

Article

Solving the Vehicle Routing Problem with Time Windows Using Modified Rat Swarm Optimization Algorithm Based on Large Neighborhood Search

Xiaoxu Wei ¹ , Zhouru Xiao ¹ and Yongsheng Wang ^{2,*} ¹ School of Automotive Engineering, Wuhan University of Technology, Wuhan 430070, China; wx2014@whut.edu.cn (X.W.); xzr798@whut.edu.cn (Z.X.)² School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China

* Correspondence: wysh@whut.edu.cn

Abstract: The vehicle routing problem with time windows (VRPTW) remains a formidable challenge, due to the intricate constraints of vehicle capacity and time windows. As a result, an algorithm tailored for this problem must demonstrate robust search capabilities and profound exploration abilities. Traditional methods often struggle to balance global search capabilities with computational efficiency, thus limiting their practical applicability. To address these limitations, this paper introduces a novel hybrid algorithm known as large neighborhood search with modified rat swarm optimization (LNS-MRSO). Modified rat swarm optimization (MRSO) is inspired by the foraging behavior of rat swarms and simulates the search process for optimization problems. Meanwhile, large neighborhood search (LNS) generates potential new solutions by removing and reinserting operators, incorporating a mechanism to embrace suboptimal solutions and strengthening the algorithm's prowess in global optimization. Initial solutions are greedily generated, and five operators are devised to mimic the position updates of the rat swarm, providing rich population feedback to LNS and further enhancing algorithm performance. To validate the effectiveness of LNS-MRSO, experiments were conducted using the Solomon VRPTW benchmark test set. The results unequivocally demonstrate that LNS-MRSO achieves optimal solutions for all 39 test instances, particularly excelling on the R2 and RC2 datasets with percentage deviations improved by 5.1% and 8.8%, respectively, when compared to the best-known solutions (BKSs). Furthermore, when compared to state-of-the-art algorithms, LNS-MRSO exhibits remarkable advantages in addressing VRPTW problems with high loading capacities and lenient time windows. Additionally, applying LNS-MRSO to an unmanned concrete-mixing station further validates its practical utility and scalability.

Keywords: rat swarm optimization algorithm; large neighborhood search; behavior operator; vehicle routing problem with time windows

MSC: 90C39; 90C27

Citation: Wei, X.; Xiao, Z.; Wang, Y. Solving the Vehicle Routing Problem with Time Windows Using Modified Rat Swarm Optimization Algorithm Based on Large Neighborhood Search. *Mathematics* **2024**, *12*, 1702. <https://doi.org/10.3390/math12111702>

Academic Editors: Shih-Wei Lin and Ioannis G. Tsoulos

Received: 16 March 2024

Revised: 17 May 2024

Accepted: 28 May 2024

Published: 30 May 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The vehicle routing problem (VRP) is a well-known combinatorial optimization problem, initially proposed by Dantzig et al. in 1954 during their study of the optimal route for gasoline transport vehicles [1]. VRP aims to address scenarios involving multiple customers with varying goods demands. Vehicles originate from one or more warehouses, traverse a network connecting all customers, provide services based on pre-planned routes, and then return to the warehouse. When each vehicle is subject to a capacity constraint, the problem transforms into a capacitated vehicle routing problem (CVRP) [2,3]. If there are constraints on the travel distance of vehicles along planned paths, it becomes a distance-constrained VRP (DVRP) [4]. Incorporating customer service time constraints leads to the formulation of a vehicle routing problem with time windows (VRPTW) [5], which better aligns with

actual production demand. In addition to CVRP's constraints, each customer point is assigned upper and lower bounds on replenishment demand service times within specific periods; the objective function seeks to minimize total travel distance and cost [6]. Due to its direct relevance to real logistics challenges and its complexity, VRPTW has garnered significant attention in the logistics field. Thorough research and resolution of this issue are not only vital for enhancing logistics industry efficiency but also essential for advancing modern intelligent logistics systems development. In an increasingly competitive market environment, logistics enterprises must seek optimal distribution solutions that minimize costs while improving efficiency and ensuring the timely and accurate delivery of goods.

With rapid advancements in unmanned and electric technologies, the logistics and distribution sector is undergoing profound transformations. Within this evolving landscape, VRPTW has regained its status as a focal research area. Essentially involving optimizing vehicle routes to meet customers' time window requirements while ensuring efficient and timely delivery of goods, earlier research primarily concentrated on minimizing the number of vehicles and travel distances to cut transportation costs [7]. However, the current technological trend is shifting focus towards minimizing individual vehicle travel distance to reduce energy consumption and emissions, enhance unmanned vehicle utilization, and further improve overall distribution efficiency. Therefore, optimizing for shortest travel distance becomes paramount in VRPTW. Delving deeper into VRPTW within the context of unmanned and electric technologies holds significant practical implications for advancing theoretical development in combinatorial optimization and intelligent logistics systems while facilitating green initiatives in logistic distributions. Addressing this issue will significantly enhance transportation efficiency, reduce costs, provide businesses with competitive advantages, and steer the logistic distribution industry towards a more sustainable trajectory, thereby offering theoretical insights along with practical applications for future intelligent logistic systems.

Various methods are available for addressing VRPTW, including exact methods, heuristic algorithms, meta-heuristic algorithms, and those based on artificial intelligence. Exact algorithms have the capability to find optimal solutions; however, due to the NP-hard nature of VRPTW, the computation time increases exponentially as the problem size grows. This exponential growth makes exact algorithms impractical for large-scale engineering applications. Consequently, exact algorithms such as dynamic programming, column generation, and branch and bound are typically limited to small-scale problems or theoretical research. Heuristic algorithms receive extensive research focus and are typically designed for specific problems with limited applicability. In contrast, meta-heuristic methods like the simulated annealing algorithm, tabu search algorithm, and genetic algorithm are well suited for a wide range of challenging combinatorial optimization problems as they iteratively operate on solutions or parts of solutions to intelligently search and improve solution spaces in order to obtain satisfactory results. The application of artificial intelligence in vehicle routing problems is still in its early stages; however, it is anticipated that more efficient solutions will be achieved through deep reinforcement learning for neural network construction in the future.

With the advancement of research, heuristic and meta-heuristic algorithms have increasingly assumed a prominent role in scientific inquiries due to their high efficiency and adaptability. While heuristic algorithms are often tailored to specific problems, which limits their widespread application, they prove particularly useful in the context of VRPTW. In VRPTW, heuristic methods can be broadly classified into two categories: path construction heuristics for generating initial feasible solutions and path improvement heuristics aimed at further enhancing the quality of existing solutions. Early literature witnessed Clarke et al. (1964) introducing the savings heuristic for vehicle routing problems, laying the foundation for subsequent VRPTW solutions [8]. Subsequently, Baker et al. (2003) built upon this foundation, introducing path construction heuristics specifically designed for VRPTW [9]. Solomon (1987) further developed the Push Forward Insertion Heuristic (PFIH), which explores improved solutions by considering time windows and inter-node

distances [10]. However, these methods still face the risk of getting trapped in local optima, especially when dealing with large-scale problems. To overcome this limitation, Potvin et al. (1993) introduced a heuristic approach that simultaneously constructs multiple paths, aiming to achieve optimal distribution plans by re-planning the farthest points from the distribution center [11]. Building upon this, Antes et al. (1995) proposed a novel method that incorporates a “reward” mechanism, allowing for the addition of multiple customer nodes to multiple paths within a single iteration [12]. This enhancement significantly improved the algorithm’s search efficiency and global optimization capabilities.

Despite significant improvements in solution quality, current methods still encounter challenges such as lengthy computation time and susceptibility to local optima, particularly when addressing large-scale and highly complex VRPTW. Many scholars have utilized heuristic and meta-heuristic algorithms for solving large-scale VRPTW problems, with GA, PSO, and ACO being widely employed [13–17]. To enhance solution quality, most improved algorithms continue to rely on simple evolutionary and local search methods, resulting in slow convergence and limited optimization effects. Researchers have started exploring hybrid algorithms that integrate heuristic and meta-heuristic methods, showing some progress [4,13–23]. However, the integration of swarm intelligence optimization algorithms with population diversity and neighborhood search algorithms is a promising research topic for improving accuracy and convergence speed when dealing with large-scale and complex VRPTW problems. The seamless integration of large neighbor search (LNS) and improved rat swarm optimization (MRSO) into a hybrid solution for the VRPTW problem aims to achieve population search using only two formulas while requiring a small number of parameters to balance the exploration and exploitation processes.

The paper makes four primary contributions:

1. This paper presents an innovative proposal for the LNS-MRSO algorithm, which combines large neighborhood search with modified rat swarm optimization to simulate rat swarm behavior and introduce a novel approach. The proposed method not only leverages the strengths of LNS in fine-grained local search but also enhances RSO’s capacity for global exploration, significantly improving search efficiency and solution quality. Moreover, it effectively addresses the challenges of slow search speed and susceptibility to local optima in the MRSO algorithm, providing new insights and methodologies for related research areas.
2. The paper proposes behavioral operators (chase, jump, rotate, attack, and escape) inspired by rat swarm behavior to overcome the limitations of the RSO algorithm in continuous optimization problems. These operators enhance population diversity and global search capabilities. Additionally, a corresponding position update mechanism is established to effectively apply the MRSO algorithm to VRPTW.
3. Thorough benchmark testing and experimental validation: by employing widely recognized VRPTW benchmark test sets, this paper empirically evaluates the performance of the LNS-MRSO algorithm in solving VRPTW problems of varying sizes and complexities. The experimental results demonstrate that the LNS-MRSO algorithm surpasses other heuristic and meta-heuristic algorithms in terms of solution quality and stability for specific instances.
4. The implementation of the LNS-MRSO algorithm in the scheduling system of unmanned electric loaders at concrete-mixing stations has resulted in significant annual electricity savings of approximately 1.6 million units.

The subsequent sections of the article are structured as follows: Section 2 involves the research work of relevant references. Section 3 provides a detailed elaboration of the VRPTW model; Section 4 presents a comprehensive introduction to the LNS-MRSO algorithm proposed in this paper; Section 5 validates the effectiveness of the algorithm through a solo calculation example and comparison with other algorithms; Section 6 conducts simulation experiments on the unmanned electric loader of the concrete-mixing station; finally, Section 7 summarizes this paper and looks forward to future research directions.

2. Related Works

Meta-heuristics demonstrate the capability to effectively handle additional constraints and generate nearly optimal solutions for pathfinding within a reasonable computational timeframe, applicable to networks of varying scales. Meta-heuristic approaches such as GA, PSO, and ACO algorithms have been extensively utilized in addressing shortest path problems across diverse research domains. For instance, Ayesha et al. (2024) present an innovative Hybrid Genetic Algorithm–Solomon Insertion Heuristic (HGA-SIH) solution, enhanced by the robust Solomon insertion constructive heuristic for solving the NP-hard VRPTW problem [13]. Khoo et al. (2021) introduce a genetic algorithm specifically tailored for tackling the multi-objective vehicle routing problem with time windows (MOVRPTW). This specialized GA employs a two-stage distributed hybrid destruction and reconstruction strategy that integrates sequential processing and parallel processing to enhance overall algorithm performance [14]. Sedighizadeh et al. (2018) propose a hybrid algorithm combining PSO with artificial bee colony (ABC) algorithm for addressing the multi-objective vehicle routing problem with inter-client priority constraints [15]. Dib et al. (2017) propose an approach that combines GA with variable neighborhood search (VNS) [18]. Furthermore, they develop an advanced GA-VNS heuristic method to address the multi-criteria shortest path problem in multimodal networks [19]. Mohiuddin et al. (2016) design a fuzzy evolutionary particle swarm optimization (FEPSO) algorithm to optimize routing paths and improve network efficiency [20].

In recent years, there have been significant advancements in addressing VRPTW through the use of hybrid algorithms that integrate heuristic and meta-heuristic approaches. These algorithms have emerged as prominent methods for tackling large-scale and intricate VRP and scheduling challenges. For instance, Wu et al. (2024) introduced the neighborhood comprehensive learning particle swarm optimization (N-CLPSO), which enhances global search capability by incorporating a neighborhood search mechanism encompassing removal and reinsert operators to effectively resolve VRPTW [16]. Teng et al. (2024) developed an evolutionary algorithm based on the ant colony system and Kuhn–Munkres (ACS-KM) bipartite graph matching to address dynamic VRPTW [17]. Zakir et al. (2023) proposed a modified football game algorithm (MFGA), simulating footballer behaviors under coach guidance to identify optimal scoring positions and devising a more efficient player position generation method, thereby optimizing VRPTW solutions [21]. Additionally, Yassen et al. (2017) put forth an adaptive harmony search hybrid algorithm (HSA), incorporating an adaptive selection mechanism to autonomously choose appropriate local search algorithms [22]. Banos et al. (2012) proposed a multi-objective variant of the VRPTW that aims to minimize travel distance and route imbalance using a procedure called multiple temperature Pareto simulated annealing (MT-PSA) [23]. Ursani et al. (2011) proposed a Local Genetic Algorithm (LGA) to address VRPTW, integrating tabu search techniques to enhance algorithm performance. The application of LGA to various benchmark instances demonstrated superior performance compared to other state-of-the-art algorithms [4]. In essence, hybrid algorithms that integrate heuristic algorithms with neighborhood search techniques have demonstrated robust performance in solving VRPTW.

Meta-heuristic algorithms can be categorized into two main groups: single-solution-based and population-based [13]. The single-solution-based heuristic algorithms encompass classical methods such as SA, TS, and VNS, as well as prominent algorithms like large neighborhood search (LNS) and adaptive large neighborhood search (ALNS). Conversely, population-based approaches are further divided into evolutionary computation (EC) techniques (such as GA, ES, EP, GP, and PR) and swarm intelligence (SI) methods (such as ACO, PSO, and BFOA). In Table 1 of this paper, we present a summary of the algorithms utilized in recent VRPTW studies based on this classification. Extensive literature reviews in the aforementioned domains consistently underscore the efficacy of integrating meta-heuristic algorithms with local search techniques or hybrid approaches for addressing diverse vehicle routing problems. The application of these methods has yielded remarkable success

in the context of VRPTW. Consequently, we have developed a novel hybrid algorithm, referred to as large neighborhood search with modified rat swarm optimization.

Table 1. Summary of hybrid algorithms in VRPTW-related references.

References	Year	Algorithms	Single-Based Heuristics	LNS/ALNS	EC	SI	Other
[13]	2024	HGA-SIH	✓		✓		
[14]	2021	HRRGA			✓		
[15]	2018	PSO-ABC				✓	
[18,19]	2017	GA-VNS	✓		✓		
[20]	2016	FEPSO				✓	✓
[16]	2024	N-CLPSO		✓		✓	
[17]	2024	ACS-KS			✓	✓	
[21]	2023	MFGA				✓	✓
[22]	2017	HSA	✓				
[23]	2012	MT-PSA	✓				✓
[4]	2011	LGA	✓		✓		
Current study	2024	LNS-MRSO	✓	✓		✓	

This paper introduces a hybrid algorithm that seamlessly integrates large neighborhood search (LNS) with modified rat swarm optimization (MRSO), building upon existing literature. Rat swarm optimization (RSO) is an innovative bio-inspired algorithm, introduced by Gaurav Dhiman et al. (2020), which mimics the hunting and fighting behaviors of rats in nature to emulate their social intelligence and aggressiveness, effectively addressing global optimization challenges [24]. Oruba et al. (2023) propose a modified rat swarm optimization algorithm for node localization in wireless sensor networks, resulting in significantly reduced positioning error [25]. Walid et al. (2023) present a trust-aware clustering technique based on the rat swarm optimization algorithm for the secure selection of cluster heads in wireless sensor networks for intelligent transportation systems [26]. Ibrahim et al. (2022) utilize the search capability of the rat swarm optimization algorithm to identify optimal cluster centers, demonstrating its effectiveness over other clustering techniques [27]. Xie et al. (2022) introduce a multi-strategy modified rat swarm optimization algorithm that effectively addresses path planning for mobile robots and enhances global search capability and optimization efficiency [28]. In conclusion, the engineering application potential of the rat swarm optimization algorithm is demonstrated through its innovative application to VRPTW.

