University of Tehran

Earth Observation and Geomatics Engineering Online ISSN: 2588-4360

Homepage: https://eoge.ut.ac.ir/

An Enhanced Three-Phase Heuristic Approach for the Capacitated Vehicle Routing Problem with Time Windows: Comparing Insertion-Based and Savings-Based Initialization

Arash Fathi [1]✉, **Mohammad Saadi Mesgari** [2]

1. *Corresponding author, M.Sc. GIS Department, Faculty of Geodesy and Geomatics, K. N. Toosi University of Technology. E-mail: afathi@email.kntu.ac.ir*
2. *Associate Professor, GIS Department, Faculty of Geodesy and Geomatics, K. N. Toosi University of Technology. E-mail: mesgari@email.kntu.ac.ir*

Article Info

Article type:
Research Article

Article history:
Received 2025-04-20
Received in revised form 2025-05-29
Accepted 2025-08-12
Published online 2025-10-19

Keywords:
Vehicle routing problem,
Three-phase method,
Initialization algorithm,
Perturbation,
Local search.

ABSTRACT

Vehicle routing optimization in goods distribution helps reduce traffic congestion, lower air pollution, and cut operational costs. This study evaluates two three-phase heuristic methods for solving the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW), focusing on minimizing both service time and the number of vehicles used. Each method consists of initialization, perturbation, and local search phases, differing primarily in their initialization algorithms: one employs an insertion-based heuristic, while the other uses a savings-based algorithm. To enhance efficiency, structured initialization was applied instead of random initialization, accelerating convergence to high-quality solutions. Additionally, solution feasibility was maintained at every step to avoid the need for a repair function. To escape local optima, the Variable Neighborhood Search (VNS) algorithm introduced controlled perturbations, while the Variable Neighborhood Descent (VND) algorithm refined solutions during the local search phase. The methods were tested on the Solomon benchmark dataset, which includes 100 customers distributed in random, clustered, and semi-clustered (random-cluster) patterns. Results showed that the insertion-based method produced better solutions in 66.1% of cases, whereas the savings-based method was computationally faster. Furthermore, the insertion-based approach outperformed a reference Genetic Algorithm (GA) in 53.6% of instances, demonstrating its effectiveness for time-sensitive distribution scenarios.

Cite this article: *Fathi, A., Saadi Mesgari, M. (2025). An Enhanced Three-Phase Heuristic Approach for the Capacitated Vehicle Routing Problem with Time Windows: Comparing Insertion-Based and Savings-Based Initialization. Earth Observation and Geomatics Engineering, Volume 8, Issue 2, Pages 81-95. http//doi.org/10.22059/eoge.2025.393777.1176*

© *The Author(s).*

Publisher: University of Tehran.

DOI: http//doi.org/10.22059/eoge.2025.393777.1176

1. Introduction

The explosive growth of urban centers intensifies last-mile logistics challenges, creating pressure to meet delivery deadlines, reduce costs, minimize transit times, and maximize fleet utilization (Liu, 2022). As transportation constitutes approximately 20% of total production costs, even marginal improvements yield significant economic benefits (Savić et al., 2020). This challenge is formalized as the Vehicle Routing Problem (VRP), where fleets of constrained vehicles service geographically scattered customers from a central depot (Polat et al., 2015). Two key variants address core practical tensions: the Capacitated VRP with strict loading limits and the VRP with Time Windows (VRPTW) incorporating customer-specified delivery intervals (Ghoseiri & Ghannadpour, 2010; Karakatič & Podgorelec, 2015). These formulations balance operational efficiency against physical and temporal constraints, making them essential for supply chain optimization (Dieter et al., 2023).

Solution methodologies vary with problem complexity and scale (Zhang, 2024). Exact methods guarantee optimality for smaller instances but become computationally infeasible as problems scale (Eiben & Smith, 2015; Guo et al., 2024; Yoshizaki, 2009). Consequently, metaheuristics are widely adopted, efficiently navigating the solution space to deliver near-optimal solutions within practical timeframes (Irnich et al., 2014). Examples include multi-objective genetic algorithms for VRPTW (Ghoseiri & Ghannadpour, 2010), disturbance-based VNS for time-constrained routing (Polat et al., 2015), saving-based heuristics for stochastic scenarios (Wang & Zhou, 2016), VNS for maritime logistics (Todosijević et al., 2017), GA-PSO hybrids for VRPTW (Ahkamiraad & Wang, 2018), and random-key GAs for Open VRP (Ruiz et al., 2019).

Effective VRP solutions often employ a structured three-phase methodology: initial solution construction, controlled randomization to avoid local optima, and intensive local search. However, current initialization strategies exhibit limitations. Random initial solutions often require time-consuming feasibility repairs (Ghoseiri & Ghannadpour, 2010; Ahkamiraad & Ruiz et al., 2019; Todosijević et al., 2017). Additionally, many studies focus solely on minimizing travel distance, neglecting operationally significant vehicle waiting times (Polat et al., 2015; Ruiz et al., 2019; Shahbazian et al., 2024; Todosijević et al., 2017).

This study presents two improved three-phase metaheuristic methods for the Capacitated VRPTW. Both share a common framework but employ distinct initialization strategies: insertion-based versus savings-based heuristics. Following initialization, both incorporate a perturbation phase for diversification and an intensive local search. The objective function minimizes total service time (including vehicle waiting times) and fleet size within a single scalar formulation, addressing key gaps in practical relevance and solution comprehensiveness compared to prior work. Performance is evaluated using Solomon's benchmark dataset (Solomon, 1987) and compared against established Genetic Algorithms.

The paper is structured as follows: Section 2 details the problem formulation. Section 3 presents the methodology. Section 4 describes the benchmark dataset. Section 5 outlines the experimental setup and metrics. Section 6 discusses findings and concludes.

2. Problem Description and Mathematical Modeling

The accelerating pace of urbanization and population growth has dramatically expanded service areas, increased customer bases, and necessitated larger vehicle fleets for distribution. These developments have rendered traditional routing approaches increasingly inadequate, creating a pressing need for advanced metaheuristic optimization methods (Karakatič & Podgorelec, 2015). The Vehicle Routing Problem (VRP) addresses this challenge by optimizing delivery routes for fleets servicing customers from a central depot while minimizing operational costs (Irnich et al., 2014). The problem incorporates several critical constraints: each vehicle has a finite capacity that cannot be exceeded, customers have specific demand quantities and service durations, and deliveries must occur within predefined time windows. Notably, vehicles arriving early must wait until a customer's time window opens - a key operational consideration that this study explicitly incorporates into its objective function. All routes must begin and end at the depot, creating closed-loop itineraries.

We adopt the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW) as our modeling framework due to its strong alignment with real-world distribution challenges. It captures both the physical constraint of limited vehicle capacity and the temporal constraint of customer time windows, both of which are essential in last-mile urban logistics. This formulation is also widely used in the literature and supported by standard benchmarks, making it suitable for rigorous comparative evaluation.

Effective mathematical modeling is paramount for successful optimization. In our formulation, we represent each vehicle's route as an ordered sequence of customer IDs, with complete solutions comprising sets of such routes. Fig. 1 illustrates this representation through a three-vehicle solution example, where Vehicle 1's route (depot → customer 5 → customer 3 → customer 4 → depot) demonstrates the sequential structure. This modeling approach captures both the assignment of customers to vehicles and their service sequence, providing a computationally tractable framework for optimization while maintaining fidelity to real-world operational constraints.

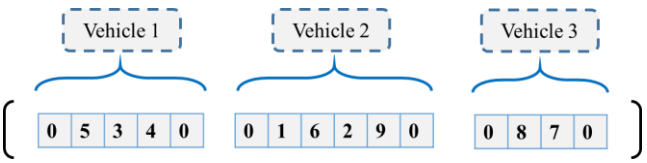


Figure 1. Problem model

The CVRPTW addressed in this study includes several key characteristics: each customer has a specific demand and a time window $[a_i, b_i]$ during which service must begin; each vehicle has a limited capacity q and must serve customers in a continuous tour that starts and ends at the depot; and vehicles may arrive early but must wait until the customer's time window opens. These constraints reflect real-world logistics operations where time-sensitive deliveries and fleet limitations are paramount.

