mathematics

MDPI

Article

Investigating Capacitated Vehicle Routing Problem Using Clustered Simulated Annealing Algorithm

Mingyu Yang †, Yifei Wang †, Yining Lu, Linfei Yin and Fang Gao *

School of Electrical Engineering, Guangxi University, Nanning 530004, China; 2012401008@st.gxu.edu.cn (M.Y.); 2212391050@st.gxu.edu.cn (Y.W.); 2312391051@st.gxu.edu.cn (Y.L.); yinlinfei@gxu.edu.cn (L.Y.)

* Correspondence: fgao@gxu.edu.cn; Tel.: +86-18119686504

† These authors contributed equally to this work.

Abstract

The Capacitated Vehicle Routing Problem (CVRP) has a wide range of applications in logistics and transportation. Current metaheuristics typically rely on manually added constraints. A hyper-heuristic framework can reduce the dependency on domain-specific knowledge. Therefore, this research proposes a Clustered Simulated Annealing algorithm (CSA). When generating the initial solution of the distribution path, the CSA adopts the Clustered Clarke–Wright Savings algorithm (CCW), the core of which is to use the K-means algorithm to cluster according to the Euclidean distances between the distribution points. The CCW can reduce the search range of the optimization problem by clustering and generating the initial solution quickly, enabling the CSA to perform better in data processing and real-time updates. The CSA then optimizes the initial solution using the Improved Simulated Annealing Hyper-Heuristic algorithm (ISAHH), divided into upper and lower layers. The Improved Simulated Annealing High-Level Heuristic strategy (ISAHLH) is used to select the Low-Level Heuristic operators (LLHs). At the same time, LLHs are used to generate new distribution paths. This research designs an Improved Tabu Low-Level Heuristic operator (ITabuLLH), which can search for several different paths simultaneously in a single iteration, thus improving the convergence speed of the algorithm. ISAHLH and ITabuLLH both use the Unequal Probability Selection mode (UEPS) to speed up the search process. The CSA is tested on the Uchoa benchmark set, and the results verify that the optimal value improvement of the CSA solution is higher than 20% when compared to eleven other algorithms.

Keywords: CVRP; hyper-heuristic; simulated annealing

MSC: 90C27; 90C59; 90B06

Check for updates

Academic Editor: Janez Žerovnik

Received: 12 January 2026
Revised: 4 February 2026
Accepted: 6 February 2026
Published: 8 February 2026

Copyright: © 2026 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.

1. Introduction

The Capacitated Vehicle Routing Problem (CVRP) stands as a pivotal subset of Vehicle Routing Problems, drawing considerable interest for CVRP's practical applications in areas such as household waste management [1], urban logistics and distribution [2], and emergency rescue and relief operations. CVRP seeks to ascertain the optimal routes for a given number of vehicles. Each vehicle has a defined capacity limit. The goal is to accommodate various customers' unique demands and minimize the total distance covered. The Capacitated Vehicle Routing Problem (CVRP) is recognized as a combinatorial optimization challenge and classified as NP-hard [3–5]. CVRP is closely related to multi-route variants

such as mTSP, and clustering–routing is a widely used strategy in the mTSP literature to reduce problem complexity by partitioning customers before route construction. In real-world logistics, routing decisions are often affected by uncertainties (e.g., demand and travel-time variability) and unexpected events; a recent survey in Mathematics systematically summarizes VRP models and solution methods under uncertainties and highlights the gap between deterministic assumptions and operational reality [6]. Moreover, disruptions during execution are particularly relevant to humanitarian and military logistics; Huang and Xu modeled a multi-period multi-depot VRP under disruption risk (VRPMD) and optimized expected deliveries/costs under probabilistic vehicle disruptions [7]. Recent developments have broadly categorized methodologies for addressing CVRPs into four main types. These categories reflect the varied strategies employed to navigate the complexities of this intricate issue.

The first category includes traditional mathematical planning methods such as column generation, branch-and-bound methods [8–10], Benders decomposition [11–21], and dynamic programming [22–25]. These methods can often provide optimal solutions for small-scale CVRPs within a feasible timeframe. However, as the scale of the problem grows—especially with a significant increase in customers—the computational complexity of these traditional methods surges rapidly. With the problem's complexity escalating, the time required to discover a solution grows exponentially. In addition to classical CVRP formulations, works in Mathematics have also reported exact-modeling efforts for cluster-structured VRP variants; for example, Sabo et al. studied the Selective Vehicle Routing Problem as a more general cluster-based extension of GVRP and developed MIP-based formulations with strengthening strategies [26].

The second category revolves around heuristic algorithms [27–30]. These algorithms seek approximately optimal solutions, focusing on practicality rather than pinpoint accuracy for CVRP. Notable traditional heuristic methods include Tabu Search (TS) [31], Simulated Annealing [32], evolutionary algorithms [5] and genetic algorithms [33–35]. For CVRPs involving fewer than 200 clients, heuristic algorithms often strike an efficient balance between the computational effort and the quality of the solution. They are exceptionally well-equipped to tackle problems of this magnitude. In other words, they might achieve excellent results for specific problems while performing inadequately on others, even when the same algorithm is applied. While heuristic algorithms generally offer quicker solutions than more rigorous mathematical planning approaches, they do not guarantee the discovery of a globally optimal solution. Recently, Arce-Ortega et al. presented a Simulated Annealing approach for homogeneous CVRP enhanced with local search (e.g., hill climbing and a double neighborhood), reporting strong performance on a large set of benchmark instances [36].

The third category focuses on advanced algorithms that utilize reinforcement learning and deep learning methods [37,38]. These algorithms typically require a considerable investment in time and data for practical training. However, their performance in test environments and real-world applications is noteworthy once adequately trained. For example, SIL [39] attempts to solve CVRPs with up to 100,000 clients in a short amount of time. In such instances, deep and reinforcement learning algorithms excel at efficiently navigating the complexities of the problem within a relatively short timeframe. This capability is crucial for rapid adaptation and response to changing demands. Nevertheless, these sophisticated algorithms face significant challenges, including the need for substantial computational resources. The process of tuning the myriad of hyperparameters inherent in deep and reinforcement learning and finding the most effective combination can be complex and time-consuming.

The fourth category of algorithms involves hyper-heuristic approaches, combining intelligent and heuristic algorithms for effective complex optimization challenge resolution. Chen et al. [40] introduced a layered search algorithm based on threshold acceptance, devised specifically for the Capacitated Electric Vehicle Routing Problem (CEVRP). The layered search algorithm, structured in three stages, initially processed a streamlined CVRP version derived from the original CEVRP through an iterative threshold search, producing various CVRP solutions. The subsequent stage employed a threshold selection mechanism to refine and select superior CVRP solutions from the initial outputs. The concluding stage integrated removal heuristic and enumeration strategies, incorporating an optimized charging schedule to evaluate each CVRP solution, thereby identifying the optimal solution for CEVRP. Wang et al. [41] developed a hyper-heuristic algorithm rooted in TS, with upper-tier heuristic operators originating from the TS algorithm and Low-Level Heuristic operators (LLHs) designated as tabu subjects. Hyper-heuristic algorithms are noted for exceptional versatility and adaptability, enabling efficient navigation across various optimization problems. This adaptability is particularly evident in the ability to address new problems with minimal adjustments to LLHs. A key advantage of hyper-heuristic algorithms lies in the ability to swiftly move beyond local optima, exploring a broader solution landscape in pursuit of a global optimum or closely approximated solutions. This research integrates intelligent and heuristic algorithms to develop hyper-heuristic solutions, employing diverse heuristic methods at the lower level to mitigate the inconsistency issues frequently encountered in solution quality when using standalone heuristic algorithms. Additionally, the streamlined nature of these heuristic operators significantly reduces the number of adjustable parameters compared to purely intelligent algorithms, substantially speeding up the solution discovery process.

Hyper-heuristic algorithms [42], reinforcement learning, and deep learning methods often fail to secure an optimal solution. Nevertheless, acquiring an approximate solution closely mirroring the optimal one within permissible time constraints is commonly achievable. Due to the computational time demands, hyper-heuristic algorithms, reinforcement learning, and deep learning methods are regularly employed to address CVRPs in practical scenarios.

The four methodologies above necessitate an initial solution before optimizing the distribution scheme. The Clarke–Wright Savings (CW) algorithm emerges as the predominant method for generating an initial solution for addressing CVRPs with capacity constraints, attributed to its conceptual simplicity and ease of implementation. Can et al. [43] employed the CW to formulate an initial solution, subsequently utilizing a variable neighborhood search algorithm for enhanced solution-seeking. Similarly, Wang et al. [44] initiated the CW to establish the preliminary solution and then integrated the variable neighborhood search algorithm with a multi-objective particle swarm optimization algorithm for improved solution identification. Generating an initial solution for CVRPs with more than 200 clients poses a significant challenge, highlighting the need for further refinement in the application of the CW algorithm. This research proposes the CSA for solving CVRP with the following main contributions.

- CVRP is essential to establish an initial solution before proceeding with the optimization search. The CW is traditionally employed, but generating initial solutions for CVRPs with more than 200 clients is time-consuming. This research introduces the Clustered Clarke–Wright Savings algorithm (CCW) to address this issue. Experiments have demonstrated that this method can generate initial solutions more than tenfold faster than the CW without compromising solution quality. The CCW accomplishes this by clustering the original problem and then solving each cluster separately, effectively reducing the scale of the problem and accelerating the solution process.

- Chen et al. [40] introduced a multi-level search algorithm based on threshold acceptance for obtaining high-quality solutions. However, threshold acceptance was a complex and time-consuming process and the equal probability random search used to select LLHs resulted in inefficient algorithm performance. This research adopts a hyper-heuristic structure with two levels, where the upper level utilizes metaheuristic algorithms, and the lower level includes intra-route local optimization heuristic operators, inter-route local optimization heuristic operators, and metaheuristic optimization heuristic operators. Due to the fewer adjustable parameters of metaheuristic algorithms and their adaptive nature, the process of threshold acceptance can be accelerated. Additionally, this research employs an Unequal Probability Selection mode (UEPS) for choosing LLHs, which increases the likelihood of selecting those that reduce the total delivery route length. Therefore, this approach prevents aimless searches within the solution space and accelerates finding the optimal solution.
- Wang et al. [41] developed a hyper-heuristic algorithm based on the TS, which efficiently selected LLHs through advanced heuristic strategies. However, the Low-Level Heuristic operators were traditional intra-route and inter-route local optimization operators, leading to suboptimal search efficiency. This research introduces an Improved Tabu Low-Level Heuristic operator (ITabuLLH) that enables extensive searches in a single iteration, addressing the issue of search inefficiency.

