

A HYBRID GENETIC ALGORITHM–TABU SEARCH METHOD FOR THE STORAGE LOCATION ASSIGNMENT PROBLEM IN ROBOTIC MOBILE FULFILLMENT SYSTEMS

Yung-Chia Chang¹, Kuei-Hu Chang^{2,*}, and Shao-Ting Yen¹

¹Department of Industrial Engineering and Management
National Yang Ming Chiao Tung University
Hsinchu, Taiwan

²Department of Management Sciences,
R.O.C. Military Academy,
Kaohsiung, Taiwan

*Corresponding author's e-mail: evenken2002@yahoo.com.tw

With the advent of artificial intelligence, the robotic mobile fulfillment system (RMFS) has emerged as a novel automated warehousing system. Using a goods-to-person approach, RMFS employs robots to transport pods to workstations, where human pickers retrieve the required items, thereby significantly reducing walking distance. In addition, both material assignment to pods and pod storage location assignment play critical roles in improving picking efficiency. This study addresses these two decision problems through a two-stage solution framework under a simplified single-AGV, single-workstation RMFS. In the first stage, Apriori-based association analysis and DBSCAN clustering are employed to identify highly associated materials and assign them to pods. In the second stage, a hybrid genetic algorithm (GA) and tabu search (TS) algorithm is developed to optimize pod storage locations by minimizing the total round-trip picking distance required to fulfill orders. To verify the feasibility and effectiveness of the proposed approach, the developed algorithms were applied to one week of real production-order data from a Taiwanese computer equipment manufacturing company. Experimental results show that, compared with the company's current method, the proposed approach reduces the average total pod movements by approximately 47.12% and the average total picking distance by approximately 29.60% under the studied operational setting.

Keywords: Robotic Mobile Fulfillment System; Storage Location Assignment Problem; Apriori Algorithm; Density-Based Spatial Clustering of Applications with Noise Algorithm; Genetic Algorithm; Tabu Search Algorithm

(Received on March 31, 2026; Accepted on August 31, 2026)

1. INTRODUCTION

With the growing adoption of Industry 4.0 concepts, manufacturers are increasingly investing in smart factories where intelligent logistics and warehouse automation play essential roles. By integrating technologies such as artificial intelligence (AI), the internet of things (IoT), and big data, intelligent logistics improves operational efficiency and reduces labor costs. Meanwhile, the rapid growth of e-commerce has led to highly diverse and frequent picking demands, making traditional manual picking increasingly inefficient. Consequently, automated warehousing systems, such as automated storage and retrieval systems (AS/RS) and robotic mobile fulfillment systems (RMFS), have been widely adopted.

The RMFS is an innovative automated warehousing solution first developed in 2003 by the German company Kiva Systems. It is an automated warehousing solution that integrates mobile robots, movable storage pods, workstations, and storage areas to automate storage, picking, and transportation processes (Boysen *et al.*, 2019). When an order is received, an automated guided vehicle (AGV) retrieves the corresponding pod (movable storage shelves), delivers it to a workstation for item picking, and subsequently returns the pod to the storage area, enabling efficient order fulfillment.

Among the various decision problems in RMFS, the storage location assignment problem (SLAP) has received considerable attention because it directly affects warehouse operational efficiency. SLAP refers to the task of assigning items to optimal storage locations within a warehouse, with the goal of minimizing handling costs and maximizing space utilization (Gu *et al.*, 2007). In the context of RMFS, the outcome of storage location assignments directly influences the total travel distance of AGVs, order picking times, and overall order fulfillment time (de Koster *et al.*, 2007). Frazee and Sharp (1989) confirmed that the SLAP is an NP-hard problem. Various mathematical modeling approaches have been proposed to solve

SLAP, including mixed-integer programming (e.g., Yang *et al.*, 2015; Chen *et al.*, 2010), binary programming (e.g., Chiang *et al.*, 2011), dynamic programming (e.g., Bodnar and Lysgaard, 2014; Dijkstra and Roodbergen, 2017), and branch-and-bound methods (e.g., Muppani and Adil, 2008; Gu *et al.*, 2010). However, these mathematical models often require substantial computational time to obtain optimal solutions, particularly for large-scale problem instances. To obtain feasible solutions within a reasonable computational time for complex and large-scale SLAP instances, researchers have turned to metaheuristic algorithms. For example, He *et al.* (2019) applied particle swarm optimization (PSO), while Udvardy *et al.* (2024) and Senaras *et al.* (2025) employed genetic algorithms (GA) to optimize the SLAP. Recent studies have investigated different decision problems in RMFS. For example, Jiao *et al.* (2026) focused on the online joint optimization of order assignment, pod selection, and task allocation, whereas Rizqi *et al.* (2025) investigated energy-efficient RMFS operations through traffic policy, routing, and robot configuration optimization. Li *et al.* (2025) optimized robot efficiency and picker energy consumption, while Liu *et al.* (2025) jointly considered storage assignment and order batching under dynamic storage conditions.

Most existing studies on the SLAP in RMFS have primarily focused on e-commerce retail environments. In contrast, few studies have investigated the SLAP in RMFS for computer hardware manufacturing, where parts are picked according to bill-of-material (BOM) requirements for specific production work orders. Unlike e-commerce environments, where orders are typically characterized by high product variety, low order quantities, and high order frequency, manufacturing environments involve BOM-driven picking tasks with high material diversity and large-volume component requirements. These fundamental differences in demand characteristics lead to different material association patterns and storage assignment requirements, making methods developed for e-commerce applications not directly applicable to manufacturing environments. Therefore, a research gap remains in developing storage assignment approaches that explicitly capture BOM-driven material associations and integrate them with pod storage location decisions in manufacturing-oriented RMFS.

To address the picking challenges faced by electronics manufacturers in their material handling processes, this study analyzes order data from a Taiwanese computer equipment manufacturing company (hereafter referred to as Company G). A two-stage solution framework is proposed to address the SLAP in the RMFS. The first stage focuses on identifying material demand associations and grouping highly related materials to derive an effective material assignment strategy tailored to BOM-driven manufacturing environments. In the second stage, a hybrid heuristic algorithm combining GA and tabu search (TS) is proposed to solve the SLAP, with the objectives of minimizing pod movement frequency and reducing the travel distance of AGVs. Through the proposed two-stage framework, the study aims to improve picking efficiency and warehouse operational performance in manufacturing-oriented RMFS environments.

Compared with existing RMFS studies, the primary contribution of this research lies in the development of an integrated solution framework for the SLAP in manufacturing-oriented RMFS. Unlike previous studies that mainly focused on e-commerce applications or general optimization problems, the proposed framework incorporates the operational characteristics of BOM-driven manufacturing into both material grouping and storage location assignments. In the first stage, Apriori-based association analysis and DBSCAN clustering are combined to identify and group highly associated materials under BOM-driven manufacturing demand. The resulting material groups are then allocated to pods according to material usage requirements and subsequently integrated with pod storage location optimization. In the second stage, GA and TS are integrated, in which TS locally refines the best GA solution using pod-swap neighborhoods. The proposed framework is validated through a real-world case study from a Taiwanese computer hardware manufacturer, demonstrating its practical applicability for improving warehouse picking efficiency.

The subsequent sections of this study are organized as follows: Section 2 discusses key issues related to the RMFS and reviews relevant literature on the SLAP, including heuristic algorithms applied within this context. Section 3 describes the SLAP examined in this study within the RMFS framework and provides a detailed explanation of the proposed hybrid heuristic algorithm. Section 4 presents the implementation data, elaborates on the experimental setup and parameters used in the proposed algorithm, and concludes with a comparative analysis of the experimental results. Finally, Section 5 summarizes the conclusions and contributions of this study and offers suggestions for future research.

2. LITERATURE REVIEW

2.1. Robotic mobile fulfillment system

With the rapid development of IoT technologies and robotic intelligence, automated warehousing systems are entering an accelerated growth phase (Lee *et al.*, 2019). The RMFS is a novel type of automated warehousing system characterized by a product-to-picker picking approach. This contrasts with traditional human-reliant warehousing systems that employ a picker-to-product picking mode (Van den Berg and Zijm, 1999). Compared to the conventional method, RMFS reduces human errors and improves picking efficiency.

Azadeh *et al.* (2019) indicated that the RMFS hardware mainly consists of three key components: robots, movable pods used for storing items and working stations. Merschformann *et al.* (2019) proposed a decision-making framework for the picking and replenishment processes in RMFS, which is divided into three hierarchical levels: strategic, tactical, and operational. The strategic level involves long-term planning and goals established during warehouse construction, such as the design of the storage area. The tactical level addresses medium- to long-term planning issues, including pod resource allocation and replenishment strategies. The operational level comprises four decision-making steps: order assignment, task creation, task allocation, and path planning. Task creation is further subdivided into two subproblems: pod selection and pod storage assignment. The SLAP addressed in this study belongs to the pod storage assignment category under task creation, which determines the placement of pods within the storage area.

