gbest} to further extract the optimal solution.

Step 9. Temperature attenuation. Attenuation of the temperature T by the cooling factor α.

Step 10. The algorithm terminates. If $t \geq M$, the algorithm terminates and outputs the global optimal solution X_{gbest} found and its fitness value f_{gbest}; otherwise, go to step 4.

For ease of reproduction and understanding, we formalize the above steps into pseudocode. The overall framework of the algorithm is presented in Algorithm 1 and the particle update mechanism is detailed in Algorithm 2.

The input to Algorithm 1 includes objective function $f(x)$, population size N_{pop}, maximum number of iterations M, Simulated Annealing parameters (T_0, α), Particle Swarm parameters $(w_{\max}, w_{\min}, (c_1, \max), (c_2, \max))$, local search probability p_{ls}, and stalled restart threshold S_{thresh}. The output comprises the global optimal solution X_{gbest} and its fitness value f_{gbest}.

Algorithm 1 The overall PS-GSA framework.

1: //1. Initialization
2: $P_{pop} \leftarrow$ Intelligent_Initialization(N_{pop})
3: **for** $i = 1$ to N_{pop} **do**
4: $f_i \leftarrow f(X_i)$
5: $X_{pbest,\,i} \leftarrow X_i$; $f_{pbest,\,i} \leftarrow f_i$ ▷ Step 3
6: **end for**
7: $f_{gbest} \leftarrow \min(f_1, \ldots, f_{N_{pop}})$; $X_{gbest} \leftarrow$ corresponding to the optimal individual; ▷ Step 3
8: $T' \leftarrow T_0$
9: //2. Main loop
10: **for** $t = 1$ to M **do** ▷ Step 4
11: Update $w(t), c_1(t), c_2(t)$; ▷ Step 5
12: **for** $i = 1$ to N_{pop} **do**
13: $X_i,\ f_i,\ (X_{pbest,\,i}),\ (f_{pbest,\,i}) \leftarrow$ Enhanced_Particle_Update$(X_i,\ f_i,\ X_{pbest,\,i},\ f_{pbest,\,i},\ X_{gbest},\ T,\ p_{ls})$; ▷ Step 6
14: **end for**
15: //3. Updating the global optimum with adaptive tuning
16: $f_{\text{current best}} \leftarrow \min(f_1, \ldots, f_{N_{pop}})$;
17: **if** $f_{\text{current best}} < f_{gbest}$ **then** ▷ Step 7
18: $f_{gbest} \leftarrow f_{\text{current best}}$
19: $S_{\text{count}} \leftarrow 0$
20: **else**
21: $S_{\text{count}} \leftarrow S_{\text{count}} + 1$
22: **end if**
23: **if** $S_{\text{count}} > S_{\text{thresh}}$ and $t < 0.8 * M$ **then** ▷ Step 8
24: Perform_Partial_Restart$(P, f, X_{pbest}, f_{pbest})$; ▷ Retain elites, reset most populations
25: $s_{\text{count}} \leftarrow 0$
26: **end if**
27: **if** $\text{mod}(t, 10) = 0$ **then** ▷ Step 8
28: $X_{gbest} \leftarrow$ Intensive_Local_Search(X_{gbest})
29: $f_{gbest} \leftarrow f(X_{gbest})$
30: **end if**
31: //4. Temperature decay
32: $T' \leftarrow T \cdot \alpha$
33: **end for**
34: **return** $X_{gbest},\ f_{gbest}$;

The input to Algorithm 2 includes current solution (X_i, f_i), individual optimal solution $(X_{\text{pbest}}, i), (f_{\text{pbest}}, i)$, global optimum solution X_{gbest}, current temperature T, and local search probability P_{ls}. The output includes updated solutions (X_i, f_i), and updated individual optimal solutions $(X_{\text{pbest}}, i), (f_{\text{pbest},}, i)$.

The above pseudo-code shows the core design ideas of PS-GSA. Algorithm 1 embodies the evolutionary flow of the algorithm, which is notable for the population stagnation judgement and restart mechanism in lines 23–26, and the elite solution depth enhancement mechanism in lines 27–30. These strategies enhance the algorithm's ability to jump out of the local optimum and are key to the robustness of the algorithm. Algorithm 2 adapts PSO for solving continuous space problems to the permutation optimization problem in this study. The key innovation is to redefine the velocity vector of the traditional PSO as a sequence of swaps, which guides the particles to perform position updates. Ultimately, the acceptance of a new solution is determined by the Metropolis criterion, which not only accepts the better solution unconditionally but also accepts the suboptimal solution with a certain probability, which decreases as the "temperature" T decreases. This mechanism gives the algorithm the ability to escape from local extremes in the early stage of the search

and gradually converge to the optimal solution region in the latter stage, which is the core mechanism to ensure the convergence performance of the algorithm.

Algorithm 2 Particle Update Process.

```
//1. Updating Discrete velocity (generation of exchange sequences)
V_i ← Generate_Swaps(X_i, X_pbest,i, X_gbest, w(t), c_1(t), c_2(t))
//2. Updating location
X' * i ← Apply_Swaps(X_i, V_i)
//3. Local Search Enhancement
if rand() < p_ls then
    X'' * i ← Variable_Neighborhood_Search(X' * i)
else
    X'' * i ← X' * i
end if
f'' * i ← f(X'' * i)
//4. Metropolis acceptance criteria
Δf ← f'' * i − f_i
if Δf < 0 or rand() < exp(−Δf/T) then
    X_i ← X'' * i
end if
//5. Updating the individual optimum
if f_i < f_pbest then
    (X_pbest,i) ← X_i, (f_pbest,i) ← f_i
end if
return X_i, f_i, (X_pbest,i), (f_pbest,i)
```

Overall, Algorithms 1 and 2 show the macro-architecture and core execution process of PS-GSA as a whole. In order to further explain the design details and theoretical basis of the key modules, the following subsections will provide a detailed mathematical elaboration of the dynamic parameter control strategy, the neighborhood generation mechanism based on the discrete PSO engine, the Variable Neighborhood Search (VNS) used to deepen the local search, as well as the population restarting and elite reinforcement strategies.

