International Journal of Engineering Research and Development
e- ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 22, Issue 2 (February 2026), PP 09-19

Optimization of Joint Order Batching and Picker Routing Problem (JOBPRP) Utilizing Modified Reduced Variable Neighborhood Search

Wael Zayat[1], Serol Bulkan[2], and Huseyin Selcuk Kilic[2]
[1] *Institute of Pure and Applied Sciences, Marmara University, Maltepe Campus, Istanbul, Turkey.*
[2] *Department of Industrial Engineering, Marmara University, Maltepe Campus, Istanbul, Turkey.*
Corresponding Author: Wael Zayat

ABSTRACT *In the highly competitive market of modern logistics and e-commerce, warehouse efficiency is one of the primary factors of supply chain success. Order picking is often cited as the most labor-intensive and costly operation within a distribution center accounting for over 50% of total warehouse operating costs. This research focuses on the Joint Order Batching and Picker Routing Problem (JOBPRP), an NP-hard optimization problem that simultaneously groups customer orders into batches and determine optimal travel paths to minimize total distance. We propose a robust two-stage solution framework. In the first stage, initial feasible solutions are constructed using First Fit Decreasing (FFD) and Aisle Saving (AS) heuristics. In the second stage, these solutions are refined using a modified version of Reduced Variable Neighborhood Search (RVNS) metaheuristic. The performance of the RVNS is evaluated across a large sample of benchmarks. The experiments included multiple neighborhood sizes ($K = 5, 10, 20, 30, 40$) to analyze the sensitivity of solution quality relative to computational effort. Our results indicate that while initial heuristics provide quick baseline solutions, the RVNS significantly enhances efficiency through systematic stochastic neighborhood exploration. The proposed model provides warehouse managers with a flexible tool to optimize high-volume picking operations.*
Keywords *warehouse management; order picking; logistics, precedence*

Date of Submission: 25-01-2026 Date of acceptance: 06-02-2026

I. INTRODUCTION

Warehousing represents a fundamental part of supply chain management. Warehouse management directly influences logistics performance, production cycles, inventory control, and the broader retail distribution network. The effective management of warehouse facilities is essential for reducing inventory holding costs, minimizing internal movement expenses, and enhancing overall responsiveness. Typically, warehouse operations are categorized into receiving, sorting, picking, packing, and shipping (Žulj et al., 2018). Among these functional areas, order picking is universally recognized as the most resource-intensive activity, often consuming 50% to 70% of total operational costs (Pinto & Nagano, 2019). Consequently, optimizing the order picking sequence is vital not only for achieving rapid response times but also for improving operational sustainability through significant energy and time savings.

The optimization of the picking process generally entails addressing two primary sub-problems. The first is the Order Batching Problem (OBP), which involves partitioning a set of customer orders into groups or "batches." Each batch is retrieved during a single picking tour, ensuring that the total volume/weight respects the physical capacity constraints of the picker while minimizing the cumulative travel distance (Pardo et al., 2024). The second sub-problem is the Picker Routing Problem (PRP), which focuses on determining the most efficient path for a picker to navigate through the warehouse aisles to retrieve the items within a specific batch. These interrelated tasks are influenced by a variety of environmental factors, including warehouse layout, storage assignment strategies, and specific order profiles (Boz & Aras, 2022).

While the OBP has traditionally utilized simplified or static routing policies (Roodbergen & de Koster, 2001a; 2001b; Pardo et al., 2024), current research supports for the Joint Order Batching and Picker Routing Problem (JOBPRP). By solving both the OBP and PRP simultaneously, this integrated approach captures the collaborations between batch composition and route optimization, which leads to more efficient logistics and warehouse performance (Pinto & Nagano, 2022; Pardo et al., 2024).

This paper investigates the application of Reduced Variable Neighborhood Search (RVNS) as a solution for the JOBPRP. RVNS is a metaheuristic known for systematically exploring different solution neighborhoods to escape local optima by applying a shaking procedure to improve the solution quality for FFD and for AS where a heuristic solution was developed to solve the routing problem for a batch of orders.

We focus on a multi-block warehouse layout and evaluate how stochastic neighborhood structures can refine traditional greedy batching methods. By expanding the search intensity through the k parameter from 5 to 40, we demonstrate the scalability and effectiveness of RVNS in reducing operational costs while identifying the solution for JOBPRP.

II. PROBLEM DESCRIPTION MATHEMATICAL MODEL

The Joint Order Batching and Picker Routing Problem addressed in this study involves the simultaneous optimization of order grouping and path construction to maximize warehouse efficiency. Following the conceptual framework established by Kulak et al. (2012) and the integrated hybrid modeling approach of Cheng et al. (2015), the problem seeks to minimize the total travel distance (Z) across all picking cycles.

The warehouse environment is defined by a set of storage locations L and a set of customer orders O. Each order $o \in O$ consists of specific items located at positions $i \in L$. Because a picker's equipment has a finite capacity C, multiple orders must be consolidated into batches B, and a routing sequence must be generated for each batch that starts and ends at the depot.

The notation used to formulate the JOBPRP is defined as follows:

- **Indices:**
  - b: Index of batches ($b = 1, 2, \dots, B$).
  - o: Index of customer orders ($o = 1, 2, \dots, O$).
  - i, j: Indices of storage locations ($i, j = 1, 2, \dots, L$).
- **Parameters:**
  - C: Maximum weight capacity of the picking vehicle.
  - W_o: Total weight of order o.
  - $d_{i,j}$:The travel distance between location i and location j.
  - $s_{i,o}$: A binary parameter where $s_{i,o} = 1$ if order o requires an item from location i, and 0 otherwise.
- **Decision Variables**

To bridge the gap between batching and routing, we define the following binary decision variables as proposed by Cheng et al. (2015):

X_o^b: 1 if order o is assigned to batch b; 0 otherwise.

Y_{ij}^b: 1 if the picker travels directly from location i to location j within batch b; 0 otherwise.

Z_i^b: 1 if location i is visited during the picking process of batch b; 0 otherwise.