The structure of this article is organized as follows: Section 2 models CVRP. Section 3 explains the CSA, including the initial solution generation via the CCW and further optimization with the Improved Simulated Annealing Hyper-Heuristic algorithm (ISAHH). Section 4 presents experimental validation using instances from the CVRPLIB [45]. Section 5 concludes the article and presents the future work.

2. Problem Definition

CVRP represents a classic issue within operations research and computer science, concentrating on allocating and scheduling vehicles to deliver a fixed quantity of goods from a central warehouse to multiple client locations efficiently [46]. CVRP considers various practical constraints, including the carrying capacity of each vehicle and the demand at delivery locations. Figure 1 illustrates CVRP: three vehicles embark from the warehouse, each servicing a distinct region, before returning to the warehouse.

CVRP is defined within an undirected graph $G(V, E)$, consisting of vertices $V = \{v_i | i = 0, 1, \ldots, N\}$ and edges $E = \{e_{ij} | i, j = 0, 1, \ldots, N\}$ connecting the vertices. The vertex v_0 represents the warehouse from which all vehicles must depart and to which they must return upon completion of their deliveries. Each of the remaining vertices represents a customer who requires service from the vehicles. For customer v_i in need of services, there is a demand $d_i, 1 \leq i \leq N$. This research assumes that all vehicles have the same capacity Q. A vehicle can successfully service all customers on the route and return to the warehouse only if the total demand of the customers on the route does not exceed the vehicle's total capacity.

CVRP aims to identify vehicle routes that minimize the total distance traveled while servicing all customers [47].

$$min \sum_{i=0}^{N} \sum_{j=0}^{N} C_{ij} x_{ij} \quad (1)$$

Here $C_{ij} \geq 0 (0 \leq i, j \leq N)$ represents the distance from vertex v_i to vertex v_j, while $x_{ij} \in \{0, 1\} (0 \leq i, j \leq N)$ indicates whether the route includes the trip from vertex v_i to vertex v_j ($x_{ij} = 1$ if included, $x_{ij} = 0$ if not included).

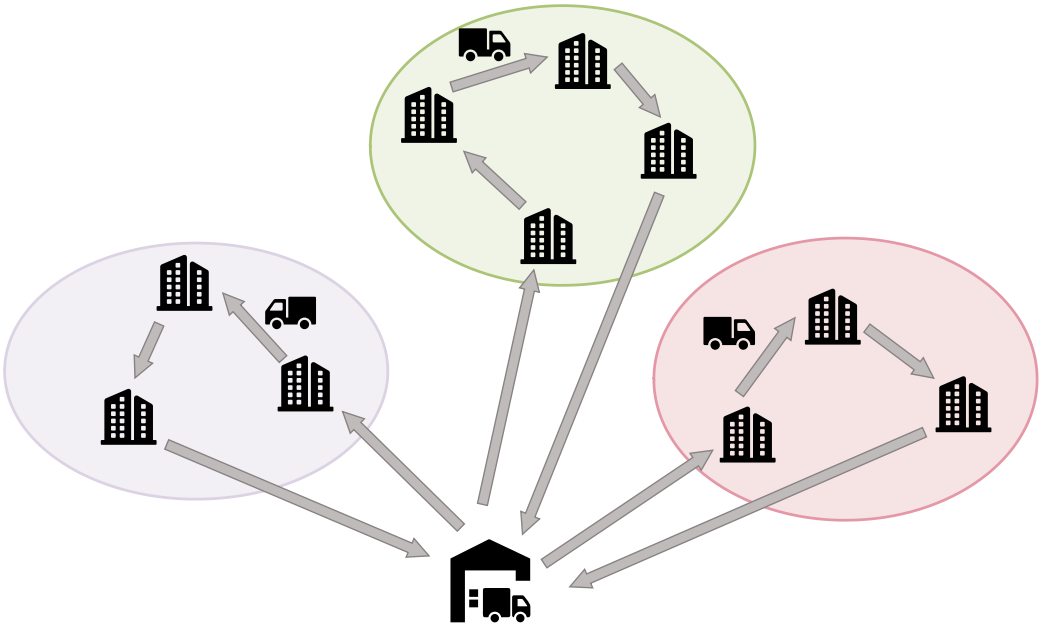


Figure 1. Schematic of the CVRP.

The constraints are:

$$\sum_{i=0}^{N} x_{ij} = 1, \forall j = 1, 2, \cdots, N, i \neq j \tag{2}$$

$$\sum_{j=0}^{N} x_{ij} = 1, \forall i = 1, 2, \cdots, N, i \neq j \tag{3}$$

$$\sum_{j=1}^{N} x_{0j} = K \tag{4}$$

$$\sum_{i=1}^{N} x_{i0} = K \tag{5}$$

$$u_i - u_j + Q x_{ij} \leq Q - d_i, \ \forall i, j = 1, 2, \cdots, N, i \neq j \tag{6}$$

$$(d_i + d_j) x_{ij} \leq Q \tag{7}$$

$$d_i \leq u_i \leq Q, \ \forall i = 1, 2, \cdots, N \tag{8}$$

$$x_{ij} \in \{0, 1\}, \forall i, j = 1, 2, \cdots, N, i \neq j \tag{9}$$

Constraints (2) and (3) ensure that precisely one edge enters and leaves each vertex in the routing. Constraints (4) and (5) stipulate that the total number of vehicles departing from and returning to the warehouse is K. Constraint (6) gives the capacity constraint, where u_i (u_j) represents the load of the vehicle upon entering the vertex v_i (v_j) and d_i denotes the demand of the customer at the vertex v_i. Constraint (7) allows a vehicle to move directly from vertex v_i to v_j only if the vehicle's capacity can accommodate the total demand of customers v_i and v_j. Constraint (8) requires that the vehicle's load upon entering the vertex v_i must meet the demand at that point and be less than or equal to the maximum carrying capacity.

The challenges of CVRP are rooted in the following aspects:

- Enormous search space: As the number of customer points increases, the route combinations rise exponentially, making it hard to traverse efficiently.
- Adaptive requirements: In CVRP, the characteristics and constraints of the problem may differ across application scenarios. For instance, although vehicles usually possess uniform capacity in standard CVRP, variants with vehicles of differing capacities may emerge. Such variation indicates that a solution devised for a particular scenario

might not be directly applicable to another, thereby requiring the formulation of adaptive strategies.

A hyper-heuristic framework can effectively overcome the above-mentioned difficulties:

- Efficient search space exploration: Hyper-heuristic algorithms can effectively navigate the vast search space by intelligently selecting and combining different heuristic rules.
- Strong adaptability and generalization: Hyper-heuristics operate at a high level of abstraction, dealing with the problem indirectly by controlling and managing related LLHs. Thus, they can be applied to different issues by modifying only the LLHs.

3. Proposed Method

Addressing the challenges of CVRP, this research introduces the CSA, which encompasses the CCW depicted in Figure 2a for swiftly generating initial solutions and the ISAHH illustrated in Figure 2b for obtaining the shortest delivery distance based on the initial solutions. The ISAHH comprises the Improved Simulated Annealing High-Level Heuristic strategy (ISAHLH) shown in Figure 2e and LLHs. Beyond traditional LLHs (intra-route and inter-route local optimization operators), this study has also designed the ITabuLLH to generate new solutions, as presented in Figure 2c. The ISAHH and the ITabuLLH both utilize the UEPS, as shown in Figure 2d.

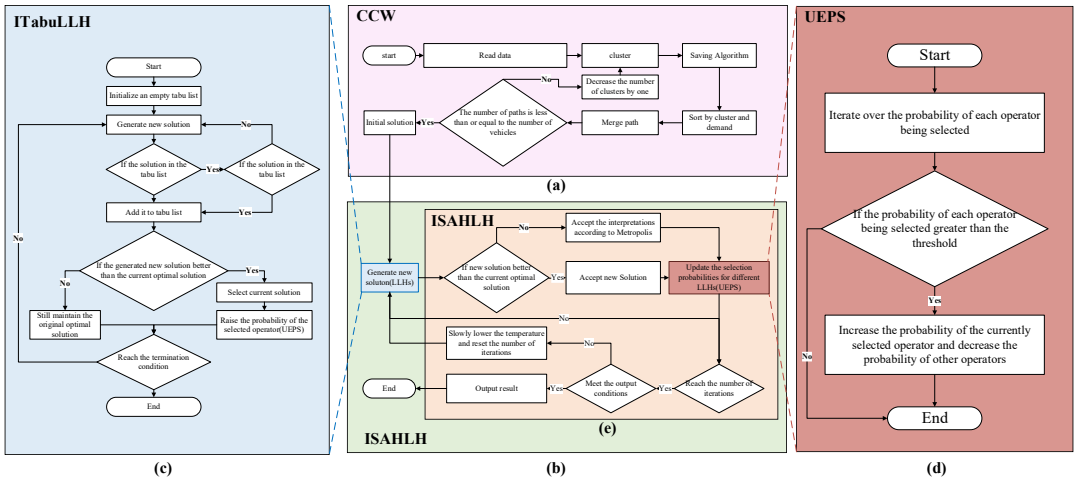


Figure 2. CSA Flowchart. The CSA consists of (**a**) the CCW for generating initial solutions, and (**b**) the ISAHH for further optimization. ISAHH is composed of (**e**) ISAHLH (the high-level strategy that controls the simulated-annealing search process) and a set of low-level heuristics (LLHs). The LLHs include intra-route and inter-route local improvement operators, while (**c**) ITabuLLH is an additional tabu-assisted low-level component designed to jointly participate in generating new solutions and accelerating convergence. Both ISAHH and ITabuLLH adopt (**d**) UEPS to adaptively update the selection probabilities of operators.

For completeness and reproducibility, the CSA framework is summarized in Algorithm 1. Figure 2 illustrates the workflow graphically, whereas Algorithm 1 provides a compact stepwise description of the two-stage process. We then elaborate the CCW initialization in Section 3.1 and the ISA(H)H improvement phase in Section 3.2.