However, applying RSO to VRPTW requires a tailored encoding scheme and operators that align with its unique characteristics. The selection of encoding rules significantly impacts the algorithm's performance. A well-designed encoding scheme can streamline operations, minimize computational complexity, enhance efficiency, and facilitate decoding. Conversely, suboptimal encoding may restrict the search space, compromise solution precision, and increase computational complexity. Encoding rules encompass various methodologies such as binary encoding, symbolic encoding, and floating-point encoding among others. It is crucial to abstract the mathematical model for the given problem while emphasizing primary contradictions over secondary ones. In the context of VRPTW, the encoding must capture both the sequence of customer nodes and the vehicle IDs. In this study, a real-number encoding approach is employed as Figure 1, with the distribution center designated as "0" and the demand points consecutively encoded as 1, 2, 3, ..., whereas the vehicle IDs are represented by real numbers 0, 1, 2, ..., thereby establishing the encoding rules for the routes.

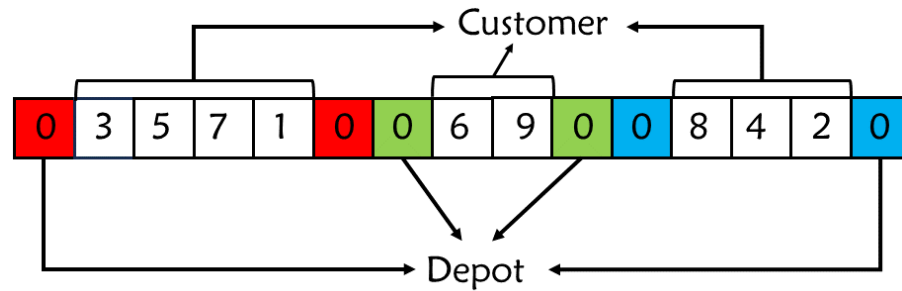


Figure 1. The coding method for VRPTW.

3. Mathematical Definition of VRPTW

VRPTW can be conceptualized as a problem involving k vehicles servicing n customers. Each vehicle commences and terminates its journey at a designated distribution center. The maximum cargo capacity assigned to each vehicle is designated as Q . The distribution center is represented by v_0 , whereas the customers are designated as v_i with positional coordinates (x_i, y_i) (where i ranges from 1 to n). The cargo demand of each customer, denoted as q_i , does not exceed the maximum load capacity of the vehicle. The service time for customer v_i is designated as s_i . Additionally, each customer has a prescribed time window $[e_i, l_i]$ for receiving service, where e_i represents the earliest permissible service start time, and l_i signifies the latest acceptable service completion time. Any vehicle arriving prior to e_i will be required to wait, and a penalty will be incurred for arrivals after l_i . The positional coordinates of the distribution center, v_0 , are (x_0, y_0) , and its time window $[0, l_0]$ must adhere to the condition that l_0 is no less than $\max(e_i)$. The set of vehicles is represented by K , the set of customers is denoted by N , and set M comprises both v_0 and N .

The primary objective of our study is to identify a feasible route that satisfies the constraints of vehicle capacity and time windows, with the dual purpose of minimizing the utilization of vehicles and reducing the overall transportation distance.

The mathematical model is delineated as follows:

$$\min f_1 = \sum_{i \in M} \sum_{j \in M} \sum_{k \in K} c_{ij} x_{ij}^k \quad (1)$$

$$\min f_2 = \sum_{k \in K} \sum_{i \in N} x_{0i}^k \quad (2)$$

s.t.

$$\sum_{k \in K} y_i^k = 1, \forall i \in N \quad (3)$$

$$\sum_{i \in N} x_{ij}^k = y_j^k, \forall j \in N, \forall k \in K \quad (4)$$

$$\sum_{j \in N} x_{ij}^k = y_i^k, \forall i \in N, \forall k \in K \quad (5)$$

$$\sum_{i \in N} y_i^k \cdot q_i \leq Q, \forall k \in K \quad (6)$$

$$\sum_{k \in K} y_0^k = |K| \quad (7)$$

$$t_i + w_j + s_i + c_{ij} = t_j, \forall i, j \in N, i \neq j \quad (8)$$

$$e_i \leq t_i \leq l_i, \forall i \in N \quad (9)$$

$$w_j = \max\{0, e_i - (t_i + s_i + c_{ij})\}, \forall i \in N \quad (10)$$

$$x_{ij}^k = \begin{cases} 1, & \text{if vehicle } k \text{ travels directly from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

$$y_i^k = \begin{cases} 1, & \text{if customer } i \text{ is served by vehicle } k \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

The objective function f_1 pertains to the transportation distance, whereas f_2 pertains to the number of vehicles utilized. Equations (3) through (5) guarantee that each customer receives service from a single vehicle. Equation (6) establishes that the carrying capacity of each vehicle must not be surpassed. Equation (7) designates that all vehicles commence and terminate their routes at the distribution center. Equations (8) to (10) encompass time window constraints, where t_i signifies the arrival time of the vehicle at v_i , w_i represents the vehicle's waiting time at v_i , and c_{ij} represents the travel duration, equivalent to the distance, from v_i to v_j .

This model demonstrates robust generality, enabling its transformation into mathematical frameworks for various combinatorial optimization challenges through parameter adjustments [16,17]. More specifically, by setting e_i to 0 and l_i to a substantial value ϕ , constraints (8) through (10) can be eliminated, effectively reducing the VRPTW model to a standard VRP model. In scenarios where a single vehicle is available, the problem morphs into a TSP problem. Furthermore, when multiple vehicles are available and additional conditions $c_{0j} = 1 (j \in N)$ and $c_{ij} = 0$ are imposed, the mathematical model for the bin packing problem can be derived.

4. Framework of LNS-MRSO

This paper presents a modified rat swarm optimization algorithm, named LNS-MRSO, which combines the strengths of both the traditional rat swarm algorithm and the large neighborhood search algorithm. Figure 2 illustrates the framework of LNS-MRSO. MRSO introduces five innovative operators: chase, jump, rotate, attack, and escape to update the positions of rats. These positions are further refined using a multi-factor-guided remove operator and a distance-increment-guided reinsert operator. In case a new position exhibits inferior quality, the algorithm incorporates a simulated annealing mechanism to evaluate its retention; if found unsuitable, it is replaced with an updated position from MRSO. Conversely, if the new position is superior, the current position is promptly updated.

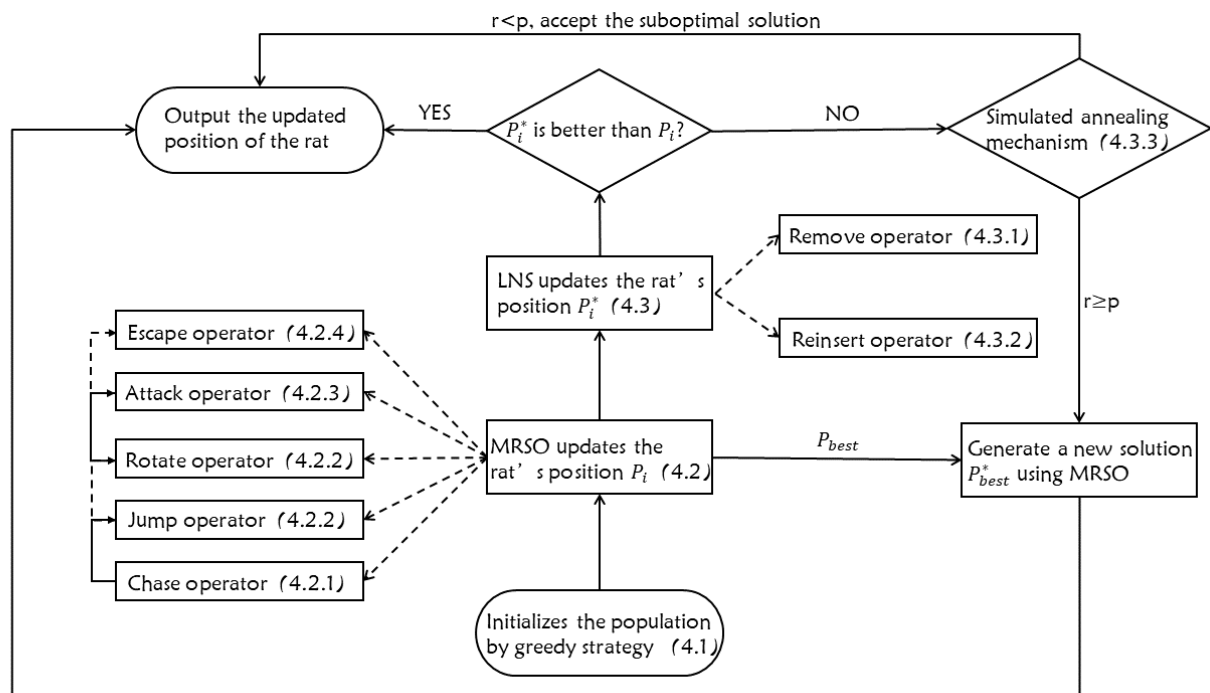


Figure 2. Framework of LNS-MRSO.

4.1. Greedy Strategy

The quality of the initial solution is crucial for the effectiveness of MRSO. However, during its operation, MRSO often encounters local optimal solutions, which restrict its search scope and hinder its convergence to a global optimal solution. This compromises its optimization capabilities and limits its practical application. To expand the search space and ensure the quality and diversity of solutions, this paper adopts a greedy strategy for constructing the initial solution. This strategy initializes vehicle delivery routes by randomly selecting a starting customer point, arranging customers based on their time windows, and adhering to vehicle capacity constraints. When the route capacity is exceeded, a new route is initiated. By generating an initial set of constraint-satisfying routes, MRSO obtains a diverse pool of candidate solutions that minimizes the number of vehicles required for initialization while significantly enhancing compliance with capacity and time window constraints. Consequently, this approach greatly improves the search efficiency, stability, and robustness of MRSO. The detailed process of initializing using the greedy strategy is outlined in Algorithm 1. Implementing this strategy provides MRSO with an extensive candidate solution space that effectively reduces the number of vehicles needed during initialization. Notably, these initially generated solutions exhibit high adherence to capacity and time window constraints, thereby mitigating potential constraint violations throughout the search process.

Algorithm 1 Population initialization

Input: Q : vehicle capacity, e : customer's left time window, q : customer demand

Output: P_{set} : Routing paths

```

1:  $P_{set} = \emptyset$ ;
2: Randomly select a customer index  $i$  from  $\{1, 2, \dots, n\}$ ;
3: Initialize path  $P$  as  $[i, i + 1, i + 2, \dots, N, 1, 2, \dots, i - 1]$ ;
4:  $k \leftarrow 1$ ;
5: for  $j \leftarrow 1$  to  $N$  do
6:   if  $Q < P_{set}[k].Q$  then
7:      $k \leftarrow k + 1$ ;
8:   end if
9:   if  $P_{set}[k]$  is empty then
10:     $P_{set}[k] \leftarrow P_{set}[k] \cup \{P(j)\}$ ;
11:   else if  $|P_{set}[k]| = 1$  then
12:     if  $P(j).e < P_{set}[k][0].e$  then
13:        $P_{set}[k] \leftarrow \{P(j)\} + P_{set}[k]$ ;
14:     else
15:        $P_{set}[k] \leftarrow P_{set}[k] + \{P(j)\}$ ;
16:     end if
17:   else
18:      $l \leftarrow 0$ ;
19:     while customer  $P(j)$  is not inserted into  $P_{set}[k]$  do
20:       if  $P(j).e \leq P_{set}[k][l + 1].e$  and  $P(j).e > P_{set}[k][l].e$  then
21:         Insert  $P(j)$  into  $P_{set}[k]$  at position  $l$ ;
22:         break;
23:       else
24:          $l \leftarrow l + 1$ ;
25:       end if
26:     end while
27:   end if
28:   Update  $P_{set}[k].Q$  as  $P_{set}[k].Q + P(j).q$ ;
29: end for
30: return  $P_{set}$ ;

```

4.2. Simulated Rat Behavior Operators

In the fundamental RSO algorithm, the behavior of the rat swarm is limited to chasing and attacking. Given our unique encoding prerequisites for this study, we have expanded the behavioral repertoire of the rat swarm by incorporating five simulated operators:

chase, jump, rotate, attack, and escape. These operators represent a novel approach to addressing discrete problems and are visually depicted in Figure 3 during simulation. The objective of these operators is to enhance population diversity and strengthen global search capabilities. Each operator possesses a distinct position updating mechanism that iteratively refines solutions to VRPTW by meticulously fine-tuning candidate solution composition and layout. Specifically, the chase operator mimics pursuit behavior within the rat swarm, improving local search proficiency by focusing on high-performing solutions. The jump operator effectively avoids local minima by introducing significant alterations to solution composition. The rotate operator explores new solution domains through segment rotation within solutions. The attack operator simulates competitive interactions among rat swarms, optimizing solutions through neighborhood exploration. Lastly, the escape operator allows for the evasion of local minima and resumption of global search when encountering stagnation in the search progress. By integrating these operators with their respective updating mechanisms, we have significantly enhanced the applicability of the rat swarm algorithm to address VRPTW. Furthermore, we utilized an encoding scheme with a length of N to represent a rat's position, where N corresponds to $n + k - 1$.

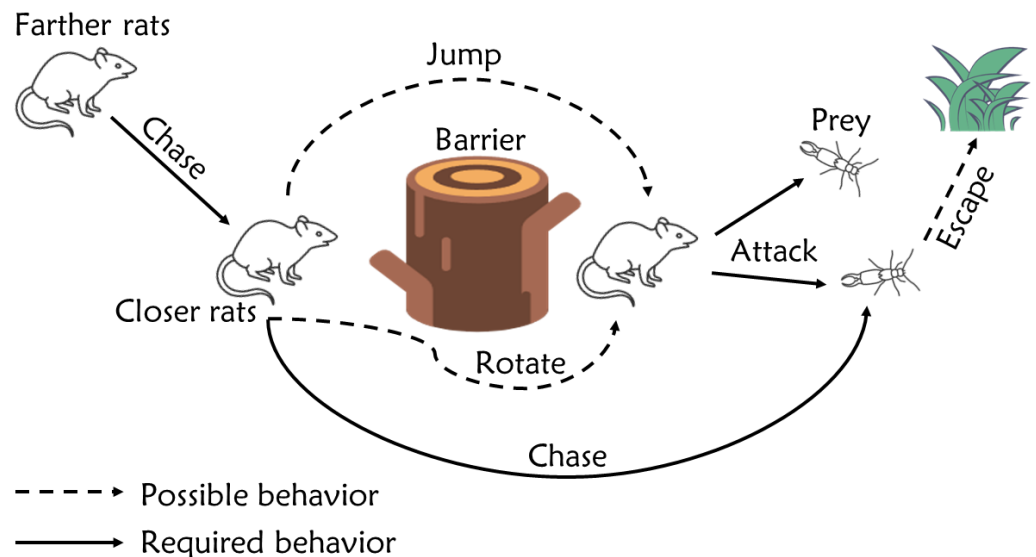


Figure 3. Schematic diagram of MRSO.

4.2.1. Chase Operator

As social animals, rats frequently engage in collaborative hunting behaviors through group interaction. To quantitatively assess this behavior, we hypothesized that the position of the most genetically superior individual within the population serves as an indicator of the prey's location, with the remaining individuals adjusting their positions accordingly. This mathematical representation is formulated as follows:

$$\vec{P} = A \cdot \vec{P}_i(t) + C \cdot (\vec{P}_r(t) - \vec{P}_i(t)) \quad (13)$$

In this equation, $\vec{P}_i$ represents the position of an individual in the rat swarm, and $\vec{P}_r$ represents the position of the current optimal individual. Parameters A and C control exploration and exploitation behaviors, respectively. Here, $A = R - R \cdot (t/T_{max})$ with R randomly chosen from $[1, 2]$ and C randomly chosen from $[0, 2]$. The iteration count t ranges from 0 to T_{max} , representing the search process of the algorithm.

The balance between exploration and exploitation is crucial in evaluating the performance of optimization algorithms. Exploration aims to explore diverse regions of the search space in order to discover potential optimal solutions, while exploitation focuses on precise searching within previously identified promising areas. MRSO achieves this equilibrium by dynamically adjusting the exploration parameter A and the exploitation

parameter C . In the original algorithm, the exploration coefficient A gradually decreases with increasing iterations, resulting in a narrowing of the exploration domain. To enhance the algorithm's performance, we propose a novel expression of the exploration coefficient.

$$A = R \cdot (\cos(\pi \cdot t / T_{max}) + 1) \quad (14)$$

The value of R is randomly selected from the interval $[0, 1]$. The newly introduced coefficient exhibits a slower rate of decrease during the initial stages, thereby expanding the exploration range and augmenting global search capabilities. However, it undergoes a faster reduction in later stages to expedite convergence.