(1) Dynamic and Adaptive Parameter Control

To accommodate the varying requirements of the search process at different stages, key parameters within the PS-GSA are dynamically adjusted [32,33]. The inertia weight $w(t)$, cognitive factor $c_1(t)$, and social factor $c_2(t)$ are varied as a function of the iteration count.

$$\begin{aligned} w(t) &= w_{\max} - (w_{\max} - w_{\min}) \cdot \frac{t}{M} \\ c_1(t) &= c_{1,\max} \cdot \left(1 - 0.5 \cdot \frac{t}{M}\right) \\ c_2(t) &= c_{2,\max} \cdot \left(0.5 + 0.5 \cdot \frac{t}{M}\right) \end{aligned} \tag{20}$$

Here, w_{max} and w_{min} represent the initial and final values of the inertia weight, respectively. This strategy enables the algorithm to favor global exploration in the early stages by employing a larger inertia weight $w(t)$ and cognitive factor $c_1(t)$, while shifting focus toward convergence on the global optimum in later stages with a smaller inertia weight and a larger social factor $c_2(t)$.

(2) Neighborhood Generation via Discrete Particle Swarm Optimization

The primary driver of the PS-GSA is a discrete PSO engine specifically adapted for permutation-based problems [34]. In this framework, a particle's "velocity," $V_i(t)$, is defined not as a continuous vector but as a sequence of swaps (SOS). The velocity

update mechanism integrates four components: inertia, personal cognitive influence, social learning, and elite-guided learning [35,36]:

$$V_i(t+1) = \omega(t) \otimes V_i(t) \oplus c_1(t) \otimes (pbest_i \ominus X_i(t)) \\ \oplus c_2(t) \otimes (gbest \ominus X_i(t)) \oplus \eta \otimes (X_{\text{elite}} \ominus X_i(t)) \tag{21}$$

where

- $pbest_i$ is the personal best position found by particle i, and $gbest$ is the global best position found by the entire population.
- X_{elite} is a superior individual selected randomly from an elite subset of the population.
- The $\ominus$ operator compares two permutations and generates a minimal sequence of swaps required to transform one into the other.
- The $\otimes$ operator denotes a probabilistic sampling of a swap sequence, thereby controlling the influence of each term.
- The $\oplus$ operator represents the application of the final swap sequence (the velocity) to a particle's current position, thereby generating a new candidate solution.
- η is the elite learning rate.

A new velocity vector $V_i(t+1)$ is generated and then applied to the current position $X_i(t)$ to produce a candidate solution $X_i'(t)$:

$$X_i'(t) = X_i(t) \oplus V_i(t+1) \tag{22}$$

To operationalize the PSO framework for this permutation-based problem, the concepts of "velocity" and "position update" are redefined in a discrete context. A particle's velocity, $V_i(t+1)$, is not a continuous vector but is represented as a sequence of swaps (SOS)—an ordered list of index pairs, where each pair signifies a swap operation to be performed on the chromosome (the solution permutation).

The generation of this SOS, represented abstractly by the $\ominus$ operator in Equation (21), is a critical step. For instance, to compute the cognitive component ($pbest_i \ominus X_i(t)$), the algorithm compares the current solution $X_i(t)$ with the particle's personal best solution $pbest_i$. It identifies all indices where the two permutations differ and then constructs a sequence of workable swaps that moves $X_i(t)$ closer to $pbest_i$. A key innovation here is the constraint-aware nature of this process: a swap between two indices is only considered valid if it preserves the feasibility of the solution, specifically by ensuring that both affected products remain in their required temperature zones after the exchange. It is important to note that this process generates a feasible sequence of swaps, not necessarily the minimal one.

The final velocity vector, $V_i(t+1)$, is composed by probabilistically selecting and concatenating swap sequences generated from the inertia, cognitive, social, and elite-learning components. The application of this velocity to update the particle's position, denoted by the $\oplus$ operator in Equation (22), is then performed. This operation involves the sequential and deterministic application of every swap pair in the final SOS to the current chromosome $X_i(t)$. The stochastic nature of the particle's movement arises from the composition of its velocity vector, not from the application of it.

To further clarify this process, let us consider a small example involving five products (P1–P5), six available locations (L1–L6), and three temperature zones (1: Ambient, 2: Refrigerated, 3: Frozen). The setup is detailed in Table 8.

Table 8. Setup for SOS example.

Product ID	Required Zone	Available Locations (Zone)	Current Position $X_i(t)$	Personal Best $pbest_i$
P1	1 (Ambient)	L1(1), L2(1), L3(1)	L1	L2
P2	1 (Ambient)		L2	L1
P3	2 (Refrigerated)	L4(2), L5(2)	L4	L5
P4	2 (Refrigerated)		L5	L4
P5	3 (Frozen)	L6(3)	L6	L6

The table supposes the current solution for a particle is $\boldsymbol{X}_i(t) = [\text{L2}, \text{L1}, \text{L4}, \text{L5}, \text{L6}]$, while its personal best is $pbest_i = [\text{L1}, \text{L2}, \text{L5}, \text{L4}, \text{L6}]$. It is important to recognize that both of these are *feasible* solutions, as every product is assigned to a location that satisfies its temperature requirement. To generate the cognitive component of velocity, the algorithm performs a randomized search for feasible exchanges among the differing positions. This approach is a deliberate design choice to maintain the explorative nature of PSO and mitigate the risk of premature convergence.

The next few steps show how the PSO operators are translated into a concrete search mechanism:

Step 1: Identify Differences. By comparing $X_i(t)$ and $pbest_i$, we find differences at indices 1, 2, 3, and 4. Product $P5$ is correctly assigned to $L6$ in both solutions.

Step 2: Generate Cognitive Swaps. The algorithm now performs a randomized search for feasible swaps among the differing indices.