The objective function aims to minimize the global travel distance, which is the summation of the distances of all edges (i, j) traversed in every batch:

$$\text{Minimize } Z = \sum_{b=1}^{B} \sum_{i=1}^{L} \sum_{j=1}^{L} d_{ij} Y_{ij}^b \quad (1)$$

Subject to the following constraints:

Order Assignment: Each order must be assigned to exactly one batch.

$$\sum_{b=1}^{B} X_o^b = 1, \quad \forall o \in O \quad (2)$$

Capacity Constraint: The total weight of orders in a batch cannot exceed the vehicle capacity C.

$$\sum_{o=1}^{O} W_o X_o^b \leq C, \quad \forall b \in B \quad (3)$$

Location Visit Requirement: A location i must be visited in batch b if any order o assigned to that batch requires an item from that location.

$$Z_i^b \geq s_{io} X_o^b, \quad \forall i \in L, \forall o \in O, \forall b \in B \quad (4)$$

Flow Conservation: If a picker enters location i in batch b, they must also depart from it.

$$\sum_{j=1, j \neq i}^{L} Y_{ij}^b = \sum_{j=1, j \neq i}^{L} Y_{ji}^b = Z_i^b, \quad \forall i \in L, \forall b \in B \quad (5)$$

Sub-tour Elimination: To ensure that each picker completes a single valid tour starting and ending at the depot (location 0) without forming isolated loops, we introduce subsidiary variables u_i^b representing the sequence order of location i in batch b:

$$u_i^b - u_j^b + L \cdot Y_{ij}^b \leq L - 1, \quad \forall i, j \in \{1, \dots, L\}, i \neq j, \forall b \in B \quad (6)$$

Integrity and Binary Constraints:

Constraints (7, 8 and 9) specify the non-negative binary characters of the decision variables.

$$X_o^b \in \{0,1\}, \quad \forall o \in O, \forall b \in B \quad (7)$$

$$Y_{ij}^b \in \{0,1\}, \quad \forall i, j \in L, \forall b \in B \quad (8)$$

$$Z_i^b \in \{0,1\}, \quad \forall i \in L, \forall b \in B \quad (9)$$

III. PROPOSED METHODOLOGY

We solve the JOBPRP by following a two-phase optimization structure, where a constructive phase provides.

3.1 Phase I: Constructive Heuristics

The initial feasible solution, and an improvement phase iteratively refines the results.We propose 2 order batching initial strategies. The first batching strategy First Fit Decreasing (FFD), the second method is Aisle Saving (AS) algorithm.

FFD: OBP aims to minimize the total number of batches required to fulfill customer orders while respecting the picker capacity. This objective allows OBP to be viewed as an extension of the classical one-dimensional bin packing problem (Pan et al., 2015; Falkenauer, 1998). In this context, FFD algorithm can be a robust approximation method for OBP as well.

In the FFD framework, customer orders are treated as "items" and the picking vehicle (or batch) is treated as the "bin." The size of each item corresponds to the total weight or volume of an individual order. The algorithm operates by first sorting all orders in descending order of weight. Following this sorting phase, each order is allocated using a First-Fit strategy: it is placed into the first existing batch that possesses sufficient remaining capacity. If no such batch exists, a new batch is initialized.

While FFD is a greedy approach that may not always yield a mathematically optimal solution, it consistently produces high-quality results that are close to optimal. Its low time complexity makes it highly suitable for industrial scenarios requiring rapid decision-making. By prioritizing larger items, the "decreasing" aspect of the algorithm effectively reduces bin fragmentation and maximizes space utilization. Efficiently minimizing the total number of batches through FFD directly correlates with a reduction in the total distance traveled by the picker. FFD algorithm is shown as follows.

Algorithm 1 FFD

Sort the set of orders $O = \{ o_1, o_2, \dots o_n \}$ by weight in a descending order
Initialize the set of batches $B = \{b_1\}$ *where* $b_1 = \{\}$
For each order $o_i \in O$:

- **Set** $j = 1$
- **Set** j_{max} = length(B)
- **while** j $\leq j_{max}$ **do**
  - i. **If** order o_i fits in batch b_j:
  - $b_j \leftarrow \{b_j, o_i\}$
  - break
  - ii. **Else if** o_i cannot fit in any existing batch ($j = j_{max}$)
  - Open new batch $B \leftarrow b_{j_{max}+1} = \{o_i\}$
  - Break
  - iii. Increment $j = j + 1$

Return B

AS: This heuristic operates on the premise that if the pick locations for orders i and j share common picking aisles (PA) or sub-aisles, it is logically advantageous to group them into a single batch. A rapid implementation of the savings algorithm can be achieved by utilizing an aisle-saving mechanism rather than traditional time or distance metrics. Within this framework, the savings S are normalized and calculated as follows:

$$S = \frac{PA(i) + PA(j) - PA(i,j)}{PA(i,j)} \times 100\% \quad (10)$$

The primary advantage of this approach lies in its computational efficiency; the objective function for distance calculation is executed only once after the batching process is finalized.

3.2 Phase II: Reduced Variable Neighborhood Search (RVNS)

Variable Neighborhood Search (VNS) is a metaheuristic method that explores the solution space by systematically altering neighborhood structures. A prominent variation is Reduced VNS (RVNS), introduced by Mladenović and Hansen (1997) to solve complex combinatorial optimization problems (Hansen et al., 2019; Brimberg et al., 2023). RVNS has been successfully implemented across various domains (Vincent et al., 2020; Rettl, Pletz, & Schuecker 2023; Yildiz, Ozcan & Cevik 2023) and is notably more computationally efficient than standard VNS. This efficiency stems from the omission of the local search (descent) phase, relying instead on a "shaking" procedure to generate new starting points across different neighborhoods. This allows the algorithm to escape local optima without the heavy overhead of exhaustive local exploration, making it ideal for the JOBPRP.

Let S represent the set of feasible solutions. A transformation applied to solution $s \in S$ results in a neighbor s'only if s' satisfies all feasibility conditions. The algorithm generates a sequence $s_1, \dots, s_m$, where each s_j is a neighbor of s_{j-1} and $f(s_j) < f(s_{j-1})$. If a neighbor fails the feasibility check, the incumbent solution s_{j-1} is retained for further perturbation.

