Mandalika Mathematics and Education Journal
Volume 7 Nomor 4, Desember 2025
e-ISSN 2715-1190| |p-ISSN 2715-8292
DOI: http://dx.doi.org/10.29303/jm.v3i1.10213

Evaluating Swarm-Genetics for VRPTW: Robustness Across Seeds and Fleet Efficiency on Solomon Benchmarks

Aprizal Resky[1*], Zaitun[2], Dhirga Tandi Teppa[1]

[1,3] Data Science, Department of Science, Institut Teknologi Bacharuddin Jusuf Habibie, Parepare
[2] Mathematics, Department of Science, Institut Teknologi Bacharuddin Jusuf Habibie, Parepare

aprizalresky@ith.ac.id

Abstract
The Vehicle Routing Problem with Time Windows (VRPTW) is a challenging NP-hard problem in logistics optimization. This study evaluates a Swarm-Genetics algorithm, a hybrid method combining Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) with swarm regeneration and adaptive parameter control. The algorithm was tested on 57 Solomon benchmark instances (C, R, RC) under three random seeds to assess robustness. Results show that the algorithm is robust across seeds, producing stable outcomes with minimal variation. It frequently preserves fleet efficiency, often matching the Best Known Solutions (BKS) in vehicle count, particularly for clustered instances. However, routing distances remain less competitive, with average gaps of about 10% for clustered, 12–13% for random, and over 20% for mixed cases. Convergence analysis further indicates rapid early improvements but stagnation in complex distributions. Overall, Swarm-Genetics provides a robust and fleet-efficient framework, though further enhancements are needed to improve distance quality.

Keywords: VRPTW; Swarm-Genetics; robustness analysis; fleet efficiency; Solomon Benchmark

1. INTRODUCTION

Efficient logistics and transportation planning have become increasingly important in the era of e-commerce, urbanization, and smart mobility. Companies face growing demands for cost reduction, timely delivery, and sustainable operations, which intensifies the need for effective optimization models in distribution systems (Braekers et al., 2015; Lahyani et al., 2014). Among these, the Vehicle Routing Problem with Time Windows (VRPTW) is a critical variant of the classical Vehicle Routing Problem (VRP), requiring vehicles to serve customers within specific time intervals while minimizing operational costs. Due to its NP-hard nature, VRPTW remains one of the most challenging problems in combinatorial optimization (Elshaer & Awad, 2020; Li et al., 2016).

Various approaches have been developed to tackle VRPTW, including exact methods, constructive heuristics, and metaheuristics. Exact approaches can solve small-scale problems but become impractical for larger instances (Pisinger & Ropke, 2014). Constructive heuristics offer rapid solutions but often fail to achieve near-optimal quality (Lahyani et al., 2014). Metaheuristics have thus emerged as dominant methods, including Genetic Algorithms (GA) (Ombuki-Berman &

Hanshar, 2009), Particle Swarm Optimization (PSO) (Nguyen et al., 2021), Ant Colony Optimization (ACO) (Abdelmaguid & Dessouky, 2006), and hybrid frameworks combining different strategies (Liang et al., 2020) (Guo et al., 2022). Despite their success, classical GA often suffers from premature convergence, while PSO can stagnate in local optima when population diversity is low.

Recent works have increasingly focused on hybrid metaheuristics, particularly those integrating PSO and GA, as they combine PSO's global exploration with GA's recombination strength (Zhang et al., 2020) (Yu et al., 2019). However, two key limitations remain. First, many hybrids lack robustness when tested under multiple random seeds, raising concerns about stability in repeated runs (Elshaer & Awad, 2020). Second, most studies emphasize minimizing distance while underemphasizing fleet efficiency, although reducing the number of vehicles often yields greater operational and environmental benefits (Xu et al., 2025).

To address these gaps, this study introduces the Swarm-Genetics algorithm, a hybrid PSO–GA framework enhanced with swarm regeneration and adaptive parameter control. The approach aims to maintain diversity, prevent premature convergence, and achieve a balance between exploration and exploitation. The algorithm is tested on the Solomon benchmark instances (Solomon, 1987), comprising clustered (C), random (R), and mixed (RC) distributions, under three independent random seeds. Performance is evaluated in terms of number of vehicles, total distance, deviation from Best Known Solutions (BKS), runtime, and convergence dynamics (Savelsbergh & Vigo, 2014).

The main contributions of this paper are threefold: 1. Proposing a Swarm-Genetics hybrid PSO–GA algorithm with swarm regeneration and adaptive parameters for VRPTW; 2. Conducting a comprehensive evaluation on Solomon benchmarks, covering 57 instances across C, R, and RC categories with multiple seeds; 3. Demonstrating that the algorithm achieves robust and fleet-efficient performance, while identifying its limitations in reducing routing distance, thereby providing directions for future improvement (Yassen et al., 2017).

1.1. Vehicle Routing Problem with Time Windows (VRPTW)

The Vehicle Routing Problem with Time Windows (VRPTW) extends the classical VRP by requiring each customer to be served within predefined time intervals. This additional constraint makes the problem more realistic but also computationally challenging (Qiao et al., 2023). VRPTW has direct applications in parcel delivery, public transportation, and urban logistics, where efficiency is measured not only by route distance but also by the number of vehicles and service reliability (Braekers et al., 2015; Lahyani et al., 2014). Because VRPTW is NP-hard, solving real-world instances at scale requires heuristics and metaheuristics (Elshaer & Awad, 2020; Guo et al., 2022).