Algorithm 1: I CSA Framework

Input: Customer coordinates c, demand dem, vehicle number limit k, vehicle capacity cap; CCW parameters;
1. Read instance data and construct distance matrix dist.
2. **Initial solution generation (CCW):**
3. Cluster customers using K-means, then apply CW savings within each cluster.
4. Merge routes across clusters to satisfy the vehicle number limit k, obtaining an initial feasible solution S_0.
5. Improvement phase (ISAHH):
6. Call Algorithm ISAHH with S_0 and parameters to obtain an improved solution S^*.
7. **Output:** S^* and its objective value $f(S^*)$.

3.1. Methods for Generation Initial Solutions

3.1.1. Clarke–Wright Savings Algorithm

The CW is a heuristic method introduced by Clarke and Wright in 1964 for addressing the Vehicle Routing Problem (VRP), aimed at minimizing the distance traveled while fulfilling demand requirements [48]. Mainly applicable to logistics and distribution, the CW is often utilized to optimize vehicle delivery routes. The core principle of the CW involves calculating the "savings" value between every pair of customer points, where the value of the savings represents the distance saved by combining two customer points into the same route. This method is also frequently employed to generate initial solutions for CVRP.

The steps of the CW proceed as follows:

- Initialization: Assign each customer point to an individual route, starting from the warehouse, visiting the customer point, and then returning to the warehouse;
- Calculation of savings: Compute the total distance saved for each pair of customer points by merging two points onto the same route. The savings value equals the distance of visiting the two points separately minus the total distance after merging;
- Sorting and merging: Arrange all savings values in order and endeavor to merge the customer points corresponding to the highest savings values onto the same route while accounting for vehicle capacity and other constraints;
- Iteration: Repeat the above process from (a) to (c) until no further savings can be achieved or other stopping criteria are satisfied.

The primary advantage of the CW lies in simplicity and ease of implementation, offering an efficient heuristic solution for complex VRPs. However, a significant limitation of the CW is a tendency to form longer routes: when customer points are unevenly distributed, the CW may overlook shorter and more compact routes. Additionally, the CW's performance is influenced by the initial solution and the method used to calculate savings values. In dealing with CVRPs, directly applying the CW can be time-consuming. As the number of customer points increases, the computation time escalates exponentially [49]. This research proposes that the CCW generate initial solutions for CVRP to address this issue.

3.1.2. Clustered Clarke–Wright Savings Algorithm

We improve the CW and propose the CCW with the following steps:

- Clustering is performed using the K-means algorithm based on the Euclidean distance between the distribution points;
- Initial solutions are generated for each class within the class using the CW;
- The initial solutions of each class are combined to generate the overall initial solution.

The specific procedure of the CCW is depicted in Figure 2a. The theoretical upper limit of clustering is reached when each point is assigned to its own cluster, whereas the theoretical lower limit occurs when all points are grouped together in a single cluster. If each point forms a separate cluster, the number of routes after merging often significantly exceeds the routes required by the problem set. Conversely, if all points are in one cluster, the subsequent running time of the CW becomes excessively long. To balance these extremes, we adopt a capacity-guided rule to initialize and search the number of clusters. Let $D = \sum_i dem_i$ denote the total demand and cap the vehicle capacity. We set the initial cluster number as $K_0 = \lfloor D/cap \rfloor + 1$, which provides a practical lower-bound-based estimate of the required number of routes. The CCW starts from $n_{clusters} = K_0$ and decreases it by one each time. For each $n_{clusters}$, we generate cluster-wise CW routes and then merge them to obtain an overall initial solution. The first feasible solution satisfying the vehicle-limit constraint ($|Routes| \leq n_{vehicles}$) is accepted and output. Otherwise, the number of clusters is reduced by one. This adaptive reduction-and-fallback procedure serves as a lightweight feasibility-repair mechanism, ensuring that a valid initial solution can still be produced when a specific clustering instance is infeasible. In the worst-case scenario for CCW, where $n_{clusters} = 1$, the algorithm reverts to the traditional Savings algorithm. In this study, we use the Euclidean distance on the 2D customer coordinates for both clustering and route cost evaluation, i.e., $d(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, which is consistent with the standard Euclidean metric used in most CVRP benchmark instances. It should be noted that the clustering procedure does not guarantee the existence of a feasible solution for every clustering instance. Not every cluster strictly represents a set of points belonging to a single path. Instead, we assess whether a more optimal path solution can be identified within the cluster. Consequently, multiple paths may exist within a single cluster in certain cases. Given the conventional Savings algorithm reliably generates a valid initial solution, the CCW ensures the production of an initial solution satisfying the requirements.

3.2. Improved Simulated Annealing Hyper-Heuristic Algorithm

This research developed the ISAHH to explore optimal solutions based on initial ones. As illustrated in Figure 2b, the algorithm is structured into two levels: the upper level utilizes the ISAHLH to select LLHs; the lower level applies these LLHs to adjust the order of customer visits within or between routes, thereby seeking potentially superior solutions. The LLHs primarily include intra-route local optimization heuristic operators, inter-route local optimization heuristics, and the ITabuLLH. ISAHLH and ITabuLLH incorporate the UEPS to update the probability of operator selection.

3.2.1. Improved Simulated Annealing High-Level Heuristic Strategy

The Simulated Annealing algorithm (SA) is a probabilistic optimization algorithm designed to find an approximate global optimum solution for a given problem [50]. The SA employs a cooling strategy derived from the annealing process to control the search procedure. The SA has been widely applied to optimization problems across various fields, including engineering, computer science, economics, and bioinformatics. The primary advantage of the SA is its effectiveness in avoiding entrapment in local optima, thereby increasing the likelihood of identifying the global optimum. However, the efficiency and performance of the SA are highly dependent on the parameter settings and the specific characteristics of the problem at hand.

$$P = \begin{cases} 1, & \Delta E < 0 \\ e^{-\Delta E/T}, & \Delta E \geq 0 \end{cases} \tag{10}$$

This research, inspired by the principles of Simulated Annealing, developed the ISAHLH. As illustrated in Figure 2e, ISAHLH selects an LLH to apply to the delivery route, generating a new total transportation distance. ISAHLH then employs the Metropolis criterion as shown in (10) to decide whether to accept this new solution based on the updated transportation distance. If accepted, the global optimum is updated. P is the probability of accepting the new solution, ΔE is the difference between the current solution and the global optimal solution, and T is the current temperature. The ISAHLH uses the UEPS when choosing LLHs, aiming to enhance the algorithm's directionality by favoring those operators that reduce the total delivery distance.

Traditional hyper-heuristic algorithms often rely on randomly selecting LLHs, assigning equal probability to each choice. This reliance on chance implies a reliance on luck to discover improved solutions. LLHs encompass intra-route local optimization operators, inter-route local optimization operators, and metaheuristic optimization operators. Traditional hyper-heuristic algorithms choose one LLH at a time to act on the delivery route. Traditional heuristics view reductions in the total delivery distance as merely coincidental and do not adjust the selection probability accordingly. In contrast, ISAHLH augments the probability of selecting LLHs associated with better outcomes, imbuing the algorithm with a memory-like feature. Accelerating the search speed aids in swiftly converging toward optimal solutions.

3.2.2. Low-Level Heuristic Operators

1. Intra-route local optimization operators

Figure 3a illustrates the intra-route exchange of route segments operator [51], which reverses the order of a segment within the delivery route (from 1-3-5-4 to 4-5-3-1). Figure 3b depicts the intra-route exchange of specific points operator [51], which swaps the delivery sequence of two points within a segment of the delivery route (points 1 and 5). Figure 3c shows the intra-route relocation of a point operator [51], which relocates a specific point (point 1) to a predetermined position within the route (between points 4 and 6).

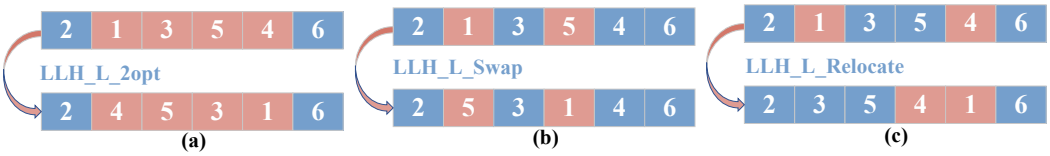


Figure 3. Intra-route local optimization heuristic operators. (**a**) Exchange of route segments (2-opt): reversing the order of a segment within the route (e.g., 1−3−5−4→4−5−3−1). (**b**) Exchange of specific points: swapping the positions of two customers in the same route (e.g., customers 1 and 5). (**c**) Relocation of a point: relocating a customer to a predetermined position within the route (e.g., moving customer 1 between customers 4 and 6). Colors are used only for visual highlighting of the affected customers/segments.

2. Inter-route local optimization operators

Figure 4a depicts the inter-route exchange of route segments operator [51], where segments from two different routes (1-3-5-4 and 7-11-10-9) are first reversed and then swapped between the two routes. Figure 4b shows the inter-route exchange of specific point operators [51], which involves swapping given points (points 1 and 11) between two different routes. Figure 4c illustrates the inter-route relocation of a point operator [51], which relocates a specific point (point 5) from one route to a predetermined position in another route (between points 7 and 11).

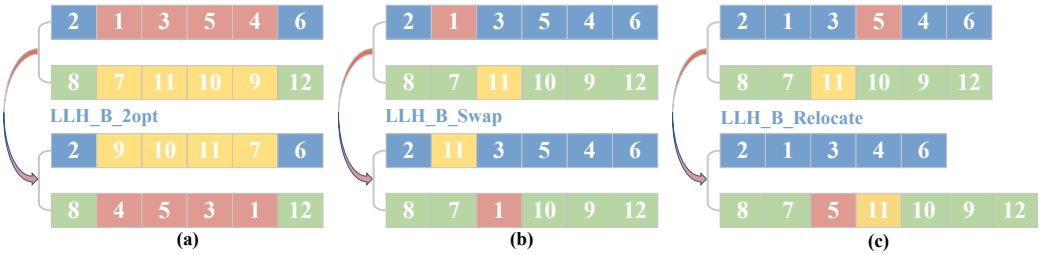


Figure 4. Inter-route local optimization heuristic operators. (**a**) segment exchange (reverse and swap two selected segments between routes); (**b**) point swap (swap two customers between routes); (**c**) point relocate (move a customer from one route to a specified position in another route). Colors are for visualization only.