When a rat is close to the prey, it can directly chase it; when the distance is greater, it may follow other rats for a coordinated chase. Drawing inspiration from the crossover operator in genetic algorithms, Algorithm 2 designs both global and local chase operators. By simulating the collaborative behavior of rats during hunting, it dynamically selects chase targets (which may be other rats or the prey) and updates the current rat's position based on the chosen target.

Algorithm 2 Chase operator

Input: Current rat position P_i , predator position P , iteration count t , maximum iteration count T_{max}

Output: Updated rat position P'_i

```

1:  $r_1, r_2 \leftarrow$  Randomly select two distinct numbers from  $\{1, 2, \dots, n\}$  with  $r_1 < r_2$ ;
2:  $C \leftarrow 2 \cdot \text{rand}([0, 1])$ ;
3:  $R \leftarrow \text{rand}([0, 1])$ ;
4:  $A \leftarrow R \cdot (\cos(\pi \cdot t / T_{max}) + 1)$ ;
5: if  $C \leq A$  then
6:    $j \leftarrow i + 1$ ;
7:    $P_r \leftarrow [P_j(r_1), P_j(r_1 + 1), \dots, P_j(r_2)]$ ;
8: else
9:    $P_r \leftarrow [P(r_1), P(r_1 + 1), \dots, P(r_2)]$ ;
10: end if
11:  $P_i \leftarrow P_i + P_r$ ;
12: for  $k \leftarrow 1$  to  $N$  do
13:   if  $P_i(k) \in P_r$  then
14:     Tag the element  $P_i(k)$ ;
15:   end if
16: end for
17:  $P'_i \leftarrow P_i$  with the tagged elements removed;
18: return  $P'_i$ ;

```

4.2.2. Jump and Rotate Operators

In our comprehensive investigation of cooperative prey chase behavior among rats, the presence of obstacles is a pivotal aspect. These obstacles may manifest as physical barriers or environmental hazards. Upon encountering such obstacles, rats leverage their acute environmental perception and exceptional motor abilities to promptly and precisely determine whether to leap over or circumvent them. This decision-making process is deliberate, grounded in a profound comprehension of environmental shifts and a meticulous evaluation of locomotor capabilities. In this study, we introduce q_b to represent the likelihood of rats encountering obstacles during the chase, a likelihood that is shaped by variables such as obstacle density, rat locomotion velocity, and environmental intricacy. When confronted with obstacles, rats activate avoidance mechanisms, engaging in intricate calculations and strategic appraisals to select the most optimal avoidance strategy. These strategies are aimed at maximizing the prey chase success rate while minimizing the cost associated with obstacle avoidance.

To accurately simulate the process, this study introduces two obstacle avoidance strategies: the jump operator and the rotate operator. The jump operator, drawing inspiration from mutation operations in genetic algorithms, enables rats to leap over obstacles. However, in VRPTW, direct jumping can render solutions infeasible due to time window

constraints. Therefore, we specifically designed the jump operator to maintain path rationality and enhance solution diversity by randomly swapping the positions of customers with strict time window requirements. If we take Figure 4 as an example, customers 2, 4, and 3 are the initial customers on their respective routes, and their left time windows are narrower than those of other customers, facilitating the satisfaction of time window constraints through pairwise swaps. Figure 4 illustrates that swapping customer 4 with customer 3 results in a green code, swapping customer 2 with customer 4 leads to a red code, and swapping customer 2 with customer 3 produces a blue code. These three codes represent potential outcomes achieved through the jump operator. The functionality of the jump operator is illustrated in Figure 5.

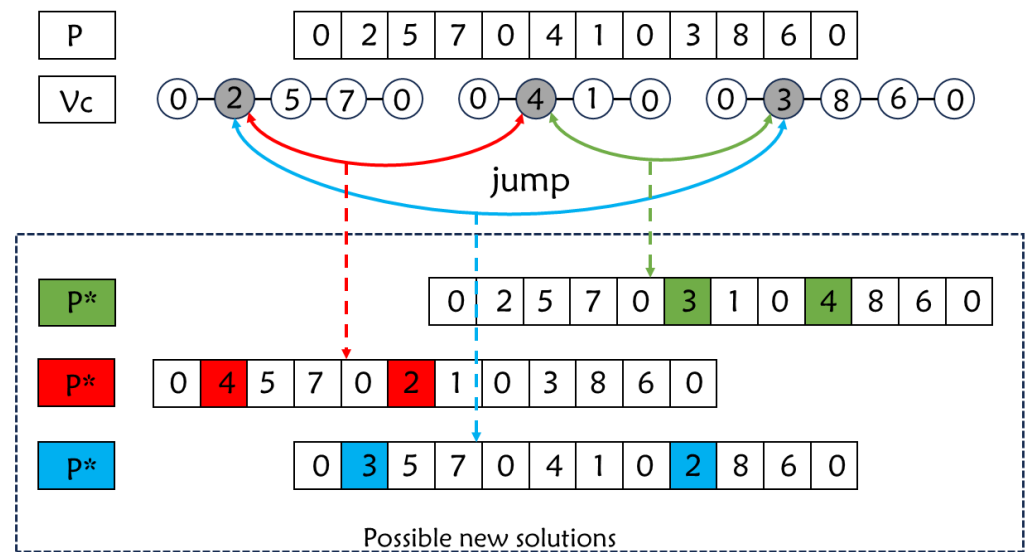


Figure 4. Jump operator diagram.

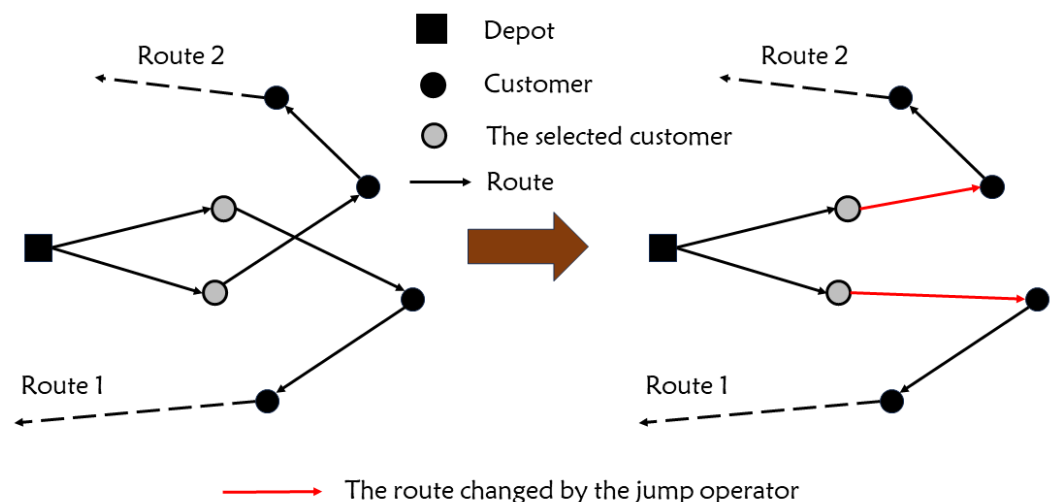


Figure 5. Working principle of jump operator.

The rotate operator, a sophisticated obstacle avoidance technique, incorporates rotation and time window adjustment steps. It achieves obstacle circumvention by reassigning customers among vehicles, while ensuring that post-transfer routes adhere to time window requirements. The challenge lies in identifying the optimal customer transfer scheme that satisfies time window constraints without escalating vehicle usage, crucial for minimizing the number of vehicles utilized. Figure 6 illustrates a unit rotation from P to $P1$, where customer 10 is inserted into the route $[9, 2, 5, 7]$ to form $[10, 9, 2, 5, 7]$. However, this new route

may violate time window constraints, necessitating adjustments. Transitioning from $P1$ to $P2$ involves repositioning customer 10 between customers 5 and 7 to meet the condition $e_5 < e_{10} \leq e_7$. Sequential adjustments can be made when rotating multiple units.

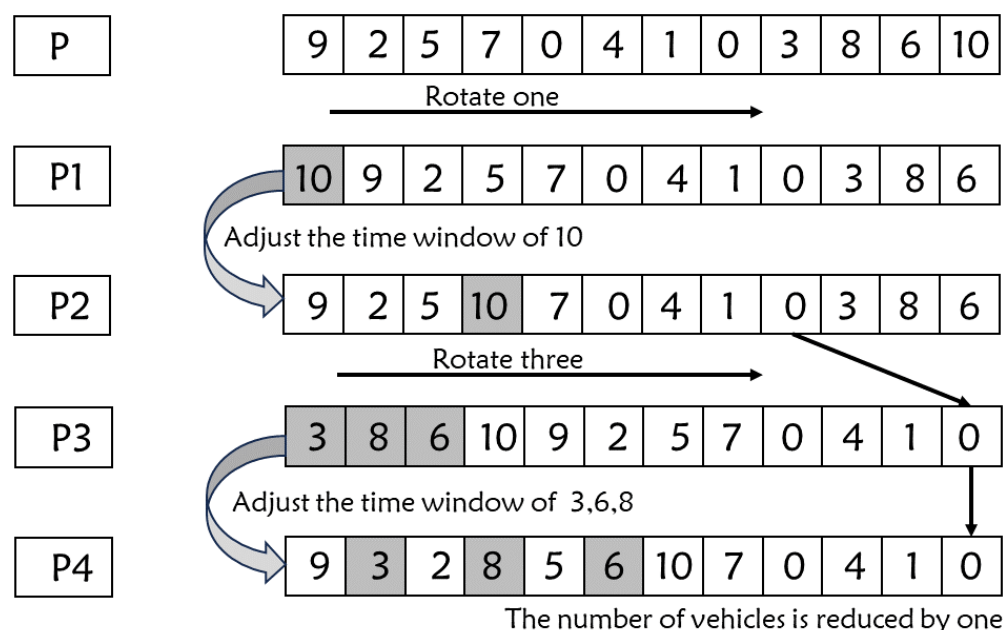


Figure 6. Rotate operator diagram.

In selecting obstacle avoidance strategies, we presume that rats possess high intelligence and can assess the relative benefits of different strategies. By computing the fitness values of these strategies, we choose the obstacle avoidance operator with the lowest fitness as the basis for position updates. This simulates the intelligent strategic decisions made by rats in their natural habitat. In summary, the jump and rotate operators designed in this paper effectively mimic the obstacle avoidance behavior exhibited by rats during prey pursuit. By flexibly applying these strategies, rats can effectively respond to obstacles and enhance the success rate of their chases. Additionally, the strategy design accounts for time window constraints and the objective of minimizing vehicle usage in VRPTW, enhancing the realism of simulation results.

4.2.3. Attack Operator

The capture of prey by rats involves relentless assaults until the prey succumbs, echoing the process of seeking optimal solutions in optimization problems through relentless attempts and adjustments aimed at achieving the best outcomes. Drawing inspiration from the variable neighborhood search (VNS) algorithm, this study introduces an innovative attack operator into the algorithm design. This operator aims to mimic the attack behavior during prey capture, thereby approximating the optimal solution to the optimization problem. Its mechanism lies in systematically adjusting the set of neighborhood structures, seeking not only local optimal solutions within the current neighborhood but also gradually expanding the search scope to explore a broader solution space, alternately yielding multiple local optimal solutions. Notably, the attack operator incorporates a feedback mechanism. In cases where the search results fail to meet the expected optimization criteria, the algorithm retains the original solution, preventing descent into inferior solution spaces. This feedback mechanism ensures the algorithm's robustness and reliability, thereby enhancing solution-finding efficiency. To generate neighborhood solutions, this paper employs the $2 - opt$ operator [29]. As depicted in Figure 7, this operator constructs new feasible solutions by eliminating two edges from the original feasible solution and subsequently reintroducing two edges. This process maintains solution feasibility while

enhancing solution diversity, thus strongly supporting the attack operator in discovering optimal solutions within a broadened solution space.

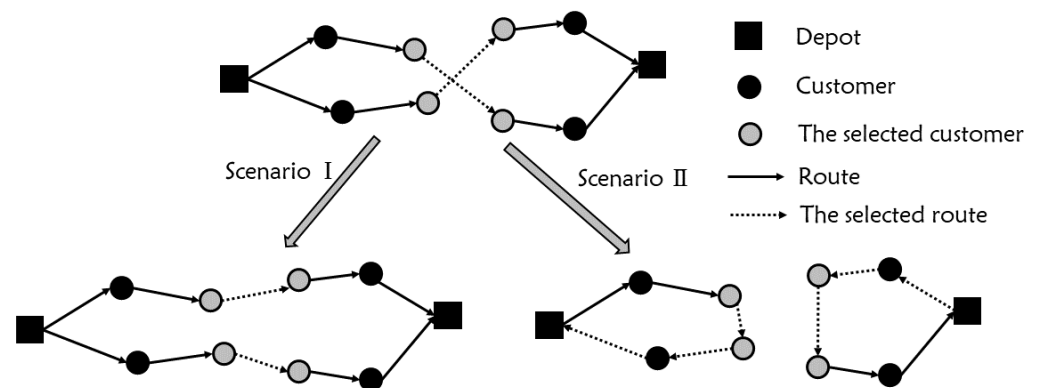


Figure 7. Working principle of attack operator.

4.2.4. Escape Operator

Although the rat swarm may not always successfully capture prey, instances of prey escaping are infrequent but nonetheless existent. To mimic this natural behavior and enhance the algorithm's global search capabilities, an escape operator has been devised. This operator utilizes the escape probability q_e as a threshold to determine whether to update the prey's position, allowing for a certain probability of a reduction in solution quality, thus enhancing the likelihood of escaping local optima. The design of this escape operator is rooted in the theory of congruence in number theory. While Algorithm 3 exhibits strong randomness, effectively facilitating the algorithm's evasion of local optima, it may also introduce newly generated solutions that violate the constraints of VRPTW. Consequently, the selection of the escape factor p is paramount. To address this challenge, we introduce the concept of a local escape operator, limiting escape updates to a subset of customers (specifically, the initial few customers based on their identification numbers). Additionally, to safeguard the solution's validity, we employ multiple escape factors p for verification. Given these considerations, the escape probability is set to a relatively low value in this study.

Algorithm 3 Escape operator

Input: Current prey position P , escape factor q_e

Output: Updated prey position P'

```

1:  $r \leftarrow$  Random value from  $[0, 1]$ ;
2: if  $r \leq q_e$  then
3:   Select a random positive integer  $p$  coprime to  $n$ ;
4:   for  $i \leftarrow 1$  to  $N$  do
5:      $P(i) \leftarrow (P(i) \cdot p) \bmod n$ ;
6:   end for
7: end if
8:  $P' \leftarrow P$ 
9: return  $P'$ ;

```

4.3. Large Neighborhood Search

LNS, a neighborhood search algorithm introduced by Shaw in 1998, exhibits structural similarities to other neighborhood search methods [30]. It initiates with the generation of an initial solution and iteratively enhances the solution's quality until satisfaction is achieved. The iterative process comprises two key steps: dismantling the current solution, often by removing certain points from the path (remove operator), followed by reconstructing the dismantled solution, typically by reinserting the removed points (reinsert

operator). Distinguishing LNS from traditional algorithms is its broadened search scope per iteration, facilitating the discovery of higher-quality local optimal solutions. Nevertheless, this broadened scope also translates into longer iteration durations. In scenarios involving numerous constraints, LNS demonstrates superior performance in generating viable neighborhood solutions. Furthermore, its utilization of depot–customer pairs as the fundamental operational unit renders it particularly apt for addressing the model underpinning this paper.

Reviewing related research, Sener Akpinar et al. proposed a hybrid algorithm combining ant colony optimization (ACO) with LNS in 2016 to solve the capacitated vehicle routing problem [3]. This hybrid approach enhanced the diversity of solutions through ACO's solution construction mechanism, thereby improving the performance of LNS. Elhassania et al. (2013) also combined ACO with LNS to address dynamic vehicle routing problems [31]. They introduced LNS into ACO to enhance its performance. Unlike Sener Akpinar, their hybrid algorithm focused on improving the quality of individual solutions and functioned as a parallel-type hybrid, where LNS operators leveraged ACO's solution construction mechanism. In contrast, the hybrid algorithm proposed in this paper aims to complement and enhance LNS with MRSO. It initially generates solutions using MRSO, then improves these solutions with LNS, and feeds the newly generated solutions back to MRSO. This characteristic makes LNS-MRSO a population-based search strategy and a sequential-type hybrid algorithm. Given the distinct search strategies of LNS and MRSO, their combination exhibits strong capabilities in both global and local searches.