We utilize two distinct neighborhood update strategies, Swap and Shift, to explore the solution space. The search intensity and the transition between these operators are governed by the maximum allowable non-improving iterations, $K \in \{5,10,20,30,40\}$.

Modified Swap Operator (N_1): In the standard version, two orders (i,j) from different batches $B_i \neq B_j$ are chosen randomly and exchanged. However, we have modified this algorithm to perform the choice of (i,j) in a probabilistic manner based on weight similarity. By selecting orders with similar weights for the swap, we significantly increase the probability that the resulting neighbor solution will remain feasible within the capacity constraints. This targeted selection minimizes wasted iterations on infeasible "random" moves, allowing the search to focus on high-quality regions of the solution space. If the weights are compatible, order i is moved to B_j and order j to B_i . The change in distance is calculated as $f_{new\ j} = f(B_i) + fB_j$, and the swap is accepted if $f_{new\ i,j} < f_{ij}$.

Shift Operator (N_2): If the Swap strategy fails to find an improvement after a set number of iterations k (in this paper $k_{max}/2$), the algorithm transitions to the Shift neighborhood. Here, a random order i is moved from B to a different batch B. This transition is accepted only if it maintains feasibility and leads to a reduction in total distance. Finding a better solution resets the search to $k = 1$, returning to the Swap phase. RVNS algorithm is shown as follows:

Algorithm 2 RVNS for JOBPRP

Input: Initial solution s (from FFD or AS), k_{max}, stopping criteria.
While stopping criteria not met:
- **While** $k \leq k_{max}$:
  - **Shaking Phase (Perturbation)**:
    - Generate a random neighbor s' from the current neighborhood $N_k(s)$
    - **If** $k <= k_{max}/2$ **(Modified Swap Operator** N_1**):**
      - Identify two batches $B_i \neq B_j$
      - Select orders (i,j) probabilistically based on weight similarity.
      - Perform swap: move order i to B_j and order j to B_i
      - **Check Feasibility**: Ensure capacity constraints are met.
      - **Evaluate**: $f_{new} = f(B_i) + f(B_j)$
    - **Else (Shift Operator** N_2**)**
      - Select a random order i from batch B.
      - Move order i to a different batch B'
      - **Check Feasibility**: Ensure capacity constraints are met.
  - **Move or Update**:
    - **If** s' is feasible **AND** $f(s') < f(s)$:
      - $s \leftarrow s'$ (Accept improvement)
      - $k \leftarrow 1$ (Reset to first neighborhood)
    - **Else**:
      - $k \leftarrow k + 1$ (Move to next neighborhood structure)

Return s^* (The best solution discovered)

IV. NUMERICAL EXPERIMENTS

The research utilizes 145 problem instances derived from the studies of Briant et al. (2020) and Valle et al. (2017), based on a two-year database of anonymized supermarket customer purchases at a Foodmart factory.

This dataset was specifically selected for its ability to simulate realistic online grocery shopping environments, where individual orders frequently consist of dozens of diverse items.
The simulated environment features a non-typical warehouse layout consisting of two blocks as seen in Figure 1. To test the scalability and robustness of the proposed algorithms, instances were generated by aggregating customer purchases over varying time horizons $d \in \{5, 10, 20\}$ days), representing three increasing levels of order density and variety.

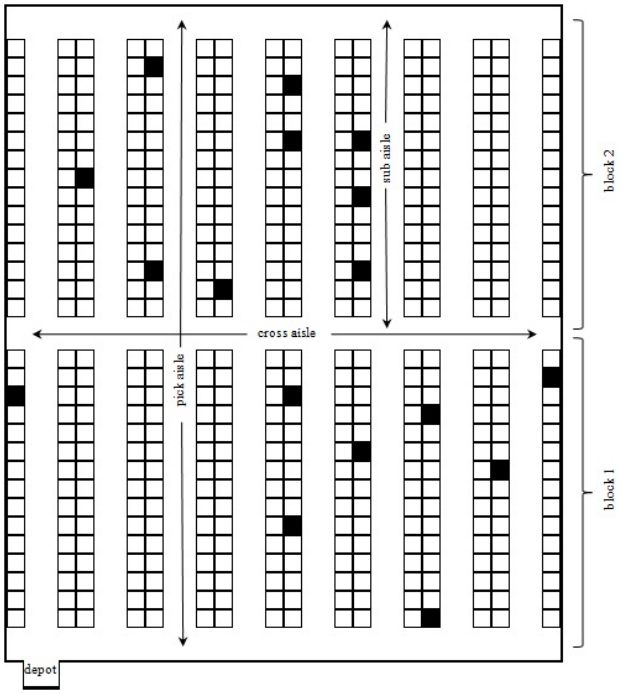


Figure 1: Warehouse layout

All algorithms were implemented and executed using the Google Colab platform. The Reduced Variable Neighborhood Search (RVNS) metaheuristic utilizes a dynamic neighborhood update strategy controlled by the parameter k:

- **Modified Swap Operator ($k <= k_{max}/2$)**: Facilitates the exchange of two orders between different batches based on weight similarity to maintain feasibility.
- **Shift Operator($k > k_{max}/2$):** Moves a single random order to a different batch to explore broader solution structures.

The search process is iterative; whenever an improved solution is identified, k is reset to 1 to restart the neighborhood exploration. In this study, we perform a sensitivity analysis by testing $K \in \{5,10, 20, 30, 40\}$. This allows for a comparison between rapid refinement and deep optimization. For each value of K, the search restarts if an improvement is found, but terminates if K consecutive iterations pass without a reduction in distance. To ensure practical relevance for warehouse managers, a hard time limit of 1,200 seconds is enforced for each instance.
The performance of each method is assessed using three primary metrics:

- **CPU Time:** Measured in seconds to determine computational efficiency.
- **Percentage Gap:** The relative difference between the result and the optimal solution.

To ensure statistical reliability, results are reported as the average and standard deviation for each problem set, capturing both mean performance and observed variability.

V. RESULTS

This section is dedicated to presenting and analyzing the experimental results of the implemented solution methods: **FFD, AS, ASRVNS** (AS improved by RVNS), and **FFDRVNS** (FFD improved by RVNS). For each problem size (d =5, 10, and 20), and each algorithm, the experiments are conducted on 14 distinct problem templates.