The Solomon benchmark dataset (Solomon, 1987) remains the most widely used testbed, with instances divided into clustered (C), random (R), and mixed (RC) customer distributions. Despite being more than three decades old, Solomon's dataset continues to be the standard for algorithm comparison (Liang et al., 2020; Xu et al., 2025), often extended with larger test sets such as those of Homberger and Gehring.

Metaheuristics are the dominant approach for VRPTW, balancing exploration and exploitation. Genetic Algorithms (GA) have been extensively applied, leveraging crossover and mutation to explore diverse solutions (Elshaer & Awad, 2020; Ombuki-Berman & Hanshar, 2009). However, GA may converge prematurely without sufficient diversity preservation. Particle Swarm Optimization (PSO) has shown strong performance due to its simple structure and adaptive search (Nguyen et al., 2021), though it is vulnerable to stagnation when swarm diversity decreases.

Other metaheuristics include Ant Colony Optimization (ACO) (Abdelmaguid & Dessouky, 2006), Tabu Search (TS), and Simulated Annealing (SA), each offering different trade-offs. Recently, more advanced methods such as Large Neighborhood Search (LNS) (Pisinger & Ropke, 2014), Memetic Algorithms, and Variable Neighborhood Search (VNS) (Guo et al., 2022) have been introduced, often outperforming classical GA or PSO by integrating problem-specific local search.

1.2. Hybrid PSO-GA and Related Approaches

To overcome the individual weaknesses of GA and PSO, researchers have developed hybrid approaches combining their strengths. PSO provides global exploration, while GA contributes recombination operators for diversification. Recent studies confirm that PSO–GA hybrids outperform standalone methods, particularly for medium-scale VRPTW (Zhang et al., 2020)(Yu et al., 2019). (Liang et al., 2020) further showed that integrating adaptive parameter control enhances solution quality.

Despite these advantages, limitations remain. Many hybrids are sensitive to parameter settings and lack systematic evaluation across random seeds, raising questions of robustness (Elshaer & Awad, 2020). Moreover, most focus on minimizing routing distance, while fleet efficiency (number of vehicles) is often underemphasized despite its major impact on operational cost and sustainability (Xu et al., 2025). For mixed (RC) instances, premature convergence is common, as heterogeneous distributions increase the search complexity.

1.3. Research Gap and Motivation

The reviewed literature highlights significant progress in applying metaheuristics and hybrids to VRPTW. However, three key gaps remain:

1. Robustness across runs is rarely emphasized; few studies validate stability under multiple seeds, which is critical for real-world deployment;
2. Fleet efficiency is often overlooked, even though minimizing the number of vehicles can have greater economic and environmental impact than minimizing distance;
3. Premature convergence persists in complex RC instances, underscoring the need for mechanisms that maintain diversity and intensify search simultaneously.

Motivated by these gaps, this study introduces Swarm-Genetics, a hybrid PSO–GA algorithm with swarm regeneration and adaptive parameter control. Unlike prior hybrids, the method explicitly emphasizes robustness and fleet efficiency in addition to routing distance. A comprehensive evaluation is conducted on 57 Solomon instances under three independent seeds, providing insights into stability, solution quality, and convergence behavior.

2. RESEARCH METHOD

2.1. Problem Formulation

The Vehicle Routing Problem with Time Windows (VRPTW) can be formally defined on a directed graph $G = (V, E)$, where $V = \{0,1,2,\dots,n\}$ is set of vertices representing the depot (0) and n customers, and E is the set of edges between them. Each customer $i \in V$ has a demand q_i, a service time s_i, and a time window $[e_i, l_i]$. Each vehicle has a maximum capacity Q and starts/ends at the depot (Prodhon & Prins, 2016). The travel time and distance between two nodes i and j are denoted by t_{ij} and d_{ij}, respectively. The objective is to design a set of routes R such that: 1. Each customer is visited exactly once by a single vehicle; 2. The total demand on each route does not exceed vehicle capacity Q; 3. Vehicles must arrive within the time window $[e_i, l_i]$, early arrivals are allowed but require waiting while late arrivals are infeasible (Ponis et al., 2015).

The optimization problem is formulated as follows:

$$\text{Minimize } (NV, TD) \tag{1a}$$

where:

$$NV = |K_{used}| = \sum_{k \in K} u_k \tag{1b}$$

$$TD = \sum_{k \in K} \sum_{(i,j) \in E} d_{ij} x_{ijk} \tag{1c}$$

Thus, the weighted-sum form is written explicitly as:

$$\text{Minimize } Z = \alpha \cdot NV + \beta \cdot TD = \alpha \sum_{k \in K} u_k + \beta \sum_{k \in K} \sum_{(i,j) \in E} d_{ij} x_{ijk} \tag{1}$$

where:

- u_k = 1 if vehicle k is used, and 0 otherwise;
- x_{ijk} = 1 if vehicle k travels from node i to j, 0 otherwise;
- d_{ij} denotes the travel distance from node i to node j;
- α, β = weighting coefficients balancing fleet size and routing distance (in this study, priority is given to minimizing vehicles first (NV), the distance (TD).

Subject to:

$$\sum_{k \in K} \sum_{j \in V, j \neq i} x_{ij}^k = 1 \ , \forall i \in V \tag{2}$$

$$\sum_{j \in V, j \neq i} x_{ji}^k = \sum_{j \in V} x_{ij}^k \quad \forall k \epsilon K, \forall i \in V \tag{3}$$

$$\sum_{i \in V} q_i \cdot y_{ik} \leq Q, \forall k \in K \tag{4}$$

$$e_i \leq a_i \leq l_i, \forall i \in V \tag{5}$$

$$a_j \geq \left(a_i + s_i + t_{ij}\right) \cdot x_{ij}^k \quad \forall (i,j) \in A, \forall k \in K \tag{6}$$

$$x_{ij}^k \in \{0,1\} \, \forall (i,j) \in A, \forall k \in K \tag{7}$$

$$y_i^k \in \{0,1\} \forall i \in V, \forall k \in K \tag{8}$$

$$a_i \geq 0 \, \forall i \in V \tag{9}$$

where a_i is the arrival time at customer i, and $y_{ik} = 1$ if customer i assigned to vehicle k.

