AN EFFICIENT INTEGRATION OF INTELLIGENT SWARM ALGORITHMS AND VARIABLE NEIGHBORHOOD SEARCH FOR OPTIMIZING ROBOT ROUTING IN 3D WAREHOUSES

Pham Dinh Thanh[1], Hoang Quang Huy[2], Tran Xuan Thanh[2], Nguyen Viet Hoang[2], Giang Thanh Trung[2], Nguyen Thi My Binh[2,*]

DOI: https://doi.org/10.57001/huih5804.2025.416

ABSTRACT

This study introduces an advanced hybrid optimization technique that integrates Particle Swarm Optimization (PSO) with Variable Neighborhood Search (VNS) to address the robot path planning problem in warehouse environments. The proposed algorithm simultaneously tackles the challenges of item distribution and shortest path computation for multiple autonomous robots operating within a complex 3D warehouse layout, while respecting each robot's load capacity constraints. Experimental evaluations reveal that the PSO-VNS hybrid significantly outperforms conventional Greedy algorithms by reducing total travel distance. Additionally, the use of precomputation and distance caching strategies enhances computational efficiency, making the approach suitable for real-time applications.

Keywords: *Particle swarm optimization, variable neighborhood search, greedy algorithm, warehouse routing, optimization, 3D warehouse.*

[1]Faculty of Natural Sciences & Technology, Tay Bac University, Vietnam
[2]School of Information & Communication Technology, Hanoi University of Industry, Vietnam
[*]Email: binhntm@haui.edu.vn

Received: 15/7/2025
Revised: 22/8/2025
Accepted: 28/11/2025

1. INTRODUCTION

The rapid advancement of e-commerce and logistics has placed significant demands on the efficiency, accuracy, and scalability of warehouse operations. As customer expectations for fast delivery and order accuracy continue to rise, traditional manual methods of warehouse management are increasingly unable to meet these evolving requirements. In response, many enterprises are turning to automated systems, particularly autonomous mobile robots (AMRs), to handle order picking and item transportation tasks within warehouses.

In modern warehouse environments - especially those with complex, multi-level (3D) layouts - one of the most critical and challenging problems is the robot routing optimization problem. This involves determining the most efficient routes for a fleet of robots to fulfill assigned tasks while minimizing total travel distance, energy consumption, and delivery time. The problem is further complicated by real-world constraints such as robot carrying capacity, traffic congestion in aisles, dynamic task assignments, and three-dimensional spatial complexity.

To address this challenge, numerous heuristic and metaheuristic approaches have been proposed. Among them, intelligent swarm algorithms - inspired by the collective behavior of social organisms - have demonstrated remarkable effectiveness in exploring high-dimensional search spaces and avoiding premature convergence. However, swarm-based methods can still suffer from local optima entrapment and may require enhancement in solution refinement capabilities.

To improve convergence quality and solution diversity, this paper proposes an efficient hybrid approach that integrates Intelligent Swarm Algorithms with Variable Neighborhood Search (VNS). While swarm intelligence provides global search capabilities and adaptability, VNS introduces systematic local search strategies to explore neighboring solutions and escape local minimum. The integration of these two techniques aims to balance exploration and exploitation, thereby enhancing the overall performance of the routing optimization process in 3D warehouse environments.

The main contributions of this study are as follows:

• A novel hybrid optimization framework combining intelligent swarm algorithms with VNS to address robotic routing challenges in 3D warehouse settings.

• A customized encoding and decoding mechanism suitable for 3D warehouse spatial representation and routing constraints.

• A comparative analysis with state-of-the-art algorithms to demonstrate the effectiveness and efficiency of the proposed method.

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 presents the problem statement. Section 4 describes the proposed algorithm. Section 5 presents experimental results and analysis. Finally, Section 6 concludes the paper and outlines future research directions.

2. RELATED WORKS

Optimization of robot routing in warehouse environments has received growing attention due to the increasing demand for efficiency in logistics and e-commerce operations. Several studies have addressed related aspects, such as order batching, path planning, and metaheuristic algorithm development.

