sustainability

Article

# Modeling and Optimization of Container Drayage Problem with Empty Container Constraints across Multiple Inland Depots

**Xuhui Yu *, Yin Feng, Cong He and Chang Liu**

College of Transportation Engineering, Dalian Maritime University, Dalian 116026, China; fengy916@dlmu.edu.cn (Y.F.); hecong@dlmu.edu.cn (C.H.); liuchagn@dlmu.edu.cn (C.L.)
* Correspondence: yuxuhui@dlmu.edu.cn

**Abstract:** Container drayage involves the transportation of containers by trucks. Although the distance is relatively short compared to maritime and rail transport, container drayage accounts for 25% to 40% of the total container transportation costs and significantly contributes to increased fuel consumption and carbon emissions. Thus, the modeling of the container drayage problem (CDP) has received a lot of attention in the last two decades. However, the three fundamental modeling factors, including the combination of trucking operation modes and empty container relocation strategies, as well as empty container constraints and multiple inland depots, have not been simultaneously investigated. Hence, this study addressed a comprehensive CDP that simultaneously incorporates the three modeling factors. The problem was formulated as a novel mixed integer linear programming (MILP) model based on the DAOV graph. Given the complexity of this problem, it was not realistic to find an exact solution for large instances. Therefore, an improved genetic algorithm (GA) was designed by integrating the "sequential insertion" method and "solution re-optimization" operation. The performance of Gurobi and GA was validated and evaluated through randomly generated instances. The results indicate that (1) the proposed algorithm can provide near-optimal solutions for large-scale instances within a reasonable running time, (2) the greatest cost savings from combining trucking operation modes and empty container relocation strategies range from 10.45% to 31.86%, and (3) the three modeling factors significantly influence the fuel consumption and carbon emissions, which can provide managerial insights for sustainable container drayage practices.

**Keywords:** container drayage; trucking operation modes; empty container relocation strategies; empty container constraints; multiple inland depots; genetic algorithm

**Citation:** Yu, X.; Feng, Y.; He, C.; Liu, C. Modeling and Optimization of Container Drayage Problem with Empty Container Constraints across Multiple Inland Depots. *Sustainability* **2024**, *16*, 5090. https://doi.org/10.3390/su16125090

Academic Editors: Mladen Krstić, Željko Stević and Snežana Tadić

Received: 20 April 2024
Revised: 21 May 2024
Accepted: 3 June 2024
Published: 14 June 2024

## 1. Introduction

Economic globalization and the concomitant technological advances in transportation modes have promoted the flourishing of container transportation, which, in turn, has made international trade more efficient and cost-effective. Generally, container transportation comprises maritime transport and land transport, where maritime transport refers to the haulage of containers by container vessels from port to port, while land transport indicates the haulage of containers by trucks or trains from shipper to port or from port to receiver. The seamless integration of various transportation modes for hauling the same container constitutes container intermodal transport. Despite the low cost and low carbon emissions of maritime and rail transport, it is difficult to achieve door-to-door services with these methods. Hence, short-haulage by trucks, which is also called container drayage, becomes an essential link in container intermodal transport [1]. However, according to statistics, although the distance is relatively short compared to maritime and rail transport, container drayage accounts for 25% to 40% of the total container transportation costs [2]. Furthermore, it also significantly contributes to shipment delays, road congestion, and increased carbon emissions [3]. Amidst the proposal of "dual carbon" goals in China [4], as well as people's pursuit of cost-effective and agile supply chain services [5], reducing carbon emissions, lowering transportation costs, and enhancing the operational efficiency of container drayage have emerged as substantial challenges.

Container drayage refers to the haulage of full or empty containers over short distances by trucks. Full containers typically originate from ports or railway hubs and are hauled to specific customers or vice versa, while empty containers usually do not have a predetermined destination and can be hauled between ports, railway hubs, depots, and customers based on demand. In the container drayage problem (CDP), two trucking operation modes are commonly adopted, i.e., "Discharge" and "Strip" [6]. In the Discharge mode, a container and its carrying truck will not be separated at the customer location, where the truck remains waiting during the cargo loading or unloading process. In response to the prolonged waiting time of trucks at customer locations, the Strip mode is introduced, which allows for the uncoupling of a container from its carrying truck and dropping it at the customer location. After completing customer service, the container can be picked up by either the same truck or a different one in consideration of the cost-effective option. Due to the limited number of truck resources, a reasonable combination of the two modes can enable more flexible and efficient truck scheduling, thus reducing unnecessary truck waiting time and traveling time.

Due to trade imbalances and logistics scheduling requirements, the inter-regional transportation of empty containers is required to balance the inventory of empty containers at various ports and depots, as well as meet the demand of customers who require the pick-up or delivery of empty containers. On average, empty containers account for about 40% of all containers in land transport, whereas, in maritime transport, it is only half that amount [7]. Traditionally, all empty containers are mandated to depart from or return to ports or depots, which is called the Depot-turn strategy [6]. The Depot-turn strategy is easy to implement but obviously increases the amount of empty container transportation. Thus, the Street-turn strategy [6], which allows for the direct haulage of empty containers that do not need maintenance or cleaning from suppliers to demanders, is proposed. If an emptied container after unloading can be directly transported to a nearby customer that demands it, the reusing of empty containers can be facilitated, and the transportation cost can be reduced significantly. However, this ideal situation is not always achievable because of trade imbalances, supply and demand timing discontinuity, and the maintenance or cleaning requirements of some empty containers. Therefore, in order to efficiently relocate such a great amount of empty containers, it is imperative to integrate the two empty container relocation strategies into the CDP.

As a variant of the vehicle routing problem (VRP), the CDP shares commonalities such as vehicle scheduling and route planning [8], but it also has unique features, notably the fact that each vehicle can carry one or two 1 TEU (Twenty-foot Equivalent Unit) containers or one 1 FEU (Forty-foot Equivalent Unit) container at a time. While there has been a substantial body of literature dedicated to addressing CDP over the past two decades, it is noteworthy that this topic has not been thoroughly investigated yet. By systematically reviewing the literature, three fundamental modeling factors for CDP are identified, i.e., the combination of trucking operation modes and empty container relocation strategies [6], the availability of empty containers [9], and multiple depots [6]. However, the existing research usually only incorporates one or two of the above three factors to reduce modeling or solving complexity, which does not describe complicated realistic scenarios. To fill the research gap, it is necessary to investigate the CDP by merging the two trucking operation modes and two empty container relocation strategies under the constraints of the limited availability of empty containers across multiple depots. This study makes three major contributions:

(1) It formally introduces a new variation of CDP with high complexity that takes into account three fundamental modeling factors for real-world scenarios. Based on the task vertex splitting, determined-activities-on-vertex (DAOV) graph, and network flow constraints, the problem is mathematically formulated as a novel mixed integer linear programming (MILP) model, which aims to minimize the carbon emissions

costs, fuel costs, and truck waiting costs of trucks completing container transportation tasks. The proposed MILP model also expands the modeling work of CDP.

(2) Methodologically, the CDP investigated in this paper is characterized by its complex and highly hybrid nature, making it challenging to address effectively by extant methodologies. Therefore, an improved genetic algorithm (GA) is proposed to adapt to the characteristics of the new CDP. Firstly, the chromosome is innovatively designed as an ordered sequence of customers using a real-number encoding approach. Secondly, to improve computational performance and result in optimality, this paper introduces the "sequential insertion" technique to construct a complete truck route by incorporating depots into the solution of customer sequence and the "solution re-optimization" technique to control the feasibility after crossover and mutation.

(3) Extensive experiments were conducted based on different scale instances to demonstrate the effectiveness and efficiency of the proposed model and algorithm. Additionally, the benefits of combining all modes and strategies and the impact of the critical model parameters were also investigated to obtain managerial insights for sustainable container drayage practices.

The remainder of this paper is organized as follows. The relevant literature is thoroughly reviewed in Section 2. The problem is formally described in Section 3, followed by a DAOV graph-based mathematical formulation. Section 4 proposes the improved GA, and numerical experiments are presented in Section 5. Section 6 concludes this paper and suggests future work.

## 2. Literature Review

The CDP has received a large amount of attention from academics in the last two decades [10], and abundant findings have arisen from the investigation of the CDP and its variants. According to the research focus, studies closely aligned with the scope of our paper can be briefly divided into three categories: the CDP involving trucking operation modes and empty container relocation strategies, the CDP with empty container constraints, and the CDP with multiple depots.

### *2.1. The CDP Involving Trucking Operation Modes and Empty Container Relocation Strategies*

The merging of trucking operation modes and empty container relocation strategies was not emphasized in the initial studies on CDP. Early articles either only concentrated on the transport of full containers or did not distinguish between full and empty containers [11–15]. Zhang et al. [16] first began to differentiate transportation between full and empty containers, leading to the merging of trucking operation modes and empty container relocation strategies gradually receiving attention. The existing research can be classified into three types from the perspective of combining modes and strategies.

The first type combines one mode and one strategy. Sterzik and Kopfer [17] and Vidović et al. [18] studied the CDP combining the Discharge mode and the Depot-turn strategy. The former proposed a MILP model with multiple depots and solved it using a taboo search algorithm, while the latter constructed a MILP model with multi-size containers and designed a variable neighborhood search algorithm. In order to increase the utilization of trucks, Lai et al. [19] and Ghezelsoflu et al. [20] investigated the CDP combining the Discharge mode and Street-turn strategy. Lai et al. [19] developed a node–arc-based MILP model and proposed a meta-heuristic method. Ghezelsoflu et al. [20] presented a set-covering formulation and verified the model, which significantly outperformed that proposed by Lai et al. [19].

The second type integrates one mode and both strategies. In this regard, it is worth noting that all seven relevant articles exclusively employed the Discharge mode. Zhang et al. [16] and Zhang et al. [21] defined the model as a multi-traveling salesman problem with time windows (MTSPTW). Zhang et al. [16] proposed a reactive taboo search (RTS), while Zhang et al. [21] utilized a window partition-based method. Subsequently, the research was gradually enriched by considering more detailed factors, such as multi-size containers [22], heterogeneous trucks [23], empty container constraints [24,25], and multi-trip CDP [26].

The third type consolidates both modes and both strategies. However, the modes or strategies were subject to some constraints in the following five articles. Braekers et al. [27] only allowed the Strip mode while trucks delivered a full container to a receiver, and modeled the problem as an asymmetric MTSPTW. Different from Braekers et al. [27], Song et al. [28] and Song et al. [9] imposed limitations on the Strip mode, restricting the same truck to picking up the separated container, while Zhang et al. [29] assumed that the operation mode preferred by each customer remained fixed and known in advance. Moghaddam et al. [30] delegated the task of matching Street-turn to the customers for the CDP with heterogeneous trucks, thereby mitigating modeling and solving complexity. Other articles considered a comprehensive combination of both modes and both strategies. Choi et al. [31] proposed an automated dispatching approach utilizing GA to address the CDP with heterogeneous trucks. Funke and Kopfer [32] formulated a MILP model with multi-size containers. Zhang et al. [33] designed a sequence-dependent MTSPTW model considering foldable containers. Bomboi [34] modeled the stochastic CDP by considering stochastic travel times. He et al. [35] devised a MILP model that considered carbon emissions and established an improved ant colony optimization algorithm. Yang et al. [6] established a MILP model grounding in the arc flow framework and designed a novel GA for CDP involving a heterogeneous fleet. Huang and Zhang [36,37] modeled a special CDP involving both foldable and standard containers simultaneously, based on an improved truck–state transition method. They separately designed an RTS and a large neighborhood search algorithm.

### *2.2. The CDP with Empty Container Constraints*

As is known, the high purchase cost of empty containers generally leads to the limited availability of empty containers at each depot, and, thus, empty container constraints should be incorporated into a realistic CDP. The CDP with empty container constraints was first modeled by Zhang et al. [24] based on the DAOV graph. They considered three specific categories of container movements: inbound full, outbound full, and inbound empty movements, and designed meta-heuristics based on RTS. Furthermore, Zhang et al. [25] extended Zhang et al. [24] by incorporating outbound empty movements while also considering constraints associated with truck resources. They introduced a more efficient model compared to Zhang et al. [24]. In a recent study, Fazi et al. [26] investigated a synchronized multi-trip CDP to promote the efficient reuse of empty containers, and Song et al. [9] merged all modes and strategies under empty container constraints. Huang and Zhang [36,37] considered the issue of the limited availability of empty foldable containers at depots. However, they assumed that the number of empty standard containers at depots was sufficiently large. Experimental results demonstrated that the introduction of foldable containers can result in a reduction in transportation costs when compared with the use of standard containers.

In short, there are few CDP-related studies that focus on empty container resource constraints. These studies have one common limitation as they considered only one depot, ignoring the scenario where large-scale trucking enterprises may operate multiple depots in a given region [6]. Furthermore, only Huang and Zhang [36,37] combined both Strip and Discharge modes, whereas other researchers focused solely on Discharge modes. A review of the literature reveals that while very few researchers have combined two modes and two strategies under empty container constraints, no study has investigated the availability of empty containers in a multi-depot scenario.

### *2.3. The CDP with Multiple Depots*

It is common to have multiple depots rather than just one in a given hinterland region, and this has garnered the attention of researchers and practitioners. Research on the CDP with multiple depots usually involves various combinations of trucking operation modes and empty container relocation strategies, but no studies have considered empty container constraints simultaneously. With regard to the combination of the Discharge mode and Depot-turn strategy, Sterzik and Kopfer [17] and Nguyen and Pham [38] presented a comprehensive MILP model and solved it using a taboo search algorithm and a local search algorithm, respectively. As for the combination of Discharge mode and both strategies, Zhang et al. [16] and Zhang et al. [21] defined their model as a MTSPTW. The former introduced a cluster method and an RTS, while the latter proposed a window partition-based method. Nossack and Pesch [39] presented the model as a full-truckload pickup and delivery problem with time windows and proposed a two-stage heuristic approach that outperformed the method by Zhang et al. [21]. Furthermore, to balance empty containers at multiple depots, Reinhardt et al. [40] regarded the problem as a set covering problem and solved it using column enumeration. In relation to the combination of both modes and both strategies, Braekers et al. [27] formulated an asymmetric MTSPTW and proposed a sequential method and an integrated method based on a deterministic annealing algorithm. The results confirmed that the integrated approach outperformed the sequential one. Braekers et al. [41] extended the previous work from a bi-objective perspective. Zhang et al. [29] assumed that the mode preferred by each customer remained fixed and known in advance. Their research formulated a mixed-integer nonlinear programming model based on the DAOV graph, which was solved using a window partition-based method. Shiri [42] proposed a mixed-integer quadratic programming model and solved it with an RTS. Yang et al. [6] extended the CDP with multiple depots by considering a heterogeneous fleet. Unlike the above studies, Cui et al. [2] and Jia et al. [43] studied container drayage operations with tractor–trailer distinction and trailer repositioning, but they ignored empty container constraints, empty container reallocation, and trucking operation modes.