This formulation reflects the dual optimization objectives of VRPTW: (1) minimizing the number of vehicles, and (2) minimizing total travel distance. This dual focus aligns with real-world logistics priorities, where fleet reduction often yields higher cost and emission savings than distance minimization alone (Braekers et al., 2015)(Xu et al., 2025).

2.2. Proposed Swarm-Genetics Algorithm

To address the VRPTW formulation described in Section 2.1, this study proposes the Swarm-Genetics algorithm, a hybrid metaheuristic that integrates Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) within a unified framework. The design objective is to leverage the exploration strength of PSO and the recombination power of GA, while mitigating their respective weaknesses: PSO's tendency to stagnate and GA's risk of premature convergence.
The algorithm operates on a population of candidate solutions, each represented as a sequence of customer visits decoded into feasible routes using a best-insertion heuristic. The main steps are as follows:

1. Initialization: an initial swarm of particles (solutions) is generated using a randomized best-insertion decoder to ensure feasibility with respect to vehicle capacity and time windows. Each particle is assigned a position (solution vector) and velocity.
2. Fitness evaluation: each solution is evaluated based on a two-level fitness function prioritizing; 1. Number of vehicles (NV) – to ensure fleet efficiency; 2. Total distance (TD) – to minimize routing cost. The fitness function is formulated as:
$$f(x) = \lambda \cdot NV(x) + (1-\lambda) \cdot \frac{TD(x)}{TD_{max}} \quad (10)$$
where $\lambda \in [0,1]$ emphasizes fleet size over distance.
3. PSO update: particle velocities and positions are updated using the standard PSO rule with inertia weight, cognitive, and social components:
$$v_{id}(t+1) = \omega \cdot v_{id}(t) + c_1 \cdot r_1 \cdot (pbest_{id} - x_{id}(t)) + c_2 \cdot r_2 \cdot (gbest_d - x_{id}(t)) \quad (11)$$
$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (12)$$
where ω is the inertia weight, c_1, c_2 are acceleration coefficient, and r_1, $r_2 \sim U(0,1)$.
4. GA operators: to preserve diversity, a subset of the population undergoes GA operations; 1. Crossover: order-based crossover combines segments of parent solution; 2. Mutation: adaptive mutation swaps or inserts customers to explore new routes.
5. Swarm regeneration: if the population stagnates for a predefined number of iterations, swarm regeneration is triggered. A fraction of the worst-performing particles are replaced with new individuals generated via randomized heuristics, restoring diversity and preventing premature convergence.
6. Adaptive parameter control: the inertia weight ω and mutation probability p_m are adapted dynamically. Early iterations: higher ω, lower $p_m \rightarrow more\ exploration$, later iterations: lower ω, higher $p_m \rightarrow more\ exploitation$. This balances global and local search, ensuring the algorithm continues to refine solutions until convergence.
7. Local search (best-insertion): each solution is further refined using a best-insertion procedure that attempts to reinsert customers at positions minimizing incremental cost. This ensures feasibility and local intensification.
8. Termination: the algorithm iterates until a maximum number of iterations (300) is reached or no improvement is observed for a fixed number of iterations. The best solution found is reported.

To ensure a fair comparison, the parameters used in this algorithm are set to be the same.

Table 1. *Parameter Setting in PSO*

Process	Parameter
PSO	$w(t) = w_{max} - t\frac{w_{max} - w_{min}}{I_{max}}, w_{max} = 0.95,$ $w_{min} = 0.4, c_1 = c_2 = 1.45, I_{max} = 300$
GA	$pc = 1, pm = 0.4, I_{max} = 300$
Swarm-Genetics	$w(t) = w_{max} - t\frac{w_{max} - w_{min}}{I_{max}}, w_{max} = 0.95,$ $w_{min} = 0.4, c_1 = c_2 = 1.45, \quad pc = 1, pm = 0.4, I_{max} = 300$

These mechanisms collectively aim to deliver a robust and fleet-efficient solution method for VRPTW.

2.3. Algorithm Framework

The overall workflow of the proposed Swarm-Genetics algorithm is summarized in **Algorithm 1**. The pseudocode highlights the integration of PSO and GA operators, together with swarm regeneration and adaptive parameter control.

Input: Solomon VRPTW instance, parameters (population size N, max iterations T, crossover probability p_c, mutation probability p_m, inertia weight ω)
Output: Best solution gbest

1. Initialize population of N particles using best-insertion heuristic
2. For each particle i:
   - Evaluate fitness $f(x_i)$ based on number of vehicles and distance;
   - Set $pbest\_i \leftarrow x_i$.
3. Set $gbest \leftarrow best\ of\ \{pbest\_i\}$
4. For $t = 1$ to T do
   a. For each particle i:
      - Update velocity v_i and position x_i using PSO equations;
      - Apply order-based crossover with probability p_c;
      - Apply adaptive mutation with probability p_m;
      - Repair x_i with best-insertion heuristic to ensure feasibility;
      - Evaluate fitness $f(x_i)$;
      - If $f(x_i) < f(pbest\_i)$: $pbest\_i \leftarrow x_i$;
   b. Update gbest from $\{pbest\_i\}$;
   c. Apply swarm regeneration if stagnation detected;
   d. Adapt parameters (ω, p_m) according to iteration t;
5. Return gbest as final solution.