2.2. Storage location assignment problem

The SLAP entails allocating items to appropriate storage locations with the objective of optimizing handling costs or maximizing storage space utilization. The solution to this problem depends on multiple parameters, such as the design of the storage area, availability of storage space, warehouse capacity, physical characteristics of stored items, arrival times, and order demands (Reyes *et al.*, 2019). SLAP exists across various industries, and its outcomes directly impact the operational efficiency of automated warehousing systems. Many scholars have studied and categorized strategies for storage location assignment. Zaerpour *et al.* (2013) noted that under the Random Assignment strategy, each storage location has an equal probability of being selected. The strategy of assigning items to the nearest available storage space prioritizes locations closest to the warehouse entrance or exit. The Full Turnover-based Assignment strategy places high-demand items closer to the exit, which can reduce order response time and shorten picking distances. Class-based Assignment assigns item categories to storage locations based on the distance from the entrance and the inventory turnover rate; categories with higher turnover rates are assigned to locations nearer the entrance, while items within the same category are randomly assigned within their designated locations. The Dedicated Storage strategy stores items in pre-assigned specific locations (Reyes *et al.*, 2019). Yu *et al.* (2015) discovered that under class-based assignment, fewer categories tend to yield better performance.

Reyes *et al.* (2019) summarized common approaches to solving the SLAP, including exact methods, metaheuristic algorithms, and simulation techniques. Exact methods typically refer to mathematical programming approaches such as mixed-integer programming (MIP), binary programming, non-linear programming (e.g., Yang *et al.*, 2017), and robust optimization (e.g., Ang *et al.*, 2012). Among these, MIP is the most frequently used method. Larco *et al.* (2017) proposed a storage assignment method based on MIP with a multi-objective strategy aimed at minimizing order preparation cycle time and reducing worker workload. Ene and Öztürk (2012) addressed a class-based storage assignment problem in the automotive industry by combining MIP with a GA to minimize storage and retrieval times.

Chang *et al.* (2024) demonstrated that the SLAP is NP-hard. When addressing larger-scale problems, mathematical modeling approaches require substantial computational time to find optimal solutions. Therefore, to obtain feasible solutions for complex and large-scale problems within reasonable time frames, many researchers have employed meta-heuristic algorithms to solve the SLAP. Commonly used meta-heuristics include TS, GA, simulated annealing, and ant colony optimization. Zhang *et al.* (2025) applied a GA to solve the SLAP. Chen *et al.* (2010) utilized TS to reduce travel time in automated warehouse systems. Cruz-Domínguez and Santos-Mayorga (2016) proposed a system combining artificial neural networks and GA to address storage space allocation, aiming to minimize order preparation distances. Kim and Smith (2012) applied simulated annealing to the SLAP to minimize the travel time during order preparation.

In the SLAP, when the number of items to be assigned equals the number of storage locations, the problem is classified as a quadratic assignment problem (QAP). When the number of items exceeds the number of storage locations and one location can store multiple products, the problem can be regarded as a knapsack problem. Kumar *et al.* (2021) found that, compared to other meta-heuristic algorithms, TS requires the shortest time and achieves the smallest error when solving small-scale QAP instances. Achary *et al.* (2021) compared the performance of six meta-heuristic algorithms for solving QAP and concluded that TS tends to yield better results in practical applications than other algorithms. Reyes *et al.* (2019) summarized previous studies employing meta-heuristic algorithms for SLAP and identified GA and TS as the most commonly used methods.

2.3. Genetic algorithm

GA is a type of meta-heuristic algorithm used to solve optimization problems. It was first proposed by Holland (1975). The concept is inspired by the natural evolutionary process of “survival of the fittest,” where the strong survive, and the weak are eliminated. GA simulates biological evolution through genetic and hereditary mechanisms: parent genes undergo recombination and mutation to produce offspring genes. After multiple generations of evolution, genes with higher fitness values are preserved and propagated, while those with lower fitness values are eliminated. This process gradually improves

the overall fitness of the genes, enabling the algorithm to find the optimal solution. Common genetic operators in GA include selection, crossover, and mutation. Selection” means chromosomes with better fitness values are more likely to be chosen for reproduction; “crossover” is the process by which two parents produce two offspring; and “mutation” refers to a change occurring in one gene of the offspring, typically with a very low probability.

To execute a GA, the problem parameters must first be transformed into genes and encoded. The encoded set of genes forms a chromosome. Kumar (2013) points out that the suitability of encoding methods is closely related to the nature of the problem itself. Common encoding methods include binary encoding, where chromosomes consist of strings of 0s and 1s; permutation encoding, which uses an ordered sequence of numbers suitable for solving sequencing problems; value encoding, where chromosomes can be composed of characters, integers, or real numbers; and tree encoding, where chromosomes have a tree-like structure. Next, an initial population is generated, and the population size — which corresponds to the number of chromosomes — is set. The population size greatly affects the quality of solutions and computational efficiency: a larger population size usually improves solution accuracy but increases computation time, while a smaller size may lead to premature convergence and poorer solution quality. After generating the initial population, some chromosomes are selected as parents. Common selection strategies include elitism selection, roulette wheel selection, and tournament selection. Then, pairs of parent chromosomes undergo crossover to produce offspring. The proportion of parents involved in crossover is determined by the crossover rate, which is typically set between 0.5 and 1.0 (Lin *et al.*, 2003). Common crossover methods include one-point crossover, two-point crossover, and uniform crossover. To maintain population diversity, a mutation rate is set to control the number of genes undergoing mutation. The mutation rate is usually set between 0.001 and 0.05 (Lin *et al.*, 2003). Finally, a stop criterion is established to end the iteration process. Typical stop criteria include reaching a predetermined number of generations or exceeding computational resource limits.

2.4. TS algorithm

TS is a meta-heuristic algorithm originally proposed by Glover in 1989, designed to solve combinatorial optimization problems. TS builds upon the hill climbing algorithm by incorporating an adaptive memory mechanism. It maintains a tabu list, which records previously visited solutions to prevent revisiting them, thereby avoiding cycles and local optima traps. This approach enables the algorithm to escape local optima and find near-optimal solutions within a reasonable timeframe. When employing TS to solve a problem, the process begins with the generation of an initial solution. A neighborhood of candidate solutions is then constructed based on this initial solution, and the best solution within this neighborhood is selected to serve as the starting point for the next iteration. This process is repeated iteratively, with new neighborhoods explored, until a predefined stopping criterion is satisfied.

The TS algorithm consists of six fundamental components: initial solution, neighborhood solution, move, tabu list, aspiration criterion, and stop criterion. A “move” refers to the process of transitioning from the initial solution to a neighborhood solution. Common move methods include the K-OPT method and the insertion method. The K-OPT method generates a neighborhood solution by exchanging K edges, where K is usually set to 2 or 3. The insertion method randomly selects two nodes and inserts one node either before or after the other, forming a new solution. The tabu list records the best neighboring solutions that have been explored during the search process. The duration for which a solution remains in the tabu list is referred to as the tabu tenure. The tabu list usually operates under a first-in, first-out policy; the longer the list, the longer the tabu tenure. The length of the list is typically determined based on the problem characteristics. Tsubakitani and Evans (1998) emphasized that the length of the tabu list has a substantial impact on the performance of the algorithm. A list that is too short may cause the search to cycle among previously visited local optima, thereby hindering the discovery of the global optimum. On the other hand, an excessively long list can lead to increased computational overhead and overly restrictive search behavior, potentially steering the search away from high-quality solutions. Glover (1990) suggested an optimal tabu list length of 7 in general settings. Based on experiments involving the Traveling Salesman Problem, Tsubakitani and Evans (1998) recommended a tabu list length of $N/4$ for 2-OPT-based TS algorithms, and a length between $N/16$ and $N/8$ for 3-OPT-based variants, where N denotes the number of cities. The aspiration criterion allows a move to be accepted even if it is TS when the neighboring solution’s objective function value is better than the current best solution, effectively overriding the TS status and updating the best-known solution. The stop criterion defines the termination conditions of the search. Generally, these conditions include reaching a pre-set maximum number of iterations or objective value; after several iterations, no better solution than the current best is found; or all feasible neighboring solutions have been added to the tabu list, leaving no further moves available under the TS rules.

The TS algorithm is a powerful metaheuristic designed for solving combinatorial optimization problems. It has been widely applied to classic problems such as the traveling salesman problem and the vehicle routing problem. For example, Tong *et al.* (2022) studied a traveling salesman problem involving trucks and drones and proposed a variable neighborhood TS algorithm to solve the model. Their results demonstrated that the method could optimize delivery routes and improve transportation efficiency. Xia and Fu (2019) addressed the open vehicle routing problem by incorporating an adaptive penalty

mechanism and multi-neighborhood structures into the TS algorithm, effectively reducing logistics costs. There are also studies applying TS to solve the SLAP. For instance, Otto *et al.* (2017) tackled the SLAP in traditional manual picking warehouses by proposing a two-stage constructive heuristic to find the initial solution, followed by TS to minimize workers' picking load risk. Liu (2022) focused on RMFS systems for e-commerce and large supermarkets, applying TS to maximize product affinity within pods and minimize picking distance. Similarly, Ma *et al.* (2022) addressed the commodity storage assignment problem (CSAP), with the objective of minimizing the total number of pod movements in order to accommodate the fast-changing order demands typical in e-commerce.