The Capacitated Vehicle Routing Problem with Time Windows (CVRPTW) can be formally represented as a directed graph G = (N, A), where N denotes the set of nodes corresponding to customer locations and the depot, while A represents the arcs connecting these nodes with associated travel distances. Building upon the foundational work of Desaulniers et al. (Desaulniers et al., 2014) and Bräysy et al. (Bräysy & Gendreau, 2002), we present a comprehensive mathematical formulation comprising constraints (1) through (9) that captures the problem's key operational constraints and objectives.

$min\ f_{VRPTW} = \sum_{k\in V}\sum_{i\in N}\sum_{j\in N} c_{ij}x_{ijk} + \sum_{k\in V} y_k$	*(1)*
$\forall i \in C: \sum_{k\in V}\sum_{j\in N} x_{ijk} = 1$	*(2)*
$\forall k \in V: \sum_{i\in C} d_i \sum_{j\in N} x_{ijk} \le q$	*(3)*
$\forall k \in V: \sum_{j\in N} x_{0jk} < y_k$	*(4)*
$\forall p \in C. \forall k \in V: \sum_{i\in N} x_{ipk} - \sum_{j\in N} x_{pjk} = 0$	*(5)*
$\forall k \in V: \sum_{i\in N} x_{i.n+1.k} < y_k$	*(6)*
$\forall i.j \in N. \forall k \in V: x_{ijk}(w_{ik} + s_i + t_{ij} - w_{jk}) \le 0$	*(7)*
$\forall i \in N. \forall k \in V: a_i \le w_{ik} \le b_i$	*(8)*
$x_{ijk} \in \{0.1\}. \quad y_k \in \{0.1\}. \quad w_{ik} \ge 0$	*(9)*

Where V is the set of vehicles, and $N = C \cup \{0, n+1\}$ is the set of all nodes, including the set of customers C and the depot represented by node 0 (start) and node $n+1$ (end). The parameter c_{ij} denotes the travel cost from node i to node j, and d_i is the demand of customer i. Each vehicle has a capacity limit denoted by q. The time window during which service must begin at node i is defined by the interval $[a_i, b_i]$, and s_i is the service time at node i. The travel time from node i to node j is represented by t_{ij}, and w_{ik} is the arrival time of vehicle k at node i.

The mathematical formulation introduces two key decision variables to model the vehicle routing and scheduling problem with time windows. The binary variable $x_{ijk} \in \{0,1\}$ indicates whether vehicle k travels directly from node i to node j, while the binary variable $y_k \in \{0,1\}$ denotes whether vehicle k is used in the solution. The continuous variable $w_{ik} \ge 0$ represents the arrival time of vehicle k at node i, subject to the time window constraints.

The mathematical model consists of the following key relations. Relation (1) defines the objective function to be minimized: it integrates both the total routing cost and the number of vehicles used in a single-objective framework. The first term, $\sum_{k\in V}\sum_{i\in N}\sum_{j\in N} c_{ij}x_{ijk}$, represents the total distance-based routing cost, while the second term, $\sum_{k\in V} y_k$, penalizes the use of vehicles. The objective function integrates two critical components: total travel cost and number of vehicles used. This reflects a trade-off between minimizing operational distance and reducing fleet size, both of which are cost drivers in logistics operations. Additionally, the model implicitly penalizes vehicle waiting times through the arrival time variable, which improves realism in urban settings where early arrivals are frequent. All travel distances are computed using the Euclidean metric, consistent with Solomon benchmark specifications.

Relation (2) ensures that each customer is visited exactly once by a single vehicle, avoiding split deliveries. Relation (3) enforces the vehicle capacity constraint: for each vehicle, the total demand of the served customers must not exceed its capacity q. Relation (4) guarantees that each vehicle, if used, departs exactly once from the depot (node 0), while Relation (5) enforces flow conservation for each customer: the number of vehicles entering a customer node must equal the number of vehicles leaving it, ensuring route continuity. Relation (6) requires that each vehicle returns to the depot (node $n+1$) if it is used. Relation (7) establishes time consistency between service at two consecutively visited nodes: if vehicle k travels from node i to node j, then the arrival time at j must be no earlier than the departure time from i, accounting for service time s_i and travel time t_{ij}. Relation (8) ensures that the arrival time at each node lies within its designated time window $[a_i, b_i]$, maintaining schedule feasibility. Relation (9) defines the binary nature of the routing and vehicle usage variables, constraining x_{ijk} and y_k to take values from {0,1}. Together, these relations provide a comprehensive mathematical framework for the VRPTW that not only seeks to minimize the total cost of routing but also discourages excessive use of vehicles, thereby producing more efficient and practical solutions.

This formulation employs binary decision variables to represent route assignments and vehicle usage, and continuous variables to model arrival times. This mixed-integer structure captures the key logistical features of the CVRPTW and enables direct modeling of waiting times, an aspect often neglected in previous studies. The formulation ensures feasible, time-consistent routes while minimizing operational cost and fleet size.

3. Methodology

We design a three-phase metaheuristic approach to solve the CVRPTW effectively. This structure enables a balance between solution quality, computational efficiency,

and feasibility. The first phase constructs an initial solution using either an insertion-based or savings-based heuristic to ensure feasibility from the outset. The second phase introduces controlled random perturbations to escape local optima, while the third phase applies local search to intensify and refine the solution. This combination leverages the strengths of both constructive heuristics and neighborhood-based metaheuristics.

This study proposes a three-phase metaheuristic approach for solving the Vehicle Routing Problem (VRP), comprising (1) initial solution construction, (2) perturbation, and (3) local search. The first phase generates a feasible initial solution using either an insertion heuristic (Hassin & Keinan, 2008) (Lu & Dessouky, 2006) or a savings heuristic (Polat et al., 2015) (Wang & Zhou, 2016) , ensuring all constraints are satisfied from the outset. This strategic initialization promotes faster convergence by eliminating the need for subsequent repair operations common in random initialization approaches. To escape local optima, the second phase incorporates a perturbation mechanism through Variable Neighborhood Search (VNS) (Hansen & Mladenović, 2001) (Hansen et al., 2017), which intentionally introduces controlled randomization to diversify the solution space. The third phase employs Variable Neighborhood Descent (VND) (Polat et al., 2015) (Todosijević et al., 2017) for intensive local search, systematically exploring neighborhood structures to identify quality improvements. This phased methodology balances exploration and exploitation, where VNS broadens the search while VND intensively refines promising solutions, collectively driving the algorithm toward superior solutions.

Fig. 2 presents the comprehensive flowchart of the proposed three-phase methodology. The algorithmic process initiates with the generation of an initial feasible solution, followed by the systematic preparation of two ordered lists: neighborhood structures for perturbation (Section 3.2) and search operators for local improvement (Section 3.3). The perturbation phase first applies the initial neighborhood structure from the predefined list, intentionally introducing controlled diversification that may not yield immediate improvements. Subsequently, the local search phase sequentially applies its operator list until either an improvement is found or all operators are exhausted.

The search mechanism implements an improvement-driven restart policy: whenever an operator successfully enhances the solution, the process reverts to the first operator, while unsuccessful attempts progress linearly through the operator list. Similarly, improved solutions trigger a reset to the initial neighborhood structure, whereas unimproved solutions advance to subsequent neighborhoods. Crucially, all modifications undergo continuous feasibility verification, with strict enforcement of vehicle capacity and time window constraints after each alteration. This embedded constraint validation eliminates the computational overhead of separate repair mechanisms.