Li et al. [1] investigated the joint optimization of order batching and picker routing in Chinese online retail warehouses. Their work emphasized the practical challenges of coordinating human pickers and efficient order grouping, laying the groundwork for robot-based alternatives. Zhen et al. [2] explored the application of Autonomous Mobile Robots (AMRs) in warehouse operations and proposed optimization frameworks to enhance robot coordination and warehouse throughput, highlighting the relevance of intelligent algorithms in real-world warehouse settings.

From an algorithmic perspective, Almufti [3] provided a comprehensive historical overview of metaheuristic algorithms, discussing their evolution and application scope. Among these, the capacitated vehicle routing problem (CVRP) has been widely studied as a foundational model for routing optimization. Praveen et al. [4] reviewed major approaches to solving CVRP, providing insights into how similar techniques can be extended to multi-robot warehouse routing problems.

A variety of hybrid and swarm-based metaheuristic algorithms have shown promising results in routing and scheduling domains. Hansen et al. [5] introduced the Variable Neighborhood Search (VNS) metaheuristic and outlined its effectiveness when applied to combinatorial optimization problems. Building on this, Wang and Xu [6] developed a hybrid PSO-VNS algorithm tailored for CVRP, demonstrating improved solution quality and convergence speed. Similarly, Niu et al. [7] proposed a hybrid tabu search algorithm to solve open vehicle routing problems with fuel consumption constraints, highlighting the importance of integrating real-world limitations into routing models.

More recently, Kongkidakhon et al. [8] proposed a hybrid PSO-VNS algorithm to tackle the machinery allocation and scheduling problem in shared environments. Their method illustrates how combining swarm intelligence with local search strategies can yield efficient solutions for complex logistics problems, reinforcing the potential of such approaches for robot routing in warehouses.

These studies collectively underline the growing role of intelligent swarm algorithms in solving routing and logistics challenges. However, the application of such algorithms specifically to autonomous robot coordination in warehouse environments remains an area with significant room for further exploration and enhancement.

3. PROBLEM STATEMENT

3.1. Verbal statement

In a large warehouse, goods are stored on multiple shelves, where each shelf consists of several tiers, and each tier contains multiple slots. The shelves are arranged in rows, forming aisles through which robots can navigate. The warehouse layout can be represented as a two-dimensional grid, where grid cells with a value of 0 denote traversable paths for the robots, and cells with a value of 1 represent shelves that are non-traversable [1].

A fleet of robots is employed to collect a list of requested items. Each robot has a starting point (counter) and a limited carrying capacity in terms of the number of items it can transport. Robots are allowed to move only along the accessible paths and are not permitted to pass through shelving units. To retrieve an item located on a shelf, a robot must move to the nearest adjacent path cell next to the shelf containing that item [1].

The objective of the problem is to assign the required items to individual robots and determine the optimal route for each robot such that the total travel distance of all robots is minimized. Each robot must start from its designated starting point, collect all assigned items in an

optimal sequence, and return to its starting point. The total number of items assigned to each robot must not exceed its carrying capacity [1].

3.2. Mathematical formulation statement

The robot path optimization problem in a warehouse can be formally stated as follows:

Given a warehouse consisting of **m** shelves (denoted by $S = \{S_1, S_2, ..., S_m\}$), where each shelf has **t** tiers (denoted by $T = \{T_1, T_2, ..., T_t\}$), and each tier contains **l** slots (denoted by $L = \{L_1, L_2, ..., L_l\}$). The warehouse is represented by a two-dimensional matrix map in which cells with a value of **0** indicate walkable paths that robots can traverse, and cells with a value of **1** represent shelving units that robots cannot pass through. There are *n* items in the warehouse (denoted by $M = \{M_1, M_2, ..., M_n\}$), where each item M_i contains information about its name ($name_i$), quantity ($quantity_i$), and position ($position_i$) specified as a triplet ($shelf_i$, $tier_i$, $slot_i$) indicating the shelf, tier, and slot location of the item. A request list of k items needs to be taken (denoted by $R = \{R_1, R_2, ..., R_k\}$). Each requested item R_i includes a name ($name_i$) and the required quantity ($req\_quantity_i$). It is assumed that all requested items are available in the warehouse, and the total requested quantity for each item does not exceed its available stock. There are r robots available (denoted by $B = \{B_1, B_2, ..., B_r\}$). Each robot B_i has a limited capacity ($capacity_i$) and a starting point located at the counter position (counter_position), typically set at coordinates [0,0,0] [1].