Moreover, many studies combine TS with other heuristic algorithms to achieve better results. Zhang *et al.* (2020) proposed a hybrid heuristic algorithm combining GA and TS to solve the QAP. Sangaiah and Khanduzi (2022) studied transportation hub networks by simulating the effects of open hub quantities, resource allocation, and logistics disruption risks on network configuration. They employed various metaheuristic algorithms aiming to minimize total transportation costs. The results indicated that TS and simulated annealing performed better than other algorithms in terms of accuracy and computational time, respectively. Additionally, the study developed a fast and robust hybrid algorithm combining the strengths of TS and simulated annealing, successfully addressing large-scale practical problems.

3. RESEARCH METHODOLOGY AND SOLUTION FRAMEWORK

3.1. Problem description

The subject of this study is the automated picking system used by Company G, a well-known computer hardware manufacturer in Taiwan. Company G primarily produces motherboards, graphics cards, desktop computers, laptops, and other computer-related peripherals and components. Taking the production of computer motherboards as an example, each motherboard may require between 200 and over 400 different types of components. In terms of quantity, a single product model may demand thousands of individual parts. These components are typically supplied in the form of reels, resulting in more than 1,200 reels per product model. Before initiating the production process, all required materials must be picked in advance to avoid any material shortages during manufacturing. Previously, the picking process at Company G was manual, requiring approximately ten workers to spend three to four hours concurrently completing the picking tasks for a single work order. This manual approach led to low picking efficiency and a high risk of human error, which could delay the start of production, extend overall manufacturing time, and postpone the completion schedule. To address these issues, Company G implemented an RMFS to handle the complex and diverse material requirements. The adoption of RMFS has helped reduce the need for manual labor, streamline the picking process, and significantly improve operational efficiency.

Company G's RMFS warehouse space is primarily divided into three main areas: operational areas, the storage area, and charging stations. The RMFS system consists of four operational zones, each simultaneously handling picking and replenishment processes. The storage area contains a total of 216 storage locations, with each location holding one pod. Each pod has two usable sides, and each side comprises 72 compartments, resulting in a total of 31,104 compartments available for storing materials. At any given time, picking or replenishment operations can be performed on only one side of a pod; the opposite side must be rotated into position before it can be accessed. Additionally, there are two charging stations for AGVs to recharge. As a manufacturer of computer hardware, Company G primarily uses the RMFS system to store components for the surface mount technology production process. Due to the nature of surface mount technology, the materials are packaged in reels—uniform in size—commonly referred to as “reel-type components.” Each compartment in a pod can hold up to 30 reels. Currently, the warehouse stores over 4,000 different stock-keeping units, with a total inventory of more than 42,000 reels, representing a high level of material variety and complexity.

Currently, Company G adopts a random assignment strategy for pod allocation. Warehouse operations are managed manually using heuristic or experience-based rules, where available storage locations are selected at random during the put-away process. This approach may lead to suboptimal picking efficiency. For instance, storing high-demand materials in pods located far from the operational zones increases travel distance, which directly extends picking times. Furthermore, if materials with low correlation or unrelated demand patterns are stored together in the same pod, the AGV might need to retrieve that pod multiple times to fulfill orders. This results in increased pod movements, longer overall picking distances, and extended picking times, ultimately reducing the efficiency of the picking process.

In the storage area, movable pods are used to store reels. When a task is triggered—such as the generation of a customer order or a replenishment request—the AGV navigates to the pod containing the required materials, transports the pod to the designated operational area, and then returns the pod to its original storage location upon task completion. AGVs navigate through aisles between pods by scanning two-dimensional QR codes embedded on the floor. To accurately model Company G's operational environment, this study employs the Manhattan distance metric to calculate the total travel distance of AGVs. Specifically, in a two-dimensional plane, the Manhattan distance between two points, $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, is defined as:

$$d(P_1, P_2) = |x_1 - x_2| + |y_1 - y_2|$$

The operational mechanism of Company G's RMFS is as follows: when a workstation receives an order, the system assigns an available AGV to retrieve the corresponding pod from the storage area based on the order's material requirements. The AGV then navigates to the specified storage location, transports the required pod to the workstation, and facilitates the picking operation. To isolate the impact of storage location assignment on AGV travel distance, this study considers a single AGV and a single workstation. This simplified setting eliminates the influence of task allocation and AGV coordination, allowing the effectiveness of the proposed storage layout strategy to be evaluated independently. The resulting storage assignment can also serve as a basis for future extensions to multi-AGV and multi-workstation environments. Moreover, since this study focuses on material assignment and pod travel distance, the additional handling time associated with pod rotation is not explicitly considered in the current optimization model. A figure showing the warehouse layout under studied is shown in Figure 1.

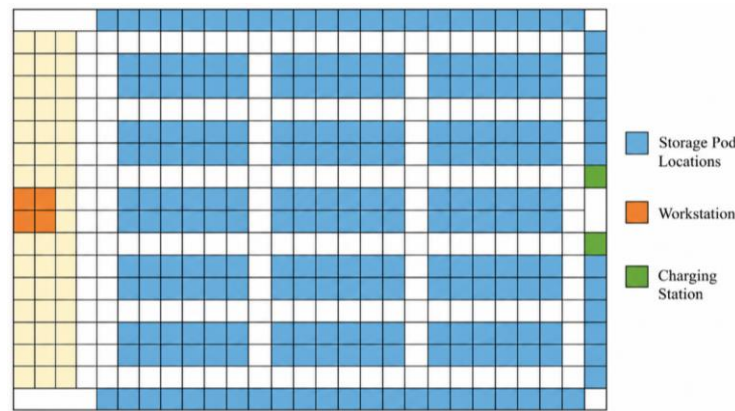


Figure 1. The warehouse layout of Company G

The following assumptions are made for the scenario considered in this study:

- (1) The initial layout of the warehouse system is known and fixed.
- (2) There is only one working station in the warehouse.
- (3) At the beginning of operations, the AGV is assumed to be located in the operational area.
- (4) The storage location of each pod is fixed; after completing a picking task, the pod must be returned to its original position.
- (5) Order information is known in advance, and picking operations are performed only when sufficient inventory is confirmed in the warehouse.
- (6) Orders cannot be split; each order is fulfilled in a single batch without partial processing.
- (7) There is only one AGV in the system, so potential conflicts between multiple AGVs are not considered.
- (8) The AGV is assumed to have sufficient battery power; charging requirements are not taken into account.
- (9) The additional handling time associated with pod rotation is not explicitly considered.

3.2. Mathematical programming model development

This study develops a two-stage mathematical programming model that builds on the two-stage problem structure of Liu (2022) and is modified to represent the operational characteristics and constraints of the manufacturing-oriented RMFS considered in this study. In Stage 1, the formulation explicitly considers material compartment requirements and the aggregate capacity of each pod, while Stage 2 incorporates order-level pod retrievals to minimize the total round-trip travel distance required to fulfill all orders. The mathematical formulation provides a formal representation of the problem structure and constraints, while the actual solution procedure described in Section 3.3 is implemented using heuristic and metaheuristic algorithms.

The first stage focuses on a material-to-pod assignment model by considering correlations among materials based on historical order data. The objective of this stage is to maximize the correlation among materials stored within the same pod, thereby reducing the total number of pod movements required during picking. The second stage focuses on a pod-to-location assignment model, in which each pod is assigned to an appropriate storage location to minimize the total picking distance.

In the proposed solution procedure, Apriori-based association analysis and DBSCAN clustering are used for material grouping and assignment in Stage 1, while a hybrid GA–TS procedure is applied to pod storage location optimization in Stage 2. Based on the assumptions outlined earlier, the material assignment model for Stage 1 is first formulated below. The corresponding parameters and decision variables are summarized in Table 1.

Table 1. Notation of decision variables and parameters in the material assignment model

	Symbol	Description
Set	M	Set of materials
	P	Set of pods
Subscript	i, j	Material indices: $i, j \in M$
	p	Pod index: $p \in P$
Parameters	s_{ij}	Correlation coefficient between materials i and j , where $0 \leq s_{ij} \leq 1$, $i, j \in M$
	D_i	Number of compartments required by material i
	L	Maximum number of compartments available in a pod ($L=144$ in the case study)
Decision variables	x_{ip}	Binary variable indicating whether material i is stored in pod p ; equals 1 if true, 0 otherwise.
	y_{ip}	Integer variable representing the number of compartments material i assigned to pod p .

For $i \neq j$, the correlation coefficient s_{ij} is defined as

$$s_{ij} = \frac{\text{Number of orders containing both material } i \text{ and material } j}{\text{Number of orders containing material } i + \text{Number of orders containing material } j}$$

For $i = j$, the correlation coefficient s_{ij} is defined as

$$s_{ij} = 1$$

The coefficient s_{ij} is symmetric ($s_{ij} = s_{ji}$) and bounded between 0 and 1, where a higher value indicates a strong association between material i and j for co-storage.