To ensure that all generated solutions remain feasible with respect to vehicle capacity and customer time windows, our methodology incorporates embedded feasibility checks at critical stages of the algorithm. Feasibility is explicitly validated during the initial solution construction phase, where both the insertion-based and savings-based heuristics are designed to add customers only when doing so does not violate capacity or time window constraints. This approach eliminates the need for post-hoc repair mechanisms, which are often computationally expensive and prone to introducing unintended bias in the solution space (Li et al., 2022). During this phase, every candidate insertion or merge is evaluated for constraint adherence before it is applied, guaranteeing that the initial solution is fully feasible by construction. In subsequent phases—perturbation and local search—feasibility is maintained by design. Each neighborhood operator in both the Variable Neighborhood Search (VNS) and Variable Neighborhood Descent (VND) algorithms performs feasibility validation before committing any solution-altering operation (Li et al., 2022). This ensures that no infeasible move is accepted into the current solution set. As a result, there is no need for separate feasibility repair procedures after each move, which significantly improves computational efficiency and preserves the structural integrity of the solution. This embedded feasibility enforcement is especially advantageous in tightly constrained problems like CVRPTW, where infeasible solutions can lead to prolonged search times or convergence failure.

To enhance the effectiveness of the proposed solution method, the selection and ordering of neighborhood structures in both the perturbation and local search phases were guided by principles of algorithmic diversity, computational efficiency, and empirical performance. In the perturbation phase (VNS), the neighborhood operators were chosen to introduce increasing levels of structural modification to the current solution. Starting with the simplest operations (e.g., one-point crossover and relocation) and progressing toward more disruptive transformations (e.g., inverted crossover), this sequencing supports a controlled diversification mechanism. The rationale is that smaller changes help escape shallow local optima with minimal disruption, while progressively larger changes enable the algorithm to explore distant regions of the solution space when necessary. This strategic ordering ensures a balance between intensification and diversification, which is central to the design philosophy of VNS.

Similarly, the local search phase (VND) employs neighborhood structures arranged in increasing complexity and computational cost. The ordering from L1 to L5 prioritizes operators that are faster to evaluate and have a high likelihood of improving solution quality early in the search. For example, the inversion and 1-opt moves (L1 and L2) are applied first because they offer rapid refinement with low overhead, often correcting small inefficiencies in route structures. More complex moves such as 3-opt and

crossover (L4 and L5) are applied later, allowing the algorithm to exploit deeper structural improvements only when simpler adjustments no longer yield benefits. This order also supports the improvement-driven restart mechanism used in VND, ensuring that more computationally expensive operators are only applied when necessary. Overall, the design and ordering of these operations reflect a deliberate trade-off between local optimization efficiency and global search capability.

The following sections detail the methodological components in sequence: initial solution construction techniques, perturbation mechanisms with associated neighborhood structures, and finally the local search optimization framework. This structured approach ensures both rigorous constraint satisfaction and efficient solution space exploration.

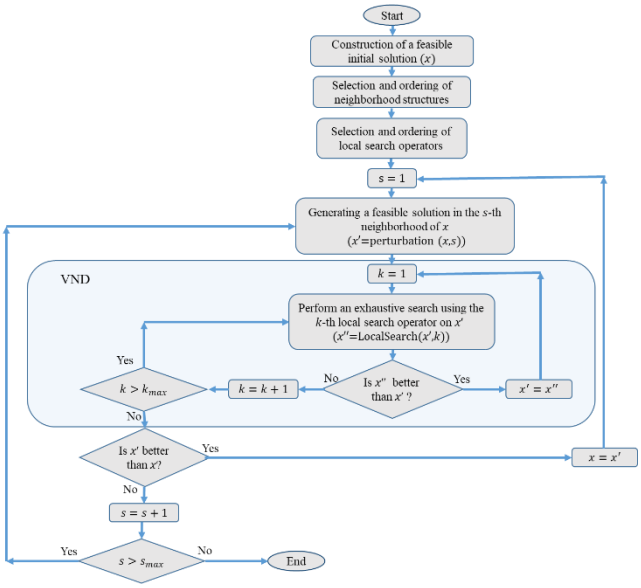


Fig. 2. Flowchart of the Three-Phase Algorithm: Initial construction of a feasible solution, perturbation using neighborhood structures, and local search via a VND scheme

3.1. Initial Solution Construction

As established earlier, a valid solution comprises an ordered set of customer sequences representing vehicle routes. However, feasibility requires strict adherence to all problem constraints, which presents significant computational challenges in constrained routing problems such as the CVRPTW, where both vehicle capacity and customer time windows must be simultaneously satisfied (Da Silva & Urrutia, 2010). To address this complexity, we employ two distinct heuristic approaches for generating initial feasible solutions: an insertion-based method utilizing insertion heuristics and a savings-based method applying savings heuristics. These approaches are subsequently evaluated through comparative performance analysis, providing insights into their respective strengths under varying problem conditions.

Insertion Heuristic

The insertion heuristic initiates by constructing an initial route where a vehicle departs from the depot, services the most distant unvisited customer, and returns directly to the depot. Subsequent customers are systematically inserted into existing routes based on three critical criteria: temporal feasibility (time windows), spatial proximity (distances), and residual vehicle capacity. The algorithm progresses iteratively, evaluating potential insertions through a tripartite cost assessment that considers: (a) candidate customers eligible for insertion, (b) feasible insertion positions within existing routes, and (c) the corresponding route cost implications (Hassin & Keinan, 2008) (Lu & Dessouky, 2006).

At each iteration, the algorithm selects the insertion yielding the minimal cost increase, with the process continuing until either vehicle capacity constraints prevent further insertions or all customers are successfully routed. When no additional customers can be accommodated in existing routes, new routes are initialized following the same farthest-first principle. Fig. 3 illustrates the complete algorithmic workflow, demonstrating this systematic balance between spatial efficiency and constraint satisfaction.

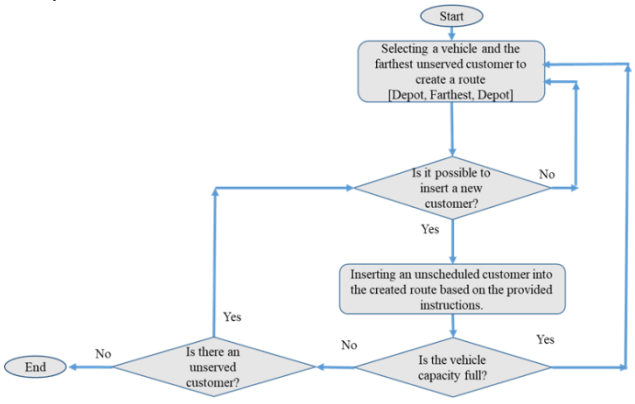


Fig. 3. Flowchart of the Insertion Heuristic Algorithm

Savings Heuristic

The savings heuristic, first introduced by Clarke and Wright (Wang & Zhou, 2016), is a classical approach for constructing cost-efficient vehicle routes through the iterative merging of customer-specific tours. Initially, each customer is assigned a dedicated route where a vehicle departs from the depot, visits a single customer, and returns—yielding routes of the form Depot → Customer → Depot.

The core idea of the algorithm lies in computing a savings value for each pair of customers i and j, which quantifies the potential cost reduction achieved by combining their individual routes into a single tour. This savings value is computed as:

$$Saving\ Cost = Dist_{i0} + Dist_{0j} - Dist_{ij} \quad (12)$$

where:

$Dist_{i0}$ is the distance from customer i to the depot,
$Dist_{0j}$ is the distance from the depot to customer j,
$Dist_{ij}$ is the distance between customers i and j.