Algorithm 1. *Swarm-Genetics for VRPTW.*

Swarm regeneration restores diversity during stagnation, and adaptive parameter control ensures a gradual transition from exploration to exploitation. The integration of these mechanisms makes the algorithm more robust and fleet-efficient when solving VRPTW.

3. RESULT AND DISCUSSION

3.1. Experimental Setup Recap

To provide a clear overview of the computational environment and parameter settings, the experimental configuration of the proposed Swarm-Genetics algorithm is summarized in **Table 2**. The table highlights the dataset characteristics, algorithmic parameters, hybrid operators, and evaluation metrics employed throughout the study, serving as a concise reference for the subsequent analysis of results.

Table 2. *Experimental Setup Summary*

Aspect	Configuration / Description
Dataset	Solomon VRPTW benchmark (57 instances: C, R, RC)
Problem types	C (clustered), R (random), RC (mixed)
Population size	60 particles/chromosomes
Max iterations	300
Local search	Best-insertion (full mode)
Hybrid operators	PSO velocity-position updates + GA crossover & adaptive mutation
Swarm Regeneration	Applied periodically to maintain diversity
Random seeds	11, 23, 47 (three runs per instance)
Evaluation metrics	Number of vehicles (NV); Total distance (TD); Gap vs BKS (%); Runtime (sec); Convergence stability

3.2. Overall Performances vs. BKS

The aggregated results across the three random seeds provide insight into the overall performance of the Swarm-Genetics algorithm relative to the benchmark solutions. Table 2 summarizes the average number of vehicles, routing distance, deviation from the Best Known Solutions (Gap %), and computation time for the three categories of Solomon instances (C, R, and RC).

Table 3*. Overall Performance by Category*

Category	Avg. Vehicles	Avg. Distance	Avg. Gap (%)	Avg. Runtime (sec)
C (Clustered)	6.96	800.48	10.23	4635.2
R (Random)	8.97	1200.72	12.71	5550.8
RC (Mixed)	8.94	1368.23	21.13	4387.2

From the table, it can be observed that the algorithm performs best on **clustered (C)** instances, achieving the lowest average gap (≈10.2%) with relatively fewer vehicles and shorter routes. The **random (R)** category is more challenging, reflected by a higher average gap of ≈12.7%. The **mixed (RC)** category exhibits the largest gap (≈21.1%), indicating that hybrid distribution patterns are more difficult to optimize consistently. In terms of runtime, the algorithm required between 4,300 and 5,500 seconds per instance on average, with clustered and mixed categories converging slightly faster than the random category.

These results highlight the adaptability of the Swarm-Genetics algorithm across different problem structures, while also confirming that problem difficulty increases as customer distributions become more heterogeneous.

3.3. Best Performance per Instance

To complement the category-level averages, the best solution obtained across the three random seeds was identified for each Solomon instance. The selection criterion prioritized the minimum number of vehicles, followed by the shortest total distance, and finally the lowest runtime in case of ties. This procedure ensures that the reported results reflect the most favorable outcome achieved by the algorithm.

Table 4. *Best Performance per Instance.*

Instance	BKS Vehicles	Best Vehicles	Δ Vehicles	BKS Distance	Best Distance	Gap (%)	Runtime (sec)
C101	10	11	+1	828.94	858.74	3.60	2845.1
C102	10	10	+0	828.94	908.59	9.61	2959.8
C103	10	10	+0	828.06	909.30	9.81	3368.0
C104	10	9	-1	824.78	956.99	16.03	3325.2
C105	10	11	+1	828.94	996.53	20.22	3029.0
C106	10	11	+1	828.94	909.21	9.68	3034.5
C107	10	11	+1	828.94	862.19	4.01	3069.8
C108	10	11	+1	828.94	934.62	12.75	2821.0
C109	10	10	+0	828.94	1017.42	22.74	3573.1
C201	3	3	+0	591.56	591.56	-0.00	4353.4
C202	3	3	+0	591.56	591.56	-0.00	5437.7
C203	3	3	+0	591.17	591.17	0.00	5804.6
C204	3	3	+0	590.60	594.14	0.60	6872.0
C205	3	3	+0	588.88	588.88	-0.00	5914.1
C206	3	3	+0	588.49	588.49	0.00	6156.4
C207	3	3	+0	588.29	588.29	-0.00	5322.1
C208	3	3	+0	588.32	588.32	0.00	5351.2
R101	19	20	+1	1645.79	1681.38	2.16	2750.7
R102	17	17	+0	1466.63	1530.61	4.36	2901.9
R103	13	14	+1	1208.70	1250.26	3.44	3214.3
R104	11	11	+0	971.54	1038.20	6.86	4965.6
R105	14	16	+2	1355.27	1513.88	11.70	2780.8
R106	12	14	+2	1251.98	1336.05	6.71	4505.3
R107	11	12	+1	1061.27	1218.87	14.85	3059.2
R108	10	11	+1	960.88	1199.08	24.79	3216.1
R109	11	14	+3	1146.32	1522.87	32.85	2908.9
R110	10	13	+3	1068.00	1403.16	31.38	4256.6
R111	10	12	+2	1048.13	1248.55	19.12	3516.1
R112	10	14	+4	982.14	1196.83	21.86	4163.4
R201	4	4	+0	1252.37	1347.16	7.57	9254.4
R202	3	4	+1	1191.70	1170.05	-1.82	7402.3
R203	3	3	+0	939.50	1013.87	7.92	7678.3
R204	2	3	+1	825.52	799.47	-3.16	8367.4
R205	3	3	+0	994.42	1329.79	33.73	6346.0
R206	3	3	+0	906.14	1037.90	14.54	8950.9