Maximize

$$\sum_{i \in M} \sum_{j \in M} \sum_{p \in P} s_{ij} x_{ip} x_{jp}, \quad (1)$$

Subject to

$$\sum_{p \in P} x_{ip} \geq 1, \forall i \in M \quad (2)$$

$$\sum_{i \in M} y_{ip} \leq L, \forall p \in P \quad (3)$$

$$\sum_{p \in P} y_{ip} = D_i, \forall i \in M \quad (4)$$

$$y_{ip} \leq L x_{ip}, \forall i \in M, \forall p \in P \quad (5)$$

$$y_{ip} \geq x_{ip}, \forall i \in M, \forall p \in P \quad (6)$$

$$y_{ip} \in \mathbb{Z} \geq 0, x_{ip} \in \{0,1\}, \forall i \in M, \forall p \in P \quad (7)$$

The objective function (1) aims to maximize the correlation among materials stored within the same pod. Constraint (2) ensures that each material is assigned to at least one pod. Constraint (3) ensures that the total number of compartments occupied by all materials assigned to pod p does not exceed its physical capacity L . Constraint (4) enforces consistency between the required number of compartments for each material and the number of compartments assigned to that material within the pods. Constraint (5) ensures that material i can occupy compartments in pod p only if it is assigned to that pod. Constraint (6) ensures that if material i is assigned to pod p , at least one compartment is allocated to that material in the pod. Finally, constraint (7) defines the feasible ranges for the decision variables.

Based on the material-to-pod assignment obtained from Stage 1, the second-stage model determines the storage location of each pod and the pods required to fulfill each order. The objective is to minimize the total round-trip travel distance required for all pod retrievals. Table 2 organized required notations for this location assignment model.

Table 2. Notations of decision variables and parameters in the location assignment model

	Symbol	Description
Set	S	Set of storage locations.
	Q	Set of working stations.
	N	Set of orders.
	P	Set of Pods.
	M	Set of materials.
Subscript	s	Pod index, $s \in S$.
	q	Working station index, $q \in Q$.
	n	Order index, $n \in N$.
	p	Pod index, $p \in P$.
	i	Material index, $i \in M$.
Parameters	R_{sq}	Distance from storage location s to working station q .
	x_{ip}	Binary parameter equal to 1 if material i is assigned to pod p in Stage 1, and 0 otherwise.
	w_{ni}	Binary parameter equal to 1 if material i is required by order n , and 0 otherwise.
Decision variables	z_{npq}	Binary decision variable equal to 1 if pod p is retrieved to workstation q to fulfill order n , and 0 otherwise.
	b_{ps}	Binary decision variable equal to 1 if pod p is assigned to storage location s , and 0 otherwise.
	u_{npsq}	Auxiliary binary variable equal to 1 if pod p , assigned to storage s , is retrieved to workstation q for order n and 0 otherwise.

Minimize

$$\sum_{n \in N} \sum_{p \in P} \sum_{s \in S} \sum_{q \in Q} 2R_{sq} u_{npsq}, \quad (8)$$

Subject to

$$\sum_{s \in S} b_{ps} = 1, \quad \forall p \in P \quad (9)$$

$$\sum_{p \in P} b_{ps} = 1, \quad \forall s \in S \quad (10)$$

$$\sum_{q \in Q} \sum_{p \in P} z_{npq} x_{ip} \geq w_{ni}, \quad \forall i \in M, \forall n \in N \quad (11)$$

$$u_{npsq} \leq z_{npq}, \quad \forall n \in N, \forall p \in P, \forall s \in S, \forall q \in Q \quad (12)$$

$$u_{npsq} \leq b_{ps}, \quad \forall p \in P, \forall s \in S \quad (13)$$

$$u_{npsq} \geq z_{npq} + b_{ps} - 1, \quad \forall n \in N, \forall p \in P, \forall s \in S, \forall q \in Q \quad (14)$$

$$z_{npq}, b_{ps}, u_{npsq} \in \{0, 1\}, \quad \forall p \in P, \forall n \in N, \forall s \in S, \forall q \in Q \quad (15)$$

Objective function (8) minimizes the total round-trip travel distance required to fulfill all orders. For each order, the distance contribution of a retrieved pod is determined by its assigned storage location and the workstation to which it is transported. The factor of two accounts for the retrieval-and-return cycle, since each pod must be returned to its original storage location after picking. Constraint (9) ensures that each pod is assigned to exactly one storage location. Constraint (10) restricts that each storage location contains exactly one pod. Constraint (11) guarantees that every material required by an order is available on at least one pod retrieved to a workstation. Constraints (12)–(14) link the pod-retrieval variable z_{npq} and the pod-location variable b_{ps} through the auxiliary variable u_{npsq} . Specifically, $u_{npsq}=1$ only when pod p is retrieved for order n and is assigned to storage location s . Finally, Constraint (15) defines the binary domains of the decision variables.

3.3. Algorithm workflow

To solve the two-stage SLAP described above, a corresponding two-stage algorithmic framework is developed in this study. The mathematical models presented in Section 3.2 provide the conceptual formulation of the problem, while the algorithms are designed to obtain practical solutions for the large-scale RMFS studied in this research.

In the first stage, the material assignment algorithm determines how materials should be allocated to pods according to their correlation relationships and usage proportions. The output of this stage is the material-to-pod allocation result, which defines the composition of each pod. Based on this result, the second stage focuses on determining the optimal storage locations for the pods within the warehouse. Since the problem size is large and difficult to solve directly using exact optimization methods, a hybrid heuristic algorithm combining a GA and TS is proposed to efficiently search for high-quality solutions. Through this two-stage process, the output of the material assignment stage directly serves as the input for the storage location assignment stage, thereby establishing a clear connection between the mathematical models and the proposed solution algorithms.

The flowchart of the material assignment algorithm used in this study is shown in Figure 2. The following sections provide a detailed explanation of the algorithmic steps.

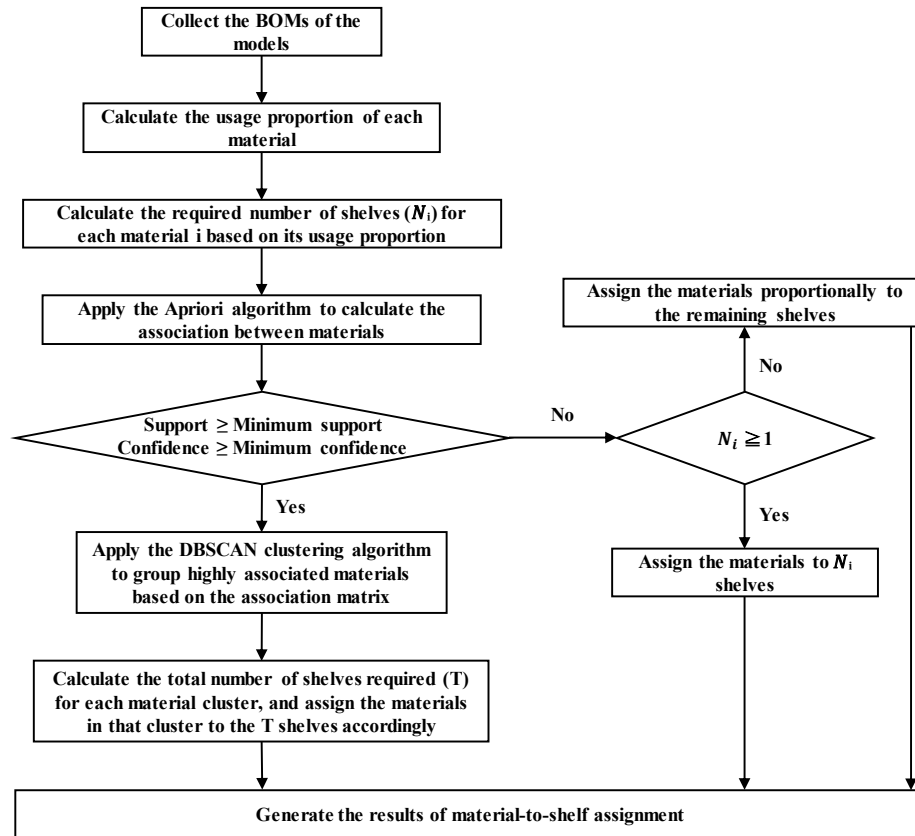


Figure 2. Flowchart of the material assignment algorithm in this study

Step 1: Collect BOM Data for Product Models

First, collect the BOM data for the product models. The dataset should include the following fields: product model, material number, and required quantity to facilitate subsequent algorithms and analyses.

Step 2: Calculate the Number of Pods to Be Assigned to Each Material in the Warehouse

Calculate the usage proportion of each material based on its required quantity in the BOM data.