All possible savings values are calculated and stored in a Savings List, which is then sorted in descending order so that the most beneficial route mergers (those with the highest savings) are considered first.
The merging process, illustrated in Fig. 4, proceeds as follows:

1. *Select the top entry from the Savings List, corresponding to a potential merge between customers i and j.*
2. *Check feasibility of the merge by verifying that:*
   - *The customers are at the ends of their respective routes.*
   - *Combining the routes does not violate vehicle capacity or time window constraints.*
3. *If feasible, merge the two routes into a single tour.*
4. *Update the solution and the Savings List:*
   - *Remove or update entries involving customers i or j, as their routes have changed.*
   - *Recalculate savings values for any new feasible route combinations involving the merged route.*
5. *If not feasible, discard the current savings entry and proceed to the next highest one.*

This process continues until the Savings List is exhausted—that is, when no further feasible merges can be performed. The final solution consists of a set of routes that aim to minimize the overall travel cost while adhering to all operational constraints, including vehicle capacity and customer time windows.

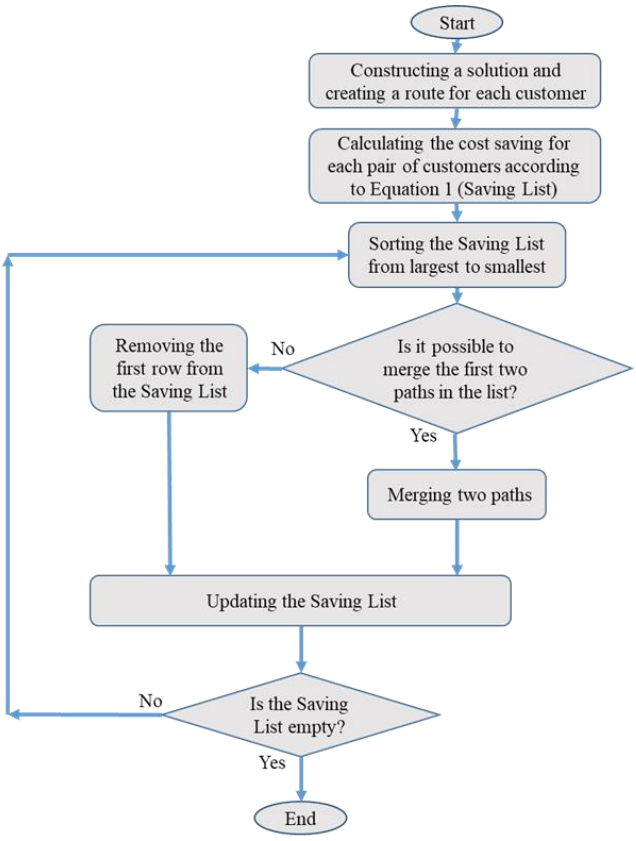


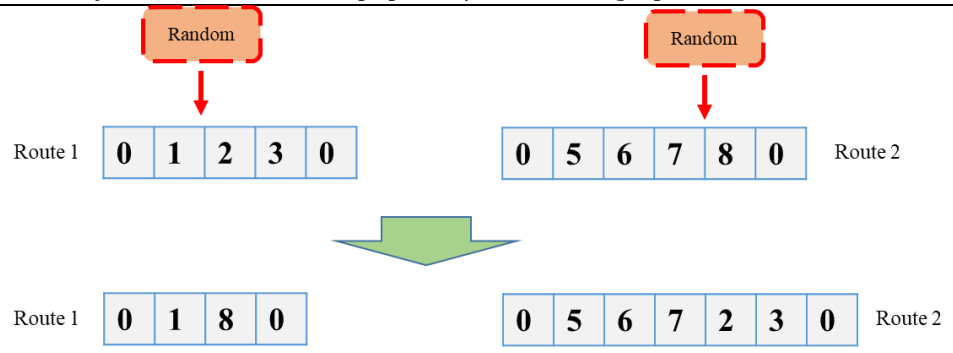
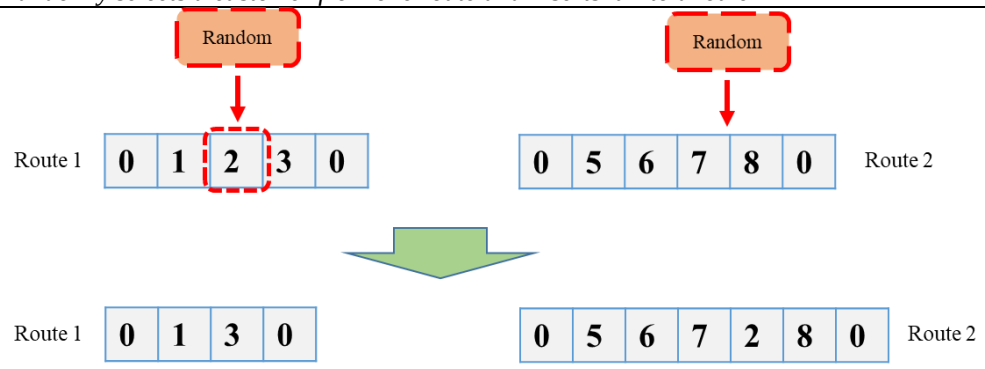
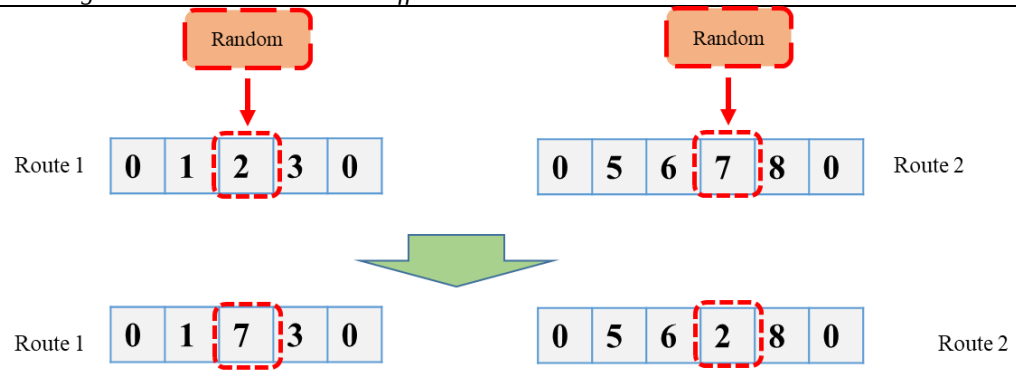
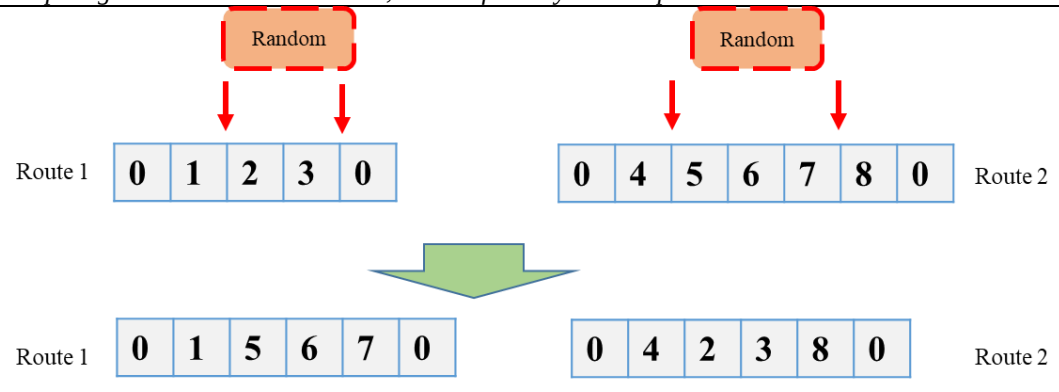
Fig. 4. Flowchart of the Heuristic Savings Algorithm