4.3.1. Remove Operator

The symbol σ represents the current solution, while v stands for the customer designated for removal. The set of already removed customers is denoted by S , and TR signifies the total number of customers to be eliminated. The partial solution resulting from the removal of TR customers is represented by σ' . The removal process is detailed in Algorithm 4. Initially, a customer is randomly chosen from σ and appended to S . Subsequently, for the remaining $TR - 1$ iterations, a customer u is randomly selected from S in each round. Subsequently, the remaining customers in σ are sorted in ascending order based on their relevance to u . The customer v with the highest relevance is then chosen for removal and added to S . This process is reiterated until all $TR - 1$ customers have been selected. Relevance is defined as follows:

$$Cov(i, j) = d_{ij} + \alpha h_{ij} + \beta t_{ij} \quad (15)$$

d_{ij} represents the normalized Euclidean distance between customers i and j . The variable h_{ij} signifies whether customers i and j are serviced by the same vehicle, with a value of 0 indicating they are on the same route and 1 indicating they are on different routes. Additionally, t_{ij} denotes the degree of time window conflict. More specifically, if the vehicle traversing from customer i to j is able to adhere to the designated time window l_j of customer j , the departure time t from customer i must comply with the condition $t \leq l_j - s_j - c_{ij}$, where

$$t_{ij} = \begin{cases} 0, & l_i \leq l_j - s_j - c_{ij} \\ \frac{l_i + s_j + d_{ij} - l_j}{l_i - e_i}, & e_i \leq l_j - s_j - c_{ij} < l_i \\ 1, & l_j - s_j - c_{ij} < e_i \end{cases} \quad (16)$$

In summary, customers with similar geographical locations, on the same route, and with minimal time window conflicts have high relevance. The importance of each factor can be adjusted by modifying α and β . In this paper, $[\alpha, \beta]$ is set to $[3, 2]$. However, there is no perfect relevance function in reality. Over-reliance on it can lead to local optimality.

To avoid this, Shaw introduced a random element D , which selects a customer from the relevance-sorted list [30]. Specifically,

$$index = \lfloor (Rand([0, 1]))^D \cdot |\sigma| \rfloor \quad (17)$$

Subsequently, the remaining customers are arranged in descending order of relevance, forming the *List*. The customer designated for removal corresponds to $List[index]$. As the value of D increases from 1, where removal is randomized, towards positive infinity, the selection process increasingly favors the customer with the highest relevance. Given the robust relevance function introduced in this study, strictly adhering to the aforementioned method would undermine its efficacy. Consequently, this paper introduces a novel removal technique that harmonizes randomness with number theoretic principles. This approach minimizes the impact of D while preserving a degree of randomness, confining the search to the K customers exhibiting the highest relevance.

$$index \equiv Rand([0, K - 1]) \pmod{K} \quad (18)$$

The remaining customers are sorted in descending order of relevance to obtain *List*, and the customer to be removed is $List[index]$. Considering the decreasing number of remaining customers, K can be dynamically set, such as 2, 3, or 5.

Algorithm 4 Remove operator

Input: Delivery plan *Plan*, number of customers to remove TR , random number K

Output: Set of removed customers *Removed*

```

1:  $D \leftarrow [2, 3, 5]$ ;
2: Randomly select a customer  $v$ ;
3:  $Removed \leftarrow Removed + \{v\}$ ;
4:  $Plan \leftarrow Plan - \{v\}$ ;
5: while  $|Removed| < TR$  do
6:   Randomly select a customer  $u$  from Removed;
7:    $List \leftarrow sort(Cov(u, v), v \in Plan)$ ;
8:    $K \leftarrow rand(D(1), D(2), D(3))$ ;
9:    $index \leftarrow |Plan| \pmod{K}$ ;
10:   $Removed \leftarrow Removed + \{List(index)\}$ ;
11:   $Plan \leftarrow Plan - \{List(index)\}$ ;
12: end while
13: return Removed;

```

4.3.2. Reinsert Operator

The reinsertion process aims to enhance the solution quality by reinstating the removed customer set S into the partial solution σ' . According to Algorithm 5, this process initiates by pinpointing the optimal insertion location for each customer in S . This involves selecting the candidate position that minimizes the increment in the objective function value, which represents the least costly insertion point, and documenting the corresponding objective function values. Subsequently, the incremental objective function values are computed for each customer upon their insertion into their respective optimal positions. The customer exhibiting the largest increment is then selected as the primary candidate for reinsertion. This approach is known as the farthest insertion heuristic [30]. The process continues iteratively until all customers in S have been successfully reinstated into the partial solution σ' . The process is more time-consuming on a broader scope.

Algorithm 5 Reinsert operator**Input:** Modified delivery plan $Plan$, removed customer set $Removed$ **Output:** New delivery route $NewPlan$

```

1: while  $|Plan| < n$  do
2:   for all  $v \in Removed$  do
3:      $N \leftarrow |Plan|$ ;
4:      $\Delta x(v) \leftarrow$  a sufficiently large value;
5:     for  $j \leftarrow 0$  to  $N$  do
6:        $Plan_v \leftarrow [Plan(1), Plan(2), \dots, Plan(j), v, Plan(j+1), \dots, Plan(N)]$ , where
        $Plan(0)$  and  $Plan(N+1)$  are empty;
7:       if  $Plan_v$  satisfies constraints then
8:          $\Delta x(v) \leftarrow \min(f(Plan_v) - f(Plan), \Delta x(v))$ ;
9:       end if
10:    end for
11:  end for
12:   $u \leftarrow$  the customer that maximizes  $\Delta x(v)$ ;
13:   $Removed \leftarrow Removed - \{u\}$ ;
14:   $Plan \leftarrow Plan + \{u\}$ ;
15: end while
16:  $NewPlan \leftarrow Plan$ ;
17: return  $NewPlan$ ;

```

4.3.3. Simulated Annealing Mechanism

In determining the acceptance of newly generated neighborhood solutions, this study employed a simulated annealing mechanism as the criterion, aiming to circumvent the entrapment in local optima. Concurrently, by leveraging MRSO's solution construction mechanism, we foster diversity within the solution set, thus enhancing the large neighborhood search algorithm's performance and intricacy of its local search. The specific process can be found in Algorithm 6.

During the algorithm's iterative process, if the total cost of the new solution is inferior to that of the current optimal solution, the optimal solution is updated with the new solution's path set. If the cost of the new solution surpasses the global optimal but remains below the current feasible solution, the current feasible solution is replaced with the new solution. For instances where the new solution's cost exceeds the current feasible solution, adhering to the simulated annealing paradigm, the new solution retains a likelihood of being accepted as the current feasible solution. This likelihood is modulated by adjusting the temperature parameter T . Specifically, when $f_n < f_c$, the probability p is computed as follows [32]:

$$p = e^{\left(\frac{100}{T} \left(\frac{f_n - f_c}{f_c}\right)\right)} \quad (19)$$

$$f(x) = c(x) + \mu q(x) + \nu w(x) \quad (20)$$

Herein, $f(x)$ signifies the objective function value, encompassing the fitness measure, $c(x)$ signifies the cumulative distance traversed by the vehicles, $q(x)$ represents the aggregate of capacity constraint violations across various routes, and $w(x)$ represents the aggregate of time window constraint violations pertaining to all customers. Given the relative difficulty of violating capacity constraints, μ is assigned a value of 1; conversely, as violating time window constraints is less challenging, ν is assigned a value of 100. In the proposed algorithm, T denotes the temperature parameter, which decreases progressively as iterations ensue, thereby rendering the acceptance criteria increasingly rigorous. After each iteration, the value of T is revised via $T = T \cdot \tau$, where τ represents the cooling rate, adhering to the condition $0 < \tau < 1$. Whenever a novel optimal solution is uncovered, T is reinstated to its initial value, T_0 . Herein, T_0 is established to accommodate a neighborhood solution that is inferior to the current solution by $\eta\%$ with a 50% likelihood. To guarantee a gradual

annealing process, the initial temperature is determined utilizing the approach advocated by Ropke [33].

$$T_0 = -\frac{\eta\%}{\ln 0.5} f(x) \quad (21)$$

Algorithm 6 Simulated annealing mechanism

Input: Current solution P , maximum iterations T_{max}

Output: Optimal solution P_{best}

```

1: Initialize temperature  $T$  as  $T_0$ ;
2: for  $t = 1$  to  $T_{max}$  do
3:    $P' \leftarrow P$ ;
4:   Apply remove operator to  $P'$ ;
5:   Apply reinsert operator to  $P'$ ;
6:   if  $F(P') < F(P)$  then
7:      $P \leftarrow P'$ ;
8:   else if  $F(P') \geq F(P)$  then
9:      $r \leftarrow \text{rand}(0, 1)$ ;
10:     $p \leftarrow e^{\left(\frac{100}{T} \times \left(\frac{F(P) - F(P')}{F(P')}\right)\right)}$ ;
11:    if  $r < \min(1, p)$  then
12:       $P \leftarrow P'$ ;
13:      Update temperature  $T$  as  $T \cdot \tau$ ;
14:      Reset temperature  $T$  to  $T_0$ ;
15:    else
16:       $P \leftarrow \text{MRSO}(P_{best})$ ; // Generate a new solution using MRSO
17:    end if
18:  end if
19:  if  $F(P) < F(P_{best})$  then
20:     $P_{best} \leftarrow P$ ;
21:  end if
22: end for
23: return  $P_{best}$ ;

```

4.4. Time Complexity Analysis

In this section, we delve into the overall time complexity analysis of the LNS-MRSO algorithm. Assuming a total of M rats, a dimension of N , and a maximum number of iterations I , the LNS-MRSO algorithm comprises multiple components, including greedy strategy-based population initialization, chase operator, jump operator, rotate operator, attack operator, escape operator, and large neighborhood search. Specifically, the time complexity of each component is as follows: greedy strategy-based population initialization is $O(N^2 * M)$, chase operator is $O(N^2 * M)$, jump operator is $O(N * M)$, rotate operator is $O(N^2 * M)$, attack operator is $O(N^2 * M)$, escape operator is $O(N * M)$, and large neighborhood search is $O(TR * N * \log N * M + TR * N^2 * M)$. By integrating these operators, the time complexity of a single iteration of the LNS-MRSO algorithm is $O(N^2 * M + N * M + TR * N * \log N * M + TR * N^2 * M)$. When the number of iterations is large, the overall time complexity can be approximated as $O(I * TR * N^2 * M)$. Notably, in solving a VRPTW problem, the time complexity performs particularly well when the scale of customers is small.

5. Algorithm Performance Experiment

In this section, we delve into the relevant research contents of the LNS-MRSO algorithm. Specifically, Section 5.1 provides an in-depth analysis of the data distribution characteristics of the Solomon dataset; Section 5.2 explores in detail the impact of the TR value on the performance tuning of the algorithm; Section 5.3 compares the performance of the original LNS algorithm with the proposed LNS-MRSO algorithm; Section 5.4 presents

a thorough comparative analysis of the differences between the LNS-MRSO algorithm and the best-known solutions (BKSs); finally, Section 5.5 comprehensively compares the performance of the LNS-MRSO algorithm with sixteen other advanced algorithms.

5.1. Setup

Our algorithms were implemented with matlab. All the experiments were conducted on a machine with 11th Gen Intel(R) Core(TM) i5-11400H clocked at 2.70 GHz and 16.0 GB RAM. We conducted a comprehensive performance evaluation of LNS-MRSO using the 56 benchmark instances established by Solomon. These instances cover various problem categories, including C1, C2, R1, R2, RC1, and RC2 within the Solomon standard dataset. The instances are meticulously classified based on node distribution patterns into clustered (C-class), random (R-class), and mixed (RC-class) categories. In the “1”-type problems, nodes have narrower time windows and strict vehicle capacity constraints. Conversely, the “2”-type problems has more lenient time windows and relaxed vehicle capacity constraints. Due to its extensive scenario diversity and representativeness, the Solomon dataset effectively simulates various real-world scheduling challenges and serves as a crucial benchmark for assessing algorithm performance.

To configure the experimental setup of LNS-MRSO, we meticulously tuned a set of crucial parameters to guarantee efficient algorithm execution. More specifically, we adjusted the obstacle encounter probability q_b to 0.5, striking a balance between exploration and exploitation. Concurrently, we fixed the prey escape probability q_e at 0.02 to avert premature convergence to local minima. Furthermore, we optimized the search efficacy and solution excellence by assigning the cost factors α and β in the remove operator to 3 and 2, respectively. In our experiments, we deployed 100 rats as search agents and capped the maximum iterations at 1000, affording ample opportunities for the algorithm to discover superior solutions. Each experiment was replicated independently 10 times. These parameter configurations, informed by prior experimental insights and algorithm traits, strive to boost the performance and robustness of LNS-MRSO in tackling intricate optimization challenges.

5.2. Parameter TR Tuning

The TR value significantly impacts the large neighborhood search, thereby directly affecting the overall performance of LNS-MRSO. When the TR value is too small, it restricts the search scope and may cause the algorithm to become trapped in local optimal solutions. Conversely, a larger TR value enhances search capabilities but also increases computational complexity, resulting in prolonged search times. Therefore, selecting an appropriate TR value is crucial for achieving a balance between search effectiveness and efficiency.

In this study, we investigated the selection of appropriate TR values by utilizing three RC101 instances of varying scales as illustrative examples. Our experimental design set the upper bound of the TR value as $TR < 3\sqrt{M}$ and partitioned it with a step size of 5, with particular emphasis on the $TR = 2$ value. To ensure consistency in our experimental findings, we conducted each test ten times for each TR value. Our key evaluation metrics include average number of vehicles (ANV), average travel distance (ATD), best number of vehicles (BNV), best travel distance (BTD), time per iteration (Times), average iteration upon convergence (AITER), and frequency of achieving known optimal solutions (FR).

In Figure 8, the iteration charts of RC101-25, RC101-50, and RC101-100 reveal that the LNS-MRSO algorithm exhibits sluggish and unstable convergence speeds when TR is set to 2 or 5. Conversely, increasing the TR value, particularly to 20 or 25, results in a more rapid convergence rate. However, it should be noted that higher TR values pose challenges for the algorithm to avoid local optima as iterations increase. Therefore, selecting an appropriate TR value is crucial for maximizing the performance of the LNS-MRSO algorithm. Similarly, this trend can also be observed in the iteration charts of RC201-25, RC201-50, and RC201-100.