- Attempt 1 (Successful): Consider a swap between index **1** ($P1$) and index **2** ($P2$). $P1$ requires Zone **1** and is currently in $L2$ (Zone **1**). $P2$ requires Zone **1** and is currently in $L1$ (Zone **1**). If they swap, $P1$ will be in $L1$ (Zone **1**) and $P2$ will be in $L2$ (Zone **1**). Since both locations satisfy the requirements of the new products, the swap (1, 2) is valid and is added to the potential **SOS**.
- Attempt 2 (Successful): Similarly, a swap between index **3** ($P3$) and index **4** ($P4$) is considered. $P3$ is in $L4$ (Zone **2**), and $P4$ is in $L5$ (Zone **2**). Swapping them is also valid. The swap (3, 4) is added to the potential **SOS**.
- Attempt 3 (Failure): Suppose we attempt an invalid swap between index **1** ($P1$) and index **3** ($P3$). $P1$ requires Zone **1**, but its proposed new location $L4$ is in Zone **2**. This violates the temperature constraint. Therefore, the swap (1, 3) is invalid and is discarded.

Step 3: Form the SOS. Based on the successful searches, the generated sequence of swaps for the cognitive component is $SOS = \{(1,2), (3,4)\}$.

Step 4: Update Position. The velocity $V_i(t+1)$ is formed in this simplified case, so let us assume it is just SOS. The update operation $X_i^{'}(t)$ is performed by sequentially applying the swaps:

- Initial: $X_i(t) = [\text{L2}, \text{L1}, \text{L4}, \text{L5}, \text{L6}]$
- Apply swap (1, 2): $[\text{L1}, \text{L2}, \text{L4}, \text{L5}, \text{L6}]$
- Apply swap (3, 4): $[\text{L1}, \text{L2}, \text{L5}, \text{L4}, \text{L6}]$
- Final: $X_i^{'}(t) = pbest_i = [\text{L1}, \text{L2}, \text{L5}, \text{L4}, \text{L6}]$

(3) Local Search Enhancement via Variable Neighborhood Search (VNS)

To improve the local optimality of the candidate solution [37,38], a Variable Neighborhood Search (VNS) is applied to X_i' with probability P_{ls}. VNS facilitates an in-depth exploration by systematically alternating among a predefined set of neighborhood structures, for $k = 1, \ldots, k_{max}$. This process can be formalized as follows:

$$X_i'' = \text{VNS}(X_i', \{N_1, \ldots, N_{k_{\max}}\}) \tag{23}$$

The resulting solution, X_i'', which has been enhanced by VNS, proceeds to the Simulated Annealing acceptance phase.

(4) Acceptance Criteria based on Simulated Annealing (SA)

Instead of the deterministic update rule typical of PSO, PS-GSA incorporates the Metropolis acceptance criterion from Simulated Annealing to probabilistically determine the acceptance of the new solution X_i''. Given the change in the objective function value, $\Delta f = f(X_i'') - f(X_i(t))$, the acceptance probability P is defined as

$$P(X_i(t) \to X_i'') = \begin{cases} 1 & \text{if } \Delta f \leq 0 \\ \exp\left(-\frac{\Delta f}{T_t}\right) & \text{if } \Delta f > 0 \end{cases} \quad (24)$$

Here, T_t is the system "temperature" at iteration t, which follows a geometric cooling schedule $T_{t+1} = \alpha \cdot T_t$, with α being the cooling rate. This mechanism endows the algorithm with the ability to escape from local optima.

To counteract premature population convergence observed in standard genetic algorithms, two specific mechanisms are integrated into the PS-GSA.

The first is a multi-operator mutation strategy. Throughout the iterative process, the algorithm probabilistically applies a mutation operator, randomly selected from a predefined library $M_{\text{ops}} = \{\text{swap}, \text{insert}, \text{inversion}, \dots\}$, to individuals, enhancing the diversity of perturbations within the population.

The second is a partial population restart mechanism [39]. The algorithm monitors the improvement of the global best solution using a stagnation counter, S_{count}. A restart is triggered if S_{count} surpasses a predefined threshold, S_{thresh}. This mechanism preserves the top p_{elite} (e.g., 20% of the population) as elites and replaces the remaining $1 - p_{\text{elite}}$ individuals with new, randomly generated solutions, rapidly reinvigorating the population.

Finally, to meticulously refine the current best solution, the algorithm executes a computationally intensive and more thorough local search on the global best solution, *gbest*, at a fixed frequency of f_{elite} iterations. This procedure is designed to systematically explore the neighborhood of *gbest* to identify solutions of superior quality.

4. Computational Results and Analysis

This chapter presents an evaluation of the proposed Particle Swarm-guided hybrid Genetic-Simulated Annealing (PS-GSA) algorithm. To validate its effectiveness and robustness, a series of computational experiments were conducted on multiple problem instances of increasing scale. The performance of PS-GSA is systematically compared against four established benchmark algorithms: Genetic Algorithm (GA), Simulated Annealing (SA), a Hybrid Particle Swarm Optimization (HPSO), and a baseline hybrid Genetic-Simulated Annealing (GSA) algorithm. The analysis encompasses solution quality, statistical significance and algorithmic convergence. All algorithms were implemented in MATLAB R2022a and executed on a laptop computer equipped with a 12th Gen Intel Core i7-12700H CPU, 16 GB RAM, and an NVIDIA GeForce RTX 3050Ti Laptop GPU.

The original data for this experiment comes from YouDaji (Wuhan) Supply Chain Co. (Wuhan, China). The company focuses on meeting the market demand of the whole fresh food industry chain, constantly broadening its service areas, and devotes itself to providing commodity trading, fresh food collection, specialized warehousing and logistics, circulation processing and other services for the participants of the whole fresh food industry chain, and has completed its strategic layouts in Wuhan, Xiangyang, Shenzhen, Yichang, and Ningbo in various regions successively. The company is committed to building a "S2B2B" agricultural trading platform, and its fresh produce warehousing and sorting process is centered on the core system of "preserving freshness and improving efficiency".

To assess the performance and scalability of the algorithms when faced with increasing problem complexity, the instances are scaled by the number of products N_p, with $N_p \in \{70, 140, 210, 280, 350\}$. The benchmark algorithms selected for comparison are GA, SA, HPSO, and GSA. The justification for each is

- Genetic Algorithm (GA) and Simulated Annealing (SA): These are foundational metaheuristics for combinatorial optimization. They are included to establish a performance baseline against classic methods.
- Hybrid Particle Swarm Optimization (HPSO): This algorithm is included as a representative of advanced, contemporary PSO-based methods. Given that PSO is a core component of the proposed algorithm, comparing against a PSO variant is essential for a fair evaluation.
- Genetic-Simulated Annealing (GSA): This hybrid serves as a crucial ablation baseline. By comparing PS-GSA to GSA, the specific performance contribution of the high-level Particle Swarm Optimization guidance mechanism can be isolated and quantified, showing the value added by the novel hierarchical structure of the proposed algorithm.