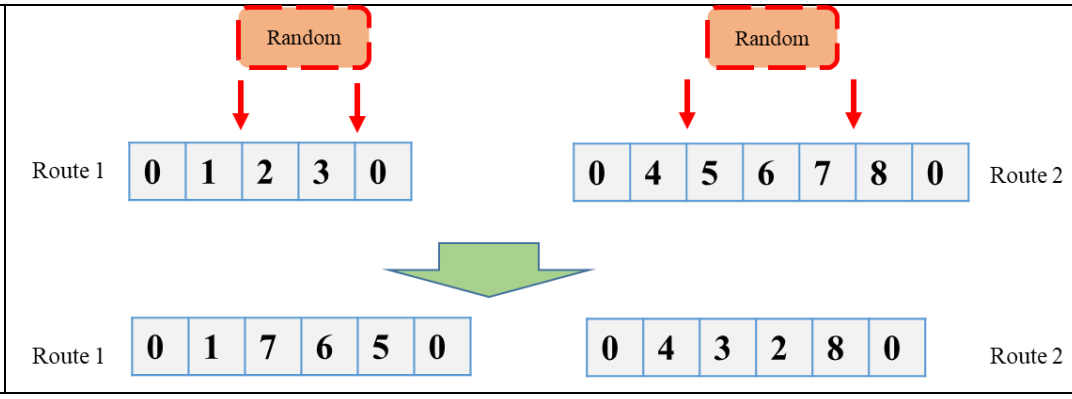
3.2. Perturbation Algorithm

Following initial solution construction, we employ the Variable Neighborhood Search (VNS) algorithm (Mladenović & Hansen, 1997) to enhance solution quality through systematic diversification. The VNS approach, originally developed by Mladenović and Hansen, operates by cyclically exploring predefined neighborhood structures that each impose distinct modifications to the current solution (Mladenović & Hansen, 1997). This strategic alternation between different neighborhood types facilitates comprehensive exploration of the solution space while mitigating premature convergence to local optima.

As detailed in Table 1, our implementation utilizes five specific neighborhood structures, applied sequentially in the following order: N1 → N2 → N3 → N4 → N5. This ordered progression ensures methodical exploration of increasingly complex solution modifications while maintaining algorithmic efficiency. Each neighborhood transition occurs only after exhaustive search within the current neighborhood structure, following the fundamental VNS principle of balanced intensification and diversification.

Table. 1. Neighborhood Structures

(N1) *One-point crossover*	*Randomly selects two routes, swaps parts of them at a single point* 
(N2) *Relocation*	*Randomly selects a customer from one route and inserts it into another* 
(N3) *Exchange*	*Exchanges customers between two different routes* 
(N4) *2 Point Cross Over*	*Swaps segments between two routes, each defined by two cut points* 
	Similar to two-point crossover, but the segments are reversed before being swapped

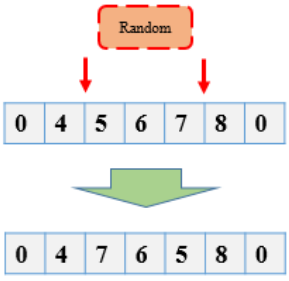
(N5) Inverted crossover	

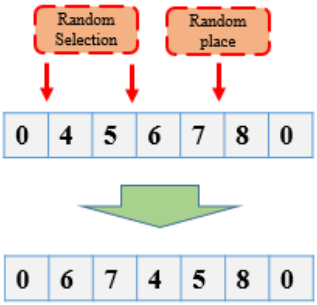
The selection of five neighborhood structures was driven by the need to balance diversification and intensification in the search process. Each operator introduces a distinct form of perturbation, allowing for broader exploration of the solution space without introducing excessive computational burden. Empirical evaluation indicated that this number provides a good trade-off between solution quality and runtime. Therefore, the number of local search operators has been deliberately limited to five to maintain algorithmic efficiency while avoiding diminishing returns from additional neighborhoods.

3.3. Local Search Algorithm

Following the perturbation phase, the Variable Neighborhood Descent (VND) algorithm performs systematic solution refinement through exhaustive neighborhood exploration. Unlike its VNS counterpart which introduces stochastic perturbations, VND employs a deterministic approach, sequentially applying local search operators in a predefined order (L1 → L2 → L3 → L4 → L5) to ensure comprehensive local optimization (Hansen et al., 2017). The algorithm implements an improvement-driven restart mechanism: whenever any operator yields an enhanced solution, the search reverts to the initial operator (L1), enabling deeper exploration of promising solution spaces. Crucially, all neighborhood moves maintain solution feasibility by design, eliminating the computational burden associated with repair mechanisms while preserving constraint satisfaction. This embedded feasibility guarantee contributes significantly to the method's computational efficiency throughout the intensification process.

Table 2. Local Search Operators

(L1) ***Inversion***	 *This operator systematically examines all feasible inversion operations across every route in the current solution. For each candidate route, the algorithm evaluates every possible pair of customer positions, calculating the potential improvement in the objective function that would result from reversing the customer sequence between each position pair. The optimal inversion, defined as the pair yielding the maximum improvement, is then implemented. The inversion mechanism operates by selecting a target route and two distinct customer positions within it, then reversing the subsequence bounded by these positions. This transformation is applied exhaustively across all routes in the solution, ensuring a comprehensive exploration of the inversion neighborhood space. The operator ultimately selects and retains only the most beneficial inversion found during this process, thereby guaranteeing local optimality to this particular neighborhood structure.*

(L2 == 1-opt) *(L3 == 2-opt)* *(L4 == 3-opt)* *k-opt*	 *This operator performs an exhaustive evaluation of all possible k-opt moves across each route in the current solution. For every route, it identifies the optimal k-opt transformation, the relocation of a sequence of k consecutive customers to a new position within the same route, that maximizes improvement in the objective function while maintaining their original order. The selected move is then implemented to refine the solution. The operator systematically explores the neighborhood space by assessing potential relocations (either forward or backward along each route) for every valid k-length customer sequence. This comprehensive search ensures identification of the most beneficial intra-route restructuring, thereby guaranteeing local optimality to the k-opt neighborhood structure.*
(L5) One-point crossover	*This operator systematically explores all feasible one-point crossover operations between route pairs in the current solution. For each potential route combination, it identifies the optimal crossover point that maximizes improvement in the objective function. The operation selects two distinct routes and a customer position within each, then generates new route configurations by exchanging the subsequences preceding the selected positions while preserving the remaining segments. The operator exhaustively evaluates all possible route pairings and crossover positions, implementing only the most beneficial configuration found. This comprehensive search strategy ensures thorough exploration of the crossover neighborhood space, while the preservation of subsequence order maintains solution feasibility. Through this process, the operator guarantees identification of locally optimal solutions to the one-point crossover transformation.*

To ensure optimal performance of the proposed three-phase metaheuristics, a systematic parameter tuning procedure was conducted. The goal was to determine effective configurations for the neighborhood structures used during the perturbation phase (VNS) and the local search operators in the refinement phase (VND), as well as to optimize other key parameters such as the number of iterations, perturbation strength, and runtime limits. Tuning was performed using a representative subset of Solomon's benchmark instances (R101, C101, RC101) to capture varying spatial distributions. A full-factorial grid search was applied, and each configuration was evaluated under a fixed time limit of 120 seconds using two performance metrics: total travel cost and the number of vehicles used.

The final configuration—500 iterations, perturbation strength of 3, and a sequential VNS structure (N1 → N3 → N2 → N4 → N5)—proved most effective. The VND operators were applied in a fixed order (L2 → L3 → L1 → L4 → L5), progressing from simple to more complex heuristics. This structure balanced diversification and intensification, enabling robust solution quality while maintaining computational efficiency. Increasing iterations beyond 500 yielded minimal improvement, and moderate perturbation strength avoided destabilizing the search. The selected settings enabled consistent convergence across instance types, demonstrating the importance of structured tuning in heuristic design.