Robots can only travel along walkable paths (cells with value 0 in the map) and are not allowed to pass through shelves. To collect an item located at (*shelf, tier, slot*), a robot must move to the nearest walkable cell adjacent to the shelf location, referred to as the **access point** [1].

Objectives:

- Item Allocation: Assign requested items to robots such that the total number of items allocated to each robot does not exceed its capacity.
- Optimal Pickup Sequencing: Determine the optimal sequence in which each robot should pick up its assigned items.
- Route Planning: Compute the shortest path for each robot starting from the counter, visiting the access points of its assigned items in the determined order, and returning to the counter.

Optimization Goal: Minimize the total travel distance of all robots. The travel distance is defined as the total number of grid cells traversed by each robot, including any additional transition costs when moving between tiers (if applicable) [1].

3.3. Mathematical model

To formally model the problem, we define the following parameters, variables, and constraints:

Parameters and variables

- Set of requested items: $R = \{R_1, R_2, ..., R_k\}$
- Set of robots: $B = \{B_1, B_2, ..., B_r\}$
- Capacity of robot *i*: $capacity_i$
- Location of item *j*: $position_j = (shelf_j, tier_j, slot_j)$
- Requested quantity of item *j*: $req\_quantity_j$
- Counter position: counter_position = (0, 0, 0)
- Distance matrix: distance[*i*][*j*] represents the shortest path distance from location *i* to location *j*
- Decision variables:
  - X_{ij}: Binary variable, $X_{ij} = 1$ if item *j* is assigned to robot *i*, and 0 otherwise
  - Y_{ijk}: Binary variable, $Y_{ijk} = 1$ if robot *i* travels from location *j* to location *k*, and 0 otherwise
  - Z_{ij}: Integer variable representing the order in which robot *i* visits location *j*

Input

The input to the problem includes the following components:

- Warehouse structure:
  - Number of shelves (*S*)
  - Number of tiers per shelf (*T*)
  - Number of slots per tier (*L*)
- Warehouse map: A matrix of size (2 * *S* + 1) x *L* representing the layout of the warehouse, where:
  - A value of 0 indicates a walkable aisle
  - A value of 1 represents a shelf location that is not passable by robots
- Inventory item list: Each item in the warehouse contains:
  - Item name
  - Available quantity
  - Location defined by a triplet (*shelf, tier, slot*)
- List of requested items: Each requested item includes:
  - Item name
  - Required quantity

- Robot information:
  - Number of robots *r*
  - Maximum capacity of each robot

Output

The output of the problem includes:

- Item allocation: The list of items assigned to each robot.
- Picking order: The sequence in which each robot retrieves its assigned items.
- Route: The detailed path each robot follows, starting from the counter, visiting each item's location in the specified order, and returning to the counter.
- Total distance: The total travel distance is covered by all robots.

Example output format:

- TOTAL_DISTANCE: 27.0
- ROBOT 1:
  - Items: Laptop (10), Smartphone (4)
  - Picking order: Smartphone -> Laptop
  - Route:
    - Start from Counter at [0,0,0]
    - Go to Smartphone at [1,1,2]
    - Go to Laptop at [1,2,2]
    - Return to Counter at [0,0,0]
  - Travel distance: 12.0
- ROBOT 2:
  - Items: Mouse (1), Camera (2), Printer (5), Monitor (8)
  - Picking order: Monitor -> Printer -> Camera -> Mouse
  - Route:
    - Start from Counter at [0,0,0]
    - Go to Monitor at [2,1,2]
    - Go to Printer at [3,1,2]
    - Go to Camera at [2,2,3]
    - Go to Mouse at [1,1,3]
    - Return to Counter at [0,0,0]
  - Travel distance: 15.0

Fitness funtion

The fitness is to minimize the total travel distance of all robots:

$$\min \sum_{i=1}^{r} \sum_{j=0}^{k} \sum_{l=0}^{k} y_{ijl} * distance_{jl} \quad (1)$$

Where:

- *r*: The number of robots
- *k*: The number of items need to be taken
- $distance_{jl}$: Cached shortest path using A* algorithm