Step 3: Calculate the Correlations Among Materials Using the Apriori Algorithm, Classifying Materials into Two Categories: High Correlation and Low Correlation

The Apriori algorithm is one of the commonly used algorithms in association analysis, first proposed by Rakesh Agrawal *et al.* in 1993. This study employs the Apriori algorithm to calculate the correlation for each material. Borgelt and Kruse (2002) pointed out that association rules are typically evaluated using two metrics:

- (1) **Support:** Represents the frequency proportion of material i occurring within the entire material set. The parameter $min_support$ is a predefined minimum support threshold set prior to running the Apriori algorithm. During the correlation calculation, materials with support below $min_support$ are filtered out, effectively removing infrequent materials. Support is calculated as:

$$Support(i) = \frac{freq(i)}{N}$$

$$Support(i, j) = \frac{freq(i, j)}{N}$$

- (2) **Confidence:** Represents the conditional probability that material j appears given that material i appears, thereby indicating the strength of association between i and j . The parameter $min_confidence$ is a predefined minimum confidence threshold set before applying the Apriori algorithm. Confidence is calculated as:

$$Confidence(i \rightarrow j) = \frac{Support(i, j)}{Support(i)} = \frac{freq(i, j)}{freq(i)}$$

Additionally, another commonly used metric is Lift.

- (3) **Measures the strength of association between materials i and j .** A lift value greater than 1 indicates a positive correlation between two materials, whereas a value less than 1 indicates a negative correlation. In the proposed Apriori procedure, Lift is used as the third filtering criterion, following support and confidence. The threshold value ($min_lift = 2$) is specified in Section 4.2. The formula is as follows:

$$Lift(i \rightarrow j) = \frac{Confidence(i \rightarrow j)}{P(j)}$$

Step 4: Cluster Highly Correlated Materials Using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) Algorithm

DBSCAN is applied only to the high-correlation materials identified by the Apriori algorithm in the preceding step. Materials not identified as highly correlated are excluded from the DBSCAN procedure, treated as low-correlation materials, and subsequently assigned to pods according to the allocation rules described in Steps 5 and 6. The core concept of DBSCAN is to identify regions with sufficiently high point density as clusters, while treating low-density regions as noise. Clusters are defined using two key parameters: Eps (ϵ) and $MinPts$. Eps specifies the neighborhood radius, and $MinPts$ is the minimum number of points required within the Eps radius for a point to be considered a core point. Core points are those with at least $MinPts$ neighbors within the Eps radius. Border points are those that fall within the Eps radius of a core point and belong to the cluster, but do not have enough neighbors themselves to be considered core points. Noise points refer to those located in low-density regions and are considered outliers. In this study, the correlation matrix generated in Step 3 is first transformed into a distance matrix, which serves as the input for this step. DBSCAN is then applied to cluster the highly correlated materials.

Step 5: Calculate the Number of Pods to Be Assigned to Each Cluster of Highly Correlated Materials

After completing Step 4, the materials are grouped into clusters based on their correlation, with highly correlated materials assigned to the same cluster. Next, using the information obtained in Step 2 regarding the number of pods assigned to each individual material, the total number of pods required for each material cluster is calculated. To ensure that the number of pods allocated to each cluster is an integer, any fractional values are rounded up. Finally, each material cluster is assigned the corresponding number of pods to meet the demand.

Step 6: Allocation of Low-Correlation Materials with High Usage

Based on the number of pods allocated to each individual material as determined in Step 2, if the allocated number of pods is greater than or equal to one, it indicates that the material requires one or more pods. In this study, such materials are assigned to pods according to their allocated pod quantities.

Step 7: Allocation of Low-Correlation Materials with Low Usage

According to the pod allocation information from Step 2, if the allocated number of pods is less than one, it indicates lower demand for the material, allowing it to share a pod with other materials. These materials are sorted in descending order by their allocated pod quantities and sequentially assigned to pods.

Step 8: Combine the Results to Obtain the Final Allocation

The material allocation results from Step 5, Step 6, and Step 7 are merged to produce the final material assignment outcome. The second stage is the Storage Location Assignment Algorithm, which assigns pods to designated storage locations, aiming to minimize the total picking travel distance. This study focuses on optimizing the storage location assignment for G Company's RMFS with the objective of minimizing the total picking distance. To achieve this goal, the BOM and historical order data from Company G were first collected. After performing material clustering and analyzing the correlations between clusters, the inbound and outbound frequencies of these material clusters were examined. The optimal storage location assignment was then determined using a hybrid algorithm. This algorithm primarily employs a GA combined with the local search capabilities of TS. While searching for the global optimum, the integration of TS enhances the ability to explore local optima, increasing solution diversity and improving results through iterative optimization. The flowchart of the storage location assignment algorithm developed in this study is shown in Figure 3. The following sections provide a detailed explanation of the algorithm process.

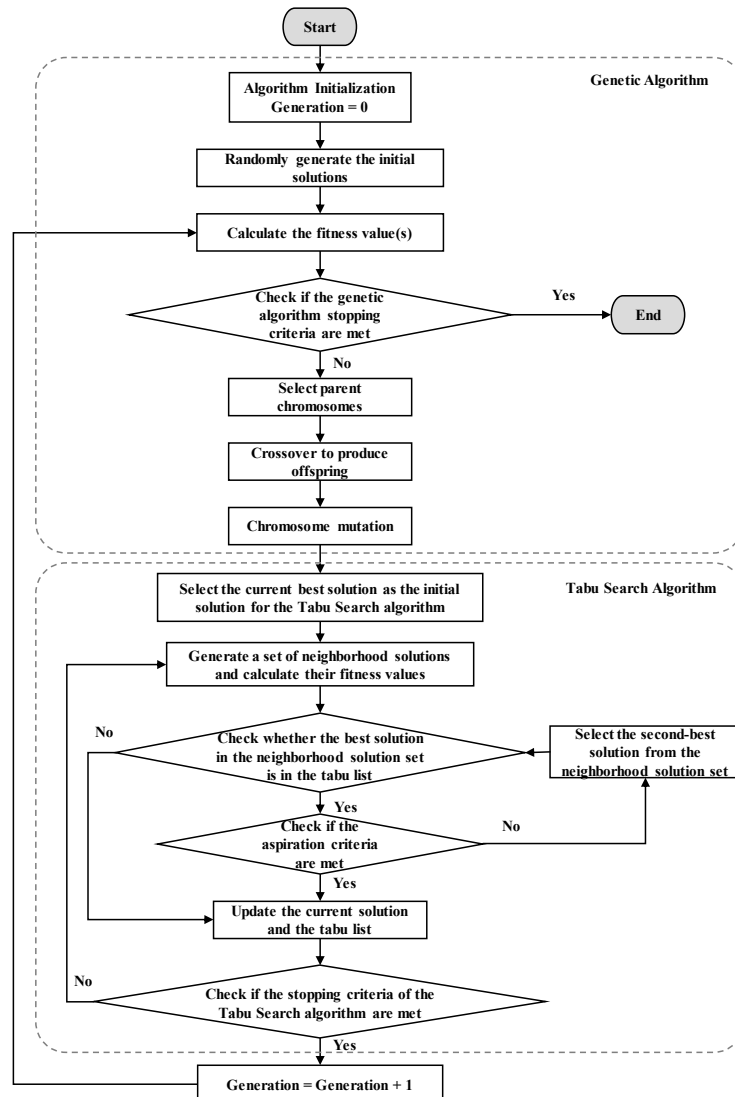


Figure 3. Flowchart of the storage location assignment algorithm developed in this study

Step 1: Encoding Pods and Storage Locations

To represent the storage location assignment results, the pods and storage locations must first be encoded. This study employs permutation encoding to assign codes to both pods and storage locations, which serve as the basis for subsequent problem-solving.

Step 2: Initialization and Parameter Setting

Before executing the algorithm, it is necessary to set various parameters. These include the selection, crossover, and mutation strategies for the GA, as well as the population size, crossover rate, mutation rate, and maximum number of iterations. Additionally, parameters required for the TS algorithm, such as the length of the TS list and the maximum number of iterations, must also be specified.

Step 3: Generate Initial Solution and Calculate Fitness Value

In this study, the initial solution is generated randomly and serves as the initial population chromosome. Material categories with higher inbound and outbound frequencies are placed in storage locations closer to the working stations to reduce the total picking distance. To evaluate the quality of a solution, a fitness value is calculated as the objective measure for storage location assignment. The fitness value corresponds to the total picking distance required to complete all orders. To reflect actual warehouse operations, the distance is calculated using the Manhattan distance metric. The order data used for evaluation is provided by Company G, covering one week.

Step 4: Stopping Criteria of the GA

Two stopping criteria are established for the algorithm in this study: (1) reaching the maximum number of iterations, or (2) after a specified number of iterations, the best objective value obtained does not improve upon the current best solution.

Step 5: Execute GA Operator — Selection

This study employs the roulette wheel selection method to choose parent chromosomes. Chromosomes with higher fitness values have a greater probability of being selected as parents.

Step 6: Execute GA Operator — Crossover

Uniform crossover is used in this study to enhance solution diversity. The crossover rate is set between 0.5 and 1.0, based on the recommendations of Lin *et al.* (2003).

Step 7: Execute GA Operator — Mutation

To prevent premature convergence during the computation process—which may result in finding only local optima rather than the global optimum—this study adopts a one-point mutation strategy. The mutation rate is set at 0.001.