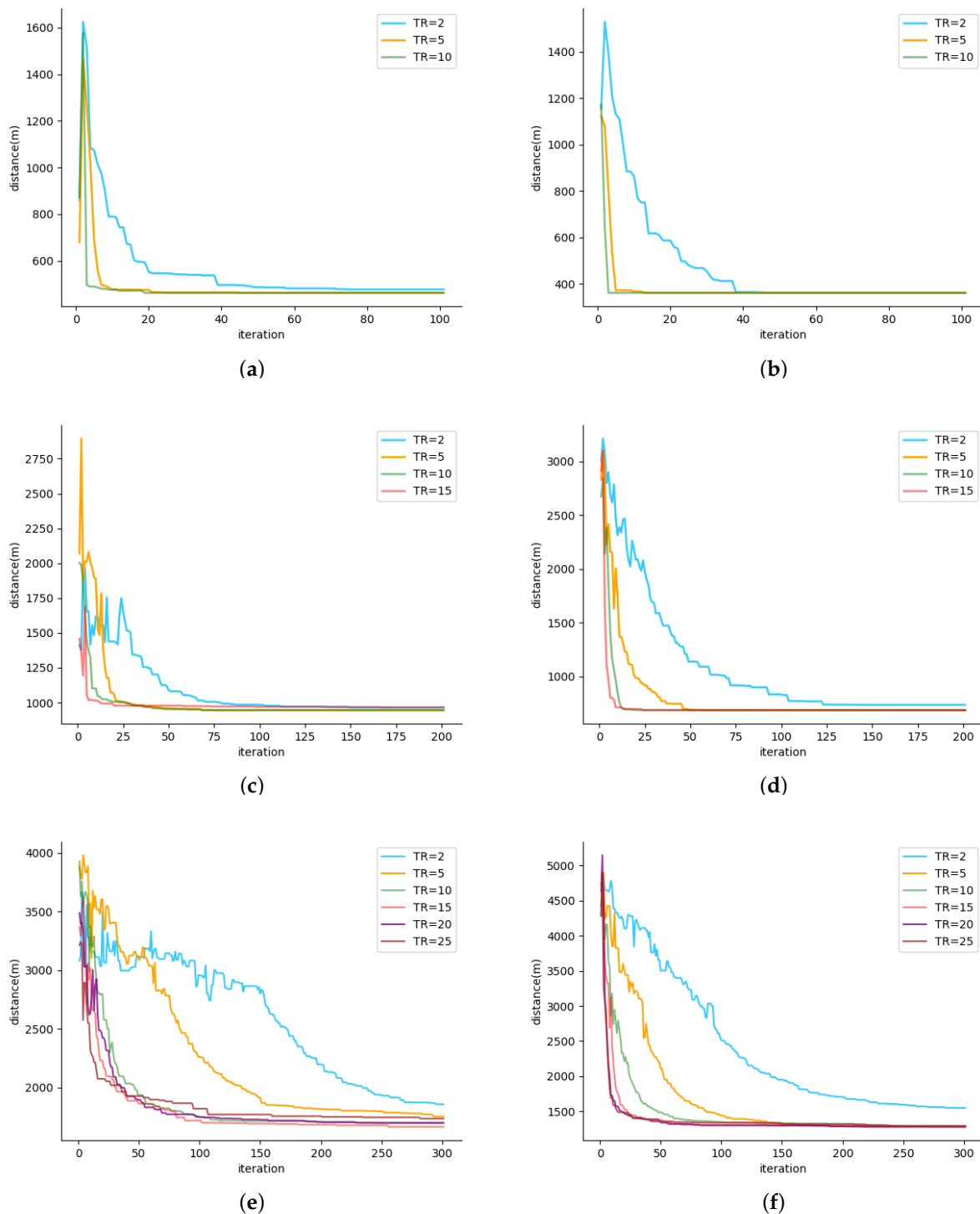


Figure 8. Iterative convergence graph of RC101 and RC201 with different TR values. (a) Convergence process of RC101-25. (b) Convergence process of RC201-25. (c) Convergence process of RC101-50. (d) Convergence process of RC201-50. (e) Convergence process of RC101-100. (f) Convergence process of RC201-100.

Upon meticulous examination of the experimental data presented in Table 2, we deduce the following observations: Firstly, the algorithm's solution time exhibits a non-linear increase with escalating TR values, necessitating longer computation durations for larger problem scales under identical TR settings. Secondly, the selection of the TR value

critically influences the algorithm's convergence capacity, as excessively high or low TR values can impede convergence or search proficiency, potentially hindering the algorithm's ability to achieve optimal solutions. Lastly, an optimally chosen TR value confers robust search capabilities and stability to the algorithm, facilitating its efficient convergence to optimal solutions within a reduced number of iterations. Through rigorous experimental validations, we have derived a general principle that can be formulated as follows:

$$TR = \lfloor 5 \log_2 M - c \rfloor \quad (22)$$

The formula utilizes M to represent the scale of customers, while c represents a constant with an approximate value of 18.22. This constant is derived from the calculation $5(\log_2 25 - 1)$. By utilizing this formula, we can determine suitable TR (threshold or performance metric) values for different instance scales. In our research, we have identified 5, 10, and 15 as appropriate TR values for three distinct instance scales. These selections not only ensure satisfactory algorithm performance but also effectively regulate computational complexity, thereby optimizing overall solution efficiency.

Table 2. Results on RC101 with different TR values.

	RF	ANV	ATD	Time (s)	AITER	FR	BNV	BTD
RC101-25	2	4	476.87	0.10	83	70%	4	462.16
	5	4	462.16	0.21	33	100%		
	10	4	462.16	0.47	18	100%		
RC101-50	2	10	964.63	0.18	-	0%	8	945.6
	5	8	948.55	0.47	92	80%		
	10	8	945.88	1.09	68	90%		
	15	8	965.69	1.94	156	10%		
RC101-100	2	19.4	1857.82	0.24	-	0%	15	1663.36
	5	16.9	1753.49	1.89	-	0%		
	10	15.3	1696.93	2.74	296	10%		
	15	15.2	1663.36	5.25	258	90%		
	20	15.8	1695.57	7.71	-	0%		
	25	15	1699.41	11.06	-	0%		

5.3. Comparison with LNS-Based Algorithms

The present study conducted a comparative analysis between the original LNS algorithm and the enhanced LNS-MRSO algorithm, utilizing 56 test instances from the Solomon benchmark. Our objective was to validate the superior performance of the proposed LNS-MRSO algorithm. Table 3 presents comprehensive experimental data, where "NV" denotes the number of vehicles utilized, "TD" represents the cumulative travel distance of the vehicles, and "Gap" quantifies the percentage of improvement in TD achieved by LNS-MRSO compared to LNS. TD1 denotes the total distance covered by the vehicle in the LNS algorithm, whereas TD2 signifies the total distance traveled in the LNS-MRSO algorithm.

$$Gap = \frac{TD1 - TD2}{TD2} \times 100 \quad (23)$$

The data presented in Table 3 unambiguously demonstrate that both TD and NV exhibit the superiority of the LNS-MRSO algorithm over its counterpart, LNS. Specifically, with regards to TD, nine test instances exhibit improvements exceeding 5%, with notable enhancements observed in C104 (9.25%), C204 (8.29%), and RC101 (7.77%). Only three instances show slightly inferior performance compared to LNS. In terms of NV, LNS-MRSO

outperforms LNS in seventeen test instances while trailing slightly behind in only six cases. Overall, across multiple key performance indicators, including but not limited to TD and NV metrics, it can be effectively concluded that our proposed algorithm LNS-MRSO exhibits superior performance, validating its efficacy.

Table 3. Comparison with LNS-based algorithms.

LNS			LNS-MRSO			LNS			LNS-MRSO		
Instances	NV	TD	NV	TD	Gap	Instances	NV	TD	NV	TD	Gap
C101	10	828.94	10	828.94	0.00	R112	11	1037.84	11	984.84	5.38
C102	10	828.94	10	828.94	0.00	R201	8	1189.11	7	1183.90	0.44
C103	10	849.06	10	828.06	2.47	R202	6	1066.65	6	1044.45	2.13
C104	10	933.49	10	847.15	9.25	R203	5	901.65	6	900.20	0.16
C105	10	828.94	10	828.94	0.00	R204	5	778.37	5	775.59	0.36
C106	10	828.94	10	828.94	0.00	R205	6	1002.92	5	962.02	4.25
C107	10	828.94	10	828.94	0.00	R206	7	938.15	4	916.30	2.38
C108	10	828.94	10	828.94	0.00	R207	5	816.33	5	832.16	−1.90
C109	10	828.94	10	828.94	0.00	R208	5	720.07	4	721.04	−0.13
C201	3	591.56	3	591.56	0.00	R209	7	905.97	5	875.95	3.43
C202	4	629.31	3	591.56	6.00	R210	7	942.80	5	925.47	1.87
C203	4	620.30	3	591.17	4.70	R211	5	794.09	5	783.68	1.32
C204	4	643.95	3	590.60	8.29	RC101	18	1792.60	15	1663.36	7.77
C205	3	588.88	3	588.88	0.00	RC102	15	1602.35	14	1498.21	6.95
C206	4	612.54	3	588.49	3.93	RC103	13	1368.54	12	1346.00	1.67
C207	3	588.29	3	588.29	0.00	RC104	11	1197.65	11	1186.50	0.94
C208	3	588.32	3	588.32	0.00	RC105	16	1614.31	16	1596.70	1.10
R101	20	1667.35	19	1659.34	0.48	RC106	14	1498.23	13	1408.83	6.35
R102	18	1484.81	18	1476.50	0.56	RC107	14	1401.16	12	1335.05	4.95
R103	15	1261.01	14	1240.97	1.59	RC108	12	1238.83	12	1227.24	0.94
R104	11	1033.68	12	1033.52	0.02	RC201	8	1329.39	8	1285.08	3.45
R105	15	1407.73	15	1405.18	0.18	RC202	7	1111.37	8	1106.84	0.41
R106	13	1287.07	14	1270.87	1.26	RC203	5	939.21	6	931.45	0.83
R107	12	1125.64	12	1122.96	0.24	RC204	4	807.44	4	824.79	−2.10
R108	11	994.43	11	986.20	0.83	RC205	7	1180.12	7	1180.12	0.00
R109	13	1227.42	12	1207.18	1.65	RC206	6	1080.40	7	1072.29	0.76
R110	13	1188.11	12	1128.21	5.04	RC207	6	1013.87	6	977.06	3.77
R111	12	1131.88	12	1062.26	6.15	RC208	6	833.58	5	805.21	3.52

5.4. Comparison with Best-Known

In this section, we conducted two experiments to comprehensively assess the effectiveness of LNS-MRSO in comparison with BKSs across varying problem sizes. The experimental outcomes are summarized in Table 4. In the table, “NV” and “TD” stand for the number of vehicles and the total travel distance, respectively. “Gap = $100 \times (TD - \text{BTD})/\text{BTD}$ ” signifies the percentage deviation from the optimal solution, and “Ave” represents the average value for a specific category of instances.

When dealing with 25 customers, although LNS-MRSO slightly lags behind BKS [34], the margin of error is restricted to 0.5%, highlighting the algorithm’s precision in addressing small-scale challenges. However, no algorithm has been developed to date that can solve to optimality all VRPTW instances; LNS-MRSO is not an exception; this is known as the NFL (No Free Lunch) theory [35]. As the number of customers rises to 50, LNS-MRSO demonstrates superior performance in R1, R2, and RC1 scenarios. More specifically, it surpasses BKS in 5, 6, and 7 instances, respectively, in terms of total travel distance. Additionally, R2, RC1, and RC2 show advantageous outcomes compared to BKS in terms of average travel distance. Furthermore, LNS-MRSO exhibits commendable performance across other datasets, maintaining a deviation within 2%. Notably, 20 instances achieve solutions superior to the known optimal ones, with 11 instances achieving optimizations in

both vehicle usage and total travel distance. This unequivocally validates the effectiveness of the LNS-MRSO algorithm, particularly in optimizing the total travel distance of vehicles.

To further corroborate the performance of LNS-MRSO, we undertook a comprehensive comparison with established optimal outcomes, as detailed in Table 5. In the tabular representation, “NV” and “TD” denote the vehicle count and the minimal distance traveled, respectively. Conversely, “BNV” and “BTD” signify the optimal vehicle count and the corresponding minimal distance, whereas “MNV” and “MTD” represent the average vehicle count and average distance traveled, respectively. Furthermore, we introduced the “Gap = $100 \times (TD - BTD) / BTD$ ” metric to quantify the divergence of our algorithm’s performance from the optimal outcomes, given an equivalent vehicle count. Concurrently, we documented the average execution time of the algorithm to evaluate its efficiency.

The analysis of Table 5 demonstrates that LNS-MRSO successfully achieves the best known solution (BKS) in 16 instances, aligning with previous literature and showcasing its superior performance. It is worth noting that when maintaining a constant vehicle count, LNS-MRSO effectively identifies the BKS for both C1 and C2 problem types, except for C104, highlighting its optimization capabilities under various constraints. Furthermore, LNS-MRSO outperforms the BKS in terms of shortest distance in 23 instances, resulting in significant reductions of 5.1% and 8.8% in average distance for R2 and RC2, respectively. Although there are some cases where it falls short compared to the BKS, these deviations remain within 10%, which can be attributed to the algorithm’s efficient large neighborhood search strategies and population update methods. However, it should be acknowledged that while pursuing optimal driving distances, LNS-MRSO may compromise some optimality in vehicle utilization. Nevertheless, the proximity of the Minimum Number of Vehicles (MNV) and Minimum Total Distance (MTD) to their respective best number of vehicles (BNV) and Best Total Distance (BTD) counterparts indicates the stability and robustness of this algorithm. Additionally, we observed improved performance by LNS-MRSO on “2”-type problems with extended time windows due to increased scheduling flexibility provided by these windows; conversely, its performance slightly diminished on “1”-type problems with narrower time windows due to heightened complexity imposed by tighter time constraints.

Overall, LNS-MRSO exhibits exceptional performance across a range of metrics, demonstrating notable effectiveness, stability, and robustness. Although certain limitations may arise under specific conditions, it consistently produces high-quality solutions in diverse scenarios. While LNS-MRSO may involve a slightly longer computation time for optimization effectiveness and stability gains, this tradeoff is deemed worthwhile.

5.5. Comparison with Other Algorithms

In Table 6, we have meticulously selected four cutting-edge, high-performing algorithms as benchmarks to evaluate the performance of LNS-MRSO. Our primary evaluation criterion was to minimize the overall path length, followed by reducing the number of vehicles utilized. Experimental results demonstrate that LNS-MRSO achieves an optimal performance in 27 out of 56 tested datasets, showcasing its robust optimization capabilities. Comparatively, ASC-BSO [36], MOLNS [37], and N-CLPSO [16] outperform in 19, 17, and 19 test instances, respectively. While MFGA [21] excels in 39 test cases and performs similarly to LNS-MRSO in minimizing total travel distance, LNS-MRSO exhibits superior performance specifically on C2, R2, and RC2 problems. Notably, LNS-MRSO excels particularly well with “2”-type problems, which typically involve larger loading capacities and longer time windows. During the large neighborhood search process, LNS-MRSO effectively utilizes these time windows to enhance the accuracy of the correlation function for discriminating customer sequences. Consequently, the algorithm can more precisely consider customer demands and constraints when planning vehicle routes, resulting in more efficient solutions. In contrast, MFGA performs well on “1”-type problems but is limited when dealing with “2”-type problems. On the other hand, LNS-MRSO demonstrates its adaptability and efficiency in handling complex resource-intensive scenarios, stressing its strength under such circumstances.

Table 4. Comparison with BKS of Solomon-25 and Solomon-50.