Constraints

Item Allocation: Each requested item must be assigned to exactly one robot:

$$\sum_{i=1}^{r} x_{ij} = 1 \forall j \in \{1,2,...,k\} \quad (2)$$

The total quantity of items assigned to each robot must not exceed its capacity:

$$\sum_{j=1}^{k} x_{ij} * req\_quantity_j \le capacity_i, \forall i \in \{1,2,...,r\} \quad (3)$$

Each robot must enter and exit each item location exactly once, if the item is assigned to that robot:

$$\sum_{l=0}^{k} y_{ijl} = x_{ij} \forall i \in \{1,2,\ldots,r\}, \forall j \in \{1,2,\ldots,k\} \quad (4)$$

$$\sum_{l=0}^{k} y_{ilj} = x_{ij} \forall i \in \{1,2,\ldots,r\}, \forall j \in \{1,2,\ldots,k\} \quad (5)$$

Counter Constraints: Each robot must start and end its route at the counter:

$$\sum_{j=1}^{k} y_{ij0} = 1, \quad \forall i \in \{1,2,...,r\} \quad (6)$$

$$\sum_{j=1}^{k} y_{i0j} = 1, \quad \forall i \in \{1,2,...,r\} \quad (7)$$

Order constraints (Miller–Tucker–Zemlin): The visiting order variables must follow the MTZ constraints to prevent sub-tours:

$$z_{ij} - z_{il} + k * y_{ijl} \le k - 1, \quad \forall i \in \{1,2,...,r\}, \forall j,l \in \{1,2,...,k\}, j \ne l \quad (8)$$

Logical Constraints: The decision variables must be binary:

$$y_{ijl} \in \{0,1\}, \quad \forall i \in \{1,2,...,r\}, \ \forall j,l \in \{0,1,...,k\} \quad (9)$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in \{1,2,...,r\}, \ \forall j \in \{1,2,...,k\} \quad (10)$$

3.4. Warehouse map model

The warehouse map model plays a critical role in computing the shortest path for each robot. In this study,

we adopt a two-dimensional grid model to represent the layout of the warehouse. This approach has been proposed in several previous works, such as the study published by ScienceGate on a novel path planning algorithm for warehouse robots based on grid modeling [2].

The warehouse map is represented as a 2D matrix of dimensions $(2 \times S + 1) \times L$, where:

- S is the number of shelves,
- L is the number of slots on each tier.

In this matrix:

- Even-numbered rows represent shelves (impassable by robots),
- Odd-numbered rows represent walkways (accessible by robots).

This structured layout allows us to clearly distinguish between passable (value 0) and blocked (value 1) cells, enabling accurate path planning.

The location of each item, originally expressed as a triple (shelf, tier, slot), is converted into grid coordinates (x, y) using the following transformation formula:

$$x = 2 * shelf - 1 \quad (11)$$

$$y = slot \quad (12)$$

This formula maps item positions to their corresponding accessible access points on the grid.

With this grid model, shortest path computations between two positions on the map can be efficiently carried out using classical path-finding algorithms such as:

- A* (A-Star),
- Dijkstra's algorithm,
- Breadth-First Search (BFS).

These algorithms consider only valid (walkable) cells and provide the optimal travel route for robots while avoiding shelf obstacles.