4. Data Used

This study employs Solomon's benchmark datasets (Solomon, 1987), which are widely used for evaluating solutions to the Vehicle Routing Problem with Time Windows (VRPTW). These datasets offer a range of problem instances with varying sizes and configurations, each including precise geographical coordinates for the depot and customer locations. Operational parameters such as fleet size, vehicle capacity, individual customer demands, and time window constraints—including earliest service start time, latest acceptable arrival time, and service duration—are clearly specified. Fig. 5 illustrates a representative data file structure from Solomon's collection, demonstrating the standardized format used for these benchmark problems.

```
<Instance name>
<empty line>
VEHICLE
NUMBER     CAPACITY
  K           Q
<empty line>
CUSTOMER
CUST NO.  XCOORD.    YCOORD.    DEMAND   READY TIME  DUE DATE   SERVICE TIME
 <empty line>
    0        x0         y1         q0         e0          l0           s0
    1        x1         y2         q1         e1          l1           s1
  ...       ...        ...        ...        ...         ...          ...
  100      x100       y100       q100       e100        l100         s100
```

Fig. 5. Structure of Solomon's data

The datasets are systematically organized into three main classes based on customer spatial distribution: Random (R), Clustered (C), and Random-Clustered (RC) configurations (Solomon, 1987). Each class includes two variants distinguished by the strictness of time windows. The "1" series (e.g., R1, C1, RC1) features tight time windows, limiting the number of customers per route due to narrower service periods. Conversely, the "2" series (e.g., R2, C2, RC2) incorporates relaxed time windows, which allow for more flexible scheduling and the possibility of longer routes serving more customers (Ghoseiri & Ghannadpour, 2010). This structure enables a comprehensive evaluation of algorithmic performance under varying temporal and spatial constraints.

The distinction among the R, C, and RC configurations lies in the spatial arrangement of customer locations and reflects different real-world distribution scenarios. In the Random (R) instances, customer locations are spread uniformly across the service area without any discernible spatial pattern. This setting simulates environments such as rural or low-density suburban regions, where demand points are dispersed, and routing solutions must account for greater travel distances and less predictable routing paths.

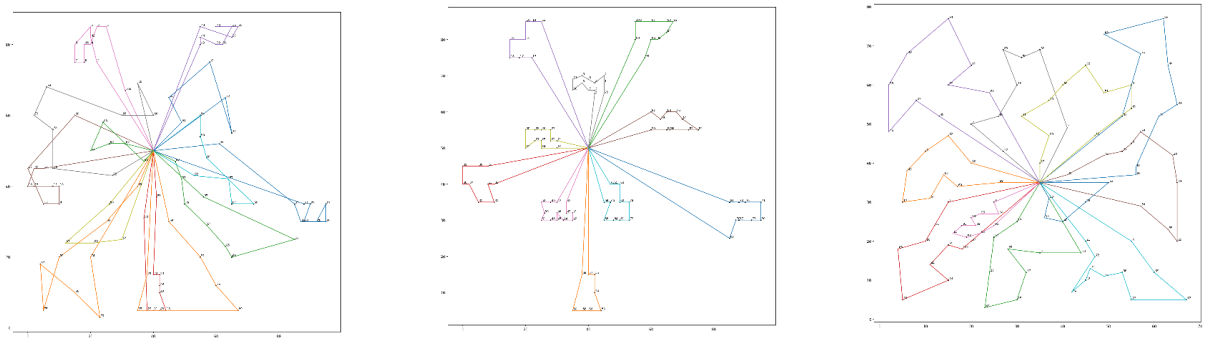
In contrast, the Clustered (C) instances consist of customers grouped into well-defined clusters or zones, often concentrated around specific points. These clusters may represent business districts, urban delivery zones, or other high-density demand centers where delivery locations are geographically close to one another. This spatial configuration allows for shorter intra-cluster travel but poses challenges in inter-cluster transitions and vehicle load balancing.

The Random-Clustered (RC) configuration is a hybrid that combines features of both previous types. In these instances, some customers are arranged in clusters while others are distributed randomly throughout the service area. This mixed layout closely resembles real-world distribution systems where urban deliveries (clusters) coexist with suburban or rural outliers (random points). As such, RC instances are generally considered more complex due to the need to simultaneously manage intra-cluster efficiency and inter-cluster routing diversity.

These variations in spatial distribution are critical to testing the robustness of VRP algorithms. Algorithms may perform well in clustered environments due to route density but struggle in random or mixed configurations that require more flexible and adaptive strategies. Therefore, the Solomon datasets remain a benchmark not only for performance comparison but also for analyzing algorithm sensitivity to different spatial structures.

5. Implementation and Numerical Evaluation

This study comparatively evaluates two three-phase solution approaches differentiated by their initial solution construction algorithms: an insertion-based method utilizing insertion heuristics and a savings-based method employing savings heuristics. Both implementations were developed in Python 3.6 and executed on standardized hardware (Intel Core i3 1.70GHz processor, 4GB RAM) to ensure consistent performance measurement. The experimental evaluation employed Solomon's benchmark datasets comprising 100 customer instances, with solution quality assessed across all problem categories. Fig. 6 demonstrates representative output visualizations for datasets R112, C105, and RC106, where distinct color-coding illustrates the optimized vehicle routes generated by each method. This comparative visualization facilitates direct observation of the routing patterns emerging from each algorithmic approach.



RC106 *R112* *C105*

Fig. 6. Example Output

Table 3: Comparison Between Insertion-Based and Savings-Based Methods

data	*Savings-Based Method*			*Insertion-Based Method*		
	Processing Time	*Total Service Time*	*Number of Vehicles*	*Processing Time*	*Total Service Time*	*Number of Vehicles*
R101	*169.98*	*1729.80*	*24*	*163.66*	*1700.70*	*20*
R102	*275.47*	*1626.03*	*22*	*390.27*	*1514.84*	*19*
R103	*119.20*	*1328.10*	*17*	*293.35*	*1276.26*	*15*
R104	*119.16*	*1051.69*	*13*	*346.58*	*1076.09*	*12*
R105	*133.36*	*1512.10*	*20*	*131.07*	*1428.03*	*15*
R106	*100.44*	*1374.96*	*18*	*221.55*	*1301.36*	*13*
R107	*134.62*	*1174.73*	*14*	*296.74*	*1144.21*	*12*
R108	*189.53*	*1012.83*	*11*	*183.36*	*1013.97*	*11*
R109	*113.12*	*1228.49*	*15*	*179.68*	*1294.33*	*14*
R110	*75.85*	*1148.17*	*13*	*216.40*	*1130.49*	*12*
R111	*110.10*	*1131.89*	*13*	*151.19*	*1154.07*	*12*
R112	*94.96*	*1016.64*	*10*	*265.08*	*1015.55*	*11*
R201	*166.50*	*1302.18*	*14*	*480.78*	*1316.75*	*4*
R202	*263.81*	*1121.37*	*9*	*714.75*	*1142.73*	*4*
R203	*312.26*	*938.69*	*8*	*1107.00*	*941.22*	*4*
R204	*466.89*	*775.15*	*6*	*1474.00*	*765.76*	*4*
R205	*184.15*	*1067.49*	*10*	*783.00*	*1015.27*	*4*
R206	*289.65*	*966.07*	*8*	*622.03*	*969.01*	*3*
R207	*492.61*	*864.60*	*6*	*1036.41*	*874.76*	*3*
R208	*506.03*	*729.51*	*4*	*1517.22*	*720.16*	*3*
R209	*469.22*	*916.59*	*7*	*731.27*	*976.88*	*3*
R210	*258.20*	*957.81*	*7*	*7870.80*	*948.65*	*4*
R211	*306.86*	*798.32*	*6*	*766.86*	*836.11*	*3*
C101	*50.36*	*828.94*	*10*	*104.10*	*828.94*	*10*
C102	*94.52*	*866.00*	*11*	*249.47*	*834.64*	*10*
C103	*68.52*	*829.86*	*10*	*284.90*	*884.48*	*11*
C104	*65.30*	*830.02*	*10*	*457.09*	*866.68*	*10*
C105	*52.78*	*866.00*	*11*	*101.99*	*828.94*	*10*
C106	*59.93*	*828.94*	*10*	*183.08*	*828.94*	*10*
C107	*50.07*	*866.00*	*11*	*120.58*	*829.70*	*10*
C108	*66.15*	*832.27*	*10*	*141.28*	*828.94*	*10*
C109	*65.06*	*849.25*	*10*	*129.69*	*836.33*	*10*
C201	*216.17*	*591.56*	*3*	*228.81*	*591.56*	*3*
C202	*463.21*	*591.56*	*3*	*530.29*	*591.56*	*3*
C203	*421.48*	*591.17*	*3*	*848.47*	*591.17*	*3*
C204	*447.84*	*590.60*	*3*	*836.91*	*602.95*	*3*
C205	*244.64*	*588.88*	*3*	*340.13*	*588.88*	*3*
C206	*289.92*	*588.49*	*3*	*394.85*	*588.49*	*3*
C207	*251.42*	*588.29*	*3*	*438.26*	*588.29*	*3*
C208	*423.55*	*588.32*	*3*	*347.66*	*588.32*	*3*
RC101	*153.63*	*1785.74*	*19*	*117.20*	*1722.51*	*17*
RC102	*157.87*	*1577.49*	*16*	*97.79*	*1548.48*	*14*
RC103	*107.63*	*1412.48*	*14*	*181.02*	*1381.06*	*13*
RC104	*89.88*	*1212.10*	*12*	*133.26*	*1223.67*	*11*
RC105	*140.76*	*1646.27*	*17*	*140.89*	*1600.80*	*17*
RC106	*86.86*	*1537.21*	*16*	*86.15*	*1421.89*	*13*
RC107	*119.77*	*1324.26*	*13*	*215.25*	*1337.86*	*13*
RC108	*79.04*	*1165.59*	*11*	*127.16*	*1199.60*	*12*
RC201	*192.00*	*1469.97*	*13*	*460.56*	*1436.35*	*5*
RC202	*366.47*	*1213.57*	*10*	*559.01*	*1174.06*	*5*
RC203	*350.84*	*1005.38*	*7*	*866.84*	*973.73*	*4*
RC204	*391.03*	*848.78*	*6*	*1263.72*	*809.36*	*4*
RC205	*330.46*	*1279.14*	*9*	*555.81*	*1249.28*	*6*
RC206	*251.60*	*1157.00*	*8*	*643.71*	*1097.04*	*5*
RC207	*295.21*	*1045.10*	*8*	*639.32*	*1048.95*	*5*
RC208	*262.23*	*791.56*	*5*	*898.74*	*869.62*	*4*
Average	*216.15*	*1033.29*	*10.04*	*590.97*	*1022.72*	*8.15*