In summary, a large body of outstanding articles dedicated to the CDP was found in the literature. Nevertheless, among the three fundamental modeling factors for CDP, namely, the combination of trucking operation modes and empty container relocation strategies, empty container constraints, and multiple depots, the previous studies only covered one or two factors. In contrast, the essential distinctions that differentiate our work from the above literature are the concurrent consideration of the three fundamental modeling factors and the development of an improved GA.

## 3. Problem Definition and Formulations

In this section, the new CDP is defined, along with the graphical and mathematical formulations of the problem.

### 3.1. Problem Description

The CDP in this paper can be characterized as follows: a trucking company owns a limited number of trucks and empty containers at multiple inland depots. The company handles various container transportation tasks within a defined area surrounding a port (or multiple ports) over a planning horizon, typically one day. Trucks are initially stationed at a depot and are mandated to return to any of these depots upon the completion of their assigned tasks. The number of empty containers at a depot dynamically changes as trucks carry out their tasks. The goal of the problem was to optimize truck scheduling and empty container relocation to fulfill the demands of all customers, while also factoring in practical constraints such as the availability of trucks and empty containers, time windows of requests, and working hours of trucks.

This problem involved the transportation of full or empty containers between a single port, multiple inland depots, and numerous customers. The depots could provide limited empty containers and take back empty containers from the customer location after unpacking or pick up empty containers from the port for future use. Import-full (empty) containers originated from the port, while export-full (empty) containers needed to be delivered to the port. From the perspectives of proposing requests, six types of task were performed in this study, including (1) delivering an empty container to a shipper or port, (2) picking up a full container from a shipper, (3) delivering a full container to a consignee, (4) picking up an empty container from a consignee or port, (5) delivering an empty container to and picking up a full container (after packing) from a shipper, and (6) delivering a full container to and picking up an empty container (after unpacking) from a consignee. If there were multiple tasks at one customer location or a port, they were divided into several customers, each maintaining consistent characteristics such as service time windows and geographical locations, and each involving a single task. In practice, if a customer required the delivery of empty (full) containers and soon afterward the pickup of full (empty) containers after packing (unpacking), the Discharge or Strip mode could be selected based on the optimal solution. As for empty container relocation, Street-turn and Depot-turn strategies were simultaneously allowed in our model.

In the context of scenarios where empty container constraints across multiple depots were considered, the integration of two trucking operation modes and two empty container relocation strategies increased the available options for creating valid truck dispatching and route planning, which, in turn, enhanced the following uncertainty and unpredictability. First, when dealing with a limited number of empty containers in multiple depots, the selection of a depot to dispatch trucks needed to consider two crucial factors: the fixed factor of the distance between the depots and the customer and the dynamic factor of the number of available empty containers and trucks at the depots. It was not feasible to pre-determine which depot would service a customer solely based on the closest principle. Second, with regard to the combination of Strip and Discharge modes, if a customer required both an empty (full) container delivery and a full (empty) container pickup (after packing or unpacking), three scenarios could unfold: (i) a single truck performed the delivery task and subsequently the pickup task; (ii) a single truck performed the delivery task, left for other customers' tasks, and then returned to perform the pickup task; and (iii) two trucks handled the delivery and pickup tasks separately. Hence, for this kind of customer, the task sequence of trucks could not be predetermined. Third, when considering both Street-turn and Depot-turn strategies, the empty containers generated by customers did not necessarily need to be transported back to depots, and they could also be routed to other customers for reuse. Consequently, unlike full containers, it was not possible to pre-determine the specific destination for each empty container. To sum up, the capture and determination of the above uncertainty and unpredictability involved in establishing valid truck dispatching and route planning needed to be handled by the model itself, which inevitably increased the complexity of the model and algorithm.

In order to develop an intuitive understanding of the disparities arising from empty container constraints across multiple inland depots, a simplified instance was introduced, as shown in Figure 1. This instance took one shift of the planning horizon (8 h) as an example and defined 10 min as one unit of time; thus, one shift contained 48 units of time. The traveling and waiting costs were measured in units of time and were assumed to be equal per unit of time. There was one port, one (two) depot, and four customers, and each customer had one task to be handled. Customer 1 was a consignee who needed a full container to be delivered and the emptied container to be picked up. Customers 2–4 were shippers. Customer 2 and customer 3 each needed an empty container to be delivered, while customer 4 needed an empty container to be delivered and the loaded container to be picked up. We assumed that the time cost required for loading (unloading) a container to (from) a truck was 1 unit (10 min) and that the time cost for packing or unpacking a container was 6 units (60 min). The numbers without parentheses on the arrow lines indicate the sequence of truck movements, while the numbers within parentheses on the arrow lines denote the corresponding transportation costs in terms of units of time. Figure 1 and Table 1 show feasible truck dispatching and route planning for four different real-world scenarios: (i) one depot with no empty container constraints, (ii) two depots with no empty container constraints, (iii) one depot with empty container constraints, and (iv) two depots with empty container constraints.

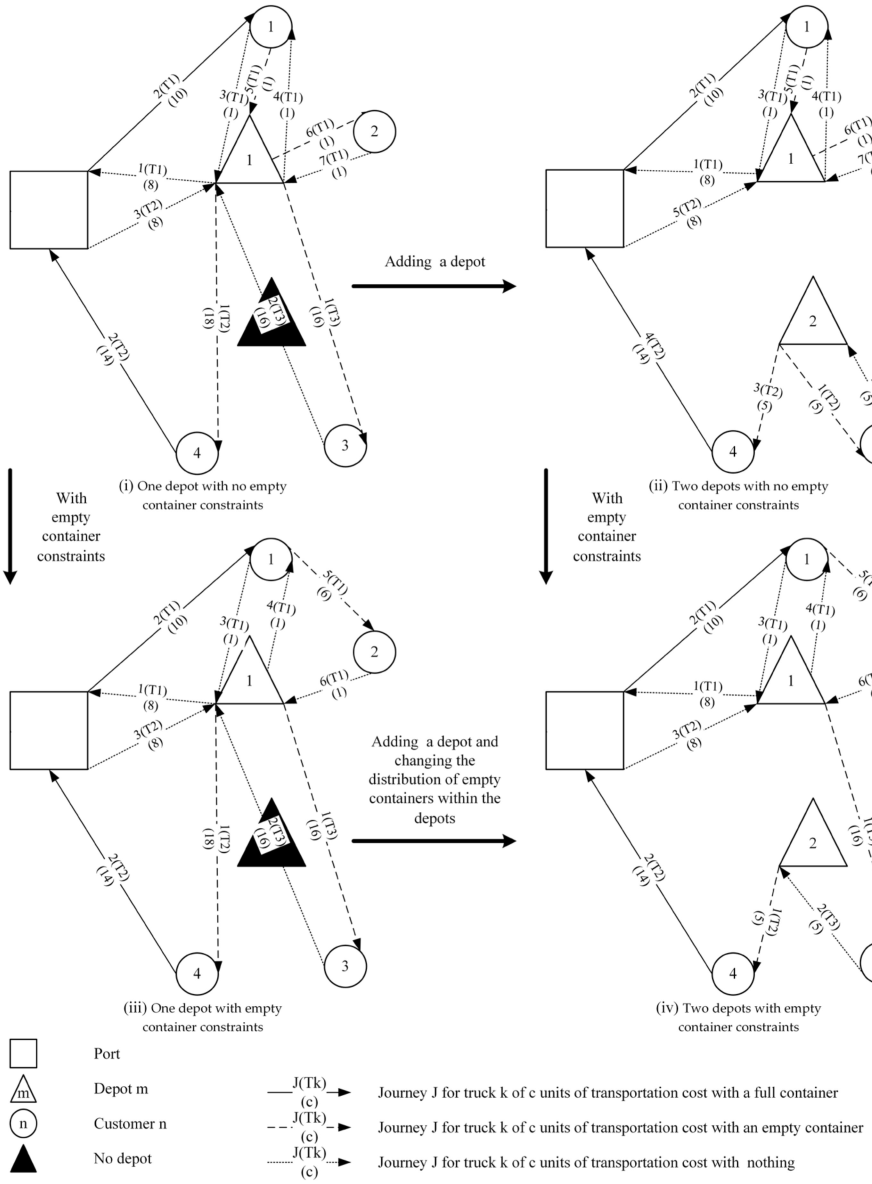

(i) One depot with no empty container constraints
(ii) Two depots with no empty container constraints
(iii) One depot with empty container constraints
(iv) Two depots with empty container constraints
Adding a depot
Adding a depot and changing the distribution of empty containers within the depots
With empty container constraints
With empty container constraints
2(T1) (10)
3(T1) (1)
5(T1) (1)
4(T1) (1)
6(T1) (1)
7(T1) (1)
1(T1) (8)
3(T2) (8)
5(T2) (8)
1(T2) (18)
2(T2) (14)
4(T2) (14)
2(T3) (16)
1(T3) (16)
3(T2) (5)
1(T2) (5)
2(T2) (5)
5(T1) (6)
2(T3) (5)
1
2
3
4
Port
Depot m
Customer n
No depot
J(Tk) (c)
Journey J for truck k of c units of transportation cost with a full container
Journey J for truck k of c units of transportation cost with an empty container
Journey J for truck k of c units of transportation cost with nothing

**Figure 1.** An instance of the CDP under four scenarios.

**Table 1.** The routes of trucks under four scenarios.

| Scenario | The Routes of Trucks | Total Cost |
|---|---|---|
| (i) | T1: depot 1→port→customer 1→depot 1→customer 1→depot 1→customer 2→depot 1<br>T2: depot 1→customer 4→port→depot 1<br>T3: depot 1→customer 3→depot 1 | 111 units |
| (ii) | T1: depot 1→port→customer 1→depot 1→customer 1→depot 1→customer 2→depot 1<br>T2: depot 2→customer 3→depot 2→customer 4→port→depot 1 | 76 units |
| (iii) | T1: depot 1→port→customer 1→depot 1→customer 1→customer 2→depot 1<br>T2: depot 1→customer 4→port→depot 1<br>T3: depot 1→customer 3→depot 1 | 113 units |
| (iv) | T1: depot 1→port→customer 1→depot 1→customer 1→customer 2→depot 1<br>T2: depot 2→customer 4→port→depot 1<br>T3: depot 2→customer 3→depot 2 | 89 units |

From Table 1 and Figure 1, it can be seen that to avoid excessively long waiting times for trucks, the trucks sometimes adopted the Strip mode rather than the Discharge mode, so some customer vertices were visited twice, which was handled by the task vertex splitting operation described in Section 3.2.1. In addition, it can also be observed that the Street-turn and Depot-turn strategies were optimally employed. Furthermore, the total cost of scenario (ii) was much lower than that of scenario (i), indicating that multiple depots could provide more options for the truck to obtain empty containers and reduce the total cost of the CDP. Conversely, the comparison between scenario (iii) and scenario (i) or scenario (iv) and scenario (ii) illustrates that due to the lack of empty containers in depots, trucks have to cover higher transportation costs to fulfill their designated tasks, even if the customer or depot owing empty containers is located far from the customer requesting empty containers. The simple instance demonstrates that except for the combination of trucking operation modes and empty container relocation strategies, empty container constraints and multiple depots not only have significant effects on the total cost of the CDP but also complicate truck scheduling. Therefore, the three modeling factors need to be carefully investigated.

## *3.2. Graphical Formulation*

### 3.2.1. Task Vertex Splitting

As described in Section 3.1, the truck performed six types of task at the customer location. Notably, when the truck delivered an empty container and simultaneously picked up a full container (after packing) for a shipper or delivered a full container and simultaneously picked up an empty container (after unpacking) for a consignee, the truck could choose between employing the Strip and Discharge modes. Representing a task vertex by a single vertex has been proven insufficient in adequately capturing the Strip mode. Therefore, it was necessary to split such task vertices into two distinct vertices to accurately capture this aspect, with the time window and geographical location of each vertex remaining the same.

As shown in Figure 2, customer $C_i$ with the fifth or sixth type of task was divided into two customers, $C_i^1$ and $C_i^2$. Specifically, $C_i^1$ had an empty (full) container delivery demand, while $C_i^2$ had a full (empty) container pickup demand. A critical constraint emerged with regard to service prioritization: $C_i^1$ needed to precede $C_i^2$ in the service sequence, and the start time for serving $C_i^2$ needed to be equal to or greater than the start time for serving $C_i^1$, taking into account the duration of packing or unpacking.

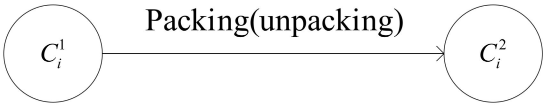


**Figure 2.** Task vertex splitting.

### 3.2.2. The DAOV Graph

In contrast to the conventional VRP, the CDP entailed both the relocation of empty containers and the planning of truck routes simultaneously, which put forward a challenge to the development of a direct mathematical model. Hence, the CDP proposed in this study was mathematically formulated using a DAOV graph [25] on the basis of task vertex splitting.

Let $G = (V, A)$ be a DAOV graph, where $V = (D, C)$ is the vertices set and $A = \{(i, j)|i \in V, j \in V, i \neq j\}$ is the arcs set. It is important to note that set $D$ represents not only the depot nodes but also departing from or returning to the depots; thus, it is referred to as departure or return vertices. Similarly, set $C$ stands for not only the customer nodes but also the determinate activities involved in the container transportation tasks for customers; it is thus called task vertices. Table 2 lists the four types of activities in set $C$. Set $A$ represents indeterminate activities of truck transfer from vertex $i$ to vertex $j$. The specific activities and associated transfer times in arc $A$ are shown in Table 3.