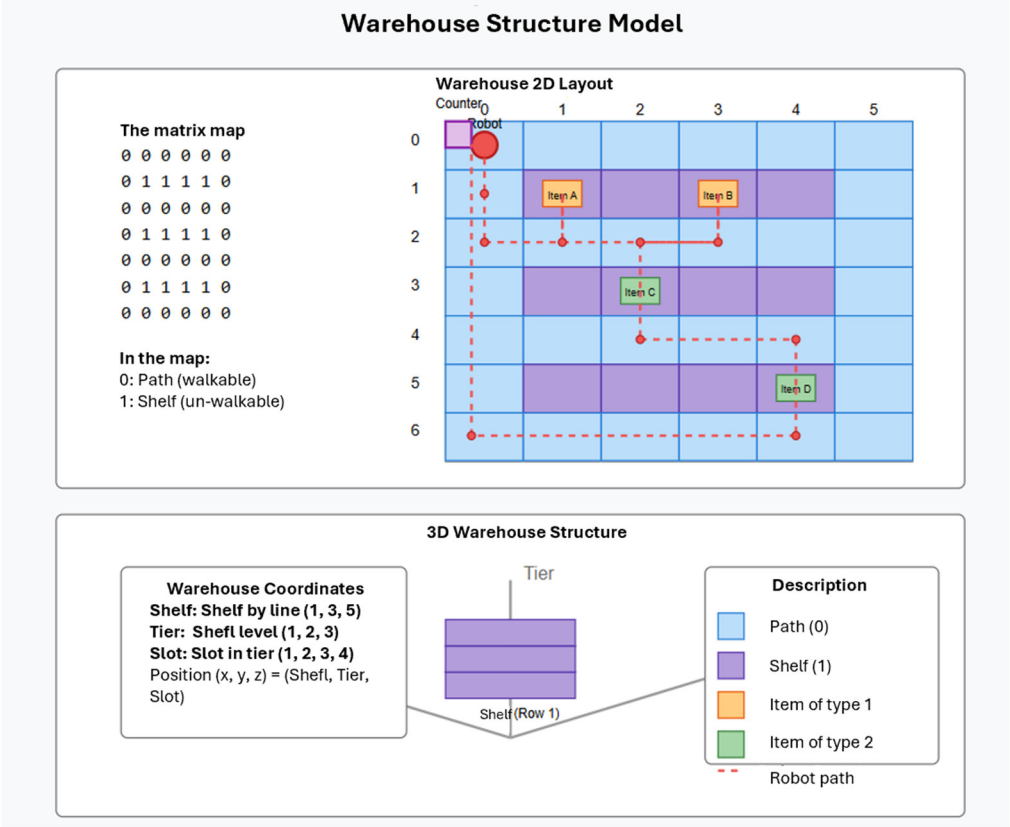

Figure 1. Warehouse structure model

4. PROPOSED ALGORITHM

4.1. Hybrid principle

The hybrid PSO-VNS (Particle Swarm Optimization - Variable Neighborhood Search) algorithm is designed to leverage the global search capability of PSO and the local exploitation strength of VNS. The goal is to improve solution quality and convergence speed in the warehouse robot path optimization problem.

- PSO
  - Performs global exploration of the search space.
  - Particles move based on their own experience and that of the swarm.
  - Help to identify promising regions in the solution space.
- VNS
  - Performs local exploitation in promising regions found by PSO.
  - Dynamically changes neighborhood structures to escape local optima and improve solutions.
- Combined Strategy:
  - Use PSO to guide the search across the solution space.

- Apply VNS in focused regions to refine and improve individual solutions.
- Use the results from VNS to update and guide the particles in the next PSO iteration.

This synergy ensures both diversification (exploration) and intensification (exploitation), increasing the chances of finding a high-quality near-optimal solution.

4.2. Algorithm procedure

The detailed procedure of the hybrid PSO-VNS algorithm is as follows:

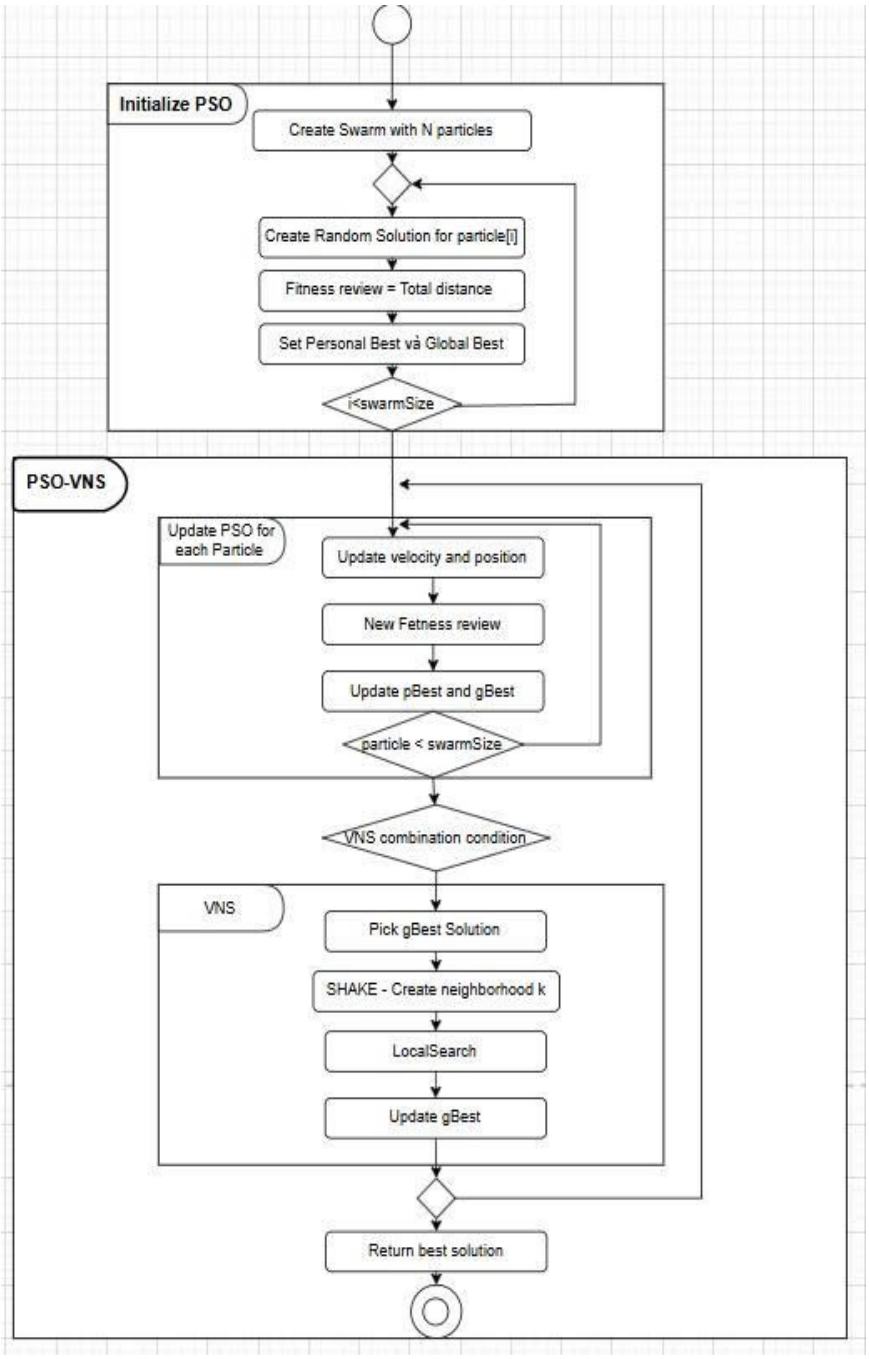

Figure 2. Algorithm flowchart of PSO-VNS

Initialization

Initialize a PSO population with n particles.

Evaluate the initial particles and identify the global best solution (globalBest).

Main loop

Update PSO: Repeat until the stopping condition is met:

- Update the position and velocity of each particle according to PSO rules.
- Evaluate new particles.
- Update each particle's personal best and the global best if a better solution is found.

Apply VNS (Periodically, e.g., every 5 iterations):

- Apply the VNS algorithm to the current global best solution to enhance it.
- If VNS finds a better solution, update the global best.

Return result

Return the best global solution (globalBest) found by the hybrid PSO-VNS algorithm.

5. EXPERIMENTAL RESULTS

5.1. Experimental environment

To evaluate the effectiveness of the proposed hybrid PSO-VNS algorithm, we conducted experiments on a computer with the following specifications:

- **CPU**: Intel Core i7-9750H, 2.6 GHz, 6 cores
- **RAM**: 16 GB DDR4
- **Operating System**: Windows 10 Professional 64-bit
- **Programming Language**: Java (JDK 11)
- **IDE**: IntelliJ IDEA 2023.1

The choice of Java as the development language ensures portability of the application, allowing it to be deployed on multiple platforms without modifying the source code.

5.2. Experimental data

To evaluate the effectiveness of the algorithm under various conditions, we used three datasets with different sizes and complexity levels:

- **Small dataset**:
  - 3 shelves, 2 tiers, 4 slots per tier
  - 13 items in the warehouse
  - 6 items to be retrieved with a total quantity of 30 units
  - 2 robots with a capacity of 25 units each
- **Medium dataset**:
  - 5 shelves, 3 tiers, 6 slots per tier
  - 25 items in the warehouse
  - 10 items to be retrieved with a total quantity of 50 units
  - 3 robots with a capacity of 20 units each
- **Large dataset**:
  - 10 shelves, 4 tiers, 8 slots per tier
  - 50 items in the warehouse
  - 15 items to be retrieved with a total quantity of 100 units
  - 5 robots with a capacity of 25 units each

These datasets were designed to simulate real-world warehouse scales - from small warehouses to large ones with multiple robots - ensuring a comprehensive performance evaluation of the proposed algorithm.