To provide a holistic assessment, performing each algorithm was evaluated based on multiple metrics derived from 30 independent runs for each algorithm-instance combination. Table 9 summarizes the overall performance of the five algorithms. For each problem size (N), the table reports the mean and standard deviation of the final objective function values, the best solution found across 30 runs, and the average runtime in seconds. The best-performing value for each metric (excluding runtime) is highlighted in bold.

Table 9. Overall performance of algorithms.

N	Algorithm	Mean	Std	Best	Worst	Median	Runtime(s)
70	GA	1010.32	**30.39**	966.93	**1064.04**	1015.59	61.89
	SA	1296.76	51.81	1220.93	1426.83	1296.19	6.24
	HPSO	1220.96	42.25	1130.75	1298.85	1226.60	68.61
	GSA	1263.60	33.61	1193.54	1321.54	1268.12	116.28
	PS-GSA	**1002.87**	61.66	**930.35**	1180.97	**997.54**	258.84
140	GA	2663.46	**61.92**	2586.27	2761.89	2659.99	72.05
	SA	2841.00	86.22	2676.97	2949.11	2862.80	13.90
	HPSO	2968.50	100.28	2839.82	3177.80	2960.95	93.13
	GSA	3015.80	82.34	2864.77	3120.45	3040.30	142.67
	PS-GSA	**2531.08**	81.76	**2424.48**	**2662.11**	**2534.56**	318.76
210	GA	5538.24	106.68	5383.56	5723.57	5523.91	97.57
	SA	5667.94	143.61	5336.31	5891.31	5700.48	12.36
	HPSO	5788.85	75.71	5609.81	5903.85	5798.95	114.51
	GSA	5947.30	80.47	5799.65	6063.92	5961.19	205.45
	PS-GSA	**5386.71**	**65.44**	**5233.68**	**5465.03**	**5406.60**	404.36
280	GA	7392.03	299.74	7037.07	7845.05	7214.16	276.55
	SA	7292.56	110.65	7031.68	7445.25	7310.22	27.62
	HPSO	7467.72	86.92	7254.98	7584.67	7469.38	316.96
	GSA	7608.20	**85.48**	7476.90	7729.57	7624.19	520.63
	PS-GSA	**6902.09**	88.53	**6727.93**	**7039.62**	**6915.98**	1036.68
350	GA	9266.27	326.83	8747.98	9644.51	9252.69	279.52
	SA	9078.99	139.35	8771.83	9351.08	9100.66	28.58
	HPSO	9267.09	98.12	9009.25	9387.73	9298.16	340.97
	GSA	9447.87	85.65	9306.75	9601.62	9454.03	498.01
	PS-GSA	**8708.58**	**77.98**	**8458.81**	**8862.88**	**8713.17**	1057.99

The results presented in Table 9 clearly show the superior performance of the proposed PS-GSA algorithm in terms of solution quality. Across all five problem instances, from $N = 70$ to $N = 350$, PS-GSA consistently achieves the lowest (best) mean objective function value. For instance, in the largest and most complex case ($N = 350$), PS-GSA obtains a mean fitness of 8708.58, which represents a 4.08 % improvement over the next-best

algorithm, SA (9078.99), and a 6.02 % improvement over HPSO (9267.09). This trend holds for the best-found solutions as well, where PS-GSA identifies solutions of significantly higher quality than any of the benchmark algorithms. The data shows that as the problem size and complexity increase, the performance gap between PS-GSA and the other algorithms widens.

To formally confirm these results, Table 10 presents the results of the Wilcoxon signed-rank test, which compares the 30 independent run results of PS-GSA against each benchmark algorithm for every instance. The table displays the calculated p-value and the percentage improvement of PS-GSA's mean fitness over comparison algorithms.

Table 10. Statistical significance of PS-GSA's performance (Wilcoxon test p-values).

N	Comparison	p Value	Significant	Mean Diff	Improvement
70	PS-GSA vs GA	0.45428	No	7.44	0.74%
	PS-GSA vs SA	0.00006	Yes	293.89	22.66%
	PS-GSA vs HPSO	0.00012	Yes	218.08	17.86%
	PS-GSA vs GSA	0.00006	Yes	260.72	20.63%
140	PS-GSA vs GA	0.00195	Yes	132.37	4.97%
	PS-GSA vs SA	0.00195	Yes	309.91	10.91%
	PS-GSA vs HPSO	0.00195	Yes	437.41	14.74%
	PS-GSA vs GSA	0.00195	Yes	484.71	16.07%
210	PS-GSA vs GA	0.00016	Yes	151.53	2.74%
	PS-GSA vs SA	0.00012	Yes	281.22	4.96%
	PS-GSA vs HPSO	0.00009	Yes	402.14	6.95%
	PS-GSA vs GSA	0.00009	Yes	560.58	9.43%
280	PS-GSA vs GA	0.00006	Yes	489.94	6.63%
	PS-GSA vs SA	0.00006	Yes	390.47	5.35%
	PS-GSA vs HPSO	0.00006	Yes	565.63	7.57%
	PS-GSA vs GSA	0.00006	Yes	706.11	9.28%
350	PS-GSA vs GA	0.00010	Yes	557.69	6.02%
	PS-GSA vs SA	0.00009	Yes	370.42	4.08%
	PS-GSA vs HPSO	0.00009	Yes	558.51	6.03%
	PS-GSA vs GSA	0.00009	Yes	739.29	7.82%

The statistical analysis provides evidence for the superiority of PS-GSA. For all problem instances with $N \geq 140$, the p-values for the comparison of PS-GSA against all four benchmark algorithms are substantially lower than the 0.05 significance level. This shows that the observed performance advantage of PS-GSA is statistically significant and not a result of random chance. However, an important nuance is observed in the smallest instance ($N = 70$). While PS-GSA still outperforms GA in mean fitness, the difference is not statistically significant ($p = 0.45428$). This outcome is not a weakness but an insight into the algorithm's domain of applicability. The search space for the $N = 70$ problem is relatively small and less complex. In such a landscape, the sophisticated global guidance and local search mechanisms of PS-GSA provide diminishing returns, as a simpler heuristic like GA can explore the space adequately.