3. Metaheuristic optimization operators

The ISAHH utilizes the ITabuLLH as a metaheuristic optimization operator. Traditional intra-route and inter-route local optimization operators are limited to searching by adjusting positions within a single route or between two routes simultaneously. In contrast, metaheuristic operators can explore across a broader scope of routes. The ITabuLLH modifies the total delivery route length by executing transformations on route positions akin to those depicted in Figures 3 and 4 and initializes an empty tabu list to record a series of moves or encountered solutions during the search process. Solutions encountered are marked as "tabu" for a specified period, preventing their repetition or re-exploration within this timeframe. This mechanism facilitates the exploration of a wider solution space, aiding in escaping local optima during the search process. Each newly generated solution is appended to the tabu list in every iteration. If a newly generated total route length is shorter than the current optimum, it satisfies the aspiration criterion, allowing the selection of this newly untabooed route as the current optimal solution. Adhering to this approach, ITabuLLH is more inclined to discover superior route combinations within a designated number of iterations. Additionally, as shown in Figure 2c, during the routing location transition, ITabuLLH utilizes UEPS to maximize the probability of selecting operators that contribute to reducing the total path length.

3.2.3. Unequal Probability Selection Mode

For CVRP, if vehicles have a higher capacity, intra-route local optimization operators should be used more frequently, as frequent use of inter-route local optimization operators could result in some vehicles being fully loaded while others carry significantly less. Conversely, if the vehicles have a lower capacity, the frequency of using both intra-route and inter-route local optimization operators may be comparable.

Based on the analysis above, this research has designed the UEPS depicted in Figure 2d: In each iteration, the ISAHLH or ITabuLLH increases the selection probability for each operator or route position transformation that can reduce the total delivery route length, while simultaneously decreasing the selection probability for other operators or route position transformations. The objective is to incentivize the algorithm to predominantly opt for LLHs that are inclined to decrease the overall length of the delivery route. To ensure that each type of operator or route position transformation has a chance of being selected and to avoid scenarios where non-selection leads to permanent exclusion, a threshold is established as the minimum selection probability. This threshold is a customizable parameter that can be adjusted to suit various circumstances. The probability increase in UEPS is also a variable parameter, with optimal settings capable of improving the search efficiency.

The pseudocode for the ISAHH algorithm is presented as Algorithm 2.

Algorithm 2: ISAHH Algorithm

Input: Each point's coordinates (c), demand (dem), number of delivery vehicles (k), and maximum vehicle capacity (cap);

```
Generate the initial solution using the CCW;
while the current temperature is above the termination threshold, and the number of vehicles required in the instance cannot complete the entire delivery task without exceeding the maximum load of each vehicle do:
    Count = 0; // Number of LLH executions within each temperature
    while Count < 100 do:
        Generate a new solution using LLH;
        if the newly generated solution is less than or equal to the current best so solution:
            Accept the new solution;
        end
        else:
            Accept the new solution based on the Metropolis;
        end
        if accept the solution:
            Count = Count + 1;
            Increase the probability of selecting this LLH;
        end
    end while;
    if the number of vehicles required in the instance can complete the entire delivery task without exceeding the maximum load of each vehicle:
        Output the result;
        break;
    end
    else:
        Reduce the temperature;
    end
end while;
```

Output: Vehicle delivery routes that meet the test instances requirements.

4. Experimental Result and Discussion

This section conducts four experiments to validate the superiority of the CSA comprehensively. First, we compare the efficiency of different methods in generating initial solutions. Next, we assess the overall performance of the CSA by comparing its solutions with those obtained from other methods using the Uchoa benchmark set in CVRPLIB [52]. Subsequently, we perform ablation studies to examine the impact of various improvements within CSA. Finally, we compare CSA with the mainstream route planning solver, OrTools [53], using the Set M benchmark set in CVRPLIB, thereby confirming CSA's superiority in achieving optimal solutions.

All experiments and comparative algorithms presented in this section were implemented in Python 3.8.20and executed on a PC configured with an Intel Core i5-10400F 2.90 GHz CPU, 16.00 GB RAM, and a Windows 10 64-bit operating system. Unless otherwise specified, each stochastic algorithm is executed 20 independent runs with different random seeds on each instance. Best denotes the best result among the 20 runs, and Avg $\pm$ Std denotes the mean $\pm$ standard deviation over the 20 runs. For all stochastic algorithms, the termination condition is defined as follows: the search stops if the improvement of

the objective value is less than 0.1 for 10 consecutive iterations, or when the maximum CPU time of 10 min is reached, whichever occurs first. This stopping policy is applied consistently across all compared methods to ensure a fair comparison. The implementation uses NumPy1.24.3, scikit-learn1.3.2 (KMeans), NetworkX3.1, and Matplotlib3.7.5.

4.1. Comparing the Efficiency of Different Methods in Generating Initial Solutions

4.1.1. CCW Algorithm Execution Flow

This subsection illustrates the process of the CCW using an instance $A - n32 - k5$ from the Set A benchmark set in CVRPLIB. The Set A benchmark set is among the classic sets in CVRPLIB, where the instances have a moderate number of customer points, moderate vehicle numbers, and moderate vehicle capacities. These characteristics make it ideal for providing a more intuitive demonstration of the CCW's execution process. Figure 5 displays the step-by-step process of generating initial solutions by the CCW for this instance. Figure 5a shows the clustering diagram with five clusters, where each grey point represents the centroid of the current cluster, and each cluster is denoted by a different color. At this stage, no satisfactory initial solution has been generated. According to the algorithm's rules, the number of clusters must be reduced by one to allow more points within a cluster. Figure 5b is the clustering diagram with four clusters. Four clusters exist, and a satisfactory initial solution has been generated. The algorithm then outputs the satisfactory initial solution, as shown in Figure 5c, where each color-coded loop represents the delivery route of one vehicle.

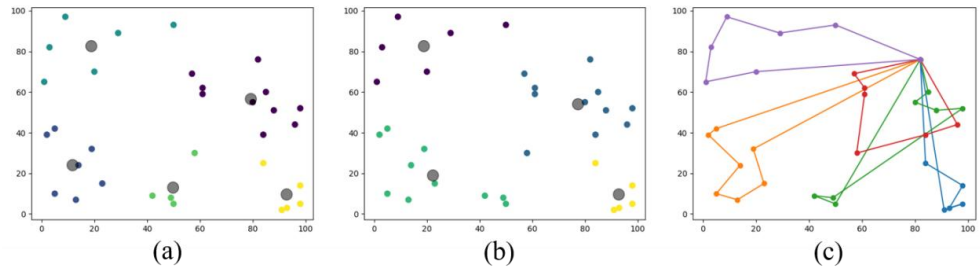


Figure 5. Diagram of the initial solution generation process. (**a**) Clustering visualization with **five clusters** (grey points denote cluster centroids; colors indicate different clusters). (**b**) Clustering visualization with **four clusters** after reducing the number of clusters by one (grey points denote cluster centroids; colors indicate different clusters). (**c**) The resulting **initial feasible routes**, where each **color-coded loop** represents the route of one vehicle.

4.1.2. Baseline Methods for Comparison

This subsection compares the CCW with the Round Trip [54] and the CW [51]. The Round Trip algorithm is a classic method for solving VRPs, commonly used in scenarios involving return requirements, where vehicles must reach designated customer points and return to their point of departure. The Round Trip algorithm typically combines heuristic methods and local search strategies to enhance the efficiency and effectiveness of route planning, finding widespread applications in logistics, distribution, and transportation planning. The core of the CW is to optimize the round-trip routes of vehicles to minimize the total driving distance, adhering to constraints such as vehicle capacity and time windows. The CW is simple and efficient, However, for CVRPs with more than 200 customer points, its performance becomes insufficient.

4.1.3. Initialization Results and Analysis

The average values and times for generating initial solutions by different algorithms are presented in Table 1, with data for Round Trip and the CW entirely sourced from Ref. [54], and the shortest times are highlighted in bold. Across all instances, the CCW

significantly reduced the time to generate initial solutions of comparable quality to the comparison algorithms. Notably, in the fifth instance, the CCW's runtime was approximately 1e-5 times that of Round Trip. The core concept of CCW is to enhance algorithm efficiency by reducing the search space. Specifically, CCW first partitions the customers in the problem into several smaller subgroups based on Euclidean distance using the K-means method, a process termed "customer clustering." This clustering process essentially acts as a data pre-processing step to simplify the initially large and complex problem into smaller, more manageable subproblems. This simplification makes the problem easier to solve and reduces computational complexity. Once customers are divided into smaller groups, the Savings algorithm can be applied within these subsets, significantly reducing the number of route combinations that need to be considered, thus speeding up the solution process. To make the runtime comparison in Table 1 well-defined, we note that the reported time corresponds to the initial solution generation stage. This is because Ref. [54] provides runtime results for the baselines (Round Trip and CW) only for the initialization stage, while the subsequent improvement stage is mainly evaluated in terms of solution quality. For CCW, the initialization time includes a K-means clustering pre-processing step, followed by cluster-wise CW route construction and route merging with feasibility checking. Although clustering introduces extra computations, this overhead is bounded and controllable in our implementation: the number of clusters is decreased progressively and the procedure stops once a feasible solution satisfying the vehicle-limit constraint ($|Routes| \leq n_{vehicles}$) is obtained; In the worst case, the method degenerates to the conventional Savings algorithm when $n_{clusters} = 1$. As a result, clustering acts as a manageable pre-processing component and does not dominate the overall runtime in practice, while helping reduce the search space and improve the initial solution quality for the subsequent optimization stage.

Table 1. Average generated initial solution values and computation time by each algorithm. Data for Round Trip and CW are sourced from [54].