Instances	25 Customers Best-Known		LNS-MRSO			50 Customers Best-Known		LNS-MRSO		
	BNV	BTD	NV	TD	Gap	BNV	BTD	NV	TD	Gap
C101	3	191.3	3	191.8	0.27	5	362.4	5	363.2	0.23
C102	3	190.3	3	190.7	0.23	5	361.4	5	362.2	0.21
C103	3	190.3	3	190.7	0.23	4	361.4	5	362.2	0.21
C104	3	186.9	3	187.4	0.29	5	359.0	5	358.9	−0.03
C105	3	191.3	3	191.8	0.27	5	362.4	5	363.2	0.23
C106	3	191.3	3	191.8	0.27	5	362.4	5	363.2	0.23
C107	3	191.3	3	191.8	0.27	5	362.4	5	363.2	0.23
C108	3	191.3	3	191.8	0.27	5	362.4	5	363.2	0.23
C109	3	191.3	3	191.8	0.27	5	362.4	5	363.2	0.23
Ave	3	190.6	3	191.1	0.27	4.889	361.8	5	362.5	0.19
C201	2	214.7	2	215.5	0.39	3	360.2	3	361.8	0.44
C202	2	214.7	2	215.5	0.39	3	360.2	3	361.8	0.44
C203	2	214.7	2	215.5	0.39	3	359.8	3	361.4	0.45
C204	1	213.1	1	213.9	0.39	2	353.4	2	355.2	0.50
C205	2	214.7	2	215.5	0.39	3	359.8	3	361.4	0.45
C206	2	214.7	2	215.5	0.39	3	359.8	3	361.4	0.45
C207	2	214.5	2	215.3	0.39	3	359.6	3	361.4	0.50
C208	2	214.5	2	215.4	0.41	2	350.5	2	352.1	0.46
Ave	1.875	214.5	1.875	215.3	0.39	2.75	357.9	2.75	359.6	0.47
R101	8	617.1	8	618.3	0.20	13	1047.0	12	1060.6	1.28
R102	7	547.1	7	548.1	0.18	12	944.9	11	924.1	−2.25
R103	5	454.6	5	455.7	0.24	9	772.9	8	784.6	1.49
R104	4	416.9	4	418.0	0.25	6	631.2	6	631.6	0.06
R105	6	530.5	6	531.5	0.20	10	906.6	10	924.3	1.91
R106	5	465.4	5	466.5	0.23	8	793.6	8	795.3	0.21
R107	4	424.3	4	425.3	0.23	7	720.4	7	713.5	−0.97
R108	4	397.3	4	398.3	0.25	6	618.2	6	630.3	1.92
R109	5	441.3	5	442.6	0.30	8	803.2	8	796.0	−0.90
R110	4	444.1	5	445.2	0.24	8	724.9	7	741.7	2.26
R111	4	428.8	4	429.7	0.21	8	724.9	7	713.1	−1.66
R112	4	393.0	4	394.1	0.28	6	651.1	6	645.8	−0.82
Ave	5	463.4	5.083	464.4	0.22	8.417	778.2	8	780.1	0.24
R201	4	463.3	4	464.4	0.23	6	800.7	5	801.8	0.13
R202	4	410.5	4	411.5	0.24	5	712.2	4	711.3	−0.13
R203	3	391.4	3	392.3	0.24	5	606.4	4	610.7	0.71
R204	2	355.0	2	355.9	0.25	2	509.5	2	511.1	0.31
R205	3	393.0	3	394.1	0.27	5	703.3	4	702.6	−0.10
R206	3	374.4	3	375.5	0.29	5	647.0	4	635.0	−1.90
R207	3	361.6	3	362.6	0.29	4	584.6	3	578.6	−1.04
R208	1	328.2	1	329.3	0.34	2	487.7	2	491.0	0.66
R209	2	370.7	2	371.6	0.23	4	600.6	4	608.1	1.23
R210	3	404.6	3	405.5	0.22	5	663.4	4	647.9	−2.40
R211	2	350.9	2	351.9	0.29	3	551.3	3	544.9	−1.17
Ave	2.727	382.2	2.727	383.1	0.25	4.182	624.2	3.545	622.1	−0.34
RC101	4	461.1	4	462.2	0.23	9	957.9	8	945.6	−1.30
RC102	3	351.8	3	352.7	0.27	8	844.3	7	827.0	−2.10
RC103	3	332.8	3	334.1	0.39	6	712.6	6	712.5	−0.01
RC104	3	306.6	3	307.1	0.18	5	546.5	5	546.5	0.00
RC105	4	411.3	4	412.4	0.26	9	888.9	8	857.0	−3.73
RC106	3	345.5	3	346.5	0.29	7	791.9	6	728.9	−8.65
RC107	3	298.3	3	298.9	0.22	6	664.5	6	649.0	−2.39
RC108	3	294.5	3	295.0	0.17	6	598.1	5	529.4	−12.98
Ave	3.25	350.2	3.25	351.5	0.36	7	750.6	6.375	724.5	−3.61
RC201	3	360.2	3	361.2	0.29	5	684.8	5	686.3	0.22
RC202	3	338.0	3	338.8	0.24	5	613.6	5	615.1	0.24
RC203	3	326.9	3	327.7	0.24	4	555.3	4	556.5	0.22
RC204	3	299.7	3	300.2	0.18	3	444.2	3	444.9	0.17
RC205	3	338.0	3	338.9	0.27	5	631.0	5	632.0	0.16
RC206	3	324.0	3	325.1	0.34	5	610.0	5	611.7	0.27
RC207	3	298.3	3	298.9	0.22	4	558.6	4	559.9	0.23
RC208	2	269.1	2	269.6	0.17	2	535.8	3	480.2	−11.59
Ave	2.875	319.3	2.875	320.1	0.26	4.125	579.2	4.25	573.3	−1.03

The bolded data is better than BKS.

Table 5. Comparison with BKS of Solomon-100.

100 Customers	Best-Known		Source	LNS-MRSO				Gap	Time (s)
	NV	TD		BNV	BTD	MNV	MTD		
C101	10	828.94	Rochat et al. [38]	10	828.94	10	828.94	0.00	6.0
C102	10	828.94	Rochat et al. [38]	10	828.94	10	851.45	0.00	5.3
C103	10	828.06	Rochat et al. [38]	10	828.06	10	830.14	0.00	5.7
C104	10	824.78	Rochat et al. [28]	10	847.15	10	847.15	2.64	5.9
C105	10	828.94	Rochat et al. [38]	10	828.94	10	828.94	0.00	5.2
C106	10	828.94	Rochat et al. [38]	10	828.94	10	828.94	0.00	5.5
C107	10	828.94	Rochat et al. [38]	10	828.94	10	828.94	0.00	5.8
C108	10	828.94	Rochat et al. [38]	10	828.94	10	828.94	0.00	5.9
C109	10	828.94	Rochat et al. [38]	10	828.94	10	828.94	0.00	6.0
C201	3	591.56	Rochat et al. [38]	3	591.56	3	591.56	0.00	5.2
C202	3	591.56	Rochat et al. [38]	3	591.56	3	591.56	0.00	5.3
C203	3	591.17	Rochat et al. [38]	3	591.17	3	591.17	0.00	5.6
C204	3	590.60	Rochat et al. [38]	3	590.60	3	590.60	0.00	5.2
C205	3	588.88	Rochat et al. [38]	3	588.88	3	588.88	0.00	5.0
C206	3	588.49	Rochat et al. [38]	3	588.49	3	588.49	0.00	5.3
C207	3	588.29	Rochat et al. [38]	3	588.29	3	588.29	0.00	5.2
C208	3	588.32	Rochat et al. [38]	3	588.32	3	588.32	0.00	5.1
R101	18	1613.59	Tan et al. [39]	19	1659.34	19.3	1667.98	2.76	5.3
R102	17	1486.12	Rochat et al. [38]	18	1476.50	18	1479.45	−0.65	5.3
R103	13	1292.68	Li et al. [40]	14	1240.97	14.3	1245.51	−4.17	6.2
R104	9	1007.24	Mester [41]	12	1033.52	11	1037.24	2.54	5.2
R105	14	1377.11	Rochat et al. [38]	15	1405.18	15	1411.69	2.00	4.9
R106	12	1251.98	Mester [41]	14	1270.87	13	1281.69	1.49	6.6
R107	10	1104.66	Shaw [30]	12	1122.96	12	1129.33	1.63	6.3
R108	9	960.88	Berger et al. [42]	11	986.20	11	989.47	2.57	5.4
R109	11	1194.73	Homberger et al. [43]	12	1207.18	12	1208.04	1.03	5.6
R110	10	1118.59	Mester [41]	12	1128.21	12	1130.02	0.85	5.6
R111	10	1006.72	Rouseau et al. [44]	12	1062.26	12	1062.26	5.2	4.9
R112	9	982.14	Gamhardella et al. [45]	11	984.84	11	987.45	0.27	5.3
R201	4	1252.37	Homberger et al. [43]	7	1183.90	7	1183.90	−5.78	4.3
R202	3	1191.70	Rouseau et al. [44]	6	1044.45	6	1045.35	−14.10	4.4
R203	3	939.54	Mester [41]	6	900.20	6	900.20	−4.37	4.5
R204	2	825.52	Bemt et al. [46]	5	775.59	5	775.59	−6.44	4.6
R205	3	994.42	Rouseau et al. [44]	5	962.02	5	964.71	−3.37	4.2
R206	3	906.14	Sckrimpf et al. [47]	4	916.30	4.3	916.30	1.11	4.4
R207	2	837.20	Bouthillier et al. [48]	5	832.16	5	834.61	−0.61	4.4
R208	2	726.75	Mester [41]	4	721.04	4	721.04	−0.79	4.6
R209	3	909.16	Homberger [49]	5	875.95	5	876.08	−3.79	5.4
R210	3	938.58	Ghoseiri et al. [7]	5	925.47	5	925.47	−1.42	5.5
R211	2	892.71	Berger et al. [42]	4	767.02	5	783.68	−13.91	5.4
RC101	14	1696.94	Taillard et al. [50]	15	1663.36	15.2	1665.93	−2.02	5.2
RC102	12	1554.75	Taillard et al. [50]	14	1498.21	14	1498.21	−3.77	5.8
RC103	11	1261.67	Taillard et al. [50]	12	1346.00	12	1346.87	6.27	5.5
RC104	10	1125.48	Fu et al. [51]	11	1186.50	11	1190.35	5.14	7.9
RC105	13	1629.44	Berger et al. [42]	16	1596.70	16	1596.70	−2.05	6.2
RC106	11	1424.73	Berger et al. [42]	13	1408.83	13	1408.83	−1.13	6.1
RC107	11	1222.10	Ghoseiri et al. [7]	12	1335.05	12	1337.21	8.46	3.8
RC108	10	1139.82	Taillard et al. [50]	12	1227.24	12	1231.58	7.12	5.7
RC201	4	1406.91	Mester [41]	8	1285.08	8	1286.17	−9.48	6.5
RC202	3	1365.65	Candh et al. [52]	8	1106.84	8	1106.84	−23.38	5.1
RC203	3	1049.62	Caech et al. [52]	6	931.45	6	931.45	−12.69	5.4
RC204	3	798.41	Mester [41]	4	824.79	4	824.79	3.20	5.3
RC205	4	1297.19	Mester [41]	7	1180.12	7	1182.39	−9.92	5.2
RC206	3	1146.32	Homberger [49]	7	1072.29	7	1075.10	−6.90	6.3
RC207	3	1061.14	Berger et al. [42]	6	977.06	6	977.06	−8.61	5.7
RC208	3	828.14	Iharaki et al. [53]	5	805.21	5	805.21	−2.85	5.1

The bolded data is better than BKS.

Table 6. Comparison with other algorithms.

Instances	ACS-BSO		MOLNS		N-CLPSO		MFGA		LNS-MRSO	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
C101	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94
C102	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94
C103	10	828.06	10	828.94	10	828.06	10	828.06	10	828.06
C104	10	824.78	10	828.94	10	824.78	10	824.78	10	847.15
C105	10	824.94	10	828.94	10	828.94	10	828.94	10	828.94
C106	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94
C107	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94
C108	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94
C109	10	828.94	10	828.94	10	828.94	10	828.94	10	828.94
C201	3	591.56	3	591.56	3	591.56	3	591.56	3	591.56
C202	3	591.56	3	591.56	3	591.56	3	591.56	3	591.56
C203	3	591.17	3	591.56	3	591.17	3	591.17	3	591.17
C204	3	591.60	3	590.60	3	590.60	3	590.60	3	590.60
C205	3	588.88	3	588.88	3	588.88	3	588.88	3	588.88
C206	3	588.49	3	588.49	3	588.49	3	588.49	3	588.49
C207	3	588.29	3	588.29	3	588.29	3	588.29	3	588.29
C208	3	588.32	3	588.32	3	588.32	3	588.32	3	588.32
R101	19	1671.16	19	1654.93	19	1648.08	19	1584.00	19	1659.34
R102	17	1504.60	18	1475.33	17	1486.12	16	1374.20	18	1476.50
R103	14	1245.86	14	1240.44	13	1292.68	13	1158.90	14	1240.97
R104	11	1010.73	10	1010.72	10	996.27	11	996.95	12	1033.52
R105	15	1366.05	15	1389.85	14	1377.11	15	1355.30	15	1405.18
R106	13	1288.84	13	1269.14	12	1252.03	13	1212.10	14	1270.87
R107	11	1101.56	11	1102.72	11	1081.17	11	1075.50	12	1122.96
R108	10	974.17	10	991.57	10	985.76	10	959.88	11	986.20
R109	12	1165.71	12	1177.76	11	1194.73	12	1155.80	12	1207.18
R110	11	1090.92	12	1129.60	11	1101.49	12	1092.40	12	1128.21
R111	11	1148.14	12	1108.70	11	1062.67	12	1059.20	12	1062.26
R112	10	1004.53	10	964.15	10	974.95	10	979.05	11	984.84
R201	4	1336.05	4	1305.25	4	1252.37	6	1168.70	7	1183.90
R202	4	1128.05	4	1093.67	3	1225.02	6	1042.40	6	1044.45
R203	3	1020.10	4	915.43	3	962.25	5	893.97	6	900.20
R204	3	834.92	3	775.99	3	766.13	4	744.02	5	775.59
R205	3	1105.38	3	1075.10	3	1027.79	5	969.42	5	962.02
R206	3	949.11	3	979.21	3	939.46	5	880.60	4	916.30
R207	4	812.35	3	851.89	3	872.40	4	822.84	5	832.16
R208	2	940.30	2	754.99	2	726.75	4	736.55	4	721.04
R209	3	1046.73	4	898.23	3	943.72	5	905.11	5	875.95
R210	3	1069.26	4	941.58	3	965.88	4	937.06	5	925.47
R211	3	836.36	3	838.14	3	828.90	4	815.09	5	783.68
RC101	16	1643.78	15	1662.56	15	1635.11	15	1595.90	15	1663.36
RC102	14	1464.63	14	1486.35	13	1503.42	14	1460.90	14	1498.21
RC103	11	1275.64	12	1291.95	11	1261.67	11	1292.60	12	1346.00
RC104	10	1156.92	10	1162.53	10	1135.48	10	1135.00	11	1186.50
RC105	14	1609.68	15	1604.53	14	1542.55	15	1510.10	16	1596.70
RC106	13	1378.45	13	1400.09	12	1388.70	13	1367.20	13	1408.83
RC107	11	1318.69	12	1259.55	11	1230.48	12	1215.90	12	1335.05
RC108	11	1134.85	11	1205.13	11	1157.12	11	1120.10	12	1227.24
RC201	4	1514.41	4	1497.89	4	1406.91	6	1274.80	8	1285.08
RC202	4	1326.71	4	1199.53	4	1169.67	5	1115.70	8	1106.84
RC203	3	1166.91	4	985.54	3	1082.57	5	945.90	6	931.45
RC204	3	929.94	3	805.46	3	828.61	4	803.91	4	824.79
RC205	4	1360.91	5	1340.38	4	1297.19	6	1209.50	7	1180.12
RC206	3	1237.21	3	1316.42	3	1146.32	5	1098.00	7	1072.29
RC207	4	1039.59	4	1031.62	3	1095.67	5	1010.40	6	977.06
RC208	3	910.59	3	859.13	3	828.14	4	810.04	5	805.21

The bolded data signifies the most optimal solution out of the five algorithms.

Table 7 comprehensively exhibits the mean optimal results of thirteen high-performing algorithms alongside the best-known solutions (BKSs) across various sub-problems (C1, C2, R1, R2, RC1, and RC2). These results are evaluated based on the number of vehicles used (NV), total distance traveled (TD), and a composite metric encompassing both (NV&TD). Additionally, we calculated the average vehicle usage (ANV) and average total distance traveled (ATD) across all datasets. Notably, LNS-MRSO attains a remarkable score of 994 in terms of total distance traveled, ranking fourth and significantly surpassing the BKS. As the bolded data shows, only three algorithms, namely MFGA [21], RRGGA [54], and Tabu-ABC [55], outperform LNS-MRSO in this metric. A common characteristic among these superior algorithms is their capacity to reduce the total travel distance by increasing the number of vehicles utilized. Nevertheless, with an ANV of 8.54, slightly exceeding RRGGA's 8.47 and Tabu-ABC's 8.34, there exists potential for further optimization in minimizing vehicle usage for LNS-MRSO. A thorough analysis of Table 7 reveals that LNS-MRSO performs comparably to most state-of-the-art algorithms on C2-type problems, matching BKS results and closely approaching them on C1-type problems. Specifically, on R2 and RC2 problems, LNS-MRSO achieved scores of 900.37 and 1022.86, ranking third and fourth, respectively, surpassing the BKS. This underscores LNS-MRSO's notable proficiency in handling VRPTW with substantial load capacities and extended time windows. Consequently, we conclude that LNS-MRSO excels in solving "2"-type problems, albeit with room for improvement in addressing "1"-type problems.

Table 7. Comparison of average levels.