Step 8: Execute TS Algorithm — Select Initial Solution and Perform Moves

The solution with the best fitness value obtained from the GA steps is selected as the initial solution for the TS algorithm. Moves are then performed to generate a set of neighborhood solutions. The move method used is the 2-OPT technique, which involves swapping two edges to produce new neighborhood solutions. In the context of the RMFS Storage Location Assignment Problem, this corresponds to randomly swapping two pods to create a new set of neighborhood solutions. The fitness values of all neighborhood solutions are calculated, and the neighborhood solution with the best fitness value is selected as the best neighborhood solution. If this solution improves upon the current best solution, it is updated accordingly.

Step 9: Execute TS Algorithm — Update Tabu List

In applying the TS algorithm, to prevent cycling and repeated selection of solutions, the best neighborhood solution selected is added to the tabu list. During the tabu tenure, this solution is prohibited from being selected again. The tabu list operates on a first-in, first-out basis, meaning that after a certain number of iterations, entries are removed from the list. Additionally, based on the aspiration criterion mechanism, if a neighboring solution with an objective function value better than the current best solution is found during the search process, this solution can be removed from the tabu list and updated as the current best solution. This prevents superior solutions from being improperly excluded due to their TS status.

The TS algorithm continuously performs local search by repeatedly executing Step 8 and Step 9 until the stopping criterion of the TS algorithm is met. At this point, a storage location assignment solution is obtained, which replaces the current best solution and becomes the parent for the next generation, thus completing the TS process. Subsequently, the global search continues by executing the GA steps (Step 5, Step 6, and Step 7), evolving to the next generation until the GA's stopping criterion is satisfied. Ultimately, this process yields an optimal storage location assignment solution that minimizes the total picking distance required to complete all orders.

3.4 Key performance indicators

This study evaluates the effectiveness of the storage location assignment using three key performance indicators, defined as follows:

- (1) **Number of Pod Movements:** Records the total number of pod movements required during the picking process, i.e., the number of pods moved to complete all orders. A pod movement is defined as one complete retrieval-and-return cycle, in which a pod is retrieved from its storage location, transported to the workstation for picking, and subsequently returned to its storage location. Thus, each round trip is counted as one pod movement.
- (2) **Total AGV Picking Distance:** Records the total distance traveled by the AGV to complete all orders, measured in meters. The objective is to minimize this distance. Since the RMFS system of Company G assigns each pod a fixed storage location, the pods must be transported back to their original storage locations after picking. Therefore, the total picking distance is calculated as twice the Manhattan distance between the working station and the pod.
- (3) **Algorithm Execution Time:** The total time required by the algorithm to generate the storage location assignment solution, measured in hours. The goal is to minimize this time.

4. ALGORITHM IMPLEMENTATION AND RESULT ANALYSIS

4.1. Description of implementation data

Since the RMFS at Company G primarily stores high-turnover and high-demand materials, the inventory within the system is updated approximately once a week. Accordingly, the dataset used in this study covers one week, specifically from May 8, 2023, to May 14, 2023. The selected production data corresponds to a normal production period and does not include any unusual production schedules or exceptional operational events. Therefore, the order mix is considered representative of the company's routine manufacturing environment. The dataset includes the following fields: work order number, product model, material part number, and quantity used.

Currently, the replenishment strategy employed by Company G follows a random placement approach. This strategy does not consider material usage frequency or demand; materials are assigned to any pod with available storage capacity. This existing random placement strategy is used as the baseline for evaluating the effectiveness of the proposed material assignment approach.

4.2. Data implementation

This study developed a two-stage storage location assignment algorithm. The following sections provide a detailed explanation of the material assignment algorithm and the storage location assignment algorithm. The algorithms were implemented using the Python programming language (version 3.9.12) on a computer equipped with a 2.10 GHz processor and 32.0 GB of RAM.

The purpose of the material assignment algorithm stage is to allocate a total of 1,121 types of materials to 216 pods within the automated warehouse. In this study, the Apriori algorithm is employed to evaluate the associations between materials, and based on the strength of these associations, all materials are classified into two groups: those with high association and those with low association. The parameters used for the Apriori algorithm in this study are set as follows: $min_support = 0.2$, $min_confidence = 0.3$, $min_lift = 2$, and $max_length = 2$. The parameters of the Apriori algorithm were determined through preliminary experiments conducted during the data preprocessing stage. Since the dataset contains a large number of materials, setting a relatively higher support threshold helps filter out infrequent combinations and reduces the generation of weak or noisy association rules. The objective of applying the Apriori algorithm in this study is to identify strongly correlated materials that frequently appear together in orders, which are then used as the basis for the subsequent clustering process. Therefore, the selected parameter settings allow the algorithm to capture the most representative material associations while maintaining computational efficiency. These parameter values are adopted as fixed settings for the case study and are not intended to represent universally optimal parameter configurations. After executing the Apriori algorithm with these settings, a total of 131 materials are identified as having strong associations with other materials. Based on these results, an association matrix is generated to represent the strength of relationships among the identified materials.

Since the DBSCAN algorithm is a density-based clustering method, the association matrix is first transformed into a distance matrix before clustering is performed. To reduce the number of outliers and obtain a reasonable number of clusters from the 131 high-association materials, the DBSCAN algorithm is applied with the following parameter settings: $eps = 0.3$ and $min_samples = 4$. After dividing the materials into six clusters, the usage volume of each material is used as the basis for determining the number of pods allocated. For the low-association materials, allocation is performed sequentially to the

remaining pods according to their respective usage volumes. Ultimately, this process yields a material assignment result that reflects both the association-based grouping and the pod allocation informed by usage volume.

The storage location assignment algorithm stage aims to allocate 216 pods within the automated warehouse to 216 storage locations. This study employs a hybrid heuristic algorithm that combines the global search capability of a GA with the enhanced local search ability of a TS algorithm. The objective is to identify an optimal storage location assignment that minimizes the total picking distance required to complete a single order. The following section provides a detailed description of the algorithmic settings used in this study.

(1) Fitness Value

The fitness value in the GA is defined as the reciprocal of the total picking distance, with the objective of maximizing this value. Since a single material may be stored across multiple pods, this study assumes that the system will automatically select the pod closest to the working station for picking to improve picking efficiency. Furthermore, if different materials required by a single work order are stored in the same pod, the AGV only needs to make one trip to complete the picking task, eliminating the need for additional transport trips. This approach enhances warehouse operational efficiency and reduces both time and cost. In addition, to align with the actual operation of Company G's RMFS, the distance is calculated using the Manhattan distance formula. Because the AGV must return the pod to its original storage location after the picking operation at the working station is completed, the total distance is computed as twice the Manhattan distance between the pod and the working station.

(2) Parameter Settings

To evaluate the effects of different parameter settings, factorial experiments were conducted using combinations of GA iteration numbers (20, 50, and 100), population sizes (30 and 50), and TS iteration numbers (20, 50, and 100). A pure GA configuration without TS was also evaluated for comparison. In total, 24 parameter combinations were tested to identify an appropriate parameter setting considering both solution quality and computational efficiency. The results are presented in Tables 3 and 4.

The results indicate a trade-off between solution quality and computational effort. Increasing the number of GA iterations generally reduces the total picking distance but requires longer computation time. Incorporating TS consistently improves solution quality compared with the pure GA, demonstrating the benefit of enhanced local search in avoiding local optima. Increasing the GA population size provides only limited additional improvement while increasing computation time. Therefore, the parameter settings should be selected by balancing solution quality and computational efficiency.

Table 3. Test results of parameter combinations for the hybrid heuristic algorithm

GA: iteration_num	TS: iteration_num	GA: pop_size	Distance (m)	execution time (hr)
20	20	30	4,504,400	0.25
		50	4,449,600	0.36
	50	30	4,485,200	0.47
		50	4,448,000	0.60
	100	30	4,471,200	1.81
		50	4,445,800	1.01
50	20	30	4,417,400	0.58
		50	4,458,000	0.90
	50	30	4,468,600	1.17
		50	4,456,800	1.58
	100	30	4,492,200	2.18
		50	4,436,000	2.47
100	20	30	4,412,600	1.73
		50	4,459,600	3.06
	50	30	4,436,200	2.37
		50	4,469,200	3.64
	100	30	4,431,200	4.59
		50	4,461,600	5.49

Note: the selected parameter combination adopted in the subsequent comparative analyses is shown in bold.

Table 4. Test results of parameter combinations for the GA

GA: iteration_num	GA: pop_size	Distance (m)	execution time (hr)
20	30	5,952,000	0.07
	50	5,859,800	0.20
50	30	5,848,200	0.18
	50	6,013,000	0.49
100	30	6,087,400	0.36
	50	5,872,800	1.01

Note: The selected parameter combination adopted in the subsequent comparative analyses is shown in bold.

Based on the overall comparison of the experimental results, the hybrid GA–TS algorithm with a GA iteration number of 100, a population size of 30, and a TS iteration number of 20 provides the best trade-off between solution quality and computational efficiency. Therefore, this parameter combination is adopted for the subsequent comparative analyses.