**Table 2.** Activities for task vertices.

| Task Vertices | Activities for Task Vertices |
|---|---|
| $C_{DEC}$ | Delivering an empty container |
| $C_{DFC}$ | Delivering a full container |
| $C_{PEC}$ | Picking up an empty container |
| $C_{PEC}$ | Picking up a full container |

**Table 3.** The DAOV graph.

| From Vertex | Activities and Transfer Time Attribute of Arcs $(i,j)$ | | | | |
|---|---|---|---|---|---|
| | **To Vertex $j \in D$** | **To Vertex $j \in C_{DEC}$** | **To Vertex $j \in C_{DFC}$** | **To Vertex $j \in C_{PEC}$** | **To Vertex $j \in C_{PFC}$** |
| $i \in D$ | / | Pick up an empty container and travel to *j* | Travel to the port, pick up a full container, and travel to *j* | Travel to *j* | Travel to *j* |
| | $\infty$ | $t_{ij}$ | $t_{ip} + t_p + t_{pj}$ | $t_{ij}$ | $t_{ij}$ |
| $i \in C_{DEC}$ | Travel to *j* | / | Travel to the port, pick up a full container, and travel to *j* | Travel to *j* | Travel to *j* |
| | $t_{ij}$ | $\infty$ | $t_{ip} + t_p + t_{pj}$ | $t_{ij}$ | $t_{ij}$ |
| $i \in C_{DFC}$ | Travel to *j* | / | Travel to the port, pick up a full container, and travel to *j* | Travel to *j* | Travel to *j* |
| | $t_{ij}$ | $\infty$ | $t_{ip} + t_p + t_{pj}$ | $t_{ij}$ | $t_{ij}$ |
| $i \in C_{PEC}$ | Travel to *j* and deliver an empty container | Travel to *j* | / | / | / |
| | $t_{ij}$ | $t_{ij}$ | $\infty$ | $\infty$ | $\infty$ |
| $i \in C_{PFC}$ | Travel to the port, deliver a full container, and travel to *j* | / | Travel to the port, deliver and pick up a full container, and travel to *j* | Travel to the port, deliver a full container, and travel to *j* | Travel to the port, deliver a full container, and travel to *j* |
| | $t_{ip} + t_p + t_{pj}$ | $\infty$ | $t_{ip} + 2t_p + t_{pj}$ | $t_{ip} + t_p + t_{pj}$ | $t_{ip} + t_p + t_{pj}$ |

### 3.2.3. Network Flow Constraints

As illustrated in the aforementioned DAOV graph, the truck was unable to perform any activity between specific vertices due to truck capacity constraints. For example, consider the scenario where one truck performs tasks between vertex $C_{PEC}$ and vertex $C_{DFC}$. If the truck is already carrying an empty container, it may not have the capacity to transport a full container to deliver to the next customer. The truck must deliver the empty container to a depot before proceeding to the port to pick up the full container and deliver it to the next customer. In our study, a truck was considered to have completed a journey once it departed from a depot and returned to a depot. If there was sufficient working time, a new index number was assigned to the truck, allowing it to depart again to serve customers. It is worth noting that although it was the same physical truck, it was treated as two separate journeys that were carried out by different trucks due to the use of different index numbers. In addition, to decrease the number of arcs in the network, hyperarcs were generated in cases where the connection between customers and customers (depots) required a visit to the port. Specifically, the port node was "bypassed" in our network, which meant that any node requiring an inbound or outbound connection to it needed to incorporate the detour through the port in its route from/to other nodes. Therefore, the arcs between other nodes and the port could be eliminated. To effectively integrate these network flow constraints, Equations (9)–(20) were introduced into the mathematical model.

### 3.3. Mathematical Formulation

In this section, the new CDP is mathematically formulated as a MILP based on the previously proposed graphical formulation.

#### 3.3.1. Assumptions

The following assumptions were introduced to formulate the proposed CDP in our study:

(1) The locations of customers, depots, and the port, as well as their demands and service time windows, are known in advance and remain static during the planning horizon.
(2) The trucks are homogeneous regarding capacity, and each truck can load one 1 TEU container at a time.
(3) The traveling speed of a truck remains constant regardless of whether it is loaded or not.

#### 3.3.2. Notations and Parameters

$N_p$: Set of ports where only one port is included
$N_c$: Set of task vertices, including $C_{DEC}$, $C_{DFC}$, $C_{PEC}$, and $C_{PFC}$
$N_d$: Set of departure or return vertices
$N$: $N = N_p \cup N_c \cup N_d$
$K$: Set of trucks, $k \in K$
$C_1$: Total fuel and carbon emissions costs per unit of traveling time
$C_2$: Penalty costs per unit of time for trucks that exceed the working time limit
$C_3$: Total dwelling cost and carbon emissions cost per unit of waiting time
$M$: A large integer
$T_{\max}$: Maximum working time for trucks
$[TS_i, TE_i]$: Service time window for task vertex $i$, $i \in N_c$
$n_i$: Frequency limit for truck dispatches from and returns to depot $i$, which was set to avoid too many trucks returning to a single depot, $i \in N_d$
$m_i$: Initial quantity of empty containers at depot $i$, $i \in N_d$
$t_{ij}$: Travel time from vertex $i$ to vertex $j$, $i, j \in N$
$t_i$: The time required for picking up or dropping off a container at vertex $i$, $i \in N$
$\varphi_i$: Predecessor vertex of vertex $i$, $i \in N_c$

#### 3.3.3. Decision Variables

$x_{ijk}$: Binary variable indicating whether truck $k$ travels from vertex $i$ to vertex $j$, $i, j \in N$
$y_{ij}$: The time when a truck departs from depot $i$ and goes to non-depot vertex $j$
$\gamma_{ij}$: The time when a truck departs from non-depot vertex $i$ and returns to depot $j$
$\tau_i$: The time when the activities of vertex $i \in N_c$ are started
$s_k$: The time when the truck $k$ starts working, $k \in K$
$e_k$: The time when the truck $k$ finishes working, $k \in K$
$w_k$: The total hours of the truck $k$ exceed the maximum working hours, $k \in K$

#### 3.3.4. Formulation

$$\min Z = C_1 \times \sum_{i \in N_c} \sum_{j \in N_c} \sum_{k \in K} x_{ijk} \times t_{ij} + C_2 \times \sum_{k \in K} w_k + C_3 \times \sum_{k \in K} \left( e_k - s_k - \sum_{i \in N_c} \sum_{j \in N_c, i \neq j} x_{ijk} \times t_{ij} \right) \tag{1}$$

$$\text{s.t.} \sum_{j \in N_c \cup N_d, i \neq j} x_{ijk} = \sum_{j \in N_c \cup N_d, i \neq j} x_{jik} \ \forall i \in N_c, \forall k \in K \tag{2}$$

$$\sum_{j \in N_d \cup N_c} \sum_{k \in K} x_{ijk} = 1 \ \forall i \in N_c \tag{3}$$

$$\sum_{j \in N_c} \sum_{k \in K} x_{ijk} \leq n_i \ \forall i \in N_d \tag{4}$$

$$\sum_{i\in N_d}\sum_{j\in N_c} x_{ijk} \leq 1 \ \forall k \in K \tag{5}$$

$$\sum_{j\in N_c}\sum_{k\in K} x_{jik} \leq n_i \ \forall i \in N_d \tag{6}$$

$$\sum_{i\in N_c}\sum_{j\in N_d} x_{ijk} \leq 1 \ \forall k \in K \tag{7}$$

$$\sum_{i\in N_d}\sum_{j\in N_c} x_{ijk} = \sum_{i\in N_d}\sum_{j\in N_c} x_{jik} \ \forall k \in K \tag{8}$$

$$\sum_{i\in N_p}\sum_{j\in N}\sum_{k\in K} x_{ijk} =0 \tag{9}$$

$$\sum_{i\in N}\sum_{j\in N_p}\sum_{k\in K} x_{ijk} =0 \tag{10}$$

$$\sum_{i\in N_d}\sum_{j\in N_d}\sum_{k\in K} x_{ijk} =0 \tag{11}$$

$$\sum_{i\in N}\sum_{j\in N, j=i}\sum_{k\in K} x_{ijk} =0 \tag{12}$$

$$\sum_{j\in C_{PEC}\cup C_{PFC}\cup C_{DFC}\cup N_d}\sum_{k\in K} x_{ijk} =1 \ \forall i \in C_{DEC} \tag{13}$$

$$\sum_{j\in C_{PEC}\cup N_d}\sum_{k\in K} x_{jik} =1 \ \forall i \in C_{DEC} \tag{14}$$

$$\sum_{j\in C_{DFC}\cup C_{PFC}\cup C_{PEC}\cup N_d, i\neq j}\sum_{k\in K} x_{ijk} =1 \ \forall i \in C_{DFC} \tag{15}$$

$$\sum_{i\in C_{DEC}\cup C_{DFC}\cup C_{PFC}\cup N_d, i\neq j}\sum_{k\in K} x_{jik} =1 \ \forall i \in C_{DFC} \tag{16}$$

$$\sum_{j\in C_{DEC}\cup N_d}\sum_{k\in K} x_{ijk} =1 \ \forall i \in C_{PEC} \tag{17}$$

$$\sum_{j\in C_{DEC}\cup C_{DFC}\cup C_{PFC}\cup N_d}\sum_{k\in K} x_{jik} =1 \ \forall i \in C_{PEC} \tag{18}$$

$$\sum_{j\in C_{PEC}\cup C_{DFC}\cup C_{PFC}\cup N_d, i\neq j}\sum_{k\in K} x_{ijk} =1 \ \forall i \in C_{PFC} \tag{19}$$

$$\sum_{j\in C_{DEC}\cup C_{DFC}\cup C_{PFC}\cup N_d, i\neq j}\sum_{k\in K} x_{jik} =1 \ \forall i \in C_{PFC} \tag{20}$$

$$\tau_{ij} = \sum_{k\in K} s_k x_{ijk} \ \forall i \in N_d, \forall j \in C_{DEC} \tag{21}$$

$$\tau_{ji} = \sum_{k\in K} e_k x_{jik} \ \forall i \in N_d, \forall j \in C_{PEC} \tag{22}$$

$$\sum_{r\in C_{PEC}, \gamma_{ri}\leq y_{ij}}\sum_{k\in K} x_{rik} - \sum_{q\in C_{DEC}, y_{iq}\leq y_{ij}}\sum_{k\in K} x_{iqk} + m_i \geq 0 \ \forall i \in N_d, \forall j \in C_{DEC} \tag{23}$$

$$\tau_j \geq \tau_i + t_{ij} + t_i - M(1 - x_{ijk}) \ \forall i \in N_c, \forall j \neq i \in N_c, \forall k \in K \tag{24}$$

$$\tau_i \geq \tau_{\varphi_i} + t_{\varphi_i} + t_{\varphi_i i} \ \forall i \in N_c, \forall \varphi_i \in N_c \tag{25}$$

$$TS_i \leq \tau_i \ \forall i \in N_c \tag{26}$$

$$\tau_i + t_i \leq TE_i \ \forall i \in N_c \tag{27}$$

$$s_k \leq \left(\tau_j - t_{ij}\right) x_{ijk} \ \forall i \in N_d, \forall j \in N_c, \forall k \in K \tag{28}$$

$$e_k \geq \left(\tau_i + t_i + t_{ij}\right) x_{ijk} \ \forall i \in N_c, \forall j \in N_d, \forall k \in K \tag{29}$$

$$w_k \geq e_k - s_k - T_{\max} \ \forall k \in K \tag{30}$$

$$e_k \geq s_k \ \forall k \in K \tag{31}$$

$$e_k - s_k - \sum_{i \in N_c} \sum_{j \in N_c, i \neq j} x_{ijk} \times t_{ij} \geq 0 \ \forall k \in K \tag{32}$$

$$x_{ijk} \in \{0,1\} \ \forall i \neq j \in N, \forall k \in K \tag{33}$$

$$w_k \geq 0 \ \forall k \in K \tag{34}$$

$$\tau_i \geq 0 \ \forall i \in N \tag{35}$$

$$y_{ij} \geq 0 \ \forall i \in N_d, \forall j \in C_{DEC} \tag{36}$$

$$\gamma_{ij} \geq 0 \ \forall i \in C_{PEC}, \forall j \in N_d \tag{37}$$

$$s_k, e_k \geq 0 \ \forall k \in K \tag{38}$$

The optimization objective function (1) minimized the total cost, including fuel costs and carbon emission costs from trucks traveling, dwelling costs and carbon emission costs from trucks waiting, and penalty costs for exceeding the maximum working time of trucks. Constraint (2) guaranteed the continuity of truck routes. Constraint (3) enforced the limitations on the number of times a task vertex was served. Constraints (4) and (6) assigned the maximum number of times that trucks were allowed to depart from and return to depots. Constraints (5), (7), and (8) regulated the procedures for trucks' departures and returns. Specifically, a truck could depart from only one depot and needed to return to one depot once it had departed. Constraints (9)–(20) ensured the network flow constraints between vertices, which prohibited trucks from traveling between certain vertices. More specifically, constraints (9)–(10) made sure that no network flow entered or left the port, as the port functioned solely as a bypass node in the hyperarcs. Constraints (11) and (12) respectively designated that there was no network flow between depots or identical vertices. Constraints (13), (15), (17), and (19) respectively reflected network flow constraints from task vertices $C_{DEC}$, $C_{DFC}$, $C_{PEC}$, and $C_{PFC}$ to other vertices, while constraints (14), (16), (18), and (20) were the corresponding opposite network flow constraints. For instance, constraint (13) meant that trucks could only travel from $C_{DEC}$ to task vertices $C_{PEC}$, $C_{PFC}$, and $C_{DFC}$ or return vertex $N_d$, and the total number of network flows was limited to 1. Constraint (21) calculated the specific time when a truck departed from a task vertex and then visited a depot, while constraint (22) determined the time when a truck departed from a depot. Both times were used to express the limitations on the number of empty containers in the depot in constraint (23). Constraint (23) specified that a depot should have a minimum of zero empty containers remaining after a truck picked up an empty container from the depot. Constraint (24) stated the continuity of the customer's service time. Constraint (25) ensured compliance with customer service priority constraints. Constraints (26) and (27) set limitations on the time windows when serving customers. Constraints (28) and (29) calculated the time for a truck to depart from and return to a depot. Constraint (30) stated how long a truck could work after exceeding the maximum working time. Constraint (31) mandated that the time a truck returned to a depot needed to be greater than or equal to the time it departed from a depot. Constraint (32) required that the total working time of a truck needed to be greater than or equal to its traveling time. Constraints (33)–(38) clarified the types of decision variables.

### 3.3.5. Linearization of Nonlinear Constraints

Constraint (23) was a nonlinear constraint. However, it could be transformed into the following linear constraints (39)–(43) by incorporating binary variables $U_{rij}$ and $U_{qij}$. $U_{rij}$ corresponded to the connection between the arc $\{(r,i)|\forall i \in N_d, \forall r \in C_{PEC}\}$ with the arc $\{(i,j)|\forall i \in N_d, \forall j \in C_{DEC}\}$, while $U_{qij}$ was the connection between the arcs $\{(i,q)|\forall i \in N_d,$ $\forall q \in C_{DEC}\}$ with the arc $(i,j)$.

$$\gamma_{ri} - y_{ij} \leq M\left(1 - U_{rij}\right) \; \forall i \in N_d, \forall j \in C_{DEC}, \forall r \in C_{PEC} \tag{39}$$

$$\gamma_{ri} - y_{ij} \geq -MU_{rij} \; + \varepsilon \; \forall i \in N_d, \forall j \in C_{DEC}, \forall r \in C_{PEC} \tag{40}$$

$$\gamma_{ri} - y_{ij} \leq M\left(1 - U_{rij}\right) \; \forall i \in N_d, \forall j \in C_{DEC}, \forall r \in C_{PEC} \tag{41}$$

$$\gamma_{ri} - y_{ij} \geq -MU_{rij} \; + \varepsilon \; \forall i \in N_d, \forall j \in C_{DEC}, \forall r \in C_{PEC} \tag{42}$$

$$\sum_{r \in PEC} \sum_{k \in K} x_{rik} U_{rij} - \sum_{q \in DEC} \sum_{k \in K} x_{iqk} U_{qij} + m_i \geq 0 \; \forall i \in N_d, \forall j \in C_{DEC} \tag{43}$$

Herein, $M$ is a large enough positive constant and $\varepsilon$ is a small enough positive constant. If $\gamma_{ri} > y_{ij}$ or $y_{iq} > y_{ij}$, which meant that the time taken by arc $(r,i)$ or $(i,q)$ to access the depot was greater than the time taken by arc $(i,j)$ to access the depot, then $U_{rij} = 0$ or $U_{qij} = 0$. Conversely, if $\gamma_{ri} \leq y_{ij}$ or $y_{iq} \leq y_{ij}$, which meant that the time taken by arc $(r,i)$ or $(i,q)$ to access the depot was less than or equal to the time taken by arc $(i,j)$ to access the depot, then $U_{rij} = 1$ or $U_{qij} = 1$. In summary, constraint (23) was equivalent to the linear constraints above.