The convergence and stability of the algorithm will be analyzed below. Figure 3 illustrates the average convergence performance of the five algorithms over 200 iterations for the largest instance ($N = 350$). The y-axis represents the average fitness value, while the x-axis represents the iteration number.

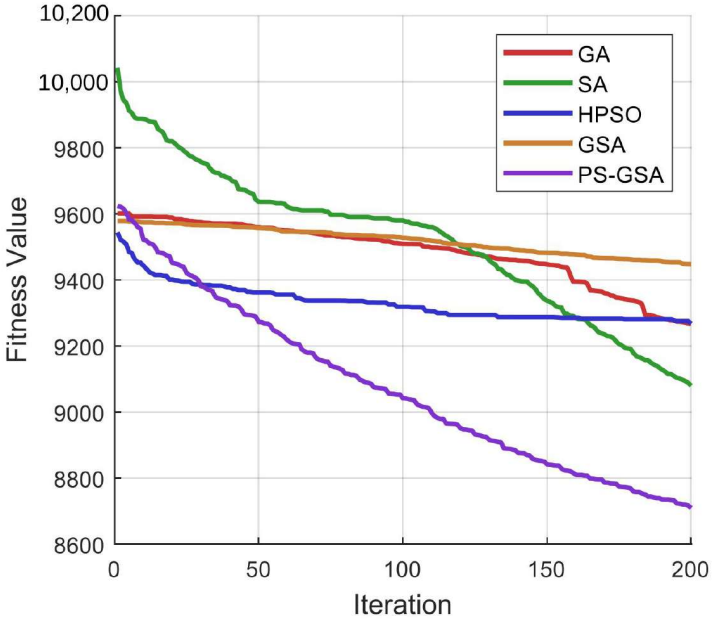


Figure 3. All algorithm convergence curves.

The convergence curves of the algorithms in Figure 3 precisely represent the different convergence performances. The curves of GA and GSA do not show any significant degradation until 150 iterations; the curves of SA and HPSO show rapid performance improvement within the first 50 iterations, but then the curves flatten out to form an obvious plateau period. In contrast, the convergence curve for PS-GSA shows a sustained and steady decline throughout the entire 200-iteration run. It does not have hard plateaus or erratic swings, suggesting that it continues to explore the solution space and identify improvements even in the later stages of the search. Figure 4 further reinforces this by showing the tight 95% confidence interval around the PS-GSA mean convergence curve, showing low variance and consistent performance across runs.

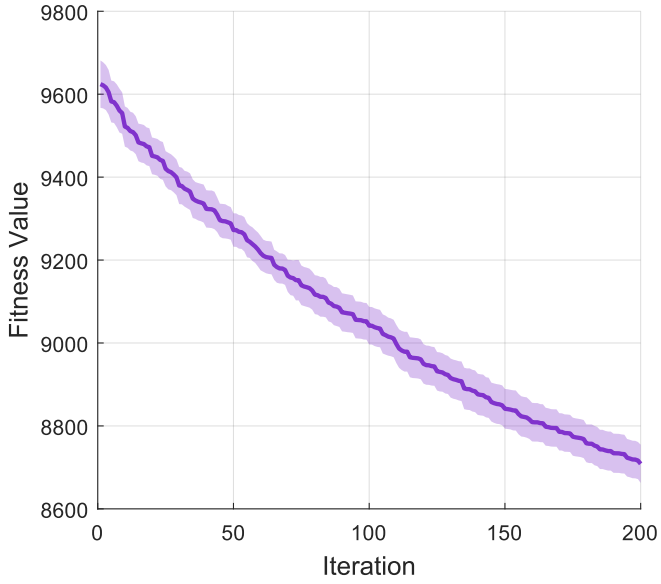


Figure 4. PS-GSA convergence (95% CI) curve.

The convergence curve for PS-GSA exhibits a slight downward trend at the end of 200 iterations, indicating that marginal improvements could be possible with an extended run. However, the rate of convergence has slowed significantly, suggesting the algorithm is

approaching the optimal solution region. Therefore, the 200-iteration limit is established as a reasonable trade-off between solution quality and computational cost.

Beyond average performance, the consistency of an algorithm is critical for its practical application. Figure 5 presents box plots illustrating the distribution of the final solutions obtained from the 30 independent runs for each algorithm on the $N = 350$ instance.

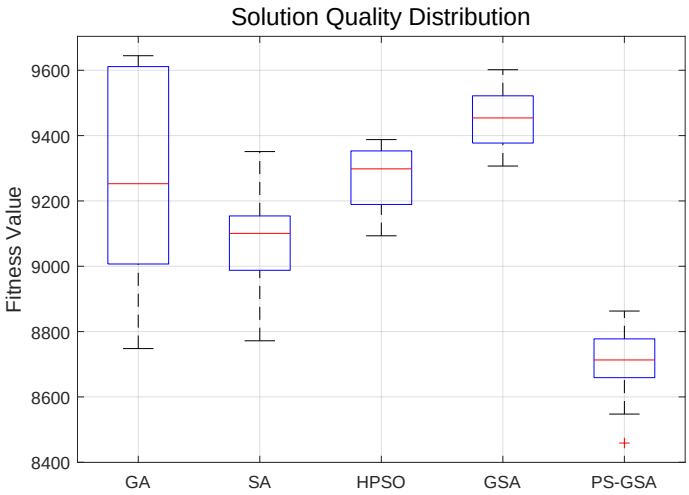


Figure 5. Distribution of final solution.

Figure 5 provides a clear visual summary of both the solution quality and stability of the algorithms. The PS-GSA box plot is positioned lowest on the y-axis, confirming that it achieves the best median fitness value. More importantly, the interquartile range, represented by the height of the box, is notably compact for PS-GSA. This shows that the middle 50% of its solutions are tightly clustered around the median. In contrast, the box plots for GA and SA are much taller and have longer whiskers, signifying a high variability in their outcomes. This high variance makes them less reliable, as a single run could yield either a good or a poor result. The performance of PS-GSA is evidenced by its low standard deviation (Table 9) and tight interquartile range. The global guidance from the PSO framework reduces the likelihood of the search deviating into poor regions of the solution space, while the VNS and SA components ensure thorough exploitation of promising areas.