5.3. Experimental result

Table 1. Experimental results on the small dataset. Improvement rate of PSO-VNS over Greedy: 23.94%. Improvement rate of PSO-VNS over PSO: 8.47%

Algorithm	Total Distance	Execution Time (ms)	Number of Convergence Iterations
Greedy	35.5	142	-
PSO	29.5	873	67
PSO-VNS	27.0	1358	42

Table 2. Experimental results on the medium dataset. Improvement rate of PSO-VNS over Greedy: 26.72%. Improvement rate of PSO-VNS over PSO: 10.53%

Algorithm	Total Distance	Execution Time (ms)	Number of Convergence Iterations
Greedy	58.0	283	-
PSO	47.5	1842	85
PSO-VNS	42.5	2514	53

Table 3. Experimental results on the large dataset. Improvement rate of PSO-VNS over Greedy: 32.68%. Improvement rate of PSO-VNS over PSO: 12.76%

Algorithm	Total Distance	Execution Time (ms)	Number of Convergence Iterations
Greedy	127.0	521	-
PSO	98.0	3517	92
PSO-VNS	85.5	4762	61

Table 4. Experimental results of PSO and PSO-VNS with different numbers of particles on the small dataset

Algorithm	Particles	Total Distance	Execution Time (ms)	Number of Convergence Iterations
PSO	5	29.5	873	67
PSO	10	27.2	980	58
PSO	25	25.3	1290	50
PSO	50	24.1	1810	42
PSO-VNS	5	27.0	1358	42
PSO-VNS	10	25.1	1510	38
PSO-VNS	25	23.5	1875	34
PSO-VNS	50	22.0	2420	29

Table 5. Experimental results of PSO and PSO-VNS with different numbers of particles on the medium dataset

Algorithm	Particles	Total Distance	Execution Time (ms)	Number of Convergence Iterations
PSO	5	47.5	1842	85
PSO	10	43.2	2120	73
PSO	25	40.1	2590	62
PSO	50	37.4	3310	51
PSO-VNS	5	42.5	2514	53
PSO-VNS	10	39.3	2840	48
PSO-VNS	25	35.5	3475	42
PSO-VNS	50	32.1	4450	36

Table 6. Experimental results of PSO and PSO-VNS with different numbers of particles on the large dataset

Algorithm	Particles	Total Distance	Execution Time (ms)	Number of Convergence Iterations
PSO	5	98.0	3517	92
PSO	10	91.3	3875	80
PSO	25	84.0	4500	66

PSO	50	77.8	5520	53
PSO-VNS	5	85.5	4762	61
PSO-VNS	10	78.0	5130	56
PSO-VNS	25	70.2	5900	48
PSO-VNS	50	63.9	7100	40

Experimental results analysis

Comparison of solution quality between the proposed algorithm and other algorithms

The experimental results in Table 1, Table 2, and Table 3 show that the proposed PSO-VNS algorithm outperforms both the Greedy algorithm and the basic PSO algorithm in terms of solution quality. Specifically:

• Improvement over Greedy: PSO-VNS reduces total travel distance by 23.94% to 32.68% compared to the Greedy algorithm, depending on the dataset size. The improvement becomes more significant as the dataset size increases, indicating that PSO-VNS is more effective on complex problems.

• Improvement over PSO: PSO-VNS achieves 8.47% to 12.76% better results than standard PSO, highlighting the crucial role of VNS in local search and avoiding local optima.

• Convergence iterations: PSO-VNS converges significantly faster than PSO, reducing the number of iterations by approximately 30 - 40%. This demonstrates how VNS accelerates the search for optimal solutions.

The analysis of item allocation and pick-up order reveals that:

• PSO-VNS achieves more balanced item distribution across robots, leading to better load balancing.

• PSO-VNS identifies more optimal pick-up sequences, significantly reducing travel distance.