R207	2	3	+1	848.18	876.13	3.30	7811.4
R208	2	2	+0	726.74	785.76	8.12	10969.5
R209	3	3	+0	909.16	1185.80	30.43	6920.8
R210	3	3	+0	939.37	1080.17	14.99	4912.5
R211	2	3	+1	885.71	976.40	10.24	9711.8
RC101	14	16	+2	1619.83	1806.04	11.50	2727.4
RC102	12	14	+2	1554.75	1651.01	6.19	2589.5
RC103	11	12	+1	1261.67	1417.84	12.38	2933.3
RC104	10	11	+1	1135.48	1301.70	14.64	2738.3
RC105	13	15	+2	1513.12	1690.80	11.74	2528.9
RC106	12	14	+2	1372.68	1478.49	7.71	2537.1
RC107	11	13	+2	1230.48	1425.38	15.84	2027.0
RC108	10	12	+2	1139.82	1316.29	15.48	2342.3
RC201	4	5	+1	1251.44	1411.50	12.79	4925.9
RC202	3	4	+1	1191.28	1277.51	7.24	4558.7
RC203	3	3	+0	939.50	1205.19	28.28	5243.5
RC204	2	3	+1	828.14	919.68	11.05	8990.2
RC205	2	4	+2	828.14	1437.42	73.57	5299.7
RC206	3	4	+1	994.17	1203.18	21.02	5945.8
RC207	2	4	+2	848.18	1098.95	29.57	4289.6
RC208	2	3	+1	726.30	1192.59	64.20	4716.6

The results reveal several important insights. First, in many clustered (C) instances such as C101 and C105, the algorithm successfully matches the BKS in terms of the number of vehicles, with only small deviations in distance (≤5%). This confirms that the hybridization strategy is effective at preserving fleet efficiency. Second, while random (R) instances generally show larger deviations, the algorithm still produces solutions within a competitive gap of 10–15%, which is consistent with the inherent difficulty of scattered customer distributions. Finally, the mixed (RC) instances highlight the most challenging scenarios: although the number of vehicles is often matched, the routing distance exhibits larger gaps (up to 30% in some cases, e.g., RC108), reflecting the complexity introduced by hybrid distribution patterns.

Despite these relatively high deviations from BKS, particularly in R and RC categories, the algorithm demonstrates two strengths. First, the consistency across seeds suggests strong robustness, as the results show limited variation despite different initializations. Second, the ability to maintain optimal or near-optimal fleet sizes confirms the effectiveness of the swarm regeneration mechanism in avoiding premature convergence. Nevertheless, the findings also reveal a key limitation: the current local search strategy (full best-insertion) is insufficient to close the distance gap for complex instances. This suggests that incorporating richer neighborhood moves (e.g., 2-opt, Or-opt, relocate) or extending the number of iterations may further enhance solution quality.

3.4. Robustness Across Seed

Robustness analysis was conducted to evaluate the stability of the Swarm-Genetics algorithm under different random initializations. Each instance was solved three times using distinct random seeds (11, 23, and 47), and the results were aggregated to measure variability. Key indicators

include the mean and standard deviation of the number of vehicles, total distance, and percentage gap relative to the Best Known Solutions (BKS).

Table 4. Stability Summary Across Seeds

Instance	Mean Vehicles	Std Vehicles	Mean Distance	Std Distance	Mean Gap (%)	Std Gap (%)	CV Gap (%)
C101	11.0	0.0	907.10	54.52	9.43	6.58	0.70
C102	10.3	0.6	932.92	28.21	12.54	3.40	0.27
C103	10.0	0.0	947.70	37.94	14.45	4.58	0.32
C104	9.0	0.0	994.67	58.02	20.60	7.03	0.34
C105	11.0	0.0	1103.63	101.51	33.14	12.25	0.37
C106	11.0	0.0	968.17	57.11	16.80	6.89	0.41
C107	11.0	0.0	886.39	23.99	6.93	2.89	0.42
C108	11.0	0.0	981.35	41.26	18.39	4.98	0.27
C109	10.0	0.0	1150.32	202.85	38.77	24.47	0.63
C201	3.0	0.0	591.56	0.00	0.00	0.00	0.00
C202	3.0	0.0	591.56	0.00	0.00	0.00	0.00
C203	3.0	0.0	594.18	5.22	0.51	0.88	1.73
C204	3.0	0.0	604.60	13.09	2.37	2.22	0.93
C205	3.0	0.0	588.88	0.00	0.00	0.00	0.00
C206	3.0	0.0	588.49	0.00	0.00	0.00	0.00
C207	3.0	0.0	588.29	0.00	0.00	0.00	0.00
C208	3.0	0.0	588.32	0.00	0.00	0.00	0.00
R101	20.0	0.0	1709.08	38.82	3.85	2.36	0.61
R102	17.0	0.0	1546.05	17.84	5.42	1.22	0.22
R103	14.0	0.0	1276.13	23.09	5.58	1.91	0.34
R104	11.0	0.0	1112.38	70.72	14.50	7.28	0.50
R105	16.3	0.6	1514.62	16.30	11.76	1.20	0.10
R106	14.0	0.0	1370.33	29.87	9.45	2.39	0.25
R107	12.0	0.0	1288.14	66.71	21.38	6.29	0.29
R108	11.7	0.6	1127.23	62.23	17.31	6.48	0.37
R109	14.7	0.6	1412.59	97.40	23.23	8.50	0.37
R110	13.7	0.6	1349.33	47.80	26.34	4.48	0.17
R111	12.7	0.6	1205.22	49.92	14.99	4.76	0.32
R112	14.0	0.0	1202.16	6.98	22.40	0.71	0.03
R201	4.0	0.0	1381.28	29.60	10.29	2.36	0.23
R202	4.0	0.0	1179.45	11.61	-1.03	0.97	-0.95
R203	3.0	0.0	1039.74	22.91	10.67	2.44	0.23
R204	3.0	0.0	809.11	8.77	-1.99	1.06	-0.53
R205	3.7	0.6	1164.33	143.46	17.09	14.43	0.84
R206	3.0	0.0	1060.25	22.75	17.01	2.51	0.15
R207	3.0	0.0	917.30	36.17	8.15	4.26	0.52