5.1 Results for Low Order Density $d = 5$

The results for applying RVNS utilizing FFD as an initial solution is presented in Table 1, whereas the results of utilizing AS as initial solution is presented in Table 2.

Table 1: A comparison of the solution methods for $d = 5$ with FFD as an initial solution

Orders	FFD		FFDRVNS (K=5)		FFDRVNS (K=10)		FFDRVNS (K=20)		FFDRVNS (K=30)		FFDRVNS (K=40)	
	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)
5	7.1%	0.47	7.1%	0.47	7.1%	0.47	7.1%	0.47	7.1%	0.47	7.1%	0.47
6	1.6%	0.86	1.6%	0.86	1.6%	0.86	1.6%	0.86	1.6%	0.86	1.6%	0.86
7	1.5%	1.06	1.5%	1.06	1.5%	1.06	1.5%	1.06	1.5%	1.06	1.5%	1.06
8	35.9%	3.17	0.8%	9.75	0.8%	12.74	0.8%	16.72	0.8%	18.56	0.8%	21.17
9	32.3%	1.79	0.7%	5.25	0.7%	7.16	0.7%	10.65	0.7%	13.07	0.7%	16.35
10	24.5%	1.86	3.1%	3.62	3.1%	5.49	3.1%	9.29	3.1%	10.6	3.1%	13.42
11	20.6%	2.28	4.7%	6.67	3.1%	15.09	1.1%	28.42	1.1%	30.26	1.1%	33.83
12	20.4%	1.39	10.3%	3.37	4.2%	11.59	4.2%	16.72	4.2%	19.74	4.2%	23.1
13	19.1%	1.94	9.9%	8.14	6.9%	13.66	3.4%	29.56	1.6%	48.96	1.6%	52.55
14	17.2%	1.81	6.4%	7.68	6.4%	12.29	6.4%	17.95	6.4%	20.7	6.4%	21.34
15	15.2%	3	2.5%	11.79	2.5%	14.61	2.5%	22.19	2.5%	22.72	2.5%	22.72
20	25.3%	5.53	11.4%	13.45	11.4%	15.97	8.5%	41.55	7.4%	57.54	7.4%	61.56
25	29.5%	11.44	17.2%	17.55	17.2%	20.69	7.2%	71.77	7.2%	72.53	7.2%	73.01
30	35.0%	14.39	29.4%	19.13	27.1%	30.74	25.1%	59.54	24.2%	74.27	24.2%	75.44
Total	23.9%	50.99	10.6%	106.4	9.5%	160.03	7.2%	324.36	6.8%	388.95	6.8%	414.49

The experimental results for the low order density group demonstrate the improvement obtained by RVNS. As expected, increasing the value of parameter k results in improving the solution quality, however, within the first 5 iterations, we can notice a significant impact of RVNS for both cases. For the first 3 experimental samples, the complete requested orders could fit within 1 batch, so RVNS just returned the value obtained by the initial solution.

Table 2: A comparison of the solution methods for $d = 5$ with AS as an initial solution

Orders	AS		ASRVNS (K=5)		ASRVNS (K=10)		ASRVNS (K=20)		ASRVNS (K=30)		ASRVNS (K=40)	
	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)
5	7.1%	0.41	7.1%	0.41	7.1%	0.41	7.1%	0.41	7.1%	0.41	7.1%	0.41
6	1.6%	0.75	1.6%	0.75	1.6%	0.75	1.6%	0.75	1.6%	0.75	1.6%	0.75
7	1.5%	0.56	1.5%	0.56	1.5%	0.56	1.5%	0.56	1.5%	0.56	1.5%	0.56
8	9.3%	0.4	6.6%	2.01	0.8%	6.72	0.8%	11.02	0.8%	13.67	0.8%	16
9	7.2%	0.42	5.8%	3.53	5.8%	6.28	5.8%	10.92	5.8%	13.63	5.8%	16.32
10	4.9%	0.75	3.4%	3.24	3.4%	4.87	3.4%	8.15	3.4%	10.73	3.4%	13.65
11	4.6%	0.66	4.6%	2.82	4.6%	4.53	4.6%	7.6	4.6%	10.49	4.6%	14.58
12	5.9%	0.36	5.9%	2.42	5.9%	5.36	3.1%	15.73	3.1%	22.15	3.1%	25.92
13	4.4%	0.35	4.4%	2.36	4.4%	5.19	4.4%	10.54	1.6%	36.03	1.6%	38.36
14	11.0%	0.38	5.6%	4.1	5.6%	6.49	5.6%	13.15	5.6%	15.21	5.6%	18.06
15	10.2%	0.57	9.4%	4.92	8.7%	12.04	8.7%	17.17	8.7%	18.09	8.7%	18.09
20	8.7%	1.05	6.8%	4.89	6.8%	7.03	6.8%	10.87	6.8%	11.92	6.8%	13.3
25	8.9%	1.25	8.9%	3.39	8.9%	5.22	8.9%	9.26	8.9%	10.07	8.9%	10.79
30	24.2%	1.49	23.4%	4.6	20.7%	17.75	20.4%	26.34	20.4%	28	20.4%	28.93
Total	9.4%	9.4	8.4%	40	7.6%	83.2	7.4%	142.47	7.2%	191.71	7.2%	215.72

Overall, for the $d = 5$ group, the FFDRVNS reduces the total average gap from 23.9% to 6.8%**.** Similarly, the ASRVNS performs strongly, maintaining a final average gap of 7.2%.

Computation time reached to 75,44 Sec for FFDRVNS whereas the maximum computation time for ASRVNS is 38.36 Sec which is acceptable for practical applications. Interestingly, AS provided a much better solution than FFD, which also lead to better performance of RVNS both for solution quality and the execution time.

5.2 Results for Medium Order Density $d = 10$

The results for applying RVNS utilizing FFD and AS for medium order density is presented in tables 3 and 4 respectively.