	Round Trip [54]		**CW [54]**		**CCW**	
Instance	**Avg**	**Time (min)**	**Avg**	**Time (min)**	**Avg**	**Time (min)**
n200-k36	61,205.2	31.39	61,167	15.11	62,607	4.11
n251-k28	41,045.73	55.82	40,576	22.29	41,256.53	2.24
n308-k13	27,453.33	191.77	28,555	49.8	29,460.18	8.217
n351-k40	28,049.7	200.54	27,123	52.2	27,992.4	20.28
n376-k94	151,778.8	487.78	149,659	279.02	184,800	0.0077
n401-k29	69,031.8	250.22	68,975	80.81	71,375.8	39.76
n420-k130	113,619.8	1002.74	112,604	517.06	113,648	45.18
n449-k29	59,424.9	322.63	58,614	75.6	60,149.57	57.76

4.2. Comparing Solutions of Different Methods on the Uchoa Benchmark Set

4.2.1. Comparison Algorithms

This subsection utilizes eight instances from the Uchoa benchmark set within CVRPLIB. The Uchoa benchmark set in CVRPLIB serves as a standardized set extensively used for researching and testing algorithms for CVRP, comprising a series of Vehicle Routing Problem instances of varying sizes and complexities. What sets the Uchoa benchmark set apart is its diversity and comprehensiveness, spanning scenarios from small to large scales, thus providing a challenging test platform for algorithm designers. Among the selected instances, the smallest involves 200 customers, while the largest has 449.

We compared the solution outcomes of several algorithms on the Uchoa instances in CVRPLIB [52], including MHTD [54], MHBU [54], CW [54], OrTools, ABSPHH [54], Genetic Algorithm (GA), Ant Colony Optimization (ACO), Elitist Ant System (EAS), Max-Min Ant System (MMAS), Variable Neighborhood Search (VNS), Adaptive Large Neighborhood

Search (ALNS) and CSA. Among these methods, GA, ACO, and VNS were implemented by us following their standard (naive) formulations, without CVRP-specific enhancements or extensive fine-tuning. In addition, improved variants such as EAS, MMAS, and ALNS were tested using publicly available implementations as baselines. All stochastic algorithms follow the same stopping policy and time budget described at the beginning of Section 4. MHTD employs a heuristic method that searches broadly before narrowing down, initiating with a broad scope before moving to more detailed searches. Conversely, MHBU adopts a heuristic strategy that begins with a narrow focus, gradually broadening from detailed searches to encompass more comprehensive ones. OrTools is an open-source framework developed by Google that is designed to tackle various optimization problems such as vehicle routing, flows, integer and linear programming, and constraint programming. ABSPHH is a perturbation-based hyper-heuristic algorithm introduced in recent years, showing promising results in handling CVRP challenges. GA is a heuristic algorithm based on the principles of natural selection and genetics. ACO is inspired by the foraging behavior of ants in nature. EAS is a variation in ACO that accelerates convergence by reinforcing the pheromone concentration on the optimal path in each iteration. EAS introduces the concept of elite ants, which represent the current best solution and have a greater influence on other ants' path choices during pheromone updates. MMAS is an improved version of ACO. It enhances search efficiency and convergence by limiting the range of pheromone concentrations and using a strategy to intensify pheromone levels on the best solutions. VNS is a metaheuristic algorithm that systematically changes the neighborhood structure to explore the solution space, helping to avoid local optima and find the global optimum. ALNS is another metaheuristic algorithm that dynamically selects and adjusts a set of destruction and repair operators. This approach allows for extensive exploration of the solution space, improving both the quality of solutions and the efficiency of the algorithm.

4.2.2. Experimental Results and Analysis

CSA parameter settings are displayed in Table 2. Solution results for different algorithms on Uchoa instances are presented in Table 3, where data for MHTD, MHBU, CW, and ABSPHH come from Ref. [54], while data for all other algorithms are produced by independent implementations. Optimal values for the best and average results across each instance are highlighted in bold, with BKS representing the best-known solution and an asterisk indicating a confirmed optimal solution. The comparison reveals that CSA outperforms the ABSPHH and other algorithms regarding average values on the second, third, fourth, fifth, seventh, and eighth instances. Regarding best values, CSA is superior to the other algorithms in the second, third, fifth, and eighth instances. CSA's advantage likely stems from several factors: CCW providing high-quality initial solutions quickly; UEPS swiftly selecting LLHs suitable for the instance, thereby accelerating the solution search; and the Metropolis criterion preventing the search from getting stuck in local optima. MHTD, MHBU, and ABSPHH employ the Round Trip [54] method for generating initial solutions. Compared to Round Trip, CCW may exhibit slightly inferior initial solution quality but offers faster generation speed, as verified in Table 1 in Section 4.1.3. Despite the slightly lower quality of initial solutions, the additional improvements beyond CCW still enable CSA to surpass algorithms like MHTD, MHBU, and ABSPHH regarding best and average values. Since the GA, VNS, ALNS, ACO, EAS, and MMAS algorithms in Table 3 typically generate initial solutions using random or greedy methods, their initial solutions are often of lower quality. Furthermore, the optimization direction of these heuristic algorithms is closely tied to their hyperparameters, which need to be carefully tuned for good results. CSA, on the other hand, builds upon a well-defined initial solution, allowing it to avoid the pitfalls of random search and achieve better optimization. The experiments demonstrate

that CSA consistently produces valid solutions for each instance, with the GAP value generally remaining within 5%.

Table 2. CSA parameter settings table.

Parameter Name	Value
Initial Temperature	100
Termination Temperature	0.01
Temperature Reduction Factor	0.99
Simulated Annealing Iterations	100
Tabu List Size	100
Tabu Search Iterations	100
Operator Selection Probability Increase Rate	0.001
Operator Selection Probability Decrease Stopping Threshold	0.13

Table 3. Optimal solutions and averages for various instances by each algorithm. Data for MHTD, MHBU, CW and ABSPHH are all sourced from [54]. Data for all other algorithms are produced by independent implementations. Unless otherwise specified, results for independently implemented stochastic algorithms are obtained from 20 independent runs with different random seeds; "Best" denotes the best result among the 20 runs and "Avg" denotes the mean over the 20 runs (the corresponding standard deviations are reported separately in Table 7 for CSA).

Instance		n200-k36	n251-k28	n308-k13	n351-k40	n376-k94	n401-k29	n420-k130	n449-k29
BKS		58,578 *	38,684 *	25,859	25,928	147,713 *	66,187	107,798 *	55,269
MHTD [54]	Best	62,306	41,787	29,012	29,253	159,458	73,098	115,910	61,982
	GAP	6.3%	8.0%	12.2%	12.8%	7.9%	10.4%	7.5%	12.1%
MHBU [54]	Best	62,067	41,960	28,374	28,263	152,638	70,183	115,640	60,376
	GAP	5.9%	8.4%	9.7%	9.0%	3.3%	6.0%	7.3%	9.2%
CW [54]	Best	61,167	40,576	28,555	**27,123**	149,659	68,975	**112,604**	58,614
	GAP	4.4%	4.8%	10.4%	4.4%	1.3%	4.2%	4.4%	6.0%
OrTools [53]	Best	~	41046	31,110	~	150,140	~	~	71,930
	GAP	~	6.1%	20.3%	~	1.6%	~	~	30.1%
ABSPHH [54]	Best	**60,959**	40,512	27,150	27,641	150,715	**68466**	112,963	58,783
	Avg	**61,205.2**	41,045.73	27,506.76	28,049.7	151,778.76	**69,031.8**	113,619.77	59,424.9
	GAP	4.4%	6.1%	6.3%	8.1%	2.7%	4.2%	5.4%	7.5%
GA	Best	68,378.55	55,265.49	52,935.68	44,279.57	165,100.22	98,824.90	139,113.35	105,449.19
	Avg	69,438.75	56,253.30	54,969.38	47,962.80	170,190.82	101,825	143,014.09	115,309.93
	GAP	18.5%	45.4%	112.6%	85.0%	15.2%	53.8%	32.7%	108.6%
VNS	Best	75,389.13	60,986.25	43,802.71	49,366.38	191,439.18	92,536.47	163,531.08	91,462.83
	Avg	78,904.97	63,838.97	45,316.30	52,362.19	196,149.59	95,846.39	168,713.03	99,586.25
	GAP	34.7%	65.0%	75.2%	102.0%	32.8%	44.8%	56.5%	80.2%
ALNS	Best	79,591.33	71,811.61	56,654.65	38,414.08	204,917.29	101,540.04	159,239.32	93,826.36
	Avg	79,593.78	71,818.34	56,660.30	38,417.98	204,923.63	101,545.86	159,239.93	93,832.04
	GAP	35.9%	85.7%	119.1%	48.2%	38.7%	53.4%	47.7%	69.8%
ACO	Best	63,658.79	44,209.23	34,950.13	29,923.67	197,987.3	72,611.74	127,094.8	64,138.84
	Avg	64,405.35	45,436.81	35,409.91	30,817.09	198,179.49	73,785.11	129,010.20	66,243.57
	GAP	9.9%	17.5%	36.9%	18.9%	34.2%	11.5%	19.7%	19.9%
EAS	Best	63,484.31	43,626.69	32,916.98	29,848.73	196,448	72,371.3	126,117.26	63,849.25
	Avg	63,927.78	44,289.45	33,925.62	30,567.39	196,962.15	73,072.52	127,389.27	64,413.36
	GAP	9.1%	14.5%	31.2%	17.9%	33.3%	10.4%	18.2%	16.5%
MMAS	Best	61,996.7	42,750.95	31,989.67	29,848.73	194,953.47	72,275.04	126,117.26	63,849.25
	Avg	63,209.54	43,222.59	33,040.15	30,315.16	195,279.47	72,883.62	127,344.31	64,399.48
	GAP	7.9%	11.7%	27.8%	16.9%	32.2%	10.1%	18.1%	16.5%
CSA	Best	61,812.74	**40,432.26**	**27,034.63**	27,253.18	**149,279.6**	69,533.48	113,185.63	**58,512.61**
	Avg	61,932.56	**40,873.85**	**27,158.38**	**27,934.74**	**150,058.76**	69,965.96	**113,354.12**	**58,538.21**
	GAP	5.7%	5.7%	5.0%	7.7%	1.6%	5.7%	5.2%	5.9%

Note: * indicates a confirmed optimal solution.

4.2.3. Convergence Behavior of CSA over Elapsed Time

To further illustrate the search dynamics of CSA beyond the final "Best/Avg" statistics reported in Table 3, we provide time-based convergence curves on two representative Uchoa benchmark instances, namely n101-k10 and n151-k12. For each instance, CSA is executed for 20 independent runs with different random seeds, following the same experimental protocol as in Section 4.2.2. During each run, we record the best-so-far objective value (i.e., the running minimum) as a function of elapsed CPU time during the ISAHH search phase. The convergence curves in Figure 6 report the average best-so-far trajectory over 20 runs, while the shaded region indicates ± 1 standard deviation, reflecting the run-to-run variability of this stochastic search process.