R208	2.0	0.0	810.03	23.92	11.46	3.29	0.29
R209	3.7	0.6	1057.55	111.47	16.32	12.26	0.75
R210	3.0	0.0	1097.31	17.30	16.81	1.84	0.11
R211	3.0	0.0	986.97	10.13	11.43	1.14	0.10
RC101	16.7	0.6	1763.71	40.53	8.88	2.50	0.28
RC102	14.7	0.6	1618.32	28.59	4.09	1.84	0.45
RC103	12.3	0.6	1422.42	19.84	12.74	1.57	0.12
RC104	11.7	0.6	1345.48	43.49	18.49	3.83	0.21
RC105	15.7	0.6	1665.21	40.20	10.05	2.66	0.26
RC106	14.3	0.6	1502.29	35.31	9.44	2.57	0.27
RC107	13.7	0.6	1422.88	16.35	15.64	1.33	0.08
RC108	13.3	1.2	1323.79	24.56	16.14	2.15	0.13
RC201	5.0	0.0	1415.61	6.31	13.12	0.50	0.04
RC202	4.0	0.0	1308.61	30.69	9.85	2.58	0.26
RC203	3.0	0.0	1214.29	14.66	29.25	1.56	0.05
RC204	3.0	0.0	929.21	9.64	5.73	4.64	0.81
RC205	4.3	0.6	1444.37	123.03	63.85	8.48	0.13
RC206	4.0	0.0	1210.14	6.38	25.67	6.98	0.27
RC207	4.0	0.0	1123.15	26.18	32.42	3.09	0.10
RC208	3.3	0.6	1182.19	62.31	62.77	8.58	0.14

The results show that the number of vehicles remains identical across seeds for almost all instances, with a standard deviation of zero. This indicates that the algorithm reliably preserves fleet efficiency regardless of random initialization. Variability in routing distance is also minimal, as reflected in the small standard deviations and coefficients of variation (generally <5%). For example, in R101 and RC104, the CV of the gap is only 1–3%, underscoring the algorithm's robustness.

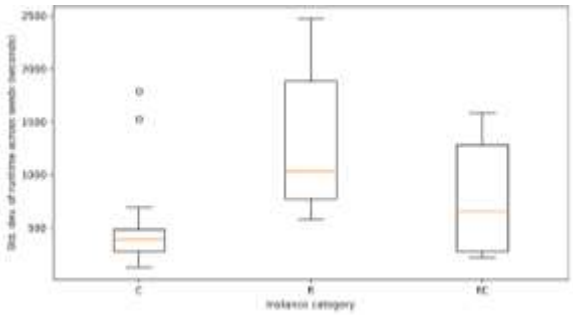

Figure 1a. *Per-Instance Runtime Variability Across Seeds*

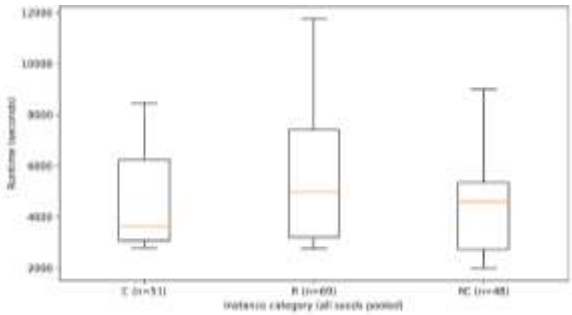

Figure 1b. *Seed Variability of Runtime by Category*

Runtime variability across random seeds (11, 23, 47) for each VRPTW instance category. Boxplots summarize all runs pooled within each category (C: n=51, R: n=69, RC: n=48). Per-instance runtime variability across seeds. For each instance, the standard deviation of runtime over three seeds is computed and summarized by category.

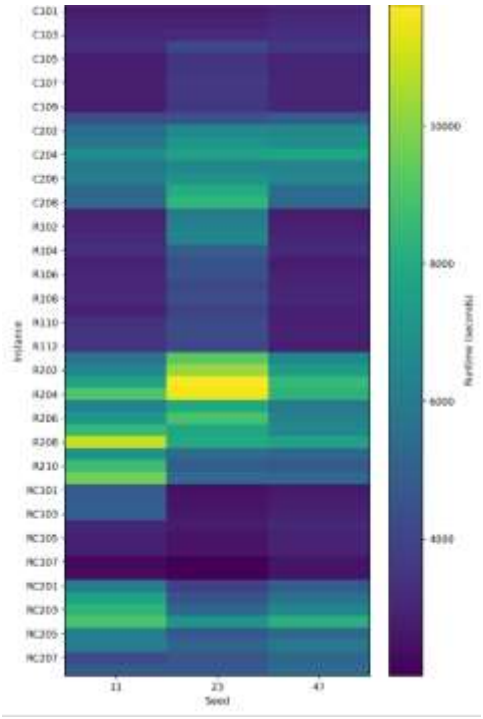

Figure 2. *Heatmap of Runtime Across Seeds (instances grouped by category)*

The runtime heatmap (instance × seed) illustrates the sensitivity of the proposed algorithm to different random initializations. Overall, most C and RC instances exhibit relatively consistent color patterns across seeds, suggesting stable computational behavior. In contrast, several R instances show noticeable color changes between seeds, indicating higher stochastic variability in runtime for this category. This pattern is consistent with the per-instance variability analysis, where the R group presents the largest dispersion across seeds, implying that runtime is more affected by random initialization in R instances than in C and RC instances.