Table 3: A comparison of the solution methods for $d = 10$ with FFD as an initial solution

Orders	FFD		FFDRVNS (K=5)		FFDRVNS (K=10)		FFDRVNS (K=20)		FFDRVNS (K=30)		FFDRVNS (K=40)	
	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)
5	7.9%	0.64	7.9%	0.64	7.9%	0.64	7.9%	0.64	7.9%	0.64	7.9%	0.64
6	8.7%	1	8.7%	1	8.7%	1	8.7%	1	8.7%	1	8.7%	1
7	31.8%	2.63	4.5%	5.99	4.5%	8.48	4.5%	15.79	4.5%	19.88	4.5%	23.58
8	22.5%	1.58	8.4%	8.65	8.4%	11.55	8.4%	17.33	8.4%	23.27	8.4%	27.18
9	18.9%	2.44	5.1%	8.87	1.0%	14.76	1.0%	22.47	1.0%	25.58	1.0%	30.14
10	17.1%	3.72	6.8%	14.86	6.8%	19.84	6.8%	27.1	5.5%	72.33	5.5%	82.22
11	15.4%	2.56	4.3%	9.52	3.8%	16.82	3.0%	34.32	3.0%	41.26	3.0%	44.89
12	8.5%	2.73	6.2%	11.25	6.2%	15.85	2.9%	49.32	2.9%	57.98	2.9%	67.99
13	7.4%	3.1	3.4%	17.57	3.4%	21.61	3.4%	30.83	3.4%	35.48	3.4%	37.34
14	5.9%	5.41	5.7%	9.34	5.7%	14.09	4.9%	39.93	4.9%	40.97	4.9%	40.98
15	23.6%	7.86	8.4%	11.66	8.4%	15.41	6.5%	32.68	6.5%	39.19	6.5%	40.88
20	15.4%	10.69	11.2%	20.97	10.0%	33.44	6.3%	63.21	6.3%	66.45	6.3%	67.84
25	22.8%	18.44	12.8%	25.35	12.8%	29.12	11.8%	49.81	8.9%	115.91	8.9%	117.06
30	24.6%	24.23	19.9%	31.15	19.9%	37.35	19.9%	44.75	19.9%	46.38	19.9%	46.38
Total	18.1%	87.03	9.3%	176.82	8.9%	239.96	7.9%	429.18	7.5%	586.32	7.5%	628.12

Due to the nature of this group of problems compared with the low-density groups, the computation time needed to solve the problem is relatively more than the previous set.
FFDRVNS demonstrates a significant improvement over the initial solution of FFD improving the total gap from 18.1% to 9.3% just within the first 5 iterations, and to 7.5% for $K = 40$.

Table 4: A comparison of the solution methods for $d = 10$ with AS as an initial solution

Orders	AS		ASRVNS (K=5)		ASRVNS (K=10)		ASRVNS (K=20)		ASRVNS (K=30)		ASRVNS (K=40)	
	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)
5	7.9%	0.4	7.9%	0.4	7.9%	0.4	7.9%	0.4	7.9%	0.4	7.9%	0.4
6	8.7%	0.49	8.7%	0.49	8.7%	0.49	8.7%	0.49	8.7%	0.49	8.7%	0.49
7	4.5%	0.46	4.5%	2.48	4.5%	4.66	4.5%	10.38	4.5%	15.66	4.5%	18.63
8	8.4%	0.48	8.4%	2.83	8.4%	5.94	8.4%	11.35	8.4%	14.68	8.4%	17.74
9	8.9%	0.82	6.6%	6.17	1.0%	16.88	1.0%	22.97	1.0%	25.43	1.0%	27.68
10	8.8%	0.86	5.7%	7.88	5.7%	10.52	5.7%	18.16	5.7%	21.86	5.7%	25.56
11	8.8%	0.66	4.9%	10.78	4.9%	14.03	3.0%	29.45	3.0%	35.12	3.0%	38.12
12	7.7%	0.71	7.7%	4.21	7.2%	14.07	2.9%	54.05	2.9%	59.13	2.9%	67.16
13	7.2%	0.7	7.2%	4.56	6.7%	14.84	6.7%	23.02	3.7%	64.06	3.7%	69.95
14	5.7%	0.79	3.7%	12.47	3.7%	15.22	3.7%	20.44	3.7%	20.44	3.7%	20.45
15	5.2%	1.4	4.8%	6.92	4.0%	12.96	4.0%	19.01	4.0%	21.68	4.0%	22.82
20	14.6%	1.12	12.1%	12.28	9.2%	25.35	9.2%	32.6	9.2%	37.17	9.2%	41.02

25	12.7%	1.31	11.7%	8.83	11.7%	12.07	11.7%	29.46	11.7%	30.46	11.7%	30.46
30	22.1%	2.32	21.8%	8.59	19.8%	21.74	19.8%	28.53	19.8%	29.82	19.8%	29.82
Total	10.6%	12.52	9.5%	88.89	8.5%	169.17	8.1%	300.31	7.9%	376.4	7.9%	410.3

The ASRVNS maintains a steady advantage, starting at a 10.6% gap and concluding at 7.9%**.** Notably, for instances with 20–30 orders, the ASRVNS converges more quickly than FFDRVNS, which imply the performance advantage of the AS as a starting solution to the problem when the complexity increases. The maximum computation time reached 117.06 Sec for FFDRVNS and 69.95 Sec for ASRVNS. A possible explanation to this is the tight neighborhood structure resulted from applying FFD as a batching phase.

5.3 Results for High Order Density $d = 20$

The results of FFD, AS, and RVNS for the high order density group is presented in Tables 5, and 6. As seen from the results, RVNS performed better in most of the experiments of this group compared to the previous groups.

This group achieves the lowest final gaps across all tested scenarios. The ASRVNS reaches a final average gap of just 4.8% at $K = 40$. Both FFD and AS gave an identical overall gap of 14.8%. This indicates that in high-density environments, the choice of initial construction heuristic is less critical than the subsequent metaheuristic refinement.