Regarding other parameter settings, the crossover rate in the GA is set to 0.6 based on the recommendation of Lin *et al.* (2003). Since this study incorporates the TS algorithm to enhance local search capabilities, the mutation rate is kept low to avoid excessive variability; it is set to 0.001. As for the length of the tabu list in the TS algorithm, a list that is too short may cause the search to cycle repeatedly around the same local optima without reaching the global optimum, while an excessively long list can lead to increased computational time. Glover (1990) suggests that an optimal tabu list length is 7; therefore, this study also sets it to 7.

The above experiments were conducted with both the material assignment algorithm and the storage location assignment algorithm running simultaneously. Since different material assignment results can lead to varying performances in storage location assignment, an additional experiment was performed using a fixed material assignment result to evaluate the effectiveness of the proposed hybrid heuristic algorithm. The experimental results are presented in Table 5. From the results, it can be observed that convergence is achieved within a certain number of iterations. Specifically, when the GA is set with 50 iterations, a population size of 30, and the TS algorithm with 20 iterations, the optimal solution can be found in the shortest execution time. It is also observed that most parameter configurations converge to the same picking distance of 4,387,200 *m* once the material assignment is fixed. This result suggests that, for the case instance considered, the Stage 2 storage location assignment is relatively stable across different GA–TS parameter settings, and additional computational effort beyond a certain level provides limited improvement in solution quality.

Table 5. Test results of parameter combinations using the same material assignment result

GA: iteration_num	TS: iteration_num	GA: pop_size	min_distance (m)	execution time (hr)
20	20	30	4,387,800	0.19
		50	4,390,000	0.29
	50	30	4,387,200	0.59
		50	4,387,400	1.23
	100	30	4,387,200	1.48
		50	4,387,200	0.80
50	20	30	4,387,200	0.48
		50	4,387,400	0.72
	50	30	4,387,200	0.95
		50	4,387,200	1.23
	100	30	4,387,200	1.77
		50	4,387,200	2.03
100	20	30	4,387,200	0.96
		50	4,387,200	1.49
	50	30	4,387,200	2.22
		50	4,387,200	2.52
	100	30	4,387,200	5.43
		50	4,387,200	4.78

4.3. Result analysis and validation

Company G currently adopts random material placement in its RMFS. To evaluate the material assignment results, the company's existing method was compared with the proposed material assignment algorithm using the number of pod movements as the performance measure. Because the company's current method relies on random placement, evaluating a single material assignment may produce an unrepresentative baseline. Therefore, 100 independent random material assignments were generated, and the mean number of pod movements was used as the baseline for comparison. The corresponding standard deviation, minimum, and maximum values were also calculated to characterize the variability of the random-placement results. As shown in Table 6, Company G's current random material placement produced an average of 3,256.3 pod movements, with a standard deviation of 40.3 across the 100 independently generated assignments. The observed number of pod movements ranged from 3,188 to 3,382. In comparison, the proposed material assignment algorithm required 1,722 pod movements, representing an average reduction of 1,534.3 movements, or approximately 47.12%. This result indicates that assigning highly associated materials to the same pods can substantially reduce the number of pod retrievals required to fulfill the orders.

Table 6. Comparison of pod movement counts

Evaluation Metric	Number of Pod Movements	Improvement Ratio (%)
Company G's Current Method: Random Material Placement	3,256 ± 40.3	-
Proposed Material Assignment Algorithm	1,722	47.12%

Note: For Company G's current random material assignment method, the reported value represents the mean ± standard deviation over 100 independently generated random material assignments. The corresponding minimum and maximum numbers of pod movements are 3188 and 3382, respectively.

Regarding the storage location assignment results, this study compares the company's current approach, a pure GA, and the proposed hybrid heuristic algorithm in terms of total picking distance and computation time. The verification data consists of order data spanning one week (May 8 to May 14, 2023). The selected production data corresponds to a normal production period and does not include any unusual production schedules or exceptional operational events. Similarly, to avoid biased outcomes, all results are averaged over 100 runs to obtain a representative baseline. The mean total picking distance, standard deviation, minimum, and maximum were calculated from these layouts. The GA and hybrid GA-TS results were obtained using the parameter settings selected from the parameter experiments. Specifically, the pure GA was executed with 50 iterations and a population size of 30, whereas the hybrid GA-TS algorithm was executed with 100 GA iterations, a population size of 30, and 20 TS iterations.

As shown in Table 7, Company G's current random-placement method produced an average total picking distance of 6,267,768 m, with a standard deviation of 164,985.2 m across 100 independently generated layouts. The observed picking distances ranged from 5,885,400 m to 6,648,200 m. Compared with this average baseline, the GA reduces the total picking distance by 6.69%, whereas the proposed hybrid GA-TS algorithm achieves a reduction of 29.60%. Although incorporating TS increases computation time, it reduces the picking distance by 1,435,600 m, corresponding to a 24.55% improvement over the pure GA. Notably, the picking distance obtained by the proposed hybrid GA-TS algorithm is lower than the minimum distance observed among the 100 random layouts, indicating that its improvement is not attributable to the selection of an unfavorable random baseline. These results demonstrate the effectiveness of the algorithm developed in this study.

Table 7. Comparison of total picking distance results

Evaluation Metric	Total Picking Distance (meters)	Computation Time (hours)	Picking Distance Improvement (%)
Company G's Current Method: Random Pod Placement	6,267,768 ± 164,985.2	-	-
Genetic Algorithm	5,848,200	0.18	6.69%
Proposed Hybrid Heuristic Algorithm	4,412,600	1.73	29.60%

Note: The result for Company G's current random-placement method is reported as the mean ± standard deviation over 100 independently generated random pod layouts. The corresponding minimum and maximum picking distances are 5,885,400 m and 6,648,200 m, respectively.

5. CONCLUSION AND RECOMMENDATIONS

With the rapid advancement of computer technology, computer components have become increasingly complex and diverse. Automating the picking process to replace manual labor can help avoid human errors and improve efficiency. This study

proposes a hybrid heuristic algorithm to address the two-stage SLAP in the RMFS. The first stage focuses on the Material Assignment Problem, where the Apriori algorithm is employed to analyze the associations between materials and cluster them accordingly, thereby reducing pod movements and enhancing picking efficiency. The second stage tackles the SLAP by developing a hybrid heuristic algorithm that combines a GA with TS to strengthen local search capabilities and determine the optimal storage location assignment.

The proposed method was applied to Company G's RMFS, using one week's worth of order data for validation and testing. Experimental results demonstrate that, compared to Company G's current random material placement approach, the proposed algorithm significantly reduces both the number of pod movements and the total AGV picking distance. These findings verify the effectiveness and feasibility of the algorithm developed in this study. The algorithm proposed in this study not only optimizes the picking process but also reduces costs and enhances operational efficiency.

In this study, the RMFS environment is simplified by assuming a single AGV and a single working station. This assumption allows the storage location assignment problem to be analyzed more clearly by isolating the influence of storage layout on travel distance. In real RMFS environments, multiple AGVs usually operate simultaneously, which may introduce additional operational issues such as path conflicts, congestion, and waiting. Although the storage layout obtained in this study focuses on minimizing travel distance, the results may still provide useful insights for multi-AGV systems because frequently accessed pods are placed closer to the workstation, which reduces the average travel distance of robots. Nevertheless, the interaction between storage layout and robot traffic management was not considered in this study. Future research may extend the model to multi-AGV and multi-workstation environments and investigate the integration of storage location assignment with robot scheduling and traffic control strategies. The storage area could also be partitioned into smaller zones to improve computational efficiency and facilitate subsequent optimization if the problem scale becomes significantly larger. Future research may also systematically investigate the sensitivity of the proposed framework to the Apriori and DBSCAN parameter settings and examine how variations in these parameters affect material grouping, pod movements, and subsequent storage location assignment performance. Furthermore, although the proposed hybrid heuristic algorithm demonstrates promising performance in the case study of Company G, its performance could be further evaluated by comparison with additional heuristic and metaheuristic algorithms commonly used for solving the SLAP in RMFS environments.

REFERENCE

- Achary, T., Pillay, S., Pillai, S.M., Mqadi, M., Genders, E., and Ezugwu, A.E. (2021). A performance study of meta-heuristic approaches for quadratic assignment problem. *Concurrency and Computation: Practice and Experience*, 33(17): e6321.
- Ang, M., Lim, Y F., and Sim, M. (2012). Robust storage assignment in unit-load warehouses. *Management Science*, 58(11): 2114-2130.
- Azadeh, K., De Koster, R., and Roy, D. (2019). Robotized and automated warehouse systems: Review and recent developments. *Transportation Science*, 53(4): 917-945.
- Bodnar, P., and Lysgaard, J. (2014). A dynamic programming algorithm for the space allocation and aisle positioning problem. *Journal of the Operational Research Society*, 65(9): 1315-1324.
- Borgelt, C., and Kruse, R. (2002). Induction of association rules: Apriori implementation. In *15th Biannual Conference on Computational Statistics (COMPSTAT)*, pp. 395-400, Berlin, Germany.
- Boysen, N., de Koster, R., and Weidinger, F. (2019). Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 277(2): 396-411.
- Chang, Y.C., Chang, K.H., Meng, H.M., and Chang, C.H. (2024). Integrating integer programming with heuristic algorithm to solve the warehouse relocation problem of automated storage and retrieval systems with multiple loading depots. *International Journal of Industrial Engineering - Theory, Applications and Practice*, 31(1): 46-64.
- Chen, L., Langevin, A., and Riopel, D. (2010). The storage location assignment and interleaving problem in an automated storage/retrieval system with shared storage. *International Journal of Production Research*, 48(4): 991-1011.
- Chiang, D.M.H., Lin, C.P., and Chen, M.C. (2011). The adaptive approach for storage assignment by mining data of warehouse management system for distribution centres. *Enterprise Information Systems*, 5(2): 219-234.