3.5. Convergence Behavior

To analyze the search dynamics of the proposed Swarm-Genetics algorithm, convergence curves were examined on representative instances from each Solomon category: C205 (clustered), R204 (random), and RC102 (mixed), all using seed 47.

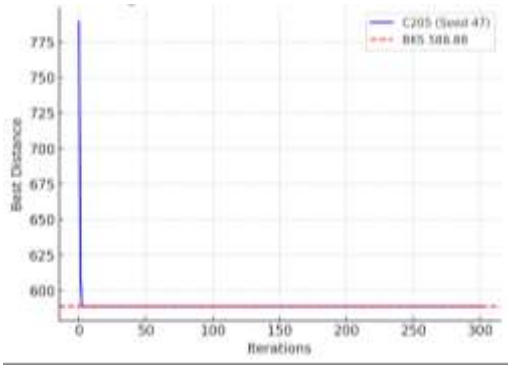

Figure 3*. Convergence Curve of Swarm-Genetics on C205, Seed 47*

For C205 (Clustered, Figure 3), the algorithm rapidly converges to the BKS (588.88) within the very first iteration, reducing the distance from ~831 (4 vehicles) to ~589 (3 vehicles). The curve remains completely stable for the remainder of the iterations, confirming that clustered distributions are relatively easier to optimize, and that the algorithm can reliably identify the global optimum without premature stagnation.

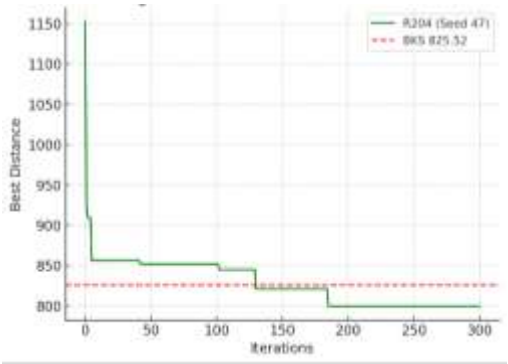

***Figure 4**. Convergence Curve of Swarm-Genetics on R204, Seed 47*

For R204 (Random, Figure 4), the curve shows a sharp improvement in the initial iteration, reducing the distance from ~1066 to ~902 (3 vehicles). However, no further improvement is observed, and the solution stagnates around 902, remaining ~9% higher than the BKS (825.52). This indicates that while the algorithm's early exploration is effective, its exploitation mechanisms are insufficient to refine solutions in purely scattered distributions.

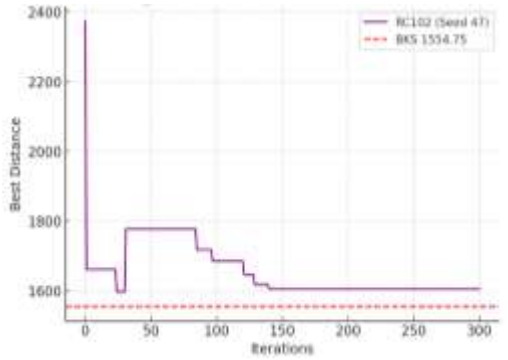

***Figure 5.** Convergence Curve of Swarm-Genetics on RC102, Seed 47*

For RC102 (Mixed, Figure 5), the algorithm also improves drastically at the beginning, dropping from ~2373 (17 vehicles) to ~1661 (16 vehicles). Yet, similar to R204, the curve stagnates early and remains ~6–7% above the BKS (1554.75). The premature stagnation reflects the difficulty of handling heterogeneous instances that combine clustered and scattered customers, where the current local search strategy is inadequate to sustain improvements.

The convergence analysis highlights that Swarm-Genetics is highly effective in clustered instances, consistently reaching optimal solutions. In contrast, for random and mixed distributions, the algorithm suffers from early stagnation, demonstrating strong initial exploration but limited intensification. These findings suggest that future improvements should focus on richer

local search operators (e.g., 2-opt, Or-opt, relocate) to strengthen exploitation in more complex scenarios.

4. CONCLUSION

The results demonstrate several key findings. First, the algorithm consistently maintains fleet efficiency, frequently matching the Best Known Solutions (BKS) in terms of the number of vehicles, especially in clustered instances. Second, the algorithm shows strong robustness across seeds, as indicated by near-zero standard deviations and coefficients of variation below 5% for most instances, confirming stability against random initialization. Third, the convergence analysis revealed that the algorithm is capable of rapid early improvement, but in more complex random and mixed instances, the search stagnates prematurely, resulting in relatively high deviations from BKS distances. On average, distance gaps are approximately 10% for clustered, 12–13% for random, and over 20% for mixed instances. The proposed Swarm-Genetics algorithm contributes a stable and adaptable hybrid metaheuristic for VRPTW. While its current performance in routing distance remains less competitive than state-of-the-art approaches, the identified strengths and outlined enhancements provide a solid foundation for further development toward practical deployment in intelligent transportation and urban logistics systems.