While solution quality and stability are paramount, computational efficiency is also a key consideration. Figure 6 plots the average computational time of each algorithm as a function of problem size. The data sourced from Table 9 demonstrates how the computational demands of each algorithm scale with increasing complexity.

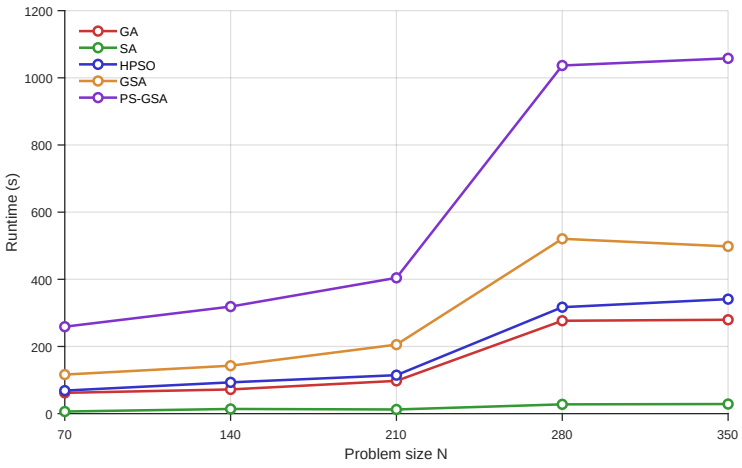


Figure 6. Average runtime vs. problem size N.

The analysis of computational time reveals that the proposed PS-GSA is the most computationally intensive algorithm among the tested set. As shown in Figure 6, its runtime increases substantially with problem size, reaching approximately 1058 s for the largest instance ($N = 350$). In contrast, SA is the fastest, requiring only approximately 29 s for the same instance. However, this higher runtime should be interpreted not as a weakness but as a justified trade-off for achieving superior solution quality. This higher computational cost is an inherent and expected consequence of PS-GSA's architectural complexity. The additional time is consumed by the sophisticated operators that simpler algorithms lack: calculating the discrete PSO swap sequences, performing the systematic VNS local searches, and executing the iterative logic of the SA acceptance criterion.

Finally, to provide a tangible illustration of the algorithm's output, this section presents a sample of the optimized storage location assignment for the largest instance ($N = 350$). Table 11 shows the initial and final storage coordinates for a selection of products, translating the abstract fitness value into a concrete physical layout.

Table 11. Optimized storage location assignment for the $N = 350$ instance

Product ID	Temperature Zone	Initial Location	Optimized Location
1	1 (Ambient)	(1, 1, 1)	(2, 14, 1)
7	2 (Refrigerated)	(6, 15, 2)	(6, 14, 1)
17	3 (Frozen)	(15, 13, 1)	(15, 18, 1)
37	1 (Ambient)	(2, 6, 2)	(4, 15, 3)
46	3 (Frozen)	(17, 3, 1)	(17, 28, 1)
52	2 (Refrigerated)	(10, 13, 2)	(8, 8, 1)
109	1 (Ambient)	(2, 28, 2)	(1, 30, 1)
121	3 (Frozen)	(13, 17, 2)	(16, 1, 1)
140	3 (Frozen)	(17, 23, 2)	(17, 11, 1)
210	3 (Frozen)	(15, 14, 3)	(15, 3, 1)
350	2 (Refrigerated)	(12, 23, 2)	(13, 1, 2)

The sample assignments in Table 11 illustrate that PS-GSA successfully reassigns products to new locations within the warehouse grid. A key observation is that all assignments adhere to the fundamental constraints of the model. For instance, Product 52, which requires a refrigerated environment (Area 2), is moved from one location to another but remains within the designated refrigerated zone. The same holds for ambient and refrigerated products.

5. Conclusions

This study aims to solve the problem of joint optimization of storage allocation and order picking for fresh products. To address this challenge, this paper constructs a comprehensive mathematical model that integrates five major costs—picking paths, slotting layout, energy consumption, First-In-First-Out principles, and personnel scheduling—and proposes an innovative Particle Swarm-guided hybrid Genetic-Simulated Annealing (PS-GSA) algorithm. The core contribution of this algorithm lies in its hierarchical and synergistic optimization framework: a Particle Swarm Optimization (PSO) algorithm acts as a global strategy guide, directing the population evolution of a lower-level Genetic Algorithm (GA), while deeply integrating the local search capabilities of Variable Neighborhood Search with the probabilistic escaping mechanism of Simulated Annealing (SA). Computational experiments based on real enterprise data have showed the superior performance of the proposed PS-GSA algorithm. In comparisons with benchmark algorithms including standard GA, SA, HPSO, and GSA, PS-GSA shows significant and statistically robust advantages in solution quality, convergence efficiency, and stability, with performance improvements ranging from 4.08% to 9.43% over the next-best algorithm, especially in large-scale instances.

This study provides significant theoretical and methodological implications for the field of combinatorial optimization. First, the "master–slave" hierarchical architecture adopted by PS-GSA, which uses the global exploration capabilities of PSO to guide the evolutionary direction of the GA population, provides the momentum for a sustained and effective search in complex solution spaces. This transcends the simple concatenation of operators found in conventional hybrid algorithms, forming a deeper synergistic mechanism. Second, by employing the Metropolis acceptance criterion of SA as a probabilistic decision gate and embedding VNS to enhance local search, this study constructs a sophisticated dynamic balancing mechanism that effectively coordinates the algorithm's behavior between global exploration and local exploitation. The success of this theoretical design is visually validated by the algorithm's convergence curve: unlike algorithms such as SA and HPSO, which enter a "plateau period" after rapid initial improvements, PS-GSA's convergence curve exhibits a sustained and steady decline, showing its ability to effectively avoid search stagnation and continuously discover better solutions in complex solution spaces.