Table 5: A comparison of the solution methods for $d = 20$ with FFD as an initial solution

Orders	FFD		FFDRVNS (K=5)		FFDRVNS (K=10)		FFDRVNS (K=20)		FFDRVNS (K=30)		FFDRVNS (K=40)	
	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)
5	24.3%	2.79	0.7%	7.8	0.7%	11.32	0.7%	16.71	0.7%	18.99	0.7%	21.95
6	12.5%	1.51	3.2%	9.5	3.2%	11.74	3.2%	16.39	3.2%	20.86	3.2%	23.89
7	9.0%	2.42	2.6%	8.86	2.6%	12.53	2.6%	17.44	1.4%	33.43	1.4%	36.11
8	9.3%	5.15	4.4%	8.82	4.4%	11.74	4.4%	21.08	4.4%	22.35	3.8%	43.87
9	5.5%	6.07	5.1%	7.6	3.3%	12.7	3.3%	14.62	3.3%	14.62	3.3%	14.62
10	21.5%	6.59	4.6%	11.29	4.6%	14.22	4.6%	21.3	4.6%	22.96	3.4%	44.03
11	15.0%	7.52	7.4%	10.35	6.0%	21.28	6.0%	26.51	3.2%	36.91	3.2%	38.06
12	13.4%	6.25	7.9%	14.23	5.5%	23.51	5.5%	29.76	4.4%	41.79	4.4%	43.4
13	12.6%	7.52	6.7%	16.45	5.3%	22.26	5.3%	30.06	5.3%	33.72	5.3%	36.26
14	12.4%	9.89	9.5%	13.62	8.4%	16.8	8.4%	20.67	8.4%	22.04	8.4%	22.04
15	11.1%	10.55	11.1%	10.55	5.5%	12.46	5.5%	14.02	5.5%	14.52	5.5%	14.52
20	13.4%	22.3	9.5%	25.42	9.5%	26.01	9.5%	26.01	9.5%	26.01	9.5%	26.01
25	16.6%	38.48	16.2%	41.48	11.0%	48.96	10.7%	59.65	10.7%	62.04	10.7%	63.64
30	20.1%	66.3	15.0%	71.55	15.0%	72.55	12.4%	103.36	12.4%	108.95	12.4%	111.81
Total	14.8%	193.3	9.1%	257.52	7.5%	318.08	7.1%	417.58	6.7%	479.19	6.6%	540.21

Even for the largest instance of 30 orders in this group, the ASRVNS manages to reduce the gap from 27.6% to 12.4%. While the complexity is high, the high density allows the RVNS to find more swaps in the neighborhood search, leading to an improved result.

Table 6: A comparison of the solution methods for $d = 20$ with AS as an initial solution

Orders	AS		ASRVNS (K=5)		ASRVNS (K=10)		ASRVNS (K=20)		ASRVNS (K=30)		ASRVNS (K=40)	
	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)	GAP	Time (Sec)
5	24.3%	577.77	0.7%	0.33	0.7%	1.87	0.7%	3.39	0.7%	6.98	0.7%	9.34
6	12.5%	678.18	8.4%	0.41	3.2%	3.03	3.2%	4.86	3.2%	8.84	3.2%	11.13
7	9.0%	699.68	2.6%	0.4	2.6%	2.7	2.6%	4.9	2.6%	10.36	2.6%	14.06
8	9.3%	719.68	3.0%	0.59	3.0%	2.26	3.0%	3.81	3.0%	7.53	3.0%	12.81

9	5.5%	749.68	3.1%	0.62	3.1%	2.94	3.1%	3.53	3.1%	10.05	3.1%	10.05
10	21.5%	946.49	5.7%	0.64	4.4%	3.06	4.4%	5.66	4.4%	12.94	4.4%	14.62
11	15.0%	1017.1	7.2%	0.81	5.0%	3.56	4.5%	6.4	3.6%	19.17	3.6%	21.71
12	13.4%	1024.4	6.5%	0.65	3.6%	4.16	3.6%	5.74	2.5%	22.87	2.5%	26.65
13	12.6%	1036.5	7.0%	0.76	3.8%	4.4	3.8%	6.58	3.8%	12.33	2.7%	29.22
14	12.4%	1054.5	5.7%	0.89	5.7%	2.07	5.7%	5.29	4.8%	13.6	4.1%	23.93
15	11.1%	1088.5	7.2%	0.87	6.8%	3.35	6.8%	4.01	6.8%	4.54	5.8%	13.36
20	13.4%	1381.6	7.5%	1.05	7.1%	4.31	5.4%	9.84	3.6%	19.2	3.6%	19.2
25	16.6%	1714.7	10.6%	2.25	9.7%	4.74	7.4%	11.37	7.4%	13.85	6.5%	24.82
30	20.1%	2066.1	13.3%	2.38	13.3%	4.56	12.3%	11.03	10.8%	28.02	10.8%	31.12
Total	14.8%	14754.8	7.4%	12.65	6.3%	47.01	5.7%	86.41	5.1%	190.28	4.8%	262.02

The total computational time for $d = 20$ increases due to the higher number of pick locations to evaluate with ASRVNS completing the K=40 with a maximum of 31.12 Sec for ASRVNS and 111.81 Sec for FFDRVNS. This suggests that RVNS can be efficient for more complex problems with even larger orders.

5.4 Sensitivity Analysis

The sensitivity graphs for all density groups ($d = 5, 10, 20$) are presented in Figures 2,3,4,5 respectively, and are concluded in Table 7. As seen from all the figures, the analysis shows a clear elbow shape, which is normally associated with the most substantial reduction in the average gap. This reduction is occurring between $K = 0$ and $K = 10$.