Table 4. Comparison Between Insertion-Based Method and Genetic Algorithm(Ghoseiri & Ghannadpour, 2010)

data	Genetic Algorithm (Ghoseiri & Ghannadpour, 2010)		Baseline Insertion Method	
	Number of Vehicles	Total Service Time	Number of Vehicles	Total Service Time
R101	19	1677	20	1700.70
R102	18	1511.8	19	1514.84
R103	14	1287	15	1276.26
R104	10	974.24	12	1076.09
R105	15	1424.6	15	1428.03
R106	13	1270.3	13	1301.36
R107	11	1108.8	12	1144.21
R108	10	971.91	11	1013.97
R109	12	1212.3	14	1294.33
R110	12	1156.5	12	1130.49
R111	11	1111.9	12	1154.07
R112	10	1036.9	11	1015.55
R201	4	1351.4	4	1316.75
R202	4	1091.22	4	1142.73
R203	3	1041	4	941.22
R204	3	1130.1	4	765.76
R205	4	1087.8	4	1015.27
R206	3	940.12	3	969.01
R207	3	904.9	3	874.76
R208	3	774.18	3	720.16
R209	4	1008	3	976.88
R210	3	938.58	4	948.65
R211	4	1101.5	3	836.11
C101	10	828.94	10	828.94
C102	10	828.94	10	834.64
C103	10	828.06	11	884.48
C104	10	824.78	10	866.68
C105	10	828.94	10	828.94
C106	10	828.94	10	828.94
C107	10	828.94	10	829.70
C108	10	828.94	10	828.94
C109	10	828.94	10	836.33
C201	3	591.56	3	591.56
C202	3	591.56	3	591.56
C203	3	591.17	3	591.17
C204	3	599.96	3	602.95
C205	3	588.88	3	588.88
C206	3	588.88	3	588.49
C207	3	591.56	3	588.29
C208	3	588.32	3	588.32
RC101	15	1690.6	17	1722.51
RC102	14	1509.4	14	1548.48
RC103	12	1331.8	13	1381.06
RC104	11	1177.2	11	1223.67
RC105	15	1611.5	17	1600.80
RC106	13	1437.6	13	1421.89
RC107	11	1222.1	13	1337.86
RC108	11	1156.5	12	1199.60
RC201	4	1423.7	5	1436.35
RC202	4	1369.8	5	1174.06
RC203	4	1060	4	973.73
RC204	3	901.46	4	809.36
RC205	4	1410.3	6	1249.28
RC206	4	1194.8	5	1097.04
RC207	4	1040.6	5	1048.95
RC208	3	898.5	4	869.62
Average	7.84	1048.84	8.15	1022.72

Table 3 compares the insertion-based and savings-based methods using the Solomon dataset. The first column lists the dataset identifiers, followed by three columns for each method: the number of vehicles used, total service time, and processing time. The results show that the insertion-based method outperforms the savings-based approach in solution quality, with lower average total costs (1022.72 vs. 1033.29) and fewer vehicles used (8.15 vs. 10.04). Additionally, the insertion-based method yielded better results in 66.1% of the datasets, demonstrating its superiority in cost-efficiency and routing performance. However, the savings-based method was significantly faster, with an average processing time of 216.15 compared to 590.97 for the insertion-based approach.

Table 4 compares the performance of the insertion-based method and a Genetic Algorithm (GA) (Ghoseiri & Ghannadpour, 2010) in terms of vehicle count and total service time. The insertion-based method achieved better solutions in 53.6% of the datasets, with lower average total costs (1022.72 vs. 1048.84) compared to the GA. However, the GA required slightly fewer vehicles on average (7.84 vs. 8.15), suggesting a trade-off between cost efficiency and fleet size optimization.

The results demonstrate that the choice between the insertion-based and savings-based methods depends on the specific priorities of the application. The insertion-based method is more effective when optimizing for solution quality and cost minimization, whereas the savings-based method is advantageous for scenarios requiring faster computational performance.

It is worth noting that the GA results used for comparison were drawn from the work of Ghoseiri and Ghannadpour (Ghoseiri & Ghannadpour, 2010), where a multi-objective VRPTW model was solved using a goal programming approach and a customized Genetic Algorithm. Their GA incorporated advanced mechanisms such as the Push Forward Insertion Heuristic and λ-interchange in the initialization phase, along with Pareto-based selection and local improvement strategies during evolution. Chromosomes represented customer sequences, and order-based crossover and mutation were used to maintain diversity. The authors tuned GA parameters empirically and tested their method on Solomon's benchmark datasets. Since our study uses their published results without re-implementation, the comparison in Table 4 reflects differences in approach and performance rather than direct execution under identical settings.

6. Discussion and Conclusion

This study addresses the Vehicle Routing Problem with Time Windows (VRPTW) under vehicle capacity constraints, with objectives to minimize total service time, reduce fleet size, and account for vehicle waiting times. The proposed solution adopts a three-phase optimization framework combining initial solution construction, perturbation, and local search. The initial phase utilizes either insertion or savings heuristics to generate feasible solutions, thereby preventing cold starts and improving convergence efficiency. The subsequent perturbation phase employs Variable Neighborhood Search (VNS), which introduces solution diversity through mechanisms comparable to mutation operators in genetic algorithms, effectively avoiding local optima. Finally, the Variable Neighborhood Descent (VND) method refines solutions through systematic local search. This combined approach leverages both exploratory capabilities and intensive local optimization to effectively solve the VRPTW.