In terms of managerial practice, this study provides a powerful and flexible decision-support tool that can deliver tangible operational value to warehouse managers in the fresh e-commerce sector. The comprehensive cost model not only covers the key factors affecting the efficiency and cost of fresh product storage allocation but, more importantly, empowers managers to customize optimization objectives based on different strategic priorities (e.g., cost control, order timeliness, and energy conservation) by incorporating the Analytic Hierarchy Process (AHP) for weight configuration. The results of the trade-off analysis clearly quantify the pros and cons of different management strategies. For instance, a "cost-centric strategy" can significantly reduce energy costs by 14.4% but at the expense of a 32.8% increase in picking path costs. Conversely, an "efficiency-driven strategy" improves FIFO execution efficiency at the cost of higher energy and layout expenses. These quantitative results enable managers to shift from reactive daily operations to proactive, data-driven strategic planning.

Despite the significant achievements of this study, several limitations remain, which also open up new directions for future research. First, the validity of this research was verified using specific warehouse layout and operational data from a single enterprise; the generalizability of its conclusions to warehouses with different layouts (e.g., fishbone), scales, or demand patterns needs further examination. Second, some assumptions in the model, such as treating picker travel speed as a constant, simplify real-world situations. Future research could introduce dynamic and stochastic factors, such as picker fatigue or demand uncertainty, to build more dynamic optimization models. Third, while the PS-GSA algorithm delivers high-quality solutions, its higher computational time represents a performance trade-off. Future work could explore the use of parallel computing to reduce solution time or develop machine learning-based surrogate models to meet real-time decision-making needs. Finally, although the weighted-sum method used in this study effectively reflects decision-maker preferences, it has inherent limitations in handling non-convex Pareto fronts.

Supplementary Materials: The following supporting information can be downloaded at https://www.mdpi.com/article/10.3390/math13213428/s1.

Author Contributions: Conceptualization, Y.Z. and J.L.; methodology, K.X.; software, Y.Z.; validation, K.X.; formal analysis, Y.X.; investigation, K.X.; resources, Y.X.; data curation, Y.X.; writing—original draft preparation, Y.Z.; writing—review and editing, Y.Z. and K.X.; visualization, Y.Z. and Y.X.; supervision, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Social Science Foundation of China (20BGL168), and the National College Student Innovation and Entrepreneurship Training Program (202410307096Z).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article and the Supplementary Materials, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Fan, Y.; de Kleuver, C.; de Leeuw, S.; Behdani, B. Trading Off Cost, Emission, and Quality in Cold Chain Design: A Simulation Approach. *Comput. Ind. Eng.* **2021**, *158*, 107411. [CrossRef]
2. Goyal, S.K.; Giri, B.C. Recent Trends in Modeling of Deteriorating Inventory. *Eur. J. Oper. Res.* **2001**, *134*, 1–16. [CrossRef]
3. Hall, R.W. Distance Approximations for Routing Manual Pickers in a Warehouse. *IIE Trans.* **1993**, *25*, 76–87. [CrossRef]
4. Petersen, C.G.; Aase, G. A Comparison of Picking, Storage, and Routing Policies in Manual Order Picking. *Int. J. Prod. Econ.* **2004**, *92*, 11–19. [CrossRef]
5. Nahmias, S. Perishable Inventory Theory: A Review. *Oper. Res.* **1982**, *30*, 680–708. [CrossRef] [PubMed]
6. Piramuthu, S.; Zhou, W. RFID and Perishable Inventory Management with Shelf-Space and Freshness Dependent Demand. *Int. J. Prod. Econ.* **2013**, *144*, 635–640. [CrossRef]
7. Bai, R.; Kendall, G. A Model for Fresh Produce Shelf-Space Allocation and Inventory Management with Freshness-Condition-Dependent Demand. *INFORMS J. Comput.* **2008**, *20*, 78–85. [CrossRef]
8. Lin, Y.S.; Li, Q.S.; Lu, P.H.; Sun, Y.N.; Wang, L.; Wang, Y.Z. Shelf and AGV Path Cooperative Optimization Algorithm Used in Intelligent Warehousing. *J. Softw.* **2020**, *31*, 2770–2784. (In Chinese) [CrossRef]
9. Yang, X.S. Review of Meta-Heuristics and Generalised Evolutionary Walk Algorithm. *Int. J. Bio-Inspired Comput.* **2011**, *3*, 77–84. [CrossRef]
10. Zhou, G.M.; Liu, R.L.; Zhang, Z.J.; Yang, C.H.; Ding, H.J. Optimization of Diesel Engine Dual-Variable Geometry Turbocharger Regulated Two-Stage Turbocharging System Based on Radial Basis Function Neural Network-Quantum Genetic Algorithm. *Energy Sources Part Recover. Util. Environ. Eff.* **2021**, *47*, 1910–1926. [CrossRef]
11. Lin, Y.H.; Liu, B.Y.; Zhang, T.Z.; Zhang, H.X.; Zhang, Z. Energy Management Strategy for Electrically-Powered Hydraulic Vehicle Based on Driving Mode Recognition. *Energy Sources, Part A Recover. Util. Environ. Eff.* **2025**, *47*, 2480–2503. [CrossRef]
12. Su, D.D.; Li, H.; Guo, J. Sustainable Multi-Objective Location-Routing Problem with Time Windows: A Case Study in China. *Int. J. Syst. Sci.-Oper. Logist.* **2025**, *12*, 2227129. [CrossRef]
13. Xu, J.; Liu, L.; Xu, W.B. Application of Immune Genetic Algorithm to Multiobjective Optimization. In Proceedings of the 2006 International Symposium on Distributed Computing and Applications to Business, Engineering and Science, Hangzhou, China, 15–16 December 2006; IEEE: Piscataway, NJ, USA, 2006; pp. 406–410. [CrossRef]
14. Hu, Y.W.; Dong, H.L.; Liu, J.H.; Zhuang, C.B.; Zhang, F. A Learning-Guided Hybrid Genetic Algorithm and Multi-Neighborhood Search for the Integrated Process Planning and Scheduling Problem with Reconfigurable Manufacturing Cells. *Robot. Comput.-Integr. Manuf.* **2025**, *93*, 102919. [CrossRef]
15. Yeo, M.F.; Agyei, E.O. Optimising Engineering Problems Using Genetic Algorithms. *Eng. Comput.* **1998**, *15*, 268–283. [CrossRef]
16. Azadeh, A.; Elahi, S.; Farahani, M.H.; Nasirian, B. A Genetic Algorithm-Taguchi Based Approach to Inventory Routing Problem of a Single Perishable Product with Transshipment. *Comput. Ind. Eng.* **2017**, *104*, 124–133. [CrossRef]
17. Hiassat, A.; Diabat, A.; Rahwan, I. A Genetic Algorithm Approach for Location-Inventory-Routing Problem with Perishable Products. *J. Manuf. Syst.* **2017**, *42*, 93–103. [CrossRef]
18. Li, J.C.; Lei, L. A Hybrid Genetic Algorithm Based on Information Entropy and Game Theory. *IEEE Access* **2020**, *8*, 36602–36611. [CrossRef]
19. Zhang, X.; Guo, P.; Zhang, H.; Yao, J. Hybrid Particle Swarm Optimization Algorithm for Process Planning. *Mathematics* **2020**, *8*, 1683. [CrossRef]
20. He, Q.; Wu, Y.L.; Xu, T.W. Application of Improved Genetic Simulated Annealing Algorithm in TSP Optimization. *Control Decis.* **2018**, *33*, 219–225. [CrossRef]
21. Liu, Y.Y.; Dai, J.J.; Zhao, S.S.; Zhang, J.H.; Shang, W.D.; Li, T.; Zheng, Y.C.; Lan, T.; Wang, Z.Y. Optimization of Five-Parameter BRDF Model Based on Hybrid GA-PSO Algorithm. *Optik* **2020**, *219*, 165147. [CrossRef]
22. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H.L. Harris Hawks Optimization: Algorithm and Applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
23. Ropke, S.; Pisinger, D. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transp. Sci.* **2006**, *40*, 455–472. [CrossRef]