	C1	C2	R1	R2	RC1	RC2	ANV	ATD
Best-known	10&828.38	3&589.86	11.83&1207.20	2.73&946.74	11.50&1383.12	3.25&1119.17	7.05	1012
N-CLPSO (2024) [16]	10&828.38	3&589.86	12.42&1205.96	3.00&955.52	12.13&1356.82	3.38&1106.89	7.32	1007
IGA (2023) [56]	10&860.05	3&602.57	11.92&1248.19	2.73&986.95	11.50&1426.84	3.25&1139.32	7.07	1044
MFGA (2023) [21]	10&828.38	3&589.86	12.83&1166.90	4.72&901.43	12.62&1337.20	5.00&1033.50	8.03	976
MOTSP (2021) [57]	10&846.91	3&598.10	12.75&1195.77	3.09&953.34	12.88&1373.06	4.00&1106.15	7.62	1012
HRRGA (2021) [14]	10&828.38	3&590.60	12.25&1211.89	3.09&966.24	11.88&1360.19	4.00&1055.53	7.37	1002
D-VND (2021) [58]	10&828.38	3&589.86	12.42&1214.02	3.09&944.71	12.13&1369.00	3.38&1069.53	7.37	1003
RRGA (2020) [54]	10&828.38	3&589.86	13.25&1180.66	5.54&878.64	12.75&1341.60	6.25&1004.35	8.47	971
MAPSO (2019) [59]	10&828.38	3&589.86	11.92&1209.99	2.73&952.06	11.50&1384.18	3.25&1119.60	7.07	1014
PDVA (2018) [60]	10&828.38	3&591.49	12.92&1228.60	3.45&1033.53	12.75&1362.09	3.75&1068.26	7.65	1019
Tabu-ABC (2017) [55]	10&828.38	3&589.86	13.75&1187.90	4.64&891.24	13.13&1361.08	5.50&1017.47	8.34	979
HGSADC (2013) [61]	10&828.38	3&589.86	11.92&1210.69	2.73&951.51	11.50&1384.17	3.25&1119.24	7.07	1014
ACO-N (2011) [62]	10&841.92	3.3&612.75	13.10&1213.16	4.60&952.30	12.70&1415.62	5.60&1120.37	8.22	1026
LNS-MRSO	10&830.86	3&589.86	13.50&1214.84	5.27&901.89	13.12&1407.74	6.37&1022.86	8.54	994

6. Simulation Scheduling Experiment of Unmanned Electric Loader

Given the exceptional performance demonstrated by LNS-MRSO in addressing large-scale loading and prolonged vehicle routing challenges, this study explores its application in scheduling unmanned loaders within concrete-mixing stations. The seamless transition from traditional fuel-powered loaders to electric loaders is primarily attributed to the seamless integration of highly efficient and dependable electric motor technology with innovative new energy technologies. This transition has led to a remarkable reduction in reliance on traditional fuels and exhaust emissions, thereby enhancing the overall efficiency of energy utilization. Additionally, the integration of autonomous driving technology enables unmanned electric loaders to navigate independently in complex construction site environments, efficiently planning routes and operational schedules. The incorporation of advanced sensors and automation systems not only mitigates the risks associated with human-induced accidents but also enables continuous, unrestricted operations. These advantages significantly address the issues of elevated labor costs and pollutant emissions encountered by LNS-MRSO when managing a large fleet of vehicles, thereby rendering it highly suitable for scheduling unmanned loaders in concrete-mixing stations. The unmanned electric loader's performance parameters are shown in Figure 9.

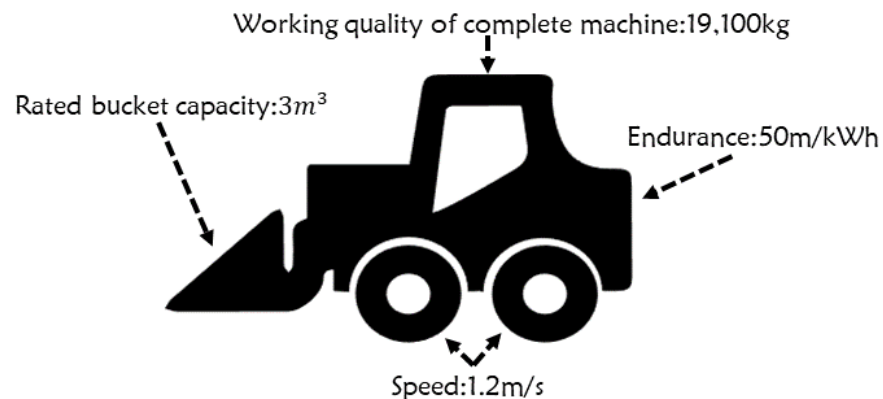


Figure 9. The characteristics of the unmanned electric loader in the concrete-mixing station.

6.1. Experimental Process

In the manufacturing process of concrete-mixing stations, the material silo plays a crucial role in efficiently storing aggregates, which is pivotal for production. To ensure the accuracy of concrete-mixing ratios, workers appropriately classify and store aggregates. On the other hand, the hopper serves as a temporary storage facility for various raw materials during the production process. Within this framework, unmanned electric loaders seamlessly transport materials between silos and hoppers, following instructions from the central scheduling system and autonomously returning to the designated parking area after completing their tasks.

In the simulation experiments, a large hopper consisting of three smaller hoppers designated for sand storage, crushed stone A storage, and crushed stone B storage, respectively, was designed. These three raw materials are consumed at an equal and steady rate, and the production of 1 cubic meter of concrete requires a specific blend of sand, crushed stone A, and crushed stone B, as detailed in Table 8. Consequently, the unmanned loaders were categorized into three groups with each group assigned to replenish a specific raw material. The central scheduling system employed LNS-MRSO to dispense instructions every 100 seconds based on real-time hopper inventory data. These instructions directed the unmanned loaders to depart from the storage silos and proceed to the designated hoppers for replenishment. Due to their system-controlled nature, the quantity of material dispensed during each operation remains constant. If the raw material level in a target hopper falls below the replenishment threshold, the unmanned loader waits until it reaches this threshold before initiating the replenishment process with an aim to fill up the hopper completely. Immediate replenishment occurs if the material level is already below this threshold. In the event of a target hopper becoming depleted, the concrete production process pertaining to that hopper is temporarily halted. This simulation experiment aims to devise an efficient replenishment route plan for the unmanned loaders, utilizing sand, crushed stone A, and crushed stone B as illustrative examples. By refining the scheduling strategy, we aspire to achieve uninterrupted concrete production across all hoppers while minimizing the power consumption of the unmanned loaders.

Table 8. Raw material ratio per cubic meter of concrete.

	Sand	Crushed Stone A	Crushed Stone B
Quality (kg)	839	307	716
Volume (m ³)	0.59	0.194	0.41
Density (kg/m ³)	1400	1600	1800
Discharge rate (m ³ /s)	0.0051	0.0043	0.0039

As depicted in Figure 10, the production of concrete involves the blending of raw materials from three interconnected small hoppers. Using sand as a representative example, during the replenishment process, the sensor monitoring system constantly oversees the remaining material in the hopper and relays the data to the central scheduling system. Leveraging the measured hopper residual quantity λ_i , the scheduling system accurately determines the replenishment time window for each hopper.

$$e_i = \max(0, \frac{\lambda_i + q_i - Q}{\rho}) \quad (24)$$

$$l_i = \frac{\lambda_i - \lambda_p}{\rho} \quad (25)$$

In this context, q_i represents the fixed demand for a certain raw material in the hopper, Q denotes the maximum capacity of the hopper, ρ refers to the discharge rate of sand, and λ_p represents the threshold for replenishment operations. Additionally, the unloading time is constant, and the unmanned electric loader will immediately proceed to the next hopper in need of replenishment upon completing a discharge operation.

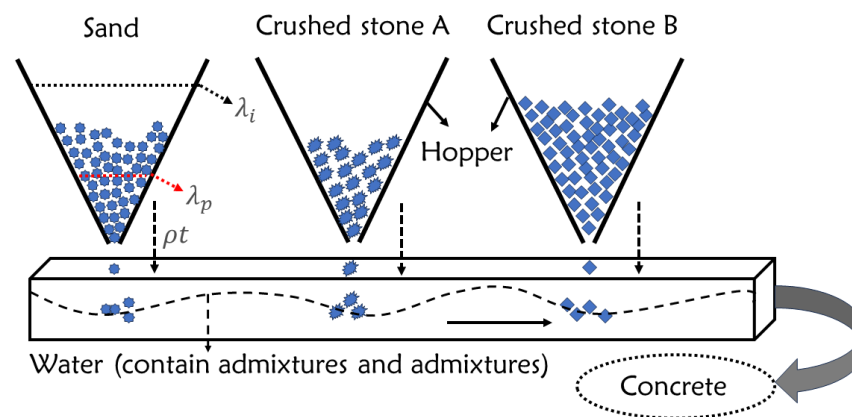


Figure 10. Concrete production diagram.

6.2. Experimental Result

We take a concrete-mixing station as an example, assuming its silo position as the origin, and that the position of hopper i is $(210 - 10i, j)$ ($i = 1, 2, \dots, 41$), where if i is odd, j is 37; otherwise, j is 52. The hopper allowance represents the monitoring value recorded by the sensor monitoring system at a specific moment, with the initial value for simulation experiments being randomly determined. The hopper demand is set at 0.5 m^3 by the central scheduling system in conjunction with the discharge capacity of unmanned electric loaders, while the filling threshold is established at 17 m^3 . Utilizing Formulas (24) and (25), we can calculate the left and right time windows for each hopper and set a discharge time of 10 s. Considering that up to eight unmanned electric loaders can be scheduled in one instruction, this paper primarily focuses on planning filling routes for unmanned electric loaders following instructions issued by the scheduling system.

The central scheduling system controlled by LNS-MRSO was compared to the initial central scheduling system utilizing ACO through a comparative analysis. As shown in Table 9 and Figure 11, the effectiveness of the LNS-MRSO algorithm in optimizing route planning for unmanned electric loaders is remarkable. During the replenishment process of the three raw materials, the LNS-MRSO-controlled central scheduling system significantly reduced the travel distance of unmanned electric loaders compared to the ACO-controlled central scheduling system. Specifically, after each command was issued, there was an approximate reduction of 385 meters in their total travel distance.

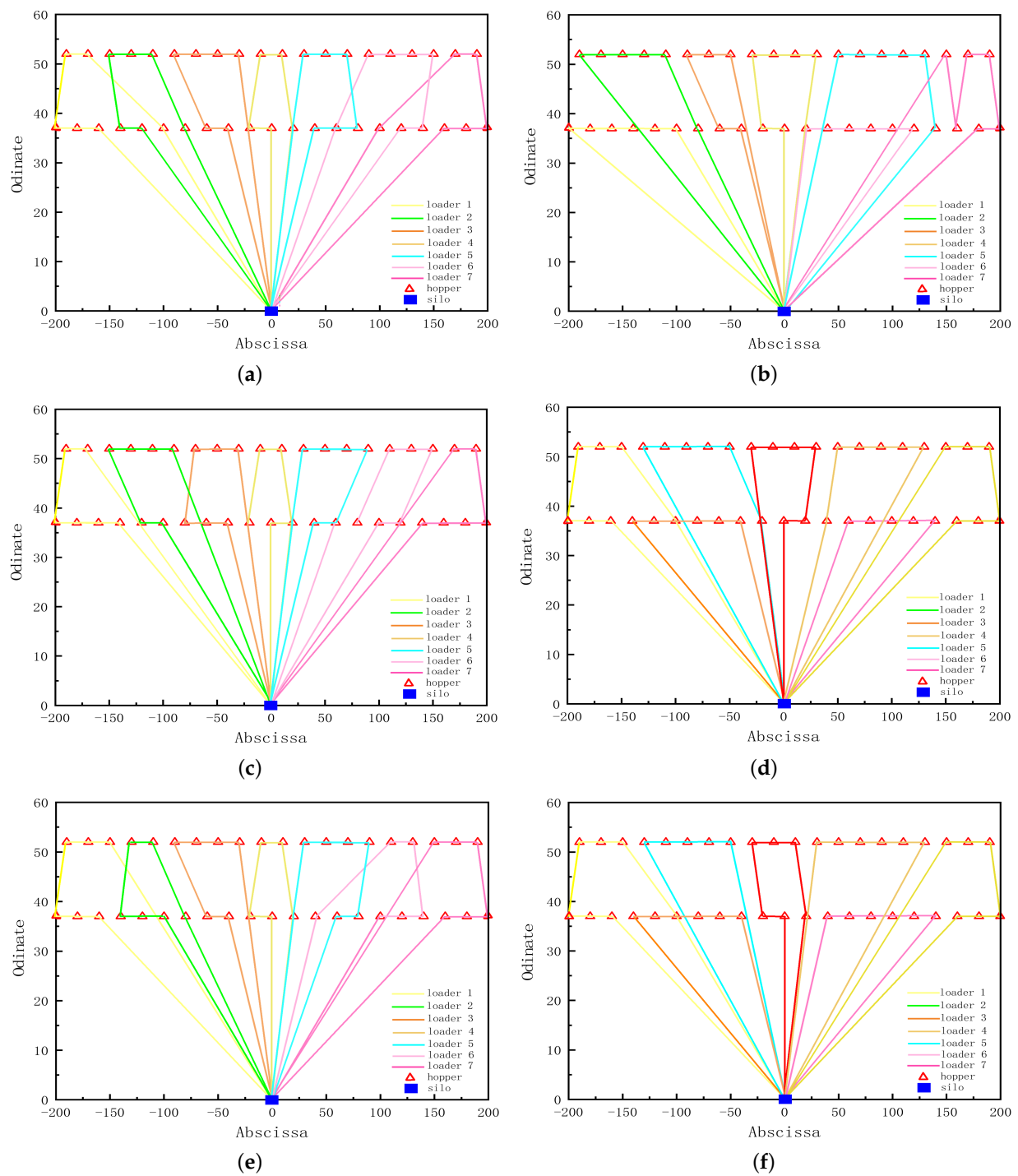


Figure 11. Comparison of feeding routes. (a) LNS-MRSO-feeding route of sand. (b) ACO-feeding route of sand. (c) LNS-MRSO-feeding route of crushed stone A. (d) ACO-feeding route of crushed stone A. (e) LNS-MRSO-feeding route of crushed stone B. (f) ACO-feeding route of crushed stone B.

Table 9. Comparison of experimental results.

Central Dispatching System	LNS-MRSO	ACO	Reduced Distance
Sand	2138 m	2290 m	152 m
Crushed stone A	2129 m	2251 m	122 m
Crushed stone B	2119 m	2240 m	111 m

6.3. Experimental Conclusions

The application of the LNS-MRSO algorithm in concrete-mixing stations enables the unmanned operation of the silo, achieving a reduction rate of up to 75%. The loader can operate continuously for extended periods, significantly enhancing the supply guarantee capability of the mixing station and reducing its management costs by over 20%. Furthermore, this optimization yields substantial environmental benefits. Equipped with a multi-angle perception module, the unmanned electric loader comprehensively monitors its operational environment, incorporating safety features such as obstacle detection, collision warning, and overturning monitoring. High-precision collection point perception technology enables detection of obstacles and pedestrians within a 10 m radius, ensuring comprehensive operational safety. As illustrated in Figure 9, the unmanned electric loader covers a distance of 50 m per 1 KWh of power consumption. Consequently, each path optimization execution saves approximately 7.7 KWh of power consumption. Considering that the central dispatching system issues instructions every 100 s, it is estimated that when operating for 16 h daily (excluding charging time), the concrete-mixing station can save nearly 1.6 million KWh annually through reduced power consumption. This indirectly mitigates greenhouse gas emissions such as carbon dioxide from power generation processes and contributes to global warming mitigation efforts. This study also underscores the advantages of utilizing LNS-MRSO algorithm in path planning applications within similar scenarios.

VRPTW is primarily utilized in the domain of logistics and transportation. As society continues to develop, an increasing number of sectors, such as distribution and logistics, express services, urban logistics, household services, and public transportation, will encounter vehicle scheduling challenges. The LNS-MRSO algorithm proposed in this study is applicable in the aforementioned scenarios. Simulation results from the Solomon dataset demonstrate that the algorithm performs better with larger vehicle scales and reasonable numbers of vehicles and minimum distances. Furthermore, this paper presented an empirical analysis using the scheduling of an unmanned loader at a concrete-mixing station as an example, concluding that the proposed algorithm can reduce driving distance, lower operating costs, and enhance production efficiency. Therefore, it is evident that the algorithm is not only suitable for traditional logistics and transportation industries but can also be extended to factory production scheduling industries to improve operational efficiency.

7. Conclusions and Future Research

In this study, we propose an algorithm called LNS-MRSO, specifically designed to address the vehicle routing problem with time windows (VRPTW). The construction of LNS-MRSO is based on a combination of large neighborhood search (LNS) and modified rat swarm optimization (MRSO), incorporating five innovative operators to enhance its evolutionary capabilities. These operators include chase and attack operators that aim to diversify the population, an escape operator designed to avoid local optima, and jump and rotate operators that strike a balance between global and local search abilities. During the initialization phase, a greedy strategy is employed to minimize the initial number of vehicles. Additionally, in order to guide neighborhood searches effectively, we have improved both the remove operator by considering multiple factors and the reinsert operator by taking into account distance increments. Within the search process, a simulated annealing mechanism is applied to accept poorer solutions with a certain probability, thereby enhancing robustness. To validate the effectiveness of LNS-MRSO, we conducted an evaluation on 56 Solomon VRPTW instances. The experimental results demonstrate that LNS-MRSO outperforms other heuristic algorithms in terms of overall distance. Particularly noteworthy is its performance on the R2 and RC2 datasets, where LNS-MRSO achieves percentage deviations that are 5.1% and 8.8% better than the best-known solutions, respectively. Additionally, the algorithm produced 16 optimal solutions and discovered 23 new optimal solutions across various dataset types.