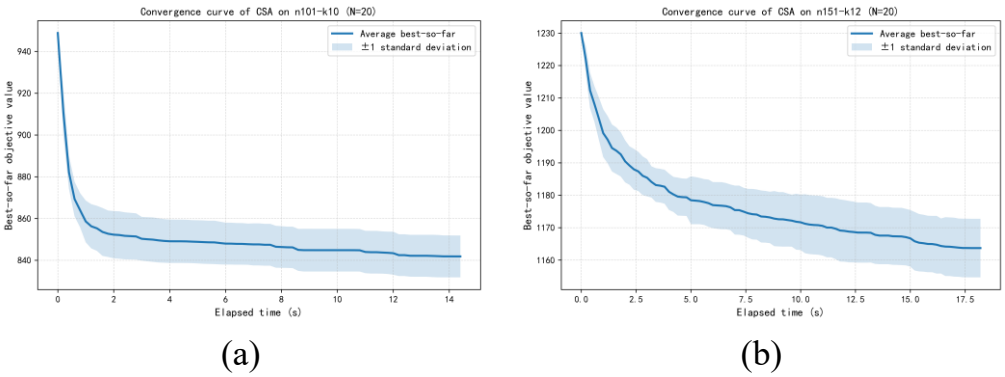


Figure 6. Time-based convergence curves of CSA on two Uchoa benchmark instances: (**a**) n101-k10 and (**b**) n151-k12. Each curve reports the average best-so-far objective value (running minimum) over 20 independent runs ($N = 20$), and the shaded area represents ± 1 standard deviation.

As shown in Figure 6a,b, CSA typically exhibits a clear two-phase convergence pattern. In the early stage (under our experimental setting), the objective value drops rapidly, indicating that the ISAHH search can quickly refine the CCW-initialized solution once the search starts and obtain substantial improvements. After this rapid descent, the curve enters a gradual improvement (plateau) phase, where the best-so-far value continues to decrease but at a slower rate, suggesting diminishing returns as the search approaches a local optimum under a fixed time budget. This behavior is consistent with the design goal of CSA: CCW provides a structured initial solution, and ISAHH further explores the neighborhood via low-level operators to deliver additional improvements.

In addition, the standard deviation bands provide insight into the stability of CSA across repeated runs. Run-to-run variability is present throughout the search due to different random seeds leading to different operator selections and neighborhood trajectories. In these instances, the variability is relatively small at the beginning but becomes more pronounced as the search progresses, mainly because different runs may converge to different local optima and exhibit different improvement timings in the later plateau phase. Overall, these time-based convergence curves complement Table 3 by revealing not only the final solution quality, but also how quickly CSA improves the objective and how stable its improvement behavior is across repeated runs.

4.3. Ablation Experiments

4.3.1. Ablation Study of CCW and UEPSComparison Algorithms

This subsection compares the impact of CCW and UEPS on the performance of CSA. The strategy employed by CCW allows for the generation of high-quality initial solutions quickly, laying a robust groundwork for subsequent route optimization and crafting the final solutions. UEPS prioritizes operators that are more likely to improve the current

solution, thereby maintaining a certain breadth of search while enhancing efficiency. The application of CCW and UEPS plays a crucial role in balancing the search's depth and breadth and improving the algorithm's overall performance. To prevent the process from becoming excessively time-consuming, a maximum duration of 10 min is set for generating initial solutions in this subsection. In the ablation study, the CSA also utilizes the parameters outlined in Table 2.

4. Experimental Results and Analysis

Table 4 showcases the results of ablation studies focused on CCW and UEPS, where BKS denotes the best-known solution, and an asterisk indicates a confirmed optimal solution. The optimal values for each instance are highlighted in bold. In the table, CW + SAHH represents a standard Simulated Annealing Hyper-Heuristic algorithm (SAHH) without CCW and UEPS; CCW + SAHH represents SAHH combined with CCW; CW + SAHH + UEPS signifies SAHH integrated with UEPS; and CSA incorporates both CCW and UEPS. The results illustrate that the CCW + SAHH algorithm outperforms the CW + SAHH algorithm, demonstrating CCW's ability to accelerate the algorithm and ensure solution quality. The CW + SAHH + UEPS algorithm surpasses the CW + SAHH algorithm, highlighting that UEPS's dynamic adjustment mechanism enhances the algorithm's flexibility and ability to tackle complex problems. UEPS effectively reduces the selection of ineffective or inefficient operations, thereby speeding up the process of finding the optimal solution in dealing with complex optimization problems.

Table 4. Ablation experiments on CCW and UEPS.

		CW + SAHH		**CCW + SAHH**		**CW + SAHH + UEPS**		**CSA**	
Instance	**BKS**	**Best**	**GAP**	**Best**	**GAP**	**Best**	**GAP**	**Best**	**GAP**
n200-k36	58,578 *	62,277.26	6.32%	62,225.43	6.22%	62,233.2	6.24%	**61,812.74**	5.52%
n251-k28	38,684 *	41,200.29	6.50%	40,898.26	5.72%	41,185	6.47%	**40,432.26**	4.52%
n308-k13	25,859	28,621.24	10.68%	27,454.25	6.17%	28,517.95	10.28%	**27,034.63**	4.55%
n351-k40	25,928	27,491.78	6.03%	27,463.1	5.92%	27,466.82	5.93%	**27,253.18**	5.11%
n376-k94	147,713 *	155,399.5	5.20%	155,343.5	5.17%	155,534.5	5.30%	**149,279.59**	1.06%
n401-k29	66,187	69,815.98	5.48%	69,603.98	5.16%	69,760.61	5.40%	**69,533.48**	5.06%
n420-k130	107,798 *	113,647	5.43%	113,484.12	5.27%	113,632.7	5.41%	**113,185.63**	5.00%
n449-k29	55,269	58,654.87	6.13%	58,602	6.03%	58,557.44	5.95%	**58,512.61**	5.87%

Note: * indicates a confirmed optimal solution.

Further comparison between CCW + SAHH and CW + SAHH + UEPS reveals that, except for the last instance where CCW + SAHH falls short, CCW + SAHH's results are better than those of CW + SAHH + UEPS in other instances, indicating a significant impact of CCW on the algorithm's best values. This result is reasonable, as initial solutions generally substantially influence initial solutions in hyper-heuristic algorithms. Combining CCW and UEPS significantly enhances the quality of solutions, as evidenced by the comparison showing that CSA's best values surpass those of CW + SAHH, CCW + SAHH, and CW + SAHH + UEPS across all instances.

4.3.2. Ablation Study of ITabuLLH

1. Comparison Algorithms

Given that high-level heuristic strategies cannot guarantee the selection of the same LLHs in each iteration or that the same LLHs conduct identical position transformations, solely demonstrating the superiority of ITabuLLH based on optimal values presents a challenge. In this research, the design goal of ITabuLLH is to accelerate the convergence speed of CSA by conducting larger-scale route searches within a single iteration. Therefore, this subsection's experiment will test the number of iterations required to converge to the

same total route length (to two decimal places) to verify the impact of ITabuLLH on the convergence speed of CSA. The experiment also utilizes the parameters shown in Table 2.

2. Experimental Results and Analysis

The ablation study results for the ITabuLLH are presented in Table 5, where the minimum number of iterations for each instance is highlighted in bold. CCW + SAHH + UEPS represents the SAHH combined with the CCW and UEPS. CSA denotes the SAHH integrated with CCW, UEPS, and ITabuLLH. The table shows that CSA requires fewer iterations to converge on all instances than CCW + SAHH + UEPS, confirming that ITabuLLH can achieve faster convergence speeds by conducting large-scale searches within a single iteration.

Table 5. Ablation experiments on ITabuLLH.

	CCW + SAHH + UEPS	**CSA**
Instance	**Iterations**	**Iterations**
n200-k36	654	405
n251-k28	223	112
n308-k13	916	779
n351-k40	222	154
n376-k94	909	216
n401-k29	865	625
n420-k130	434	276
n449-k29	845	751

4.4. Comparing the Performance of OrTools and CSA on the Set M Benchmark Set

This section compares the performance of OrTools and CSA on the Set M benchmark set, which is relatively smaller in scale. If OrTools fails to produce a feasible solution within a specific timeframe, it outputs that no solution is available. As shown in Table 3, using the OrTools algorithm, half of the Uchoa benchmark set did not yield a feasible solution. Therefore, this subsection employs instances from the comparatively smaller Set M benchmark set for a renewed comparison of the solution quality produced by OrTools and CSA. To ensure a consistent and comparable evaluation, a unified experimental protocol and reporting scheme are adopted in this subsection. For CSA, each instance is executed independently 20 times with different random seeds, and the solution quality is summarized by the best value (Best) and the mean ± standard deviation (Avg ± Std) over the 20 runs. The computational time is measured in seconds and decomposed into the initialization time t_{ccw} (CCW-based initial solution construction) and the search time t_{sa} (ISAHH search process), with the total time defined as $t_{total} = t_{ccw} + t_{sa}$. When the same CCW initial solution is reused across runs, t_{ccw} becomes deterministic and thus may have zero standard deviation. The corresponding statistics are summarized in Table 6.

Table 6. Robustness and runtime statistics of CSA (20 independent runs). Std denotes the sample standard deviation computed over 20 independent runs. Std denotes the sample standard deviation computed over 20 independent runs.

Instance	**Runs**	**BKS**	**Best**	**Avg ± Std**	t_{ccw} **± Std (s)**	t_{sa} **± Std (s)**	t_{total} **± Std (s)**
M-n101-k10	20	820 *	821	836 ± 10.5	1.6 ± 0.0	14.3 ± 0.26	15.9 ± 0.26
M-n121-k7	20	1034 *	1048	1056 ± 3.1	131.9 ± 0.0	15.5 ± 0.09	147.4 ± 0.09
M-n151-k12	20	1015 *	1105	1113 ± 6.7	51.9 ± 0.0	18.5 ± 0.58	70.4 ± 0.58
M-n200-k16	20	1274 *	1411	1425 ± 5.5	566.1 ± 0.0	22.9 ± 0.25	589.0 ± 0.25
M-n200-k17	20	1275 *	1395	1404 ± 5.9	218.9 ± 0.0	23.2 ± 0.41	242.1 ± 0.41

Note: * indicates a confirmed optimal solution.