## 4. Solution Algorithm

The proposed MILP model was amenable to solutions using commercial software, such as Gurobi 10.0.2. However, its applicability was restricted to addressing modest-scale instances. Due to the extensive memory requirements and lengthy computational time associated with large-scale instances, an alternative GA approach was devised to tackle this problem.

The concept of GA, originally proposed by Holland [44], closely emulates the mechanisms of biological evolution. GA has gained widespread acceptance and application in addressing intricate combinatorial optimization problems [45–47]. The operationalization of GA commences with the generation of an initial population of solutions, a process executed either through random generation or in accordance with predefined criteria. Subsequently, a subset of solutions within the current generation is selected based on their superior fitness scores, as determined by objective functions and feasibility assessments. The selected solutions serve as the progenitors for generating offspring via genetic operators, including crossover, mutation, and selection, thereby populating the ensuing generation. This cyclic procedure continues until predefined termination criteria are met, such as reaching a maximum number of iterations.

In order to enhance the computational efficiency, two innovative techniques were introduced into the GA, namely, the "sequential insertion" method and the "solution re-optimization" operation. Since the solution of GA was construed as an ordered sequence of customers using a real-number encoding approach, the "sequential insertion" method was designed to insert a suitable depot into the theoretical solution to demarcate it into multiple viable truck routes. While these routes were feasible, they could still incur high truck waiting costs and occasionally violate constraints on the quantity of empty containers at depots and customer service priority. Therefore, the "solution re-optimization" operation was proposed to further improve the quality of solutions. A flowchart of the improved GA is shown in Figure 3, followed by a detailed description of the implementation process.

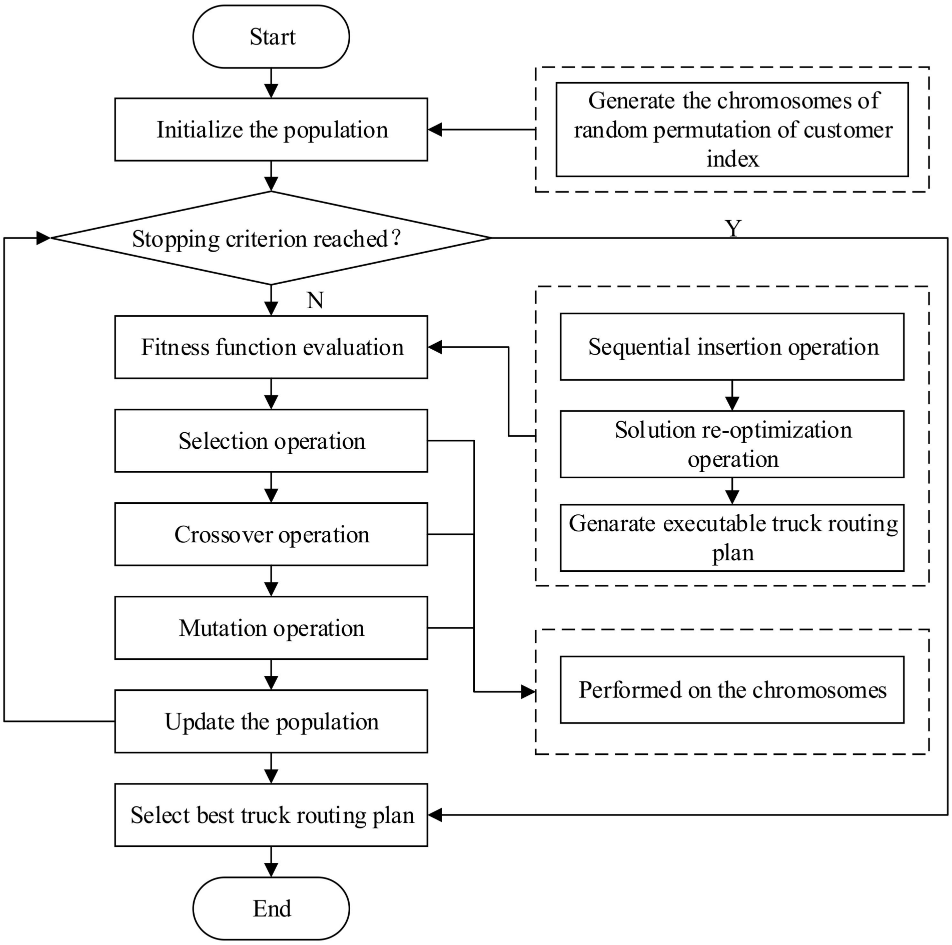
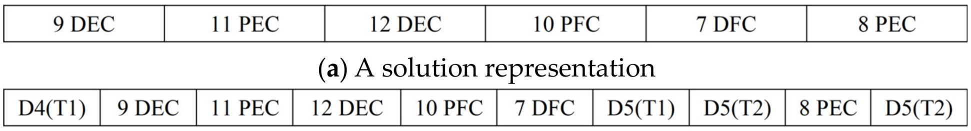

Start
Initialize the population
Generate the chromosomes of random permutation of customer index
Stopping criterion reached?
Y
N
Fitness function evaluation
Sequential insertion operation
Selection operation
Solution re-optimization operation
Crossover operation
Genarate executable truck routing plan
Mutation operation
Performed on the chromosomes
Update the population
Select best truck routing plan
End

**Figure 3.** Flowchart of the improved GA.

### 4.1. Initial Solution Representation

For the VRP, a common approach to represent a solution by a chromosome involves determining the optimal routes assigned to each truck, which proved unsuitable for the CDP proposed in our study. The reason is that the CDP introduced a multifaceted set of constraints, such as truck capacity, customer service time windows, empty container quantity at the depots, and customer service priority constraints. Under these constraints, the traditional solution representation resulted in a large number of infeasible truck routes when applying crossover and mutation operations, which affected the efficiency and convergence of the algorithm.

Furthermore, in light of the multifaceted considerations within this study, including two operation modes (Strip and Discharge), coupled with two strategies for relocating empty containers (Street-turn and Depot-turn), it became inherently challenging to accurately anticipate the length of an individual route. Therefore, a new approach was employed to solution representation, as advocated by Yang et al. [6]. In this paradigm, the solution was construed as an ordered sequence of customers using a real-number encoding approach. As depicted in Figure 4a, all customers were randomly positioned within the solution, resulting in a solution that precisely reflected the total count of customers. However, the assignment of trucks to each customer and the time of trucks visiting customers remained undetermined at this stage. To transform these theoretical solutions into executable routes, a "sequential insertion" method was implemented by aligning with the constraints of truck capacity, truck working hours, customer time window, and the frequency limit for truck dispatches from and returns to the depot.



| | | | | | |
|---|---|---|---|---|---|
| 9 DEC | 11 PEC | 12 DEC | 10 PFC | 7 DFC | 8 PEC |

(**a**) A solution representation

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| D4(T1) | 9 DEC | 11 PEC | 12 DEC | 10 PFC | 7 DFC | D5(T1) | D5(T2) | 8 PEC | D5(T2) |

(**b**) An executable truck route after transformation

**Figure 4.** Sample solution representation and the executable truck route after performing Algorithm 1.

### *4.2. Sequential Insertion Method*

The "sequential insertion" method constructed a complete truck route by incorporating depots into the solution of the customer sequence. This process entailed evaluating the current truck's capacity to accommodate the present customer's requirements, taking into account the truck's operational time limit and the customer's service time window. If the truck could effectively fulfill the customer's needs, the customer was added to the truck's service queue. Temporal considerations, including the time required for the truck to visit all nodes, were then adjusted based on the minimum total working time principle, accounting for the customer's service time window. Subsequently, the working time of the truck was updated. This iterative procedure persisted as the algorithm proceeded to evaluate each subsequent customer in the sequence.

For instances where the current customer could not be accommodated within the current truck's constraints, the algorithm triggered a selection process to determine the appropriate depot for truck return. This selection considered both the frequency limit for truck returns to a depot and the closest distance principle. A new truck was then assigned to serve the current customer, and the "sequential insertion" process recommenced.

Note that if the current customer was the first to be served by a truck, it was necessary to select a depot for truck dispatch by considering the frequency limit for truck dispatches from a depot, as well as the closest distance principle. If the customer's requirement was to deliver an empty container, then the available quantity of empty containers was also considered.

To clarify the insertion process, a scenario featuring two depots, two trucks, and six customers is illustrated. As visually depicted in Figure 4a, a solution was initially established as a random arrangement of customer indices. In order to make it an executable truck route, depots were strategically introduced into the solution, thereby demarcating it into discrete sub-solutions, each indicating a viable route for a truck to traverse, as elucidated in Figure 4b.

To explain in more detail, considering the frequency limit for truck dispatches from depots, the available number of empty containers in a depot, and the closest distance principle, the first truck (denoted as T1) started from depot D4. Its journey entailed the initial transport of an empty container for customer 9. Subsequently, T1 proceeded to customer 11 to pick up an empty container and then headed to customer 12 to deliver the empty container. Afterward, T1 continued its journey to customer 10 to pick up a full container and subsequently headed towards customer 7 to deliver a full container. It is worth noting that the truck could not directly transport from customer 10 to customer 7 because the full container from customer 10 needed to be transported to the port, and the full container customer 7 requested originated from the port. Therefore, T1 needed to detour to the port, where it dropped off the full container obtained from customer 10 and subsequently picked up another full container for delivery to customer 7. Due to the restriction of working time, T1 was unable to continue its journey from customer 7 to customer 8, and thus returned to depot D5 based on the frequency limit for truck returns to a depot and the closest distance principle. Thereupon, a new truck (T2) from D5 was dispatched to fulfill the task of serving customer 8 and finally returned to depot D5. This iterative "check and insert" process persisted until all customers were examined and accommodated within the defined operational constraints.

**Algorithm 1. Sequential Insertion Method**

```
Step 0: Let $n$ denote the total number of customers. Let $i$ denote the index of the current
  customer in the customer sequence. Let $k$ denote the index of the current truck. Set $i = 1$
  and $k = 1$.
Step 1: If there is no en-route truck $k$ to be selected for customer $i$, go to Step 2.
  Otherwise, go to Step 3.
Step 2: If customer $i$ requires a delivery of an empty container, go to Step 2.1. Otherwise,
  go to Step 2.2.
  Step 2.1: Select a depot to dispatch truck $k$, considering the frequency limit for truck
    dispatches from a depot, the available quantity of empty containers, and the closest
    distance principle. Go to Step 2.3.
  Step 2.2: Select a depot to dispatch truck $k$, considering the frequency limit for truck
    dispatches from a depot and the closest distance principle. Go to Step 2.3.
  Step 2.3: Calculate the visit time of customer $i$, the departure time of truck $k$ from the
    depot, and the total working time of truck $k$.
    If $i = n$, go to Step 5.
    If $i < n$, insert customer $i$ into the service sequence of truck $k$. Meanwhile, update the
        visit time of customer $i$ and the total working time of truck $k$. Set $i = i + 1$ and go
        to Step 1.
Step 3: Calculate the visit time of customer $i$ and determine if it is within the time
  window of customer $i$. If not, go to Step 4. If it is, calculate the latest visit time of the
  preceding customer served by the current truck to recalculate the departure time of
  truck $k$ from the depot and the total working time of truck $k$.
    If $i = n$ and truck $k$ does not exceed its working time limit, go to Step 5.
    If $i < n$ and truck $k$ does not exceed its working time limit, insert customer $i$ into the
        service sequence of truck $k$. Meanwhile, update the visit time of customer $i$ and
        the total working time of truck $k$. Set $i = i + 1$ and go to Step 1.
    If truck k exceeds its working time limit, go to Step 4.
Step 4: Select a depot for truck $k$ to return based on the frequency limit for truck returns
  to a depot and the closest distance principle. Set $k = k + 1$ and go to Step 1.
Step 5: Insert customer $i$ into the service sequence of truck $k$. Select a depot for truck $k$ to
  return based on the frequency limit for truck returns to a depot and the closest
  distance principle. The algorithm will stop and the truck routing scheme will be
  generated.
```

Note that the proposed "sequential insertion" method served as a heuristic approach and did not offer a guarantee of achieving optimality, even when the solution sequence was derived from an optimal truck routing plan. There are two primary reasons for this limitation:

(1) The algorithm prioritized utilizing the maximum working hours of the truck. However, there were cases where using the maximum working hours of each truck did not necessarily lead to optimal results.

(2) It transformed a solution sequence into a truck routing plan by specific rules to ensure feasibility and explored the huge amount of feasible space within a limited number of iterations. However, this process did not guarantee optimality.

### *4.3. Solution Re-Optimization Operation*

It is important to note that the solution produced by the “sequential insertion” method could result in high truck waiting costs and occasionally violate other constraints, such as the available quantity of empty containers at the depots and customer service priorities. Therefore, the solution needed to be improved by the proposed “solution re-optimization” operation (Algorithm 2), which consisted of two main steps.

The first step focused on adjusting the time of the truck visiting each node, while keeping the departure and return depots, served customers, and serving sequence unchanged. The objective was to minimize the total working time of the trucks while adhering to a set of constraints. If the re-optimized solution violated the customer service priority constraint, then it was considered infeasible; otherwise, the following second step was carried out.

The second step aimed to evaluate whether the re-optimized feasible solution complied with the constraint of the empty container quantity at depots. As is known, the number of empty containers at the depots dynamically fluctuated while trucks undertook the two types of actions of delivering an empty container to a depot or picking up an empty container from a depot. These actions were closely related to task types 1, 4, 5, and 6 because the destination or origin of the tasks could be the depot. However, in the “sequential insertion” method, the operation of inserting a depot was implemented according to the order of customers in the solution sequence, ignoring the correlation between the dynamic variation of the empty container quantity at depots and the specific completion time of the above two types of actions. Therefore, the re-optimize operation extracted the specific time for the delivery and pickup of empty containers by trucks at each depot based on the result of the first step. It then dynamically calculated the number of empty containers at each depot. If the number of empty containers was less than zero, the solution was considered infeasible; otherwise, the fitness value of the solution was adjusted based on the optimization result.