Comparison of algorithm execution time

Although PSO-VNS produces better results, its execution time is higher compared to the Greedy and standard PSO algorithms. Specifically:

• The execution time of PSO-VNS is approximately 9 - 10 times longer than that of the Greedy algorithm.

• The execution time of PSO-VNS is approximately 35 - 55% longer than that of the standard PSO algorithm.

However, in the context of robot path planning in warehouses, execution time is not the most critical factor because the planning is typically performed before deployment and does not require real-time computation. The total travel distance is more important, as it directly affects operational efficiency and long-term energy costs.

Scalability evaluation

The experimental results show that the effectiveness of the PSO-VNS algorithm increases with the dataset size:

• **Small dataset**: 23.94% improvement over Greedy

• **Medium dataset**: 26.72% improvement over Greedy

• **Large dataset**: 32.68% improvement over Greedy

This demonstrates the algorithm's good scalability with larger and more complex warehouse environments. While the Greedy and standard PSO algorithms struggle with large search spaces, PSO-VNS maintains high performance due to its combination of global and local search capabilities.

Comparison of particle count

The experimental results demonstrate that increasing the number of particles significantly improves the performance of both PSO and PSO-VNS algorithms:

• **PSO**: 18.31% improvement in total distance (solution quality) when increasing from 5 to 50 particles

• **PSO-VNS**: 18.52% improvement in total distance when increasing from 5 to 50 particles

• **Convergence speed**: PSO reduced the number of iterations by 37.31%, while PSO-VNS reduced it by 30.95% over the same range

These findings highlight the scalability of both algorithms in terms of solution quality and convergence efficiency as the swarm size grows. However, the execution time also increased substantially - up to 107% for PSO and 78% for PSO-VNS - indicating a computational trade-off. Notably, PSO-VNS consistently outperforms standard PSO across all particle sizes due to its hybrid global-local search design, enabling it to achieve high-quality solutions even in larger swarms.

6. CONCLUSIONS

This study introduced an innovative hybrid optimization approach that integrates Particle Swarm Optimization (PSO) with Variable Neighborhood Search (VNS) to effectively tackle the robot routing problem in 3D warehouse environments. The proposed method simultaneously optimizes the distribution of tasks among multiple robots and computes efficient routing paths within a complex three-dimensional warehouse layout, all while adhering to the robots' capacity limitations.

Experimental evaluations demonstrate that the PSO-VNS hybrid outperforms traditional methods by generating higher-quality solutions and significantly reducing overall travel distances. These findings confirm the potential of the proposed algorithm as a robust and effective tool for optimizing robotic operations in automated warehouse systems.

REFERENCES

[1]. Li J., Huang R., Dai J.B., "Joint optimisation of order batching and picker routing in the online retailer's warehouse in China," *International Journal of Production Research*, 55(2), 447-461, 2017.

[2]. Zhen L., Tan Z., de Koster R., He X., Wang S., Wang H., "Optimizing Warehouse Operations with Autonomous Mobile Robots," *Transportation Science*, 2025.

[3]. Almufti S. M., "Historical survey on metaheuristics algorithms," *International Journal of Scientific World,* 7(1), 1, 2019.

[4]. Praveen V., Keerthika P., Sivapriya G., Sarankumar A., Bhasker B., "Vehicle routing optimization problem: a study on capacitated vehicle routing problem," *Materials Today: Proceedings,* 64, 670-674, 2022.

[5]. Hansen P., Mladenović N., Brimberg J., Pérez J. A. M., *Variable neighborhood search.* In Handbook of metaheuristics (pp. 57-97). Cham: Springer International Publishing, 2018.

[6]. Wang X., Xu X., "A hybrid PSO-VNS algorithm for capacitated vehicle routing problem," *Advances in Intelligent Systems and Computing*, 502, 519-528, 2017.

[7]. Niu Y., Yang Z., Chen P., Xiao J., "A hybrid tabu search algorithm for a real-world open vehicle routing problem involving fuel consumption constraints," *Complexity*, 1-12, 2018.

[8]. Kongkidakhon Worasan, Kanchana Sethanan, Rapeepan Pitakaso, Thitipong Jamrus, Karn Moonsri, Paulina Golinska-Dawson, "A hybridization of PSO and VNS to solve the machinery allocation and scheduling problem under a machinery sharing arrangement," *Intelligent Systems with Applications,* 18, 1-16, 2023.