- Cruz-Domínguez, O., and Santos-Mayorga, R. (2016). Artificial intelligence applied to assigned merchandise location in retail sales systems. *South African Journal of Industrial Engineering*, 27(1): 112-124.
- de Koster, R., Le-Duc, T., and Roodbergen, K.J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2): 481-501.
- Dijkstra, A.S., and Roodbergen, K.J. (2017). Exact route-length formulas and a storage location assignment heuristic for picker-to-parts warehouses. *Transportation Research Part E: Logistics and Transportation Review*, 102: 38-59.
- Ene, S., and Öztürk, N. (2012). Storage location assignment and order picking optimization in the automotive industry. *International Journal of Advanced Manufacturing Technology*, 60(5-8): 787-797.
- Frazele, E.A., and Sharp, G.P. (1989). Correlated assignment strategy can improve any order-picking operation. *Industrial Engineering*, 21(4): 33-37.
- Glover, F. (1989). Tabu search—part I. *ORSA Journal on Computing*, 1(3): 190-206.
- Glover, F. (1990). Tabu search—part II. *ORSA Journal on Computing*, 2(1): 4-32.
- Gu, J.X., Goetschalckx, M., and McGinnis, L.F. (2007). Research on warehouse operation: A comprehensive review. *European Journal of Operational Research*, 177(1): 1-21.
- Gu, J., Goetschalckx, M., and McGinnis, L.F. (2010). Solving the forward-reserve allocation problem in warehouse order picking systems. *Journal of the Operational Research Society*, 61(6): 1013-1021.
- He, Y.Y., Wang, A.H., Su, H.L., and Wang, M.Y. (2019). Particle swarm optimization using neighborhood-based mutation operator and intermediate disturbance strategy for outbound container storage location assignment problem. *Mathematical Problems in Engineering*, 2019: 9132315.
- Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press.
- Jiao, G.S., Huang, M., Song, Y., Li, H.B., and Wang, X.W. (2026). Online joint optimization of order picking process in robotic mobile fulfillment systems. *Omega-International Journal of Management Science*, 138: 103374.
- Kim, B.S., and Smith, J.S. (2012). Slotting methodology using correlated improvement for a zone-based carton picking distribution system. *Computers & Industrial Engineering*, 62(1): 286-295.
- Kumar, A. (2013). Encoding schemes in genetic algorithm. *International Journal of Advanced Research in IT and Engineering*, 2(3): 1-7.
- Kumar, M., Sahu, A., and Mitra, P. (2021). A comparison of different metaheuristics for the quadratic assignment problem in accelerated systems. *Applied Soft Computing*, 100: 106927.
- Larco, J.A., de Koster, R., Roodbergen, K.J., and Dul, J. (2017). Managing warehouse efficiency and worker discomfort through enhanced storage assignment decisions. *International Journal of Production Research*, 55(21): 6407-6422.
- Lee, C.K.M., Lin, B.B., Ng, K.K.H., Lv, Y.Q., and Tai, W.C. (2019). Smart robotic mobile fulfillment system with dynamic conflict-free strategies considering cyber-physical integration. *Advanced Engineering Informatics*, 42: 100998.
- Li, Y., Wang, N., and Fan, F. (2025). Optimization of robotic mobile fulfillment system considering robot efficiency and picker energy consumption. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 32(5): 1254-1278.

- Lin, W.Y., Lee, W.Y., and Hong, T.P. (2003). Adapting crossover and mutation rates in genetic algorithms. *Journal of Information Science and Engineering*, 19(5): 889-903.
- Liu, Z., Lu, J.S., Ren, C.H., Chen, J., Xu, Z.L., and Zhao, G.L. (2025). Joint optimization of storage assignment and order batching in robotic mobile fulfillment system with dynamic storage depth and surplus items. *Computers & Industrial Engineering*, 200: 110767.
- Liu, M. (2022). Research on the storage location assignment problem in robotic mobile fulfillment systems based on association rules. *Frontiers in Economics and Management*, 3(3): 110-125.
- Ma, Z., Wu, G., Ji, B., Wang, L., Luo, Q., and Chen, X. (2023). A novel scattered storage policy considering commodity classification and correlation in robotic mobile fulfillment systems. *IEEE Transactions on Automation Science and Engineering*, 20(2): 1020-1033.
- Merschformann, M., Lamballais, T., de Koster, M.B.M., and Suhl, L. (2019). Decision rules for robotic mobile fulfillment systems. *Operations Research Perspectives*, 6: 100128.
- Muppani, V.R., and Adil, G.K. (2008). A branch and bound algorithm for class based storage location assignment. *European Journal of Operational Research*, 189(2): 492-507.
- Otto, A., Boysen, N., Scholl, A., and Walter, R. (2017). Ergonomic workplace design in the fast pick area. *OR Spectrum*, 39(4), 945- 975.
- Reyes, J.J.R., Solano-Charris, E.L., and Montoya-Torres, J.R. (2019). The storage location assignment problem: A literature review. *International Journal of Industrial Engineering Computations*, 10(2): 199-224.
- Rizqi, Z.U., Chou, S.Y., and Oscar, A.D. (2025). Towards energy-efficient Robotic Mobile Fulfillment System: Hybrid agent-based simulation with DEA-based surrogate machine learning. *Applied Soft Computing*, 176: 113141.
- Sangaiah, A.K., and Khanduzi, R. (2022). Tabu search with simulated annealing for solving a location–protection–disruption in hub network. *Applied Soft Computing*, 114: 108056.
- Senaras, O.M., Inanç, S., Senaras, A.E., and Bilir, B.Ö. (2025). Comparing the use of ant colony optimization and genetic algorithms to organize kitting systems within green supply chain management practices. *Sustainability*, 17(5): 2001.
- Tong, B., Wang, J.W., Wang, X., Zhou, F.H., Mao, X.H., and Zheng, W.L. (2022). Optimal route planning for truck–drone delivery using variable neighborhood tabu search algorithm. *Applied Sciences-Basel*, 12(1): 529.
- Tsubakitani, S., and Evans, J.R. (1998). Optimizing tabu list size for the traveling salesman problem. *Computers & Operations Research*, 25(2): 91-97.
- Udvardy, K., Görbe, P., Bódis, T., and Botzheim, J. (2024). Conceptual framework for adaptive bacterial memetic algorithm parameterization in storage location assignment problem. *Mathematics*, 12(23): 3688.
- van den Berg, J.P., and Zijm, W.H.M. (1999). Models for warehouse management: Classification and examples. *International Journal of Production Economics*, 59(1-3): 519-528.
- Xia, Y.K., and Fu, Z. (2019). Improved tabu search algorithm for the open vehicle routing problem with soft time windows and satisfaction rate. *Cluster Computing-the Journal of Networks Software Tools and Applications*, 22: 58725-58733.
- Yang, P., Miao, L., Xue, Z., and Qin, L. (2015). An integrated optimization of location assignment and storage/retrieval scheduling in multi-shuttle automated storage/retrieval systems. *Journal of Intelligent Manufacturing*, 26: 1145-1159.
- Yang, P., Peng, Y., Ye, B., and Miao, L. (2017). Integrated optimization of location assignment and sequencing in multi-shuttle automated storage and retrieval systems under modified 2n-command cycle pattern. *Engineering Optimization*, 49(9): 1604-1620.

Yu, Y., de Koster, R.B.M., and Guo, X. (2015). Class-based storage with a finite number of items: Using more classes is not always better. *Production and Operations Management*, 24(8): 1235-1247.

Zaerpour, N., de Koster, R.B.M., and Yu, Y.G. (2013). Storage policies and optimal shape of a storage system. *International Journal of Production Research*, 51(23-24): 6891-6899.

Zhang, H.Z., Liu, F., Zhou, Y.Y., and Zhang, Z.Y. (2020). A hybrid method integrating an elite genetic algorithm with tabu search for the quadratic assignment problem. *Information Sciences*, 539: 347-374.

Zhang, W.B., Deng, Z.Y., Zhang, C.J., and Shen, W.M. (2025). Integrated optimization of storage space allocation and crane scheduling in automated storage and retrieval systems. *Robotics and Computer-Integrated Manufacturing*, 93: 102918.