**Algorithm 2. Solution Re-Optimization**

```
Step0: Obtain the initial solution using the “sequential insertion” method, including the departure and destination depots, served customers, and service order of each truck $k$.
Step1: Let $i_k$ denote the vertex in the service sequence of truck ($k \in K$). Let $i'_k$ denote the subsequent vertex of $i_k$.
Step2: Set the optimization objective to minimize the total working time of all trucks:
    $\sum(e_k - s_k)\ \forall k \in K$
Step3: Add constraints:
(1) Temporal continuity constraints for truck visiting nodes sequentially:
    $\tau_{i_k} + t_{i_k i'_k} \leq \tau_{i'_k}\ \forall k \in K$
(2) Customer service time window constraints:
    $TS_i \leq \tau_i\ \forall i \in N_c$
    $\tau_i + t_i \leq TE_i\ \forall i \in N_c$
(3) Customer service priority constraints:
    $\tau_i \geq \tau_{\varphi_i} + t_{\varphi_i} + t_{\varphi_i i}\ \forall i \in N_c, \forall \varphi_i \in N_c$
Step4: Use the linprog function in Matlab R2022a to solve the model.
Step5: If the optimized solution satisfies the customer service priority constraint, check whether it also complies with the constraint on the quantity of empty containers at depots. If it does, reassign its fitness value based on the optimization result; otherwise, consider the solution infeasible.
```

### *4.4. Implementation of GA*

In order to address the MILP in our study, the implementation of GA was exhaustively delineated. The configuration of GA parameters, including population sizes, the number of generations, crossover and mutation rates, and other pertinent factors, was tailored to suit the unique characteristics of individual test instances.

#### 4.4.1. Initialization of the Population

Firstly, a specific number of chromosomes was generated through the random permutation of the customer index in accordance with the predetermined population size. It is imperative to note that these chromosomes did not directly represent tangible truck routing plans because they had the same length, thus failing to represent the number of trucks used and the number of nodes each truck visited. The crossover and mutation operations were all based on the chromosome of the random permutation of the customer index (Figure 4a), rather than the tangible truck routing plans (Figure 4b). On the contrary, to calculate the fitness value of each chromosome, they were transformed into concrete truck routing plans through the "sequential insertion" method described in Section 4.2 and the "solution re-optimization" operation described in Section 4.3.

#### 4.4.2. Fitness Function and Evaluation

The fitness function played a pivotal role in assessing the chromosomes' quality, thereby facilitating their ranking for subsequent GA operations. The primary optimization objective was to minimize the total cost. For the initial and generated population, if the individuals were infeasible, their respective fitness function values were set to a relatively infinitely small value. Conversely, the fitness function values of feasible individuals were set to be equivalent to the inverse of the total cost value of formulation (44), reflecting the quality of their corresponding solutions.

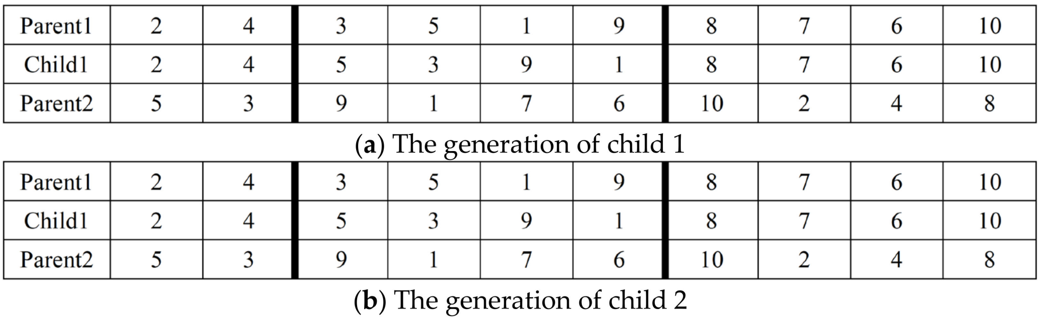
$$Fit = 1/\left(C_1 \times \sum_{k\in K}\sum_{i\in N_c}\sum_{j\in N_c} x_{ijk} \times t_{ij} + C_2 \times \sum_{k\in K} w_k + C_3 \times \sum_{k\in K}\left(e_k - s_k - \sum_{i\in N_c}\sum_{j\in N_c, i\neq j} x_{ijk} \times t_{ij}\right)\right) \tag{44}$$

#### 4.4.3. Selection Process

The generation of the new population of chromosomes was achieved through the implementation of specific recombination processes, including crossover and mutation. It was essential to apply an appropriate selection process to identify suitable parents. This study employed the rank selection approach. The process began by sorting all individuals in the population in descending order based on their fitness values, which ranked the individuals from the highest fitness to the lowest fitness. The probability of selection for any individual could be obtained by dividing the rank index by the total number of individuals. Next, a certain number of individuals were selected as the parents for the next generation by considering the population size and the selection probability of the sorted individuals. The purpose of rank selection was to ensure that the best individuals, those with higher fitness, were preserved and carried forward in the evolutionary process and copied into the initial population of the next generation to produce better new individuals.

#### 4.4.4. Crossover Operation

As a fundamental genetic operator, the crossover operation was executed as follows. At first, all individuals in the initial population were randomly paired with one another. Subsequently, a unique random number was assigned to each pair of individuals. Pairs possessing random numbers falling below the specified crossover probability threshold underwent the crossover operation. Both the parent individuals forming the pairs and the resultant offspring generated through the crossover process were preserved for subsequent mutation operations. Notably, the methodology employed in this study for crossover was rooted in subsequences. For each pair of individuals, two random numbers were generated, enabling the creation of subsequences. Within the subsequence of one individual, customers were reorganized based on their respective positions in the permutation in their paired individual. This organizational approach resulted in the generation of novel individuals, as illustrated in Figure 5.



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Parent1 | 2 | 4 | 3 | 5 | 1 | 9 | 8 | 7 | 6 | 10 |
| Child1 | 2 | 4 | 5 | 3 | 9 | 1 | 8 | 7 | 6 | 10 |
| Parent2 | 5 | 3 | 9 | 1 | 7 | 6 | 10 | 2 | 4 | 8 |

(**a**) The generation of child 1

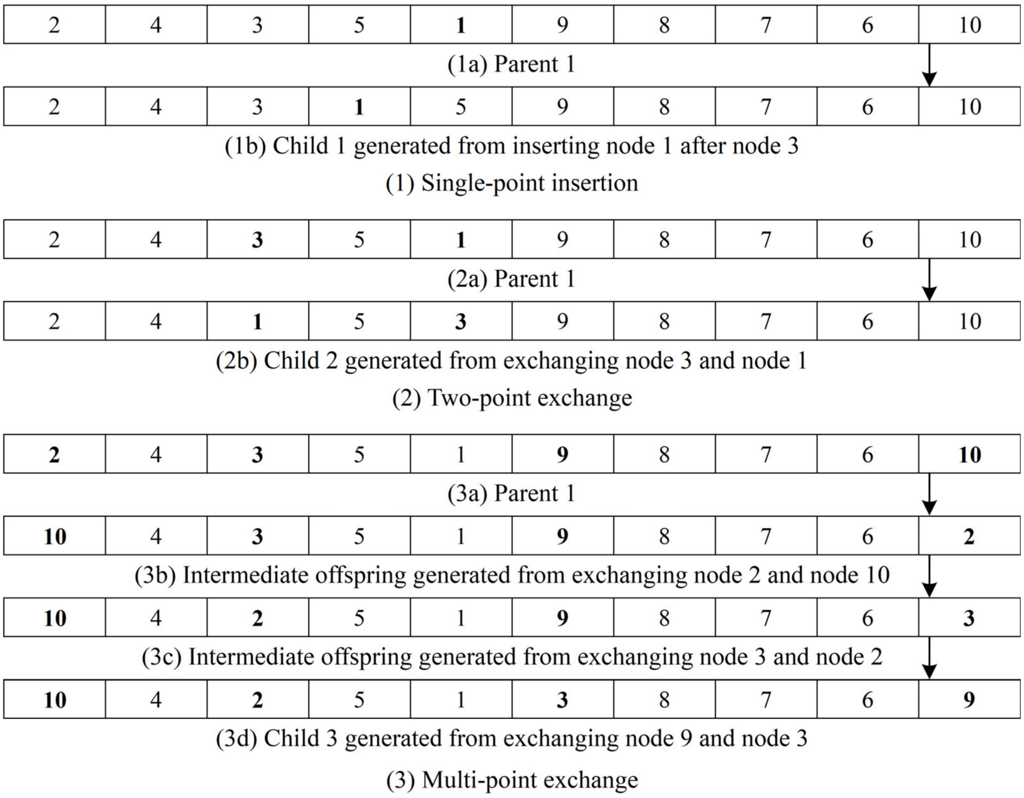
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Parent1 | 2 | 4 | 3 | 5 | 1 | 9 | 8 | 7 | 6 | 10 |
| Child1 | 2 | 4 | 5 | 3 | 9 | 1 | 8 | 7 | 6 | 10 |
| Parent2 | 5 | 3 | 9 | 1 | 7 | 6 | 10 | 2 | 4 | 8 |

(**b**) The generation of child 2

**Figure 5.** Crossover operations.

### 4.4.5. Mutation Operation

To effectively increase the solution space of GA, three distinct mutation methods were introduced: single-point insertion, two-point exchange, and multi-point exchange. The choice of mutation methods depended on the size of the problem at hand. When dealing with smaller problem sizes, one or two mutation methods were selected. Conversely, when solving larger population sizes, all three mutation methods were employed concurrently. Figure 6 provides a graphical representation of the mutation process.



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 3 | 5 | **1** | 9 | 8 | 7 | 6 | 10 |

(1a) Parent 1

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 3 | **1** | 5 | 9 | 8 | 7 | 6 | 10 |

(1b) Child 1 generated from inserting node 1 after node 3

(1) Single-point insertion

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 4 | **3** | 5 | **1** | 9 | 8 | 7 | 6 | 10 |

(2a) Parent 1

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 4 | **1** | 5 | **3** | 9 | 8 | 7 | 6 | 10 |

(2b) Child 2 generated from exchanging node 3 and node 1

(2) Two-point exchange

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **2** | 4 | **3** | 5 | 1 | **9** | 8 | 7 | 6 | **10** |

(3a) Parent 1

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **10** | 4 | **3** | 5 | 1 | **9** | 8 | 7 | 6 | **2** |

(3b) Intermediate offspring generated from exchanging node 2 and node 10

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **10** | 4 | **2** | 5 | 1 | **9** | 8 | 7 | 6 | **3** |

(3c) Intermediate offspring generated from exchanging node 3 and node 2

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **10** | 4 | **2** | 5 | 1 | **3** | 8 | 7 | 6 | **9** |

(3d) Child 3 generated from exchanging node 9 and node 3

(3) Multi-point exchange

**Figure 6.** Mutation operations.

Single-point insertion randomly selected one position within the chromosome and inserted the node in the selected position into another random position. Subsequently, the nodes at the insertion position and all subsequent positions were moved backward in sequence. The generation of child 1 was shown in Figure 6 (1a) and (1b).

Two-point exchange entailed the random selection of two positions within the chromosome, and the nodes at these positions were then exchanged. Child 2 was created as given in Figure 6 (2a) and (2b).

In the case of multi-point exchange, a random number was assigned to each position within the chromosome. Subsequently, the position with a random number lower than the specified mutation probability and the highest random number value were identified. The nodes at each position, whose random numbers were below the specified mutation probability, were sequentially exchanged with the node in the position with the highest random number value according to the order of the customer numbers in the chromosomes. Figure 6 (3a) to (3d) illustrated the detailed process of generating child 3.

#### 4.4.6. Stopping Criterion

The GA continued to iterate until a predefined stopping criterion was fulfilled. This study employed the maximum number of iterations as the criterion. The maximum number of iterations set an upper limit on the number of generations the algorithm would undergo before terminating. Different numbers of iterations could be implemented for multiple-scale instances to achieve optimal convergence.

## 5. Computational Experiments

This section validates and evaluates the mathematical model, as well as the GA, through numerical experiments. All experiments were conducted on a personal computer equipped with an Intel Xeon W-2775 28-core CPU @ 3.30 GHz and 64.0 GB of RAM and running a 64-bit Windows 11 operating system. The mathematical model was solved using Gurobi 10.0.2, and the Gurobi solver was invoked using the Python language in Pycharm Community Edition 2023.1.3. The implementation of the GA was performed using Matlab R2022a with the C++ language. To ensure the reliability of the results, each experimental instance was solved multiple times with the GA, which could differ according to the problem size, thereby mitigating the influence of randomness. The parameters of the GA were set differently for each test instance, considering their specific characteristics.

### *5.1. Experiment Instances*

The experiment instances were generated randomly, following a similar approach employed in previous studies, such as those by Zhang et al. [33] and Zhang et al. [25]. Initially, the locations of one port, six inland depots, and multiple customers were randomly generated on an Euclidean plane. The dimensions of the plane were set to match a truck's travel distance of 3 h. The time required for loading (unloading) a container to (from) a truck was set to 0.1 h, while the time for packing or unpacking a container was set to 0.4 h. Furthermore, the left boundary of the time window for customers was randomly generated within the [0, 4] hour range, and the width of the time window was distributed within the 4 to 8 h range.

Recognizing the diverse task compositions encountered within real-world operational contexts, this study generated a total of 50 varying scale experiment instances with heightened complexity, characterized by the quantity of each type of task and the quantity of empty containers at each depot. For example, instance 7 comprised 12 tasks and 2 empty containers distributed across various depots. Specifically, the number of tasks from type 1 to type 6 was respectively 3, 3, 1, 1, 3, and 1, while the number of empty containers from depot 1 to depot 6 was respectively 0, 1, 1, 0, 0, and 0.

In the context of small, medium, and large-scale instances, there were six types of task, as described in Section 3.1. However, in practice, some customers could require trucks to wait at their locations during packing or unpacking operations to deliver empty (full) and then pick up full (empty) containers after packing (unpacking), which was the fifth (sixth) type of task, but the Strip mode was not applicable. Therefore, to make the computational experiment more practical and evaluate the performance of the GA, the presence of these special customers was introduced in some of the super-scale instances. Taking instance 44 as an example, "75 (15)" meant there were 75 customers with tasks of type 5, among which, 15 customers required the truck to wait for packing or unpacking.

### 5.2. Experiment Results

#### 5.2.1. Experiments on Small-Scale and Medium-Scale Instances

A set of small-scale and medium-scale instances was generated to test the performance of the model in terms of the computation time and the quality of the solutions, as shown in Tables 4 and 5. Since Gurobi could not find the optimal solution within 1 h for medium-scale instances, the performance of the GA approach was assessed by comparing its solutions to the best ones obtained from solving the model for 1 h.