In order to validate the practical application of the LNS-MRSO algorithm in engineering, we applied the algorithm to the vehicle scheduling problem of unmanned loaders in a concrete-mixing station. The experimental results demonstrate that, with an equivalent number of unmanned loaders, LNS-MRSO requires 385m less than the ACO scheduling algorithm within 100S scheduling instructions, thereby enhancing the operational efficiency of unmanned loaders and conserving energy. Future solutions for the scheduling problem of unmanned electric loaders should possess real-time data processing capabilities and dynamically adjust vehicle paths and allocation strategies to accommodate complex and variable real-world scenarios. Furthermore, given the increasing global attention towards environmental protection and sustainable development, future research on VRPTW should prioritize achieving green logistics. By optimizing vehicle distribution paths and reducing energy consumption and exhaust emissions, it is imperative to promote environmentally friendly transformations within the logistics and distribution industry.

In the subsequent research of LNS-MRSO, an effective approach to reduce the local search space and enhance the performance of guided reinsert operators is to extract valuable local information from diverse search attributes pertaining to specific individuals. Furthermore, there exists a multitude of VRP variants, such as EVRPTW and PDPTW, that warrant investigation based on LNS-MRSO. These studies necessitate a redesign of certain key operators within LNS-MRSO in accordance with the distinct attributes of these VRP variants, including the remove–reinsert neighbor search mechanism. Additionally, our focus lies in augmenting the algorithm’s intelligence by integrating artificial intelligence and machine learning technologies to endow the LNS-MRSO algorithm with self-learning capabilities. A meaningful avenue for research involves adjusting the search strategy in discrete network scenarios to achieve efficient optimization through reinforcement learning. Moreover, we are dedicated to enhancing the algorithm’s parallel computing capability in order to address large-scale logistics problems in the era of big data while simultaneously expediting solution speed.

Author Contributions: Development or design of methodology, X.W. and Z.X.; Writing—Review & Editing, X.W.; Writing—Original Draft, Z.X.; Programming, Z.X.; Evolution of overarching research goals and aims, Y.W.; Project administration, Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: <https://www.bernabe.dorronsoro.es/vrp/>.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Dantzig, G.B.; Ramser, J.H. The Truck Dispatching Problem. *Manag. Sci.* **1959**, *6*, 80–91. [\[CrossRef\]](#)
2. Kallehauge, B. Formulations and exact algorithms for the vehicle routing problem with time windows. *Comput. Oper. Res.* **2008**, *35*, 2307–2330. [\[CrossRef\]](#)
3. Akpinar, S. Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem. *Expert Syst. Appl.* **2016**, *61*, 28–38. [\[CrossRef\]](#)
4. Ursani, Z.; Essam, D.; Cornforth, D.; Stocker, R. Localized genetic algorithm for vehicle routing problem with time windows. *Appl. Soft Comput.* **2011**, *11*, 5375–5390. [\[CrossRef\]](#)
5. Gómez S., C.G.; Cruz-Reyes, L.; González B., J.J.; Fraire H., H.J.; Pazos R., R.A.; Martínez P., J.J. Ant colony system with characterization-based heuristics for a bottled-products distribution logistics system. *J. Comput. Appl. Math.* **2014**, *259*, 965–977. [\[CrossRef\]](#)
6. Ombuki, B.; Ross, B.J. Hanshar, F. Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows. *Appl. Intell.* **2006**, *24*, 17–30. [\[CrossRef\]](#)
7. Ghoseiri, K.; Ghannadpour, S.F. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Appl. Soft Comput.* **2010**, *10*, 1096–1107. [\[CrossRef\]](#)
8. Clarke, G.; Wright, J.W. Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.* **1964**, *12*, 568–581. [\[CrossRef\]](#)

9. Baker, E.K.; Schaffer, J.R. Solution improvement heuristics for the vehicle routing and scheduling problem with time window constraints. *Am. J. Oper. Res.* **2003**, *3*, 7–20. [\[CrossRef\]](#)
10. Solomon, M.M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* **1987**, *35*, 254–265. [\[CrossRef\]](#)
11. Potvin, J.Y.; Bengio, Y.; Gendreau, M.; Rousseau, J.M. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *Eur. J. Oper. Res.* **1993**, *66*, 331–340. [\[CrossRef\]](#)
12. Antes, J.; Derigs, U. A new parallel tour construction algorithm for the vehicle routing problem with time windows. *Comput. Oper. Res.* **1995**, *22*, 785–800.
13. Maroof, A.; Ayvaz, B.; Naeem, K. Logistics Optimization Using Hybrid Genetic Algorithm (HGA): A Solution to the Vehicle Routing Problem with Time Windows (VRPTW). *IEEE Access* **2024**, *12*, 36974–36989. [\[CrossRef\]](#)
14. Khoo, T.S.; Mohammad, B.B. The parallelization of a two-phase distributed hybrid ruin-and-recreate genetic algorithm for solving multi-objective vehicle routing problem with time windows. *Expert Syst. Appl.* **2021**, *168*, 114408. [\[CrossRef\]](#)
15. Sedighizadeh, D.; Mazaheripour, H. Optimization of multi objective vehicle routing problem using a new hybrid algorithm based on particle swarm optimization and artificial bee colony algorithm considering Precedence constraints. *Alex. Eng. J.* **2018**, *57*, 2225–2239. [\[CrossRef\]](#)
16. Wu, Q.; Xia, X.; Song, H.; Zeng, H.; Xu, X.; Zhang, Y.; Yu, F.; Wu, H. A neighborhood comprehensive learning particle swarm optimization for the vehicle routing problem with time windows. *Swarm Evol. Comput.* **2024**, *84*, 101425. [\[CrossRef\]](#)
17. Teng, Y. Solving dynamic vehicle routing problem with time windows by ant colony system with bipartite graph matching. *Egypt. Inform. J.* **2024**, *25*, 100421. [\[CrossRef\]](#)
18. Dib, O.; Manier, M.-A.; Moalic, L.; Caminada, A. Combining VNS with Genetic Algorithm to solve the one-to-one routing issue in road networks. *Comput. Oper. Res.* **2017**, *78*, 420–430. [\[CrossRef\]](#)
19. Dib, O.; Moalic, L.; Manier, M.-A.; Caminada, A. An advanced GA–VNS combination for multicriteria route planning in public transit networks. *Expert Syst. Appl.* **2017**, *72*, 67–82. [\[CrossRef\]](#)
20. Mohiuddin, M.A.; Khan, S.A. Engelbrecht, A. Fuzzy particle swarm optimization algorithms for the open shortest path first weight setting problem. *Appl. Intell.* **2016**, *45*, 598–621. [\[CrossRef\]](#)
21. Ahmed, Z.H.; Maleki, F.; Yousefikhoshbakht, M.; Haron, H. Solving the vehicle routing problem with time windows using modified football game algorithm. *Egypt. Inform. J.* **2023**, *24*, 100403. [\[CrossRef\]](#)
22. Yassen, E.T.; Ayob, M.; Nazri, M.Z.A.; Sabar, N. An adaptive hybrid algorithm for vehicle routing problems with time windows. *Comput. Ind. Eng.* **2017**, *113*, 382–391. [\[CrossRef\]](#)
23. Banos, R.; Ortega, J.; Gil, C.; Fernández, A.; de Toro, F. A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. *Expert Syst. Appl.* **2013**, *40*, 1696–1707. [\[CrossRef\]](#)
24. Dhiman, G.; Garg, M.; Nagar, A.; Kumar, V.; Dehghani, M. A novel algorithm for global optimization: Rat swarm optimizer. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 8457–8482. [\[CrossRef\]](#)
25. Alfawaz, O.; Osamy, W.; Saad, M.; Khedr, A.M. Modified Rat Swarm Optimization Based Localization Algorithm for Wireless Sensor Networks. *Wirel. Pers. Commun. Int. J.* **2023**, *130*, 1617–1637. [\[CrossRef\]](#)
26. Osamy, W.; Khedr, A.M.; Vijayan, D.; Salim, A. TACTIRSO: Trust aware clustering technique based on improved rat swarm optimizer for WSN-enabled intelligent transportation system. *J. Supercomput.* **2023**, *79*, 5962–6016. [\[CrossRef\]](#)
27. Zebiri, I.; Zeghida, D.; Redjimi, M. Rat Swarm Optimizer for Data Clustering. *Jordanian J. Comput. Inf. Technol.* **2022**, *8*, 297–307. [\[CrossRef\]](#)
28. Xie, R.; Hai, B. Robot path planning based on multi-strategy improved rat swarm algorithm. *Modul. Mach. Tool Autom. Manuf. Tech.* **2022**, *10*, 50–54.
29. Zobolas, G.I.; Tarantilis, C.D.; Ioannou, G. Solving the open shop scheduling problem via a hybrid genetic-variable neighborhood search algorithm. *Cybern. Syst. Int. J.* **2009**, *40*, 259–285. [\[CrossRef\]](#)
30. Shaw, P. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. *Lect. Notes Comput. Sci.* **1998**, *1520*, 417–431.
31. Elhassania, M.J.; Jaouad, B.; Ahmed, E.A. A new hybrid algorithm to solve the vehicle routing problem in the dynamic environment. *Int. J. Soft Comput.* **2013**, *8*, 327–334.
32. Qiu, H. Research on Weapon-Target Assignment Problem Based on Adaptive Large Neighborhood Search Algorithm. National University of Defense Technology. Master's Thesis, National University of Defense Technology, Changsha, China, 2018.
33. Ropke, S.; Pisinger, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **2006**, *40*, 455–472. [\[CrossRef\]](#)
34. VRPTW. Available online: <https://www.bernabe.dorronsoro.es/vrp/> (accessed on 6 February 2024).
35. Service, T.C. A no free lunch theorem for multi-objective optimization. *Inf. Process. Lett.* **2010**, *110*, 917–923. [\[CrossRef\]](#)
36. Shen, Y.; Liu, M.; Yang, J.; Shi, Y.; Middendorf, M. A hybrid swarm intelligence algorithm for vehicle routing problem with time windows. *IEEE Access* **2020**, *8*, 93882–93893. [\[CrossRef\]](#)
37. Konstantakopoulos, G.D.; Gayialis, S.P.; Kechagias, E.P.; Papadopoulos, G.A.; Tatsiopoulos, I.P. A multiobjective large neighborhood search metaheuristic for the vehicle routing problem with time windows. *Algorithms* **2020**, *13*, 243. [\[CrossRef\]](#)
38. Rochat, Y.; Taillard, É.D. Probabilistic diversification and intensification in local search for vehicle routing. *J. Heuristics* **1995**, *1*, 147–167. [\[CrossRef\]](#)

39. Tan, K.C.; Chew, Y.H.; Lee, L.H. A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *Comput. Oper. Res.* **2006**, *34*, 115–151. [\[CrossRef\]](#)
40. Li, H.; Lim, A. Local search with annealing-like restarts to solve the vehicle routing problem with time windows. *Eur. J. Oper. Res.* **2003**, *150*, 115–127. [\[CrossRef\]](#)
41. Mester, D. An evolutionary strategies algorithm for large scale vehicle routing problem with capacity and time windows restrictions. In *Proceedings of the Conference on Mathematical and Population Genetics*; University of Haifa: Haifa, Israel, 2002.
42. Berger, J.; Barkaoui, M.; Bräysy, O. A route-directed hybrid genetic approach for the vehicle routing problem with time windows. *INFOR Inf. Syst. Oper. Res.* **2003**, *41*, 179–194. [\[CrossRef\]](#)
43. Homberger, J.; Gehring, H. Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR Inf. Syst. Oper. Res.* **1999**, *37*, 297–318. [\[CrossRef\]](#)
44. Rousseau, L.M.; Gendreau, M.; Pesant, G. Using constraint-based operators to solve the vehicle routing problem with time windows. *J. Heuristics* **2002**, *8*, 43–58. [\[CrossRef\]](#)
45. Gambardella, L.M.; Taillard, É.; Agazzi, G. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In *New Ideas in Optimization*; McGraw Hill: London, UK, 1999; pp. 63–76.
46. Bent, R.; Van Hentenryck, P. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transp. Sci.* **2004**, *38*, 515–530. [\[CrossRef\]](#)
47. Schrimpf, G.; Schneider, J.; Stamm-Wilbrandt, H.; Dueck, G. Record breaking optimization results using the ruin and recreate principle. *J. Comput. Phys.* **2000**, *159*, 139–171. [\[CrossRef\]](#)
48. Bouthillier, A.L.; Crainic, T.G. A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Comput. Oper. Res.* **2005**, *32*, 1685–1708. [\[CrossRef\]](#)
49. Homberger, J. *Verteilt-Parallele Metaheuristiken Zur Tourenplanung*; Deutscher Universitätsverlag: Wiesbaden, Germany, 2000.
50. Taillard, É.; Badeau, P.; Gendreau, M.; Guertin, F.; Potvin, J.-Y. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transp. Sci.* **1997**, *31*, 170–186. [\[CrossRef\]](#)
51. Fu, Z.; Eglese, R.; Li, L.Y.O. A unified tabu search algorithm for vehicle routing problems with soft time windows. *J. Oper. Res. Soc.* **2008**, *59*, 663–673. [\[CrossRef\]](#)
52. Czech, Z.; Czarnas, P. Parallel simulated annealing for the vehicle routing problem with time windows. In *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-Based Processing*, Canary Islands, Spain, 9–11 January 2002; pp. 376–383.
53. Ibaraki, T.; Imahori, S.; Kubo, M.; Masuda, T.; Uno, T.; Yagiura, M. Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transp. Sci.* **2005**, *39*, 206–232. [\[CrossRef\]](#)
54. Khoo, T.; Mohammad, B.B.; Wong, V.-H.; Tay, Y.-H.; Nair, M. A two-phase distributed ruin-and-recreate genetic algorithm for solving the vehicle routing problem with time windows. *IEEE Access* **2020**, *8*, 169851–169871. [\[CrossRef\]](#)
55. Zhang, D.; Cai, S.; Ye, F.; Si, Y.; Nguyen, T.T. A hybrid algorithm for a vehicle routing problem with realistic constraints. *Inf. Sci.* **2017**, *394*, 167–182. [\[CrossRef\]](#)
56. Yang, K.; Duan, P.; Yu, H. An improved genetic algorithm for solving the helicopter routing problem with time window in post-disaster rescue. *Math. Biosci. Eng.* **2023**, *20*, 15672–15707. [\[CrossRef\]](#)
57. He, Z.; Zhou, K.; Shu, H.; Chen, X.; Lyu, X. Multi-objective algorithm based on tissue p system for solving tri-objective optimization problems. *Evol. Intell.* **2021**, 1–16. [\[CrossRef\]](#)
58. Lan, Y.; Liu, F.; Ng, W.W.; Zhang, J.; Gui, M. Decomposition based multi-objective variable neighborhood descent algorithm for logistics dispatching. *IEEE Trans. Emerg. Top. Comput. Intell.* **2020**, *5*, 826–839. [\[CrossRef\]](#)
59. Marinakis, Y.; Marinaki, M.; Migdalas, A. A multi-adaptive particle swarm optimization for the vehicle routing problem with time windows. *Inf. Sci.* **2019**, *481*, 311–329. [\[CrossRef\]](#)
60. Dong, W.; Zhou, K.; Qi, H.; He, C.; Zhang, J. A tissue p system based evolutionary algorithm for multi-objective vrptw. *Swarm Evol. Comput.* **2018**, *39*, 310–322. [\[CrossRef\]](#)
61. Vidal, T.; Crainic, T.G.; Gendreau, M.; Prins, C. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Comput. Oper. Res.* **2013**, *40*, 475–489. [\[CrossRef\]](#)
62. Yu, B.; Yang, Z.Z.; Yao, B.Z. A hybrid algorithm for vehicle routing problem with time windows. *Expert Syst. Appl.* **2011**, *38*, 435–441. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.