24. Pan, J.C.H.; Shih, P.H.; Wu, M.H.; Lin, J.H. A Storage Assignment Heuristic Method Based on Genetic Algorithm for a Pick-and-Pass Warehousing System. *Comput. Ind. Eng.* **2015**, *81*, 1–13. [CrossRef]
25. Zhang, X.; Mo, T.; Zhang, Y. Optimization of Storage Location Assignment for Non-Traditional Layout Warehouses Based on the Firework Algorithm. *Sustainability* **2023**, *15*, 10242. [CrossRef]
26. Zhang, Z.Y.; Liu, X.B.; Yang, S.L. A Note on the 1-9 Scale and Index Scale in AHP. In Proceedings of the 20th International Conference on Multiple Criteria Decision Making (MCDM 2009), Chengdu, China, 21–26 June 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 1–5. [CrossRef]
27. Jakob, W.; Blume, C. Pareto Optimization or Cascaded Weighted Sum: A Comparison of Concepts. *Algorithms* **2014**, *7*, 166–185. [CrossRef]
28. Blum, C.; Roli, A. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Comput. Surv.* **2003**, *35*, 268–308. [CrossRef]
29. Shi, Y.; Eberhart, R.C. Empirical Study of Particle Swarm Optimization. In Proceedings of the 1999 Congress on Evolutionary Computation–CEC99, Washington, DC, USA, 6–9 July 1999; Volume 3, pp. 1945–1950.
30. Kennedy, J.; Eberhart, R.C. Particle Swarm Optimization. In Proceedings of the 1995 IEEE International Conference on Neural Networks (ICNN '95), Perth, WA, Australia, 27 November–1 December 1995.
31. Shi, Y.; Eberhart, R.C. A Modified Particle Swarm Optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, 4–9 May 1998; pp. 69–73.
32. Smetek, M.; Trawinski, B. Investigation of Genetic Algorithms with Self-Adaptive Crossover, Mutation, and Selection. In Proceedings of the Hybrid Artificial Intelligence Systems, Wroclaw, Poland, 23–25 May 2011; Lecture Notes in Computer Science; Volume 6679, pp. 301–308. [CrossRef]
33. Smetek, M.; Trawinski, B. Investigation of Self-Adapting Genetic Algorithms Using Some Multimodal Benchmark Functions. In Proceedings of the Computational Collective Intelligence. Technologies and Applications, Gdynia, Poland, 21–23 September 2011; Lecture Notes in Computer Science; Volume 6922, pp. 440–449. [CrossRef]
34. Liu, H.B.; Abraham, A.; Choi, O.; Moon, S.H. Variable Neighborhood Particle Swarm Optimization for Multi-Objective Flexible Job-Shop Scheduling Problems. In *Proceedings of the Simulated Evolution and Learning*; Lecture Notes in Computer Science; Wang, T.D., Li, X., Chen, S.H., Wang, X., Abbass, H., Iba, H., Chen, G., Yao, X., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4247, pp. 197–204. [CrossRef]
35. Chen, J.Y.; Xiao, Z.Q. Research on Adaptive Genetic Algorithm Based on Multi-Population Elite Selection Strategy. In Proceedings of the 2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA), Beijing, China, 8–11 September 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 112–116. [CrossRef]
36. Sun, M.X.; Luan, T.T.; Xu, J. Amphibious Vehicle Layout Optimization Based on Adaptive Elite Genetic Algorithm. In Proceedings of the 2019 16th IEEE International Conference on Mechatronics and Automation (ICMA), Tianjin, China, 4–7 August 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 106–110. [CrossRef]
37. Zhou, A.M.; Qu, B.Y.; Li, H.; Zhao, S.Z.; Suganthan, P.N.; Zhang, Q.F. Multiobjective Evolutionary Algorithms: A Survey of the State of the Art. *Swarm Evol. Comput.* **2011**, *1*, 32–49. [CrossRef]
38. Hansen, P.; Mladenovic, N. Variable Neighborhood Search: Principles and Applications. *Eur. J. Oper. Res.* **2001**, *130*, 449–467. [CrossRef]
39. Sevkli, Z.; Sevilgen, F.E. A Hybrid Particle Swarm Optimization Algorithm for Function Optimization. In Proceedings of the Applications of Evolutionary Computation, Naples, Italy, 26–28 March 2008; Lecture Notes in Computer Science; Volume 4974, pp. 865–874. [CrossRef]

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.