**Table 4.** Comparison of Gurobi and GA solutions for small-scale instances.

| Instance | No. of All Tasks | No. of Each Type of Tasks | No. of Empty Containers at Each Depot | Cost | | Time (s) | | Gap from Gurobi (%) |
|---|---|---|---|---|---|---|---|---|
| | | | | Gurobi | GA | Gurobi | GA | |
| 1 | 3 | (1, 0, 0, 0, 1, 1) | (1, 0, 1, 0, 0, 0) | 84.7 | 85.6 | 37 | 0.60 | 1.06 |
| 2 | 4 | (0, 1, 1, 0, 1, 1) | (1, 0, 1, 0, 0, 0) | 81.8 | 81.8 | 32 | 1.38 | 0.00 |
| 3 | 6 | (1, 2, 0, 1, 2, 0) | (1, 0, 1, 0, 0, 0) | 126.7 | 126.7 | 47 | 1.50 | 0.00 |
| 4 | 6 | (0, 1, 0, 1, 4, 0) | (1, 0, 1, 0, 0, 0) | 147.3 | 147.3 | 67 | 6.44 | 0.00 |
| 5 | 7 | (0, 1, 0, 2, 1, 3) | (1, 0, 1, 0, 0, 0) | 171.9 | 171.9 | 72 | 6.71 | 0.00 |
| 6 | 7 | (0, 0, 0, 2, 1, 4) | (1, 0, 0, 2, 1, 0) | 192.5 | 192.5 | 1131 | 11.42 | 0.00 |
| 7 | 12 | (3, 3, 1, 1, 3, 1) | (0, 1, 1, 0, 0, 0) | 226.4 | 231.3 | 178 | 22.18 | 2.16 |
| 8 | 12 | (0, 0, 3, 0, 8, 1) | (0, 0, 1, 2, 1, 0) | 291.2 | 305.0 | 1377 | 34.46 | 4.74 |
| 9 | 16 | (3, 2, 2, 3, 3, 3) | (1, 0, 0, 0, 2, 1) | 312.9 | 326.2 | 796 | 33.44 | 4.25 |
| 10 | 19 | (2, 2, 4, 5, 2, 4) | (4, 1, 0, 0, 3, 0) | 367.1 | 380.8 | 747 | 29.16 | 3.73 |

**Table 5.** Comparison of Gurobi and GA solutions for medium-scale instances.

| Instance | No. of All Tasks | No. of Each Type of Tasks | No. of Empty Containers at Each Depot | Cost | | Time (s) | | Gap from Gurobi (%) |
|---|---|---|---|---|---|---|---|---|
| | | | | Gurobi | GA | Gurobi | GA | |
| 11 | 19 | (0, 0, 0, 4, 4, 11) | (5, 0, 0, 3, 0, 0) | 516.9 | 525.8 | 3600 | 71.49 | 1.72 |
| 12 | 23 | (3, 2, 5, 5, 3, 5) | (0, 3, 5, 0, 0, 0) | 453.9 | 469.0 | 3600 | 58.34 | 3.33 |
| 13 | 26 | (4, 5, 5, 4, 4, 4) | (3, 0, 0, 1, 0, 2) | 455.0 | 484.8 | 3600 | 43.58 | 6.55 |
| 14 | 26 | (2, 2, 2, 0, 10, 10) | (0, 5, 2, 0, 0, 1) | 765.0 | 755.3 | 3600 | 10.35 | −1.27 |
| 15 | 29 | (6, 6, 3, 3, 7, 4) | (0, 1, 0, 0, 3, 1) | 542.3 | 567.1 | 3600 | 67.58 | 4.57 |
| 16 | 29 | (3, 2, 2, 0, 11, 11) | (4, 0, 4, 4, 0, 1) | 814.0 | 745.5 | 3600 | 57.99 | −8.41 |
| 17 | 29 | (3, 1, 2, 0, 17, 6) | (0, 0, 3, 0, 1, 0) | 815.5 | 815.6 | 3600 | 32.03 | 0.01 |
| 18 | 30 | (5, 6, 3, 6, 4, 6) | (1, 0, 4, 3, 0, 0) | 576.6 | 571.2 | 3600 | 84.64 | −0.94 |
| 19 | 34 | (6, 8, 6, 2, 8, 4) | (0, 3, 2, 0, 0, 0) | 646.4 | 663.8 | 3600 | 77.74 | 2.69 |
| 20 | 35 | (3, 2, 2, 1, 21, 6) | (0, 5, 0, 1, 1, 0) | 957.9 | 926.9 | 3600 | 55.30 | −3.23 |

The results in Tables 4 and 5 substantiate the validity of the model. Specifically, for small- and medium-scale instances, Gurobi demonstrated its ability to find the optimal or near-optimal solution within 1 h. It is noteworthy that for all the small-scale instances, when the total number of tasks after task vertices splitting did not exceed 25, Gurobi consistently produced the optimal solutions within the average solving time of 448.40 s. Furthermore, for the medium-scale instances, Gurobi exhibited the capability to produce high-quality solutions within the specified time limit of 1 h. Such a performance of the mathematical model can be deemed acceptable and practical for real-world applications.

The performance of GA is also validated from the results in Tables 4 and 5. Notably, GA was capable of achieving the optimal solutions within an average solving time of 5.49 s for small-scale instances 2 to 6, while Gurobi had an average solving time of 269.80 s. For other small-scale instances, although GA failed to find the optimal solution, the average sub-optimality gap of these instances was no higher than 3.19%. Additionally, GA was significantly faster than Gurobi, with average solving times of 23.97 s and 627 s, respectively, indicating a notable disparity in performance. Furthermore, for certain medium-scale instances (14, 16, 18, and 20), GA showed the ability to find better solutions than Gurobi, with an average solving time of 52.07 s, even when Gurobi was given 1 h. However, for other medium-scale instances, GA obtained worse solutions than Gurobi, but the average sub-optimality gap was no more than 3.15%. The reason that GA obtained worse solutions than Gurobi can be attributed to two main factors: firstly, the GA got stuck in a local optimum; secondly, because the proposed "sequential insertion" method always tried to convert a solution sequence into a truck routing plan according to specific rules aimed at ensuring plan feasibility, and it did not guarantee that the resulting solutions would be optimal, even when the solution sequence was derived from the optimal truck routing plan. Despite this limitation, GA consistently exhibited the capacity to generate solutions that were superior, comparable, or at least reasonably suboptimal compared to solving the model, and the sub-optimality gaps of the GA solution in all small- and medium-scale instances were smaller than 6.55%. These results validate the effectiveness of the GA and its potential for larger-scale instances.

### 5.2.2. Experiments on Large-Scale and Super-Scale Instances

The inherent NP-hard characteristics of the MILP model render the attainment of optimal solutions for larger-scale instances impractical. Providing even a feasible solution for such scale instances is challenging. This section reports on the performance of the GA on large-scale and super-scale instances. One constraint to consider was the operational capacity of a sub-fleet of trucks, a limitation defined in previous research by Wang and Regan [11], stipulating a maximum handling capacity of 75 container tasks per day. Taking this constraint into account, our investigation involved the generation of 20 distinct large-scale instances (Table 6). The total amount of tasks did not exceed 75 before task vertices splitting and 135 after splitting. Furthermore, for the purpose of validating the algorithm's solving capability for super-scale experiments, 10 super-scale instances (Table 7) featuring container task counts ranging from 100 to 480 were generated, with the total number of tasks after task vertices splitting ranging from 160 to 760.

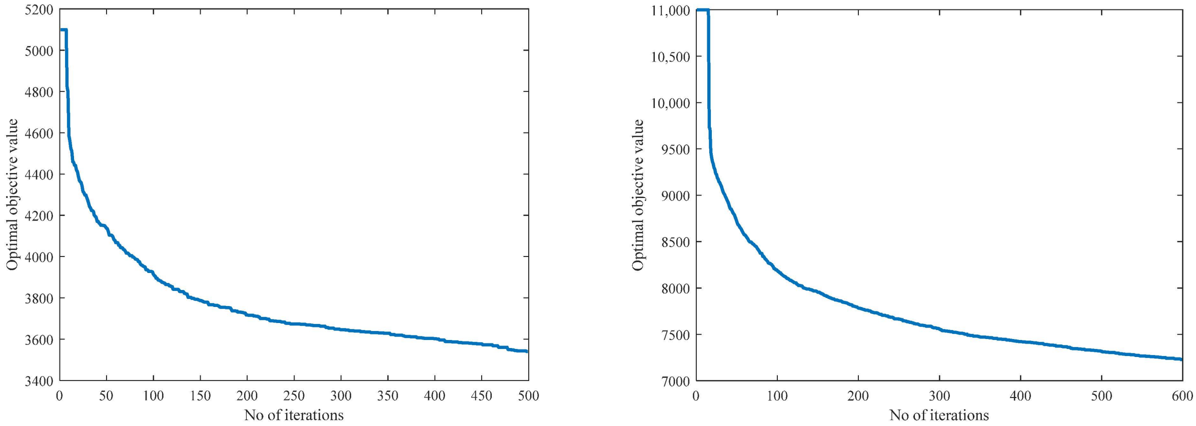
**Table 6.** Results of solving large-scale instances by GA.

| Instance | No. of All Tasks | No. of Each Type of Task | No. of Empty Containers at Each Depot | GA | |
|---|---|---|---|---|---|
| | | | | Cost | Time (s) |
| 21 | 38 | (4, 4, 9, 9, 4, 8) | (4, 4, 0, 0, 0, 5) | 796.2 | 58.36 |
| 22 | 43 | (5, 5, 10, 10, 5, 8) | (5, 3, 4, 1, 0, 2) | 875.5 | 56.66 |
| 23 | 46 | (11, 10, 6, 5, 9, 5) | (0, 1, 2, 0, 1, 2) | 898.0 | 64.53 |
| 24 | 46 | (3, 3, 3, 3, 24, 10) | (0, 2, 0, 0, 6, 0) | 1257.0 | 84.22 |
| 25 | 48 | (2, 0, 3, 0, 22, 21) | (8, 0, 6, 4, 4, 0) | 1396.9 | 76.65 |
| 26 | 49 | (9, 8, 6, 10, 8, 8) | (2, 4, 0, 2, 0, 1) | 1095.5 | 33.16 |
| 27 | 54 | (12, 12, 6, 6, 12, 6) | (0, 0, 4, 1, 0, 0) | 1067.7 | 72.27 |
| 28 | 54 | (4, 2, 4, 0, 32, 12) | (0, 0, 4, 0, 6, 0) | 1575.8 | 100.52 |
| 29 | 58 | (10, 10, 10, 10, 9, 9) | (0, 4, 5, 0, 2, 0) | 1192.6 | 74.51 |
| 30 | 58 | (4, 3, 4, 3, 22, 22) | (6, 5, 4, 0, 5, 0) | 1596.8 | 95.65 |
| 31 | 59 | (6, 6, 14, 14, 7, 12) | (5, 5, 5, 0, 0, 5) | 1201.4 | 91.76 |
| 32 | 63 | (11, 10, 11, 11, 10, 10) | (4, 5, 2, 0, 3, 0) | 1269.2 | 106.18 |
| 33 | 68 | (16, 16, 7, 8, 14, 7) | (0, 2, 0, 2, 4, 0) | 1360.3 | 102.14 |
| 34 | 68 | (4, 6, 1, 3, 37, 17) | (4, 4, 0, 0, 0, 0) | 1901.2 | 115.58 |
| 35 | 70 | (4, 2, 1, 7, 16, 40) | (7, 7, 7, 3, 7, 1) | 2023.5 | 99.95 |
| 36 | 75 | (13, 12, 12, 14, 12, 12) | (4, 2, 1, 3, 3, 0) | 1470.4 | 113.87 |
| 37 | 75 | (16, 17, 9, 9, 15, 9) | (2, 5, 5, 0, 1, 0) | 1525.5 | 87.36 |
| 38 | 75 | (8, 9, 16, 17, 9, 16) | (5, 5, 5, 5, 5, 0) | 1463.9 | 105.59 |
| 39 | 75 | (4, 2, 6, 3, 30, 30) | (7, 7, 7, 5, 5, 0) | 2136.9 | 93.52 |
| 40 | 75 | (4, 2, 6, 3, 40, 20) | (7, 7, 7, 2, 7, 5) | 2161.9 | 104.60 |

**Table 7.** Results of solving super-scale instances by GA.

| Instance | No. of All Tasks | No. of Each Type of Task | No. of Empty Containers at Each Depot | GA | |
|---|---|---|---|---|---|
| | | | | Cost | Time (s) |
| 41 | 100 | (10, 10, 10, 10, 30(0), 30(0)) | (0, 7, 6, 8, 4, 8) | 2474.6 | 375.90 |
| 42 | 120 | (5, 5, 5, 5, 50(10), 50(10)) | (8, 8, 8, 8, 9, 9) | 3487.0 | 441.60 |
| 43 | 140 | (10, 10, 10, 10, 50(0), 50(0)) | (0, 10, 10, 10, 10, 10) | 3540.3 | 898.88 |
| 44 | 190 | (10, 10, 10, 10, 75(15), 75(15)) | (15, 15, 15, 10, 10, 10) | 5630.0 | 1506.54 |
| 45 | 220 | (20, 20, 20, 20, 70(0), 70(0)) | (20, 15, 10, 10, 10, 10) | 5548.5 | 1207.03 |
| 46 | 260 | (10, 10, 10, 10, 110(20), 110(20)) | (20, 20, 20, 20, 10, 10) | 8140.3 | 1593.20 |
| 47 | 280 | (20, 20, 20, 20, 100(0), 100(0)) | (20, 20, 20, 20, 10, 10) | 7724.0 | 3598.63 |
| 48 | 360 | (15, 15, 15, 15, 150(30), 150(30)) | (30, 30, 20, 20, 20, 20) | 11,847.1 | 3642.90 |
| 49 | 420 | (25, 25, 25, 25, 160(20), 160(20)) | (30, 20, 30, 30, 30, 30) | 13,756.0 | 3655.67 |
| 50 | 480 | (30, 30, 30, 30, 180(40), 180(40)) | (40, 40, 30, 30, 30, 30) | 16,235.6 | 3623.27 |

As detailed in Tables 6 and 7, when the total number of container tasks after task vertices splitting was below 135, the proposed GA managed to find solutions with excellent convergence within 120 s. Even when handling instances with a total container task count ranging from 135 to 440 after task vertices splitting, the GA still achieved solutions with excellent convergence within 1600 s. Moreover, in instances featuring a total container task amount after task vertices splitting exceeding 440, the GA exhibited the capacity to yield solutions with relatively acceptable convergence within 3660 s. To elucidate the convergence of the GA, Figure 7 inspects the reduction of the lowest objective values for instances 43 and 47.



(**a**) Instance 43 (**b**) Instance 47

**Figure 7.** Convergence and number of iterations for the GA algorithm.

In order to test the stability of the proposed GA, Table 8 presents statistical information over the 10 repeats for five super-scale instances, among which, the last column exposes the relative difference between the highest and lowest objective values. The smallest relative difference was 0.09% and the average relative difference across all instances was 0.21%. These results demonstrate that the improved GA is very stable.

**Table 8.** Statistical information of objective values over the seven repeats.

| Instance | Minimum | Maximum | Average | Difference (%) |
|---|---|---|---|---|
| 41 | 2468.65 | 2478.72 | 2474.32 | 0.41 |
| 42 | 3483.07 | 3489.51 | 3486.19 | 0.18 |
| 43 | 3537.54 | 3544.82 | 3542.25 | 0.21 |
| 46 | 8136.37 | 8147.56 | 8141.22 | 0.14 |
| 47 | 7719.79 | 7726.67 | 7722.82 | 0.09 |

"Difference" is the relative difference between the maximum value and the minimum value.