Table 7 displays the comparative experiment results, with the best values for each instance highlighted in bold. The table reveals that CSA outperforms OrTools in three instances. Notably, CSA achieves a GAP value of only 0.12% in the first instance, nearly reaching the best-known solution. Hence, CSA achieves better solution quality compared to OrTools. In addition, Table 7 reports the runtime statistics of CSA (mean ± standard deviation over 20 independent runs). The reported times include t_{ccw}, t_{sa}, and t_{total}, which characterize both computational cost and robustness.

Table 7. Performance comparison between OrTools and CSA (ISAHH).

		OrTools		CSA	
Instance	**BKS**	**Best**	**GAP**	**Best**	**GAP**
M-n101-k10	820 *	910	11.00%	**821**	0.12%
M-n121-k7	1034 *	1065	2.90%	**1048**	1.35%
M-n151-k12	1015 *	**1148**	13.10%	**1105**	8.87%
M-n200-k16	1274 *	1563	22.60%	**1411**	10.75%
M-n200-k17	1275 *	**1460**	14.50%	**1395**	9.41%

Note: * indicates a confirmed optimal solution.

Across the 20 independent runs, CSA shows consistent performance in both solution quality and runtime on the Set M instances. As reported in Table 7, the standard deviation of the objective value remains relatively small compared with the mean, indicating that CSA produces consistent solutions under different random seeds. In addition, the runtime dispersion is also modest, suggesting that the computational cost is stable under the same platform and stopping criterion. Overall, these statistics demonstrate the robustness of CSA for the considered benchmarks.

Sensitivity to Vehicle Capacity

We investigate the robustness of CSA under different capacity tightness levels on three representative Set M instances (M-n101-k10, M-n121-k7, and M-n151-k12). Keeping the instance structure and the vehicle limit k unchanged, we scale the original capacity C by $C' = rC$ with three regimes: tight ($r = 0.9$), nominal ($r = 1.0$), and loose ($r = 1.1$). For each setting, CSA is executed for 20 independent runs ($N = 20$) with different random seeds. We report Best and Avg ± Std, together with the runtime breakdown t_{ccw}, t_{sa}, and t_{total} (Table 8).

Table 8. Sensitivity analysis of CSA under different vehicle capacities on Set M instances ($N = 20$).

Instance	**Ratio*r***	C'	**BKS**	**Best**	**Avg ± Std**	t_{ccw} **± Std (s)**	t_{sa} **± Std (s)**	t_{total} **± Std (s)**
M-n101-k10	0.9	180	-	973	989 ± 7.1	76.8 ± 0.0	14.5 ± 0.09	91.3 ± 0.09
	1.0	200	820 *	821	836 ± 10.5	1.6 ± 0.0	14.3 ± 0.26	15.9 ± 0.26
	1.1	220	-	808	825 ± 11.1	1.8 ± 0.0	14.1 ± 0.06	15.9 ± 0.06
M-n121-k7	0.9	180	-	1291	1309 ± 9.0	137.3 ± 0.0	15.7 ± 0.10	153.0 ± 0.10
	1.0	200	1034 *	1048	1056 ± 3.1	131.9 ± 0.0	15.5 ± 0.09	147.4 ± 0.09
	1.1	220	-	1011	1023 ± 7.1	2.8 ± 0.0	15.6 ± 0.15	18.4 ± 0.15
M-n151-k12	0.9	180	-	1168	1187 ± 9.3	241 ± 0.0	18.6 ± 0.28	259.6 ± 0.28
	1.0	200	1015 *	1105	1113 ± 6.7	51.9 ± 0.0	18.5 ± 0.58	70.4 ± 0.58
	1.1	220	-	1093	1113 ± 11.6	2.8 ± 0.0	18.2 ± 0.11	21.0 ± 0.11

Note: * indicates a confirmed optimal solution.

As expected, loosening capacity enlarges the feasible region and tends to shorten routes. Compared with the nominal case ($r = 1.0$), the best solutions improve under $r = 1.1$: 821→808 (M-n101-k10), 1048→1011 (M-n121-k7), and 1105→1093 (M-n151-k12).

In contrast, under tight capacity ($r = 0.9$), feasibility becomes more restrictive and cross-route moves are often blocked by capacity checks, leading to worse averages: 989 ± 7.1 (M-n101-k10), 1309 ± 9.0 (M-n121-k7), and 1187 ± 9.3 (M-n151-k12). Across all regimes, CSA remains stable with moderate standard deviations over repeated runs.

The SA search time t_{sa} is relatively insensitive to capacity changes (about 14–19 s), while the CCW initialization t_{ccw} is highly sensitive under tight capacity, dominating the total runtime (e.g., 76.8 s, 137.3 s, and 241.0 s for $r = 0.9$ on the three instances). This is consistent with CCW requiring substantially more effort to construct a feasible initial solution when the capacity constraint is tight under a fixed vehicle limit. Under nominal/loose capacity, t_{ccw} drops to only a few seconds (typically 1.6–2.8 s), so the overall runtime is mainly determined by the SA stage.

5. Conclusions and Future

This research introduces the CSA to address CVRP. Initially, CSA employs a clustered version of the CW, which, as our comparison reveals, not only enhances the quality of initial solutions but also accelerates the search process. Regarding selecting high-level heuristic strategies, the CSA incorporates an Improved Simulated Annealing method to pick LLHs precisely. To effectively guide the search toward the optimal solutions swiftly, the UEPS enhances the algorithm's directiveness. Additionally, this research developed the ITabuLLH that broadens the search scope, enhancing overall search efficiency and addressing the limitations of traditional intra-route and inter-route local optimization operators. Tests on the CVRPLIB Uchoa benchmark set demonstrate CSA's superior performance in managing CVRP challenges compared to eleven other algorithms, confirming CSA's effectiveness and superiority in solving complex CVRPs. Ablation experiments also validate the effectiveness of the CCW, UPES, and ITabuLLH modules.

The design of UEPS aims to enhance algorithm directiveness, favoring LLHs that reduce the total delivery distance, a concept well-aligned with reinforcement learning through a reward (penalty) guided learning process. Future work may explore integrating UEPS with reinforcement learning such as DQN, PPO, etc., for more precise algorithm directiveness. The Uchoa benchmark set used in this study goes up to 449 customer points, while the largest instance in the Uchoa benchmark set has 1001 customer points, and the recently introduced Arnold benchmark set reaches up to 30,000 customer points. When selecting LLHs, if intra-route or inter-route local optimization operators are chosen, only one or two routes can be modified, which becomes inefficient for solving CVRP at larger scales. Future work could extend upon the current algorithm to develop more Metaheuristic optimization operators, replacing traditional local optimization operators. Future work will also try to use deep reinforcement learning methods to construct local optimization operators that can search in parallel on a larger scale. This extension aims to handle batches of routes in a single iteration, thereby improving efficiency for solving CVRP at the scale of tens of thousands of customer points.

Author Contributions: Conceptualization, L.Y.; Methodology, L.Y.; Software, Y.W.; Validation, M.Y.; Formal analysis, M.Y.; Resources, F.G.; Writing—original draft, Y.W.; Visualization, Y.L.; Supervision, Y.L.; Funding acquisition, F.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (NSFC), grant number 62463001.

Data Availability Statement: The original data presented in the study are openly available in CVRPLIB at https://galgos.inf.puc-rio.br/cvrplib/index.php/en/instances (accessed on 7 January 2026). The benchmark instances used in this study were derived from public-domain resources provided by CVRPLIB (including the Uchoa benchmark set, Set A, and Set M). The source code is available at https://github.com/GXU-LIPE/CVRP-20250204 (accessed on 7 January 2026).