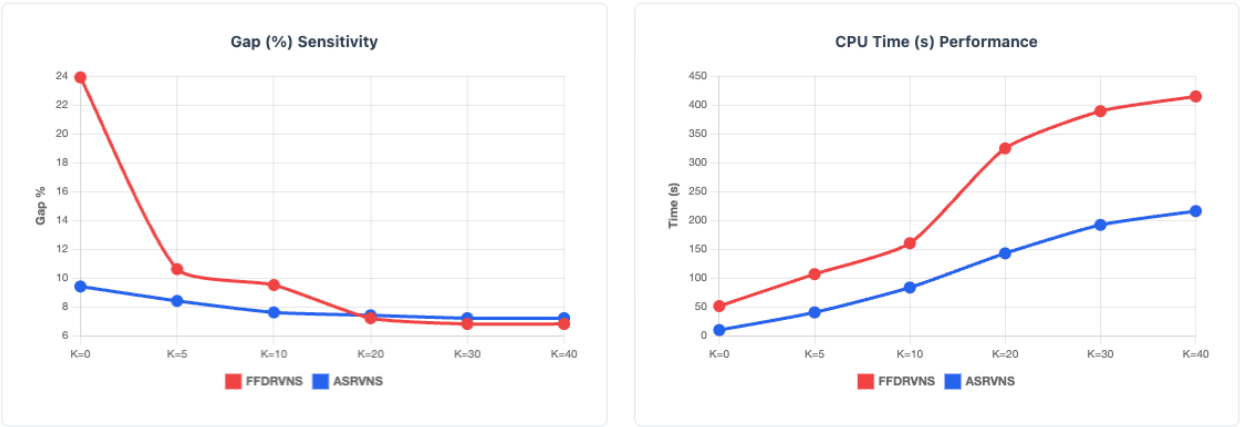


Figure 2: Sensitivity Analysis for d=5 group

Beyond $K = 20$, the improvement curve toward a more liner direction. For example, in the global summary, increasing k from 20 to 40 only yields a 0.5% improvement in the average gap (from 6.9% to 6.4%), while significantly increasing the total CPU time.

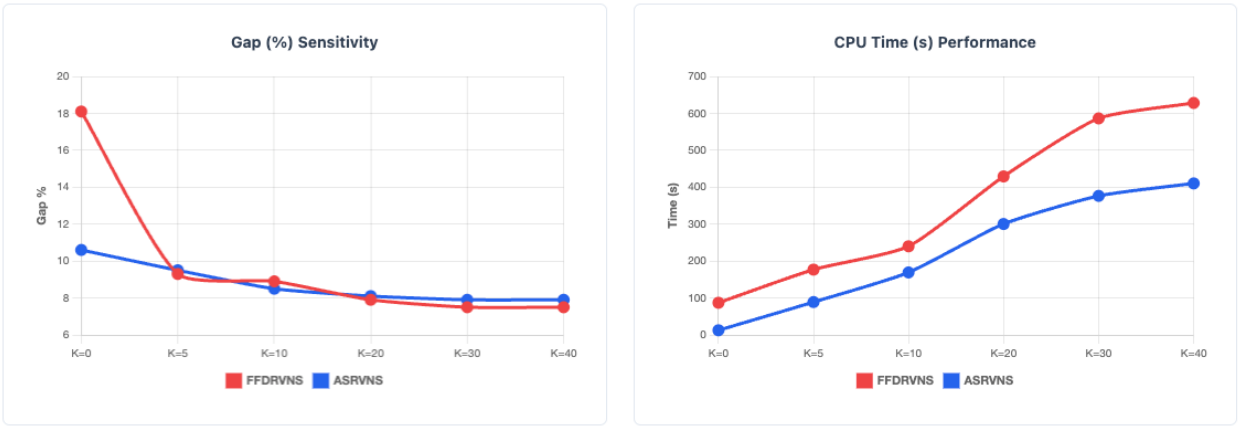
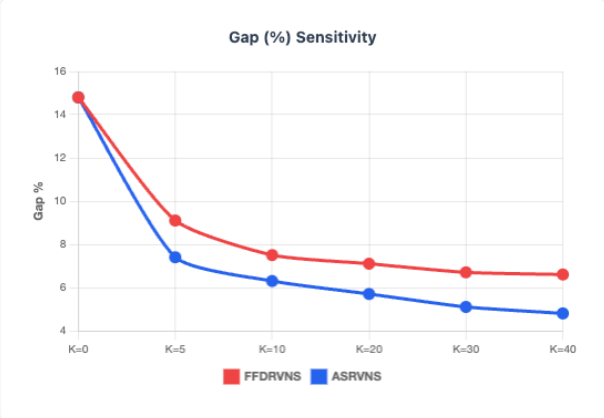


Figure 3: Sensitivity Analysis for d=10 group

The relationship between K and CPU time is near-linear for both algorithms. However, a significant discrepancy in efficiency is observed between the two starting heuristics. The total CPU time for FFDRVNS at $K = 40$ is 1,582.82 Sec, while ASRVNS achieves a much less gap on average within only 888.04 Sec. This 44% reduction in time suggests that the AS heuristic provides a more stable foundation for neighborhood exploration, allowing the RVNS to reach a quality results faster than the FFD-based approach.



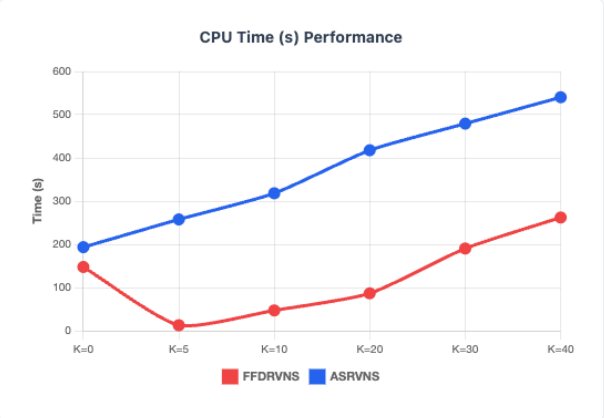
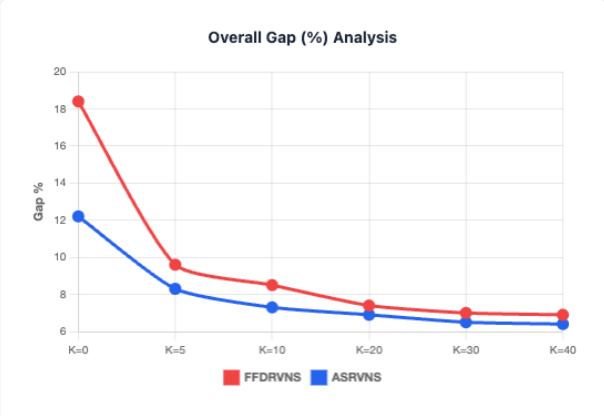