In conclusion, the numerical results for large-scale and super-scale problems demonstrate that the proposed GA is an efficient and stable approach and can thus be employed for the new practical CDP.

### *5.3. Operational Methods Comparison and Analysis*

Two trucking operation modes and two empty container relocation strategies were simultaneously considered and modeled in this study. However, the experiment instances always selected the optimal combination of the two modes and two strategies based on Gurobi or GA, which did not illustrate the advantage of this combination. Thus, this section presents a comprehensive comparison and analysis of four distinct combinations, denoted as follows:

(1) The combination of Discharge and Depot-turn (D/D);
(2) The combination of Discharge, Street-turn, and Depot-turn (D/SD);
(3) The combination of Strip, Discharge, and Depot-turn (SD/D);
(4) The combination of Strip, Discharge, Street-turn, and Depot-turn (SD/SD).

These combinations were examined independently to determine the benefits of incorporating both modes alongside both strategies. For our empirical investigation, 12 instances of distinct scales were selected from the above 50 instances. The results are illustrated in Table 9 and Figure 8.

**Table 9.** The results of four operational methods.

| Instance | Cost | | | | Time (s) | | | | Cost Saving from D/D (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **D/D** | **D/SD** | **SD/D** | **SD/SD** | **D/D** | **D/SD** | **SD/D** | **SD/SD** | **D/SD** | **SD/D** | **SD/SD** |
| 16 | 985 | 934 | 862 | 767 | 18 | 26 | 37 | 57 | 5.18 | 12.49 | 22.13 |
| 39 | 2853 | 2822 | 2423 | 2137 | 33 | 46 | 72 | 94 | 1.09 | 15.07 | 25.10 |
| 41 | 3459 | 3320 | 2537 | 2475 | 83 | 186 | 448 | 376 | 4.02 | 26.66 | 28.45 |
| 42 | 4627 | 4441 | 3609 | 3487 | 310 | 257 | 498 | 442 | 4.02 | 22.00 | 24.64 |
| 43 | 5195 | 4965 | 3849 | 3540 | 171 | 447 | 631 | 899 | 4.43 | 25.91 | 31.86 |
| 44 | 7050 | 6860 | 5805 | 5630 | 197 | 447 | 892 | 1507 | 2.70 | 17.66 | 20.14 |
| 45 | 7898 | 7551 | 5943 | 5548 | 362 | 610 | 1020 | 1207 | 4.39 | 24.75 | 29.75 |
| 46 | 10,269 | 9752 | 8342 | 8140 | 761 | 1533 | 1377 | 1593 | 5.03 | 18.77 | 20.73 |
| 47 | 10,539 | 10,068 | 7859 | 7724 | 1088 | 1508 | 3625 | 3599 | 4.47 | 25.43 | 26.71 |
| 48 | 14,094 | 13,521 | 11,920 | 11,847 | 2035 | 3292 | 3638 | 3643 | 4.07 | 15.43 | 15.94 |
| 49 | 15,983 | 15,403 | 13,869 | 13,756 | 3009 | 3158 | 3847 | 3656 | 3.63 | 13.23 | 13.93 |
| 50 | 18,130 | 17,540 | 16,413 | 16,236 | 3198 | 3616 | 3745 | 3623 | 3.25 | 9.47 | 10.45 |
| Average | 8424 | 8098 | 6953 | 6774 | 939 | 1261 | 1653 | 1725 | 3.86 | 18.90 | 22.49 |

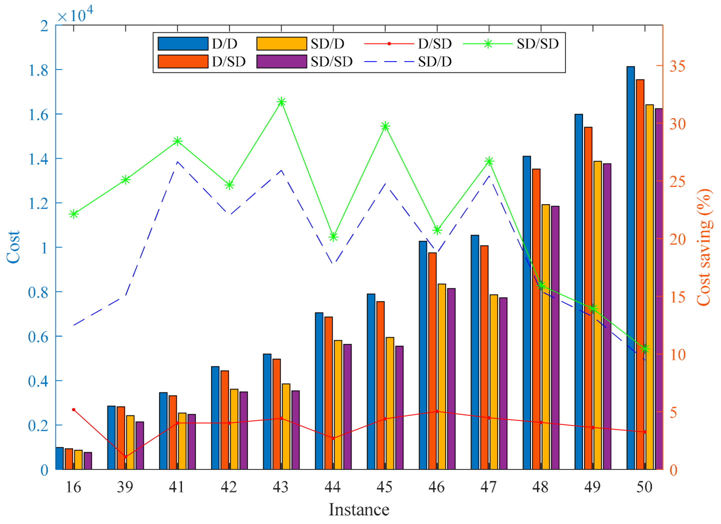


**Figure 8.** The total costs of four operational methods and cost savings from D/D.

In terms of total cost across the four operational methods, SD/SD was the best, followed by SD/D, D/SD, and D/D. Compared to D/D, the greatest cost saving range was observed for SD/SD, ranging from 10.45–31.86%. The cost reductions were more pronounced when the instance size was moderate rather than excessively large or small. It was evident that the inclusion of both Strip and Discharge modes offered a notable advantage. This advantage stemmed from the capacity it afforded to avoid unnecessary waiting during the packing and unpacking processes, thereby effectively reducing truck waiting costs. Consequently, this integration yielded a reduced total cost compared to the operational method that exclusively utilized the Discharge mode. Moreover, when both Strip and Discharge modes were integrated into an operational framework, along with the simultaneous use of both Street-turn and Depot-turn strategies, a higher level of scheduling flexibility emerged. This flexibility enabled a truck to decide if it should wait at or leave a customer location, and where to obtain an empty container if the next customer requested one, based on the cost-effective option. This enhanced flexibility contributed to the diminishment of unnecessary truck waiting time and traveling time. Consequently, this comprehensive method resulted in the lowest total cost.

Despite achieving the lowest cost, SD/SD required the highest average computing time of 1725 s. On the contrary, D/D had the lowest average computing time of 939 s. It is noteworthy that the operational method that exclusively encompassed Discharge mode with either one or both empty container relocation strategies (D/D, D/SD) exhibited a decreased total task count, leading to a smaller problem size and shorter computation time. On the other hand, for the simplest combination D/D, the problem benefited from the clear definition of departure and destination for empty containers, obviating the need for complex empty container relocation. Consequently, D/D yielded the smallest problem size and the shortest computational runtime. When considering the operational method that incorporated both Discharge and Strip modes with either one or both empty container relocation strategies, it was observed that the computational time of SD/D gradually exceeded that of SD/SD with the expansion of the instance scale. The reason for this is that incorporating the Street-turn strategy increased both the complexity of problem optimization and the scheduling flexibility. However, in smaller-scale instances, the increased complexity dominated, leading to longer computing times. Conversely, in larger-scale instances, the enhanced flexibility prevailed, improving the likelihood of identifying feasible solutions in complex scenarios and reducing computation times.

### *5.4. Sensitivity Analysis and Managerial Insights*

In this section, the sensitivities of several key parameters in the model are examined and practical managerial insights for the CDP are presented.

#### 5.4.1. The Effect of Parameters $C_1$ and $C_3$

Since the traveling and waiting costs played a crucial role in truck scheduling and route planning, the impact of $C_1$ and $C_3$ was examined on the cost savings of the operational method SD/SD in comparison to D/SD. As elucidated in Table 10, four instances were solved while keeping $C_1 = 1$ and incrementally increasing $C_3$ from 1 to 400 separately.

Figure 9 clearly illustrates the increase in cost savings as the ratio $C_3/C_1$ increased. However, when $C_3/C_1$ was less than 4, the values of cost savings were negative, which was contrary to our expectations. Through a careful comparison of solutions under two operation methods, it was found that although SD/SD could increase the solution space, it could also lead the GA to become stuck in a local optimum when the cost per unit waiting time approached or equaled the cost per unit of traveling time. When $C_3/C_1$ equaled or exceeded 4, SD/SD gradually exhibited its advantages in cost savings. When $C_3/C_1$ was 400, the average value of cost savings reached 57.08%. This intuitive phenomenon was a result of augmented flexibility endowed by the combination of Strip and Discharge modes. The truck could decide whether to wait for packing and unpacking operations depending on the specific circumstances, thus improving scheduling flexibility to reduce unnecessary truck waiting time and traveling time. The aforementioned observations permit the conclusion that if the total cost per unit of waiting time was significantly higher than the total cost per unit of transportation time, the combination of the Strip and Discharge modes emerged as the more advantageous alternative.

**Table 10.** Effect of parameters $C_1$ and $C_3$.

| $C_3/C_1$ | Cost Saving (%) | | | |
|---|---|---|---|---|
| | **Instance 21** | **Instance 39** | **Instance 41** | **Instance 43** |
| 1 | −18.73 | −13.38 | −12.55 | −16.92 |
| 2 | −6.49 | −3.78 | −0.50 | −7.70 |
| 4 | 4.54 | 5.72 | 13.83 | 11.20 |
| 6 | 16.66 | 19.57 | 22.26 | 18.57 |
| 8 | 24.68 | 21.89 | 26.18 | 23.99 |
| 10 | 28.18 | 29.31 | 34.89 | 32.42 |
| 20 | 39.81 | 40.45 | 44.86 | 43.19 |
| 50 | 48.64 | 49.56 | 54.10 | 52.21 |
| 100 | 51.70 | 54.04 | 56.49 | 55.86 |
| 200 | 53.81 | 56.13 | 57.49 | 58.44 |
| 400 | 54.96 | 57.21 | 57.53 | 58.61 |

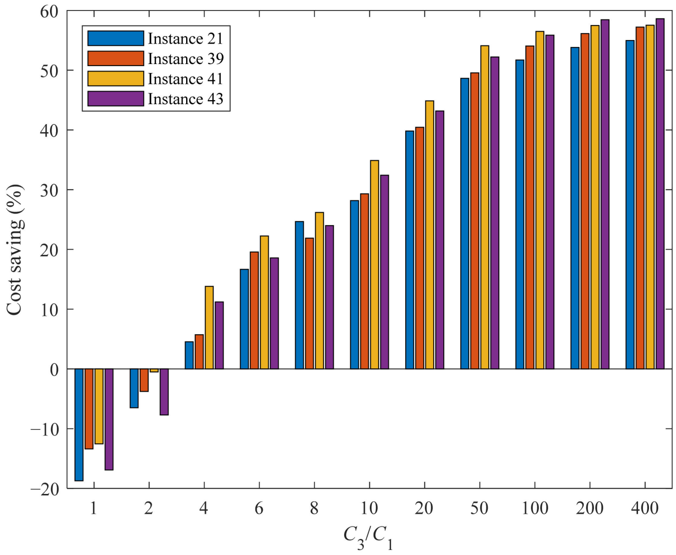


**Figure 9.** Effect of $C_3/C_1$ on cost savings of SD/SD in comparison to D/SD.

### 5.4.2. The Effect of Parameter $m_i$

For the scenario of multiple depots, the effect of the availability of empty containers was reflected not only in the total quantity but also in the quantity distributed in each depot. Thus, two types of variations were investigated. The index from 1 to 5 indicated an increase in the quantity of empty containers, while the indices A and B represented two types of distributions with the same quantity of empty containers. The distribution with lower costs was categorized as Type A, whereas that with higher costs was classified as Type B.

The results of four instances are shown in Table 11 and Figure 10. As expected, an increase in the total quantity of empty containers resulted in an obvious reduction in total costs. For example, with the change from category 1-A (1-B) to category 5-A (5-B), the cost of instance 21 decreased by 48.77 (10.33). It was also found that the distribution of empty containers was of great significance to the total cost. Without exception, distribution type A outperformed type B with the same number of empty containers in any instance. The essential reason for this phenomenon was as follows: providing additional empty containers at the depots allowed the truck to pick up these empty containers from a depot closer to the customers who needed them. This arrangement helped reduce the total cost by avoiding trucks waiting with some customers for full containers to be emptied. In conclusion, this finding implies that supplying more empty containers and a suitable distribution of empty containers to each depot can remarkably enhance the efficacy of truck scheduling when there is limited availability of empty containers.

**Table 11.** Effect of parameter $m_i$.

| Category | Instance 21 | | Instance 39 | | Instance 41 | | Instance 43 | |
|---|---|---|---|---|---|---|---|---|
| | $m_i$ | Cost | $m_i$ | Cost | $m_i$ | Cost | $m_i$ | Cost |
| 1-A | (2, 2, 2, 2, 2, 2) | 789.69 | (3, 3, 5, 5, 7, 7) | 2159.71 | (8, 8, 5, 5, 3, 3) | 2392.20 | (2, 9, 7, 7, 5, 5) | 3664.96 |
| 1-B | (1, 1, 2, 2, 3, 3) | 793.06 | (5, 5, 5, 5, 5, 5) | 2153.64 | (6, 6, 5, 5, 5, 5) | 2488.00 | (0, 7, 7, 7, 7, 7) | 3636.37 |
| 2-A | (3, 3, 2, 2, 2, 2) | 773.48 | (6, 6, 6, 5, 5, 5) | 2102.90 | (8, 8, 6, 6, 4, 4) | 2365.50 | (0, 8, 8, 8, 8, 8) | 3593.61 |
| 2-B | (2, 2, 2, 2, 3, 3) | 791.96 | (4, 4, 6, 5, 7, 7) | 2143.25 | (6, 6, 6, 6, 6, 6) | 2414.27 | (2, 10, 8, 8, 6, 6) | 3608.06 |
| 3-A | (3, 3, 3, 3, 2, 2) | 756.42 | (6, 6, 6, 6, 6, 6) | 2086.15 | (9, 9, 7, 7, 4, 4) | 2349.23 | (0, 9, 9, 9, 9, 9) | 3586.52 |
| 3-B | (2, 2, 3, 3, 3, 3) | 789.18 | (4, 4, 6, 6, 8, 8) | 2125.96 | (7, 7, 7, 7, 6, 6) | 2382.84 | (2, 11, 9, 9, 7, 7) | 3592.90 |
| 4-A | (3, 3, 3, 3, 3, 3) | 749.53 | (7, 7, 7, 6, 6, 6) | 2069.33 | (10, 10, 7, 7, 5, 5) | 2341.97 | (0, 10, 10, 10, 10, 10) | 3555.77 |
| 4-B | (2, 2, 3, 3, 4, 4) | 784.87 | (5, 5, 7, 6, 8, 8) | 2123.79 | (8, 8, 7, 7, 7, 7) | 2370.54 | (2, 12, 10, 10, 8, 8) | 3588.73 |
| 5-A | (4, 4, 3, 3, 3, 3) | 740.93 | (7, 7, 7, 7, 7, 7) | 2026.07 | (10, 10, 8, 8, 6, 6) | 2335.92 | (0, 11, 11, 11, 11, 11) | 3497.62 |
| 5-B | (3, 3, 3, 3, 4, 4) | 782.73 | (5, 5, 7, 7, 9, 9) | 2115.94 | (8, 8, 8, 8, 8, 8) | 2360.98 | (2, 13, 11, 11, 9, 9) | 3579.13 |

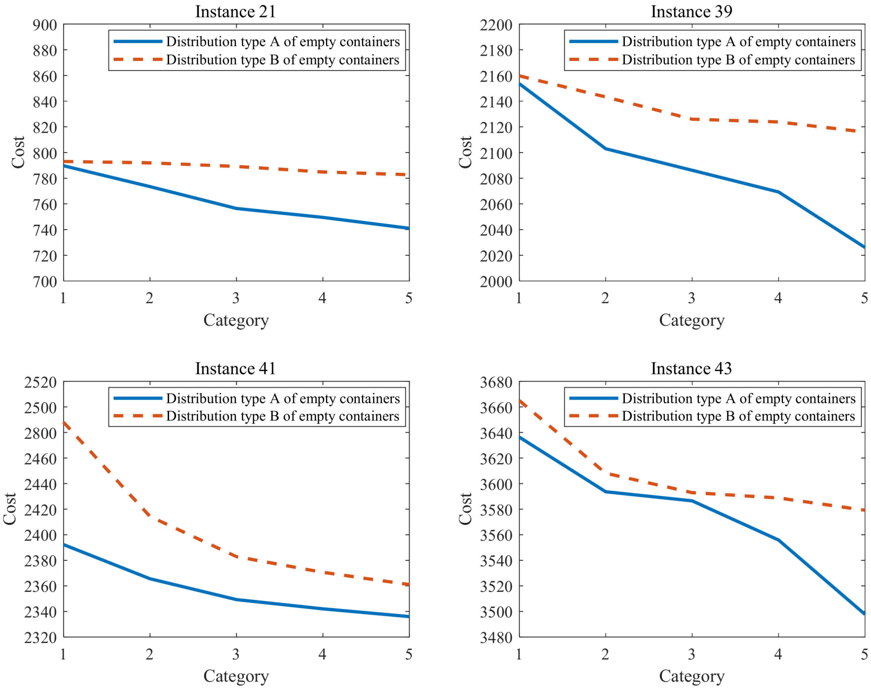


**Figure 10.** Cost under different values of $m_i$.

From the perspective of management, decision-makers are encouraged to rationally plan the inventory of empty containers and distribute empty containers among various depots to minimize the total cost of CDP.