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

References

1. Zhang, Y.; Mei, Y.; Zhang, H.; Cai, Q.; Wu, H. RoCaSH2: An effective route clustering and search heuristic for large-scale multi-depot capacitated arc routing problem. *IEEE Comput. Intell. Mag.* **2023**, *18*, 43–56. [CrossRef]
2. Zhou, J.; Wu, Y.; Cao, Z.; Song, W.; Zhang, J.; Chen, Z. Learning large neighborhood search for vehicle routing in airport ground handling. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 9769–9782. [CrossRef]
3. Liu, S.; Lin, Q.; Feng, L.; Wong, K.C.; Tan, K.C. Evolutionary multitasking for large-scale multiobjective optimization. *IEEE Trans. Evol. Comput.* **2022**, *27*, 863–877. [CrossRef]
4. Chen, Z.G.; Zhan, Z.H.; Kwong, S.; Zhang, J. Evolutionary computation for intelligent transportation in smart cities: A survey. *IEEE Comput. Intell. Mag.* **2022**, *17*, 83–102. [CrossRef]
5. Jiang, H.; Lu, M.; Tian, Y.; Qiu, J.; Zhang, X. An evolutionary algorithm for solving capacitated vehicle routing problems by using local information. *Appl. Soft Comput.* **2022**, *117*, 108431. [CrossRef]
6. Meraliyev, M.; Turan, C.; Kadyrov, S.; Sadyk, U. A Comprehensive Survey of Methods and Challenges of Vehicle Routing Problem with Uncertainties. *Mathematics* **2025**, *13*, 3782. [CrossRef]
7. Huang, K.; Xu, M. Optimization Models for the Vehicle Routing Problem under Disruptions. *Mathematics* **2023**, *11*, 3521. [CrossRef]
8. Luo, H.; Dridi, M.; Grunder, O. A branch-price-and-cut algorithm for a time-dependent green vehicle routing problem with the consideration of traffic congestion. *Comput. Ind. Eng.* **2023**, *177*, 109093. [CrossRef]
9. Torres, F.; Gendreau, M.; Rei, W. Crowdshipping: An open VRP variant with stochastic destinations. *Transp. Res. Part C Emerg. Technol.* **2022**, *140*, 103677. [CrossRef]
10. Şahin, M.K.; Yaman, H. A branch and price algorithm for the heterogeneous fleet multi-depot multi-trip vehicle routing problem with time windows. *Transp. Sci.* **2022**, *56*, 1636–1657. [CrossRef]
11. Yao, C.; Chen, S.; Salazar, M.; Yang, Z. Joint routing and charging problem of electric vehicles with incentive-aware customers considering spatio-temporal charging prices. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 12215–12226. [CrossRef]
12. Xiao, H.; Zhao, J.; Feng, J.; Liu, L.; Pei, Q.; Shi, W. Joint optimization of security strength and resource allocation for computation offloading in vehicular edge computing. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 8751–8765. [CrossRef]
13. Wang, Y.; Yan, M.; Feng, G.; Qin, S.; Wei, F. Autonomous on-demand deployment for UAV assisted wireless networks. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 9488–9501. [CrossRef]
14. Campos, V.; Mota, E. Heuristic procedures for the capacitated vehicle routing problem. *Comput. Optim. Appl.* **2000**, *16*, 265–277. [CrossRef]
15. Rivera, J.C.; Afsar, H.M.; Prins, C. A multistart iterated local search for the multitrip cumulative capacitated vehicle routing problem. *Comput. Optim. Appl.* **2015**, *61*, 159–187. [CrossRef]
16. Zhang, R.; Song, S.; Wu, C. The hot strip mill scheduling problem with uncertainty: Robust optimization models and solution approaches. *IEEE Trans. Cybern.* **2022**, *53*, 4079–4093. [CrossRef]
17. Yin, Y.; Qing, L.; Wang, D.; Cheng, T.C.E.; Ignatius, J. Exact solution method for vehicle-and-drone cooperative delivery routing of blood products. *Comput. Oper. Res.* **2024**, *164*, 106559. [CrossRef]
18. Yu, N.; Dong, B.; Qu, Y.; Zhang, M.; Chen, G.; Tan, Q.; Wang, Y.; Dai, H. Multiple-allocation hub-and-spoke network design with maximizing airline profit utility in air transportation network. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 7294–7310. [CrossRef]
19. Pu, Y.; Liu, H.; Wang, J.; Hou, Y. Collaborative scheduling of port integrated energy and container logistics considering electric and hydrogen-powered transport. *IEEE Trans. Smart Grid* **2023**, *14*, 4345–4359. [CrossRef]
20. Yin, J.; Pu, F.; Yang, L.; D'Ariano, A.; Wang, Z. Integrated optimization of rolling stock allocation and train timetables for urban rail transit networks: A Benders decomposition approach. *Transp. Res. Part B Methodol.* **2023**, *176*, 102815. [CrossRef]
21. Qi, X.; Zhao, T.; Liu, X.; Wang, P. Three-stage stochastic unit commitment for microgrids toward frequency security via renewable energy deloading. *IEEE Trans. Smart Grid* **2023**, *14*, 4256–4267. [CrossRef]
22. Hong, Y.; Yang, Y.; Luo, C.; Li, D.; Lu, Y.; Chen, Z. Spatiotemporal optimization for charging scheduling in wireless rechargeable sensor networks. *IEEE Internet Things J.* **2023**, *11*, 3056–3067. [CrossRef]
23. Lee, W.; Kim, T. Multiagent reinforcement learning in controlling offloading ratio and trajectory for multi-UAV mobile-edge computing. *IEEE Internet Things J.* **2023**, *11*, 3417–3429. [CrossRef]

24. Ren, Q.; Abbasi, O.; Kurt, G.K.; Yanikomeroglu, H.; Chen, J. Handoff-aware distributed computing in high altitude platform station (HAPS)-assisted vehicular networks. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 8814–8827. [CrossRef]
25. Guo, N.; Yuan, X.; Hu, Y.; Schmeink, A. Reliability-oriented resource allocation for wireless powered short packet communications with multiple WPT sources. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 8983–8998. [CrossRef]
26. Sabo, C.; Pop, P.C.; Horvat-Marc, A. On the Selective Vehicle Routing Problem. *Mathematics* **2020**, *8*, 771. [CrossRef]
27. Zhang, X.; Sang, H.; Li, Z.; Zhang, B.; Meng, L. An efficient discrete artificial bee colony algorithm with dynamic calculation method for solving the AGV scheduling problem of delivery and pickup. *Complex Intell. Syst.* **2024**, *10*, 37–57. [CrossRef]
28. Kritikos, M.N.; Metzidakis, T.; Ioannou, G. The cumulative vehicle routing problem with arc time windows. *Expert Syst. Appl.* **2024**, *240*, 122447. [CrossRef]
29. Gkiotsalitis, K.; Nikolopoulou, A. The multi-vehicle dial-a-ride problem with interchange and perceived passenger travel times. *Transp. Res. Part C Emerg. Technol.* **2023**, *156*, 104353. [CrossRef]
30. Wang, T.; Wu, D.; Zhao, W. Minimizing indirect contacts in urban pick-up and delivery services during COVID-19 pandemic. *IEEE Trans. Syst. Man Cybern. Syst.* **2023**, *53*, 7876–7887. [CrossRef]
31. Tiwari, K.V.; Sharma, S.K. An optimization model for vehicle routing problem in last-mile delivery. *Expert Syst. Appl.* **2023**, *222*, 119789. [CrossRef]
32. Ren, T.; Luo, T.; Jia, B.; Yang, B.; Wang, L.; Xing, L. Improved ant colony optimization for the vehicle routing problem with split pickup and split delivery. *Swarm Evol. Comput.* **2023**, *77*, 101228. [CrossRef]
33. Greco, A.; Cannizzaro, F.; Bruno, R.; Pluchino, A. A nested genetic algorithm strategy for an optimal seismic design of frames. *Comput. Optim. Appl.* **2024**, *87*, 677–704. [CrossRef]
34. De Moraes, M.B.; Coelho, G.P. A benchmark generator for scenario-based discrete optimization. *Comput. Optim. Appl.* **2024**, *88*, 349–378. [CrossRef]
35. Mohammadi, A.; Custódio, A.L. A trust-region approach for computing Pareto fronts in multiobjective optimization. *Comput. Optim. Appl.* **2024**, *87*, 149–179. [CrossRef]
36. Arce-Ortega, D.V.; Alonso-Pecina, F.; Cruz-Chávez, M.A.; Peralta-Abarca, J.D.C. A Simulated Annealing Approach for the Homogeneous Capacitated Vehicle Routing Problem. *Mathematics* **2025**, *13*, 3209. [CrossRef]
37. Bogyrbayeva, A.; Meraliyev, M.; Mustakhov, T.; Dauletbayev, B. Machine learning to solve vehicle routing problems: A survey. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 4754–4772. [CrossRef]
38. Wu, Y.; Zhou, J.; Xia, Y.; Zhang, X.; Cao, Z.; Zhang, J. Neural airport ground handling. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 15652–15666. [CrossRef]
39. Di Puglia Pugliese, L.; Guerriero, F. On the shortest path problem with negative cost cycles. *Comput. Optim. Appl.* **2016**, *63*, 559–583. [CrossRef]
40. Chen, Y.; Xue, J.; Zhou, Y.; Wu, Q. An efficient threshold acceptance-based multi-layer search algorithm for capacitated electric vehicle routing problem. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 5867–5879. [CrossRef]
41. Wang, Z.; Liu, J.; Zhang, J. Hyper-heuristic algorithm for traffic flow-based vehicle routing problem with simultaneous delivery and pickup. *J. Comput. Des. Eng.* **2023**, *10*, 2271–2287. [CrossRef]
42. Jia, F.; Zhu, Z.; Dai, W.; Le, V.V. Short-term forecasting of streamflow by integrating machine learning methods combined with metaheuristic algorithms. *Expert Syst. Appl.* **2024**, *245*, 123076. [CrossRef]
43. Atasagun, G.C.; Karaoğlan, İ. Solution approaches for the integrated production and outbound distribution scheduling problem with multiple plants and perishable items. *Expert Syst. Appl.* **2024**, *237*, 121318. [CrossRef]
44. Wang, Y.; Wang, X.; Wei, Y.; Sun, Y.; Fan, J.; Wang, H. Two-echelon multi-depot multi-period location-routing problem with pickup and delivery. *Comput. Ind. Eng.* **2023**, *182*, 109385. [CrossRef]
45. VRP Benchmark Site. Available online: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/ (accessed on 7 January 2026).
46. Zhang, D.; Zhou, S.; Ji, B.; Li, S. A two-echelon capacitated vehicle routing problem with sharing satellite resources. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 12216–12227. [CrossRef]
47. Küçük, M.; Yildiz, S.T. Constraint programming-based solution approaches for three-dimensional loading capacitated vehicle routing problems. *Comput. Ind. Eng.* **2022**, *171*, 108505. [CrossRef]
48. Wang, Y.; Luo, S.; Fan, J.; Xu, M.; Wang, H. Compensation and profit allocation for collaborative multicenter vehicle routing problems with time windows. *Expert Syst. Appl.* **2023**, *233*, 120988. [CrossRef]
49. Tsao, Y.C.; Tai, J.Y.; Vu, T.L.; Chen, T.H. Multiple bin-size bin packing problem considering incompatible product categories. *Expert Syst. Appl.* **2024**, *247*, 123340. [CrossRef]
50. Yao, Y.; Lei, S.; Guo, Z.; Li, Y.; Ren, S.; Liu, Z.; Guan, Q.; Luo, P. Fast optimization for large scale logistics in complex urban systems using the hybrid sparrow search algorithm. *Int. J. Geogr. Inf. Sci.* **2023**, *37*, 1420–1448. [CrossRef]
51. Wu, Y.; Song, W.; Cao, Z.; Zhang, J.; Lim, A. Learning improvement heuristics for solving routing problems. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 5057–5069. [CrossRef]

52. CVRPLIB: Capacitated Vehicle Routing Problem Library (Instances). Available online: https://galgos.inf.puc-rio.br/cvrplib/index.php/en/instances (accessed on 3 February 2026).
53. Google OR-Tools: Optimization Tools. Available online: https://developers.google.com/optimization?hl=zh-cn (accessed on 3 February 2026).
54. Costa, J.G.C.; Mei, Y.; Zhang, M. Adaptive search space through evolutionary hyper-heuristics for the large-scale vehicle routing problem. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, Australia, 1–4 December 2020; IEEE: New York, NY, USA, 2020; pp. 2415–2422.

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.