The two methods differ primarily in their initial solution construction: the insertion-based approach employs the insertion heuristic, while the savings-based method utilizes the savings heuristic. Both approaches share identical perturbation and local search phases, employing neighborhood structures including one-point crossover, relocation, swap, two-point crossover, and inverse crossover during perturbation, and operators such as inversion, 1-opt (including repeated application), and one-point crossover during local search.

Evaluation on Solomon's benchmark datasets (random, clustered, and random-clustered configurations, each containing 100 customers) revealed distinct performance characteristics. The insertion-based method demonstrated superior solution quality, achieving lower average total costs (1022.72 vs. 1033.29) and requiring fewer vehicles (8.15 vs. 10.04), while producing optimal solutions in 66.1% of cases. However, the savings-based method showed significantly faster computation times (216.15 seconds vs. 590.90 seconds per instance), as shown in Table 3.

This performance difference stems from fundamental algorithmic distinctions. The savings heuristic maintains computational efficiency by computing the savings list once (with subsequent resorting) and simplifying route merges by considering only route endpoints. Conversely, the insertion heuristic's exhaustive evaluation of all possible customer insertion positions results in greater computational overhead, though yielding better quality solutions.

Comparative analysis reveals the insertion-based method achieved superior cost efficiency (average total cost: 1022.72 vs. 1048.84) compared to the genetic algorithm approach (Ghoseiri & Ghannadpour, 2010), though the genetic algorithm demonstrated marginally better vehicle utilization (7.84 vs. 8.15 vehicles). The three-phase methodology developed in this study offers several advantages over conventional approaches. By employing heuristic-based initial solutions rather than random starts, the method achieves faster convergence and more efficient optimization. The incorporation of a perturbation phase effectively prevents convergence to local optima, with the framework occasionally outperforming existing genetic algorithm implementations.

A key innovation lies in the systematic validation of solution feasibility throughout each phase, eliminating the need for post-hoc correction procedures and their associated computational costs. The comprehensive exploration of neighborhood structures and local search operators in this work suggests several productive directions for future research, including: (1) systematic evaluation of individual

operator contributions, (2) investigation of alternative operator combinations, and (3) integration with real-world routing constraints such as dynamic traffic conditions. Furthermore, the potential synergies between the proposed approach and established metaheuristics (e.g., particle swarm optimization, tabu search) warrant examination in subsequent studies.

References

Ahkamiraad, A., & Wang, Y. (2018). Capacitated and multiple cross-docked vehicle routing problem with pickup, delivery, and time windows. Computers & Industrial Engineering, 119, 76-84. https://doi.org/10.1016/j.cie.2018.03.007

Bräysy, O., & Gendreau, M. (2002). Tabu search heuristics for the vehicle routing problem with time windows. Top, 10(2), 211-237. https://doi.org/10.1007/BF02579017

Da Silva, R. F., & Urrutia, S. (2010). A General VNS heuristic for the traveling salesman problem with time windows. Discrete Optimization, 7(4), 203-211. https://doi.org/10.1016/j.disopt.2010.04.002

Desaulniers, G., Madsen, O. B., & Ropke, S. (2014). Chapter 5: The vehicle routing problem with time windows. In Vehicle Routing: Problems, Methods, and Applications, Second Edition (pp. 119-159). SIAM. https://doi.org/10.1137/1.9781611973594.ch5

Dieter, P., Caron, M., & Schryen, G. (2023). Integrating driver behavior into last-mile delivery routing: Combining machine learning and optimization in a hybrid decision support framework. European Journal of Operational Research, 311(1), 283-300. https://doi.org/10.1016/j.ejor.2023.04.043

Eiben, A. E., & Smith, J. E. (2015). Introduction to evolutionary computing. Springer.

Ghoseiri, K., & Ghannadpour, S. F. (2010). Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. Applied Soft Computing, 10(4), 1096-1107. https://doi.org/10.1016/j.asoc.2010.04.001

Guo, S., Hu, H., & Xue, H. (2024). A Two-Echelon Multi-Trip Capacitated Vehicle Routing Problem with Time Windows for Fresh E-Commerce Logistics under Front Warehouse Mode. Systems, 12(6), 205. https://doi.org/10.3390/systems12060205

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. European Journal of Operational Research, 130(3), 449-467. https://doi.org/10.1016/S0377-2217(00)00100-4

Hansen, P., Mladenović, N., Todosijević, R., & Hanafi, S. (2017). Variable neighborhood search: basics and variants. EURO Journal on Computational Optimization, 5(3), 423-454. https://doi.org/10.1007/s13675-016-0075-x

Hassin, R., & Keinan, A. (2008). Greedy heuristics with regret, with application to the cheapest insertion algorithm for the TSP. Operations Research Letters, 36(2), 243-246. https://doi.org/10.1016/j.orl.2007.05.001

Irnich, S., Toth, P., & Vigo, D. (2014). Chapter 1: The family of vehicle routing problems. In Vehicle Routing: Problems, Methods, and Applications, Second Edition (pp. 1-33). SIAM. https://doi.org/10.1137/1.9781611973594.ch1

Karakatič, S., & Podgorelec, V. (2015). A survey of genetic algorithms for solving multi depot vehicle routing problem. Applied Soft Computing, 27, 519-532. https://doi.org/10.1016/j.asoc.2014.11.005

Li, Y., Chen, M., & Huo, J. (2022). A hybrid adaptive large neighborhood search algorithm for the large-scale heterogeneous container loading problem. Expert Systems with Applications, 189, 115909. https://doi.org/10.1016/j.eswa.2021.115909

Liu, Y. (2022). Cognitive Smart City Logistics: a new approach for sustainable last mile in the era of digitization Université Paris sciences et lettres].

Lu, Q., & Dessouky, M. M. (2006). A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. European Journal of Operational Research, 175(2), 672-687. https://doi.org/10.1016/j.ejor.2005.05.012

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. Computers & operations research, 24(11), 1097-1100. https://doi.org/10.1016/S0305-0548(97)00031-2

Polat, O., Kalayci, C. B., Kulak, O., & Günther, H.-O. (2015). A perturbation based variable neighborhood search heuristic for solving the vehicle routing problem with simultaneous pickup and delivery with time limit. European Journal of Operational Research, 242(2), 369-382. https://doi.org/10.1016/j.ejor.2014.10.010

Ruiz, E., Soto-Mendoza, V., Barbosa, A. E. R., & Reyes, R. (2019). Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm. Computers & Industrial Engineering, 133, 207-219. https://doi.org/10.1016/j.cie.2019.05.002

Savić, B., Petrović, M., & Vasiljević, Z. (2020). The impact of transportation costs on economic performances in crop production. Економика пољопривреде, 67(3), 683-697. https://doi.org/10.5937/ekoPolj2003683S

Shahbazian, R., Pugliese, L. D. P., Guerriero, F., & Macrina, G. (2024). Integrating Machine Learning Into Vehicle Routing Problem: Methods and Applications. IEEE Access. https://doi.org/10.1109/ACCESS.2024.3422479

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

Todosijević, R., Mjirda, A., Mladenović, M., Hanafi, S., &

Gendron, B. (2017). A general variable neighborhood search variants for the travelling salesman problem with draft limits. Optimization Letters, 11, 1047-1056. https://doi.org/10.1007/s11590-014-0788-9

Wang, Z., & Zhou, C. (2016). A Three‐Stage Saving‐Based Heuristic for Vehicle Routing Problem with Time Windows and Stochastic Travel Times. Discrete Dynamics in Nature and Society, 2016(1), 7841297. https://doi.org/10.1155/2016/7841297

Yoshizaki, H. T. Y. (2009). Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in Brazil. European Journal of Operational Research, 199(3), 750-758. https://doi.org/10.1016/j.ejor.2008.08.003

Zhang, R. (2024). Branch-and-Price for the Capacitated Autonomous Vehicle Assisted Delivery Problem. INFORMS Journal on Computing. https://doi.org/10.1287/ijoc.2023.0177