Figure 4: Sensitivity Analysis for d=20 group



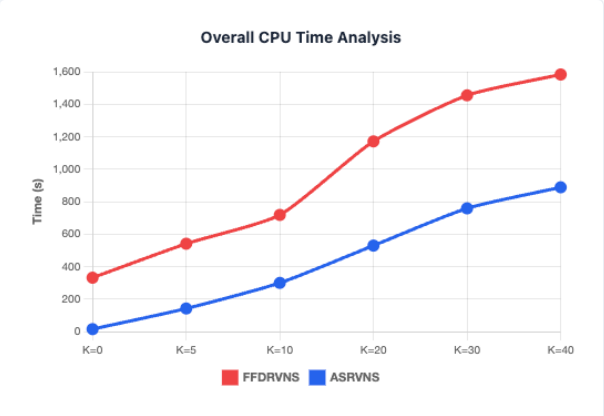


Figure 4: Sensitivity Analysis for the total summary of problems

Regardless of the initial order density, the Modified RVNS demonstrates high robustness. While the $d = 5$ group starts with the highest initial error (23.9%), the metaheuristic brings all groups into a final gap range of 4.8% to 7.9% at $K = 40$. This indicates that the proposed two-stage framework is flexible and can be applied effectively in warehouses regardless of order sizes or item distributions.

Table 7: A comparison of the solution methods for all the problems

Parameter K	FFDRVNS Gap (%)	ASRVNS Gap (%)	FFDRVNS Time (s)	ASRVNS Time (s)
5	9.6%	8.3%	540.74	141.54
10	8.5%	7.3%	718.07	299.38
20	7.4%	6.9%	1,171.12	529.19
30	7.0%	6.5%	1,454.46	758.39
40	6.9%	6.4%	1,582.82	888.04
Total	18.4%	12.2%	331.36	14,776.72*

VI. CONCLUSION

In this paper, we proposed a robust two-stage solution framework applied to the Foodmart factory layout, utilizing FFD and AS heuristics for initial construction, followed by a RVNS for iterative refinement.

The experimental results, validated across multiple order densities and neighborhood sizes ($K = 5, 10, 20, 30, 40$), demonstrate the effectiveness of the proposed framework. The findings show that RVNS achieved significant reductions in travel distance and solution gaps, particularly in complex scenarios where initial heuristics couldn't reach good results.

The comparative analysis showed that starting the search with AS heuristic (ASRVNS) generally outperformed the FFDRVNS in both solution quality and computational efficiency. ASRVNS consistently reached better

optimality gaps in nearly half the CPU time. This highlights the importance of neighborhood structure of the solution space. Furthermore, the sensitivity analysis identified $K = 20$ for balancing computational effort with solution quality.

In conclusion, this research provides warehouse managers and practitioners with a scalable and robust tool for optimizing the JOBPRP. By systematically exploring stochastic neighborhoods, the RVNS ensures high-quality picker planning for both small and large size problems. Future research directions may include extending this framework to dynamic environments with real-time order arrivals or integrating heterogeneous picking devices to further enhance distribution center productivity.

REFERENCES

[1]. Boz, E., & Aras, N. (2022). The order batching problem: A state-of-the-art review. *Sigma Journal of Engineering and Natural Sciences*, 40(2), 402–420.

[2]. Brimberg, J., Salhi, S., Todosijević, R., & Urošević, D. (2023). Variable Neighborhood Search: The power of change and simplicity. *Computers & Operations Research*, 155, 106221.

[3]. Cheng, C. Y., Chen, Y. Y., Chen, T. L., & Yoo, J. J. W. (2015). Using a hybrid approach based on the particle swarm optimization and ant colony optimization to solve a joint order batching and picker routing problem. *International Journal of Production Economics*, 170, 805-814.

[4]. Falkenauer, E. (1998). *Genetic algorithms and grouping problems.* John Wiley & Sons, Inc.

[5]. Hansen, P., Mladenović, N., Brimberg, J., & Pérez, J. A. M. (2019). Variable neighborhood search. In *Handbook of Metaheuristics* (pp. 57-97). Springer International Publishing.

[6]. Kulak, O., Sahin, Y., & Taner, M. E. (2012). Joint optimization of order batching and picker routing in picker-to-part systems. *International Journal of Production Research*, 50(6), 1691-1704.

[7]. Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11), 1097-1100.

[8]. Pan, J. C. H., Shih, P. H., & Wu, M. H. (2015). Order batching in a pick-and-pass warehousing system with group genetic algorithm. *Omega*, 57, 238-248.

[9]. Pardo, E. G., Gil-Borrás, S., Alonso-Ayuso, A., & Duarte, A. (2024). Order batching problems: Taxonomy and literature review. *European Journal of Operational Research*, 313(1), 1–24.

[10]. Pinto, A. R. F., & Nagano, M. S. (2019). An approach for the solution to order batching and sequencing in picking systems. *Production Engineering*, 13, 325-341.

[11]. Pinto, A. R. F., & Nagano, M. S. (2022). A review and analysis of the joint order batching and picker routing problem. *Logistics*, 6(3), 48.

[12]. Rettl, M., Pletz, M., & Schuecker, C. (2023). Evaluation of combinatorial algorithms for optimizing highly nonlinear structural problems. *Materials & Design*, 230, 111958.

[13]. Roodbergen, K. J., & De Koster, R. (2001a). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9), 1865-1883.

[14]. Roodbergen, K. J., & De Koster, R. (2001b). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, 133(1), 32-43.

[15]. Vincent, F. Y., Maglasang, R., & Tsao, Y. C. (2020). A reduced variable neighborhood search-based hyperheuristic for the shelf space allocation problem. *Computers & Industrial Engineering*, 143, 106420.

[16]. Yildiz, S. T., Ozcan, S., & Cevik, N. (2023). Variable neighborhood search-based algorithms for the parallel machine capacitated lot-sizing and scheduling problem. *Journal of Engineering Research*, 100145.

[17]. Zhang, J., Zhang, Y., & Zhang, X. (2021). The study of joint order batching and picker routing problem with food and nonfood category constraint in online-to-offline grocery store. *International Transactions in Operational Research*, 28(5), 2440-2463.

[18]. Žulj, I., Kramer, S., & Schneider, M. (2018). A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. *European Journal of Operational Research*, 264(2), 653-664.