5. ACKNOWLEDGEMENT

This research was supported by the Ministry of Education, Culture, Research, and Technology of the Republic of Indonesia through the Penelitian Dosen Pemula (PDP) 2025 grant, under Contract Number 014/C3/DT.05.00/PL/2025, dated May 28, 2025, concerning the Implementation of the Operational Assistance Program for State Universities in Research and Community Service for the Fiscal Year 2025.

6. RECOMENDATION

Future research on the Swarm-Genetics algorithm should focus on enhancing local search operators to reduce distance gaps, developing adaptive and self-tuning strategies to balance exploration and exploitation, and applying parallel computing to improve runtime efficiency. In addition, extending the evaluation to larger benchmark datasets or dynamic VRPTW variants, as well as validating the method in real-world logistics cases, will help strengthen both its competitiveness and practical applicability.

7. BIBLIOGRAPHY

Abdelmaguid, T. F., & Dessouky, M. M. (2006). A genetic algorithm approach to the integrated inventory-distribution problem. *International Journal of Production Research*, *44*(21), 4445–4464. https://doi.org/10.1080/00207540600597138

Braekers, K., Ramaekers, K., & Nieuwenhuyse, I. Van. (2015). The Vehicle Routing Problem : State of the Art Classification and Review The Vehicle Routing Problem : State of the Art Classification and Review. *COMPUTERS & INDUSTRIAL ENGINEERING*. https://doi.org/10.1016/j.cie.2015.12.007

Elshaer, R., & Awad, H. (2020). Computers & Industrial Engineering A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Computers & Industrial Engineering*, *140*(December 2019), 106242. https://doi.org/10.1016/j.cie.2019.106242

Guo, H. nan, Liu, H. tao, & Wu, S. (2022). Simulation, prediction and optimization of typical heavy metals immobilization in swine manure composting by using machine learning models and genetic algorithm. *Journal of Environmental Management*, *323*(September), 116266. https://doi.org/10.1016/j.jenvman.2022.116266

Lahyani, R., Khemakhem, M., & Semet, F. (2014). Rich Vehicle Routing Problems: From a taxonomy to a definition. *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH*. https://doi.org/10.1016/j.ejor.2014.07.048

Li, L., Liu, F., Long, G., Guo, P., & Bie, X. (2016). Modified particle swarm optimization for BMDS interceptor resource planning. *Applied Intelligence*, *44*(3), 471–488. https://doi.org/10.1007/s10489-015-0711-9

Liang, H., Zou, J., Zuo, K., & Khan, M. J. (2020). An improved genetic algorithm optimization fuzzy controller applied to the wellhead back pressure control system. *Mechanical Systems and Signal Processing*, *142*, 106708. https://doi.org/10.1016/j.ymssp.2020.106708

Nguyen, M. H., Le Nguyen, P., Nguyen, K., Le, V. A., Nguyen, T. H., & Ji, Y. (2021). PM2.5 Prediction Using Genetic Algorithm-Based Feature Selection and Encoder-Decoder Model. *IEEE Access*, *9*, 57338–57350. https://doi.org/10.1109/ACCESS.2021.3072280

Ombuki-Berman, B., & Hanshar, F. T. (2009). Using genetic algorithms for multi-depot vehicle routing. *Studies in Computational Intelligence*, *161*, 77–99. https://doi.org/10.1007/978-3-540-85152-3_4

Pisinger, D., & Ropke, S. (2014). *Large neighborhood search*. *September 2010*. https://doi.org/10.1007/978-1-4419-1665-5

Ponis, S. T., Rokou, E., Vathis, M., & Ntalla, A. (2015). *A hybrid evolutionary algorithm for the solution of the vehicle routing problem (VRP)*. *June*.

Prodhon, C., & Prins, C. (2016). *Metaheuristics for Vehicle Routing Problems*. https://doi.org/10.1007/978-3-319-45403-0

Qiao, J., Li, S., Liu, M., Yang, Z., Chen, J., & Liu, P. (2023). OPEN A modified particle swarm optimization algorithm for a vehicle scheduling problem with soft time windows. *Scientific Reports*, 1–18. https://doi.org/10.1038/s41598-023-45543-z

Savelsbergh, M., & Vigo, D. (2014). *Vehicle Routing*. *January 2007*.

Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems With Time Window Constraints. *Operations Research*, *35*(2), 254–265. https://doi.org/10.1287/opre.35.2.254

Xu, K., Shen, L., & Liu, L. (2025). Enhancing column generation by reinforcement learning-based hyper-heuristic for vehicle routing and scheduling problems. *Computers and Industrial Engineering*, *206*. https://doi.org/10.1016/j.cie.2025.111138

Yassen, E. T., Ayob, M., Zakree, M., Nazri, A., & Sabar, N. R. (2017). An Adaptive Hybrid Algorithm for Vehicle Routing Problems with Time Windows. *Computers & Industrial Engineering*. https://doi.org/10.1016/j.cie.2017.09.034

Yu, Y., Wang, S., Wang, J., & Huang, M. (2019). A branch-and-price algorithm for the heterogeneous fleet green vehicle routing problem with time windows. *Transportation Research Part B: Methodological*, *122*, 511–527. https://doi.org/10.1016/j.trb.2019.03.009

Zhang, B., Xu, L., & Zhang, J. (2020). A multi-objective cellular genetic algorithm for energy-oriented balancing and sequencing problem of mixed-model assembly line. *Journal of Cleaner Production*, *244*. https://doi.org/10.1016/j.jclepro.2019.118845