### 5.4.3. The Effect of the Number of Depots

Further research was conducted to examine how the number of depots affected the total cost. Five instances were used for this experiment. The process commenced with the random generation of eight depots and subsequently eliminated one depot at a time to perform separate experiments while keeping the total number of empty containers and trucks at all depots unchanged. Table 12 shows the apparent tendency towards a decrease in the total cost as the number of depots increased. The magnitude of the decline was accentuated when there were fewer depots available, and this phenomenon became more evident with the increase in customers. One possible interpretation of this result is as follows. With a limited number of depots, increasing the number of depots allowed for choosing closer locations for trucks to depart, return, and handle empty containers, thereby facilitating shorter transport distances. However, once the number of depots reached a certain threshold, depot saturation occurred. This suggests that the increase in the number of depots was disproportionate to the reduction of total costs. Moreover, the greater the number of customers, the more depots were needed, leading to a more noticeable cost reduction trend as the number of depots increased.

**Table 12.** Effect of the number of depots.

| **Number of Depots** | **Cost** | | | |
|---|---|---|---|---|
| | **Instance 21** | **Instance 39** | **Instance 41** | **Instance 43** |
| 1 | 926.97 | 2880.25 | 3292.26 | 4959.27 |
| 2 | 869.41 | 2571.80 | 3066.30 | 4298.82 |
| 3 | 812.54 | 2347.85 | 2830.62 | 4137.08 |
| 4 | 783.85 | 2207.23 | 2595.31 | 3855.19 |
| 5 | 762.95 | 2182.35 | 2535.63 | 3746.03 |
| 6 | 758.92 | 2147.53 | 2473.79 | 3589.27 |
| 7 | 756.17 | 2115.44 | 2445.88 | 3503.87 |
| 8 | 753.88 | 2109.50 | 2417.44 | 3482.68 |

It should be noted that while the construction cost of depots remains unaddressed in this study, it is recommended that decision-makers take the number of depots into consideration when devising an efficient container drayage system.

## 6. Conclusions

This study focused on a complex CDP under the constraints of the limited availability of empty containers across multiple inland depots, taking into account various trucking operation modes and empty container relocation strategies. A novel MILP model, which expands upon previous research by incorporating more complex and realistic factors, was formulated to minimize carbon emissions costs, fuel costs, and truck waiting costs. To improve the computational efficiency, an improved GA was proposed. Exhaustive experiments provided the following conclusions and managerial insights for sustainable container drayage practices.

(1) The improved GA is capable of rapidly providing optimal or near-optimal solutions for small to medium-sized instances, while also addressing large-scale instances within a reasonable timeframe.

(2) The combination of both trucking operation modes and both empty container relocation strategies increases operational efficiency and reduces costs compared to the other three operational methods. The greatest cost savings are observed in the range of 10.45% to 31.86%.

(3) The cost savings obtained from combining two trucking operation modes increase when the related cost per unit of waiting time increases. If the total cost per unit of waiting time is significantly higher than the total cost per unit of transportation time, combining two trucking operation modes emerges as the more advantageous alternative.

(4) The inventory of empty containers and their distribution among various depots have an obvious impact on the total cost of the CDP under the same number of requests, and should thus be carefully determined by decision-makers. Furthermore, the number of depots is also of great significance in reducing operational costs and should be considered when devising an efficient container drayage system.

This study has several limitations and suggests multiple directions for future research. One direction is to consider the use of dynamic orders. In practice, some orders are received dynamically during the planning horizon, meaning that they are not fixed and known in advance. Another direction is to consider foldable containers. In this case, a truck can handle one full container or several empty folded containers at a time. These factors increase the complexity of the CDP. Consequently, the proposed model and algorithm must be extended and tailored to the specific characteristics of the novel CDP, representing a key area of future research.

**Author Contributions:** Conceptualization, X.Y.; methodology, Y.F.; software, Y.F.; validation, X.Y. and Y.F.; formal analysis, Y.F. and C.H.; investigation, X.Y.; resources, X.Y.; data curation, Y.F. and C.H.; writing—original draft preparation, Y.F.; writing—review and editing, X.Y.; visualization, Y.F. and C.L.; supervision, X.Y.; project administration, X.Y.; funding acquisition, X.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Natural Science Foundation of China (no. 42001122).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are available upon request from the corresponding author.

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

## References

1. Macharis, C.; Bontekoning, Y.M. Opportunities for OR in intermodal freight transport research: A review. *Eur. J. Oper. Res.* **2004**, *153*, 400–416. [CrossRef]
2. Cui, H.; Chen, S.; Chen, R.; Meng, Q. A two-stage hybrid heuristic solution for the container drayage problem with trailer reposition. *Eur. J. Oper. Res.* **2022**, *299*, 468–482. [CrossRef]
3. Li, S.; Wu, W.; Ma, X.; Zhong, M.; Safdar, M. Modelling medium- and long-term purchasing plans for environment-orientated container trucks: A case study of Yangtze River port. *Transp. Saf. Environ.* **2023**, *5*, tdac043. [CrossRef]
4. Ran, C.; Zhang, Y. The driving force of carbon emissions reduction in China: Does green finance work. *J. Clean. Prod.* **2023**, *421*, 138502. [CrossRef]
5. Erhun, F.; Kraft, T.; Wijnsma, S. Sustainable Triple—A Supply Chains. *Prod. Oper. Manag.* **2021**, *30*, 644–655. [CrossRef]
6. Yang, X.; Daham, H.A.; Salhi, A. Combined strip and discharge delivery of containers in heterogeneous fleets with time windows. *Comput. Oper. Res.* **2021**, *127*, 105141. [CrossRef]
7. Lee, S.; Moon, I. Robust empty container repositioning considering foldable containers. *Eur. J. Oper. Res.* **2020**, *280*, 909–925. [CrossRef]
8. Tan, S.Y.; Yeh, W.C. The Vehicle Routing Problem: State-of-the-Art Classification and Review. *Appl. Sci.* **2021**, *11*, 10295. [CrossRef]
9. Song, Y.; Zhang, Y.; Wang, W.; Xue, M. A Branch and Price Algorithm for the Drop-and-Pickup Container Drayage Problem with Empty Container Constraints. *Sustainability* **2023**, *15*, 5638. [CrossRef]
10. Yan, X.; Xu, M.; Xie, C. Local container drayage problem with improved truck platooning operations. *Transp. Res. Part E Logist. Transp. Rev.* **2023**, *169*, 102992. [CrossRef]
11. Wang, X.; Regan, A.C. Local truckload pickup and delivery with hard time window constraints. *Transp. Res. Part B Methodol.* **2002**, *36*, 97–112. [CrossRef]
12. Gronalt, M.; Hartl, R.F.; Reimann, M. New savings based algorithms for time constrained pickup and delivery of full truckloads. *Eur. J. Oper. Res.* **2003**, *151*, 520–535. [CrossRef]
13. Jula, H.; Dessouky, M.; Ioannou, P.; Chassiakos, A. Container movement by trucks in metropolitan networks: Modeling and optimization. *Transp. Res. Part E Logist. Transp. Rev.* **2005**, *41*, 235–259. [CrossRef]
14. Imai, A.; Nishimura, E.; Current, J. A Lagrangian relaxation-based heuristic for the vehicle routing with full container load. *Eur. J. Oper. Res.* **2007**, *176*, 87–105. [CrossRef]
15. Chung, K.H.; Ko, C.S.; Shin, J.Y.; Hwang, H.; Kim, K.H. Development of mathematical models for the container road transportation in Korean trucking industries. *Comput. Ind. Eng.* **2007**, *53*, 252–262. [CrossRef]
16. Zhang, R.; Yun, W.Y.; Moon, I. A reactive tabu search algorithm for the multi-depot container truck transportation problem. *Transp. Res. Part E Logist. Transp. Rev.* **2009**, *45*, 904–914. [CrossRef]
17. Sterzik, S.; Kopfer, H. A Tabu Search Heuristic for the Inland Container Transportation Problem. *Comput. Oper. Res.* **2013**, *40*, 953–962. [CrossRef]
18. Vidović, M.; Popović, D.; Ratković, B.; Radivojević, G. Generalized mixed integer and VNS heuristic approach to solving the multisize containers drayage problem. *Int. Trans. Oper. Res.* **2017**, *24*, 583–614. [CrossRef]
19. Lai, M.; Crainic, T.G.; Di Francesco, M.; Zuddas, P. An heuristic search for the routing of heterogeneous trucks with single and double container loads. *Transp. Res. Part E Logist. Transp. Rev.* **2013**, *56*, 108–118. [CrossRef]

20. Ghezelsoflu, A.; Di Francesco, M.; Frangioni, A.; Zuddas, P. A set-covering formulation for a drayage problem with single and double container loads. *J. Ind. Eng. Int.* **2018**, *14*, 665–676. [CrossRef]
21. Zhang, R.; Yun, W.Y.; Kopfer, H. Heuristic-based truck scheduling for inland container transportation. *OR Spectr.* **2010**, *32*, 787–808. [CrossRef]
22. Zhang, R.; Yun, W.Y.; Kopfer, H. Multi-size container transportation by truck: Modeling and optimization. *Flex. Serv. Manuf. J.* **2015**, *27*, 403–430. [CrossRef]
23. Daham, H.A.; Yang, X.; Warnes, M.K. An efficient mixed integer programming model for pairing containers in inland transportation based on the assignment of orders. *J. Oper. Res. Soc.* **2017**, *68*, 678–694. [CrossRef]
24. Zhang, R.; Yun, W.Y.; Moon, I.K. Modeling and optimization of a container drayage problem with resource constraints. *Int. J. Prod. Econ.* **2011**, *133*, 351–359. [CrossRef]
25. Zhang, R.; Huang, C.; Wang, J. A novel mathematical model and a large neighborhood search algorithm for container drayage operations with multi-resource constraints. *Comput. Ind. Eng.* **2020**, *139*, 106143. [CrossRef]
26. Fazi, S.; Choudhary, S.K.; Dong, J. The multi-trip container drayage problem with synchronization for efficient empty containers re-usage. *Eur. J. Oper. Res.* **2023**, *310*, 343–359. [CrossRef]
27. Braekers, K.; Caris, A.; Janssens, G.K. Integrated planning of loaded and empty container movements. *OR Spectr.* **2013**, *35*, 457–478. [CrossRef]
28. Song, Y.; Zhang, J.; Liang, Z.; Ye, C. An exact algorithm for the container drayage problem under a separation mode. *Transp. Res. Part E Logist. Transp. Rev.* **2017**, *106*, 231–254. [CrossRef]
29. Zhang, R.; Lu, J.; Wang, D. Container drayage problem with flexible orders and its near real-time solution strategies. *Transp. Res. Part E Logist. Transp. Rev.* **2014**, *61*, 235–251. [CrossRef]
30. Moghaddam, M.; Pearce, R.H.; Mokhtar, H.; Prato, C.G. A generalised model for container drayage operations with heterogeneous fleet, multi-container sizes and two modes of operation. *Transp. Res. Part E Logist. Transp. Rev.* **2020**, *139*, 101973. [CrossRef]
31. Choi, H.R.; Park, B.; Kang, S.H.; Lee, J.W.; Park, C. Dispatching of container trucks using genetic algorithm. In Proceedings of the 4th International Conference on Interaction Sciences, Memphis, TN, USA, 9–12 October 2011; pp. 146–151.
32. Funke, J.; Kopfer, H. A model for a multi-size inland container transportation problem. *Transp. Res. Part E Logist. Transp. Rev.* **2016**, *89*, 70–85. [CrossRef]
33. Zhang, R.; Zhao, H.; Moon, I. Range-based truck-state transition modeling method for foldable container drayage services. *Transp. Res. Part E Logist. Transp. Rev.* **2018**, *118*, 225–239. [CrossRef]
34. Bomboi, F. New Routing Problems with Possibly Correlated Travel Times. Ph.D. Thesis, Universitá degli Studi di Cagliari, Cagliari, Italy, 2020.
35. He, W.; Jin, Z.; Huang, Y.; Xu, S. The Inland Container Transportation Problem with Separation Mode Considering Carbon Dioxide Emissions. *Sustainability* **2021**, *13*, 1573. [CrossRef]
36. Huang, C.; Zhang, R. Modeling and Optimization of a Drayage Problem with Foldable and Standard Containers. In Proceedings of the 2020 Chinese Control And Decision Conference (CCDC), Hefei, China, 22–24 August 2020; pp. 1505–1600.
37. Huang, C.; Zhang, R. Container Drayage Transportation Scheduling with Foldable and Standard Containers. *IEEE Trans. Eng. Manag.* **2023**, *70*, 3497–3511. [CrossRef]
38. Nguyen, V.S.; Pham, Q.D. Solving a Real-World Problem of Truck-Trailer Scheduling in Container Transportation by Local Search. *JST Smart Syst. Devices* **2022**, *32*, 64–73.
39. Nossack, J.; Pesch, E. A truck scheduling problem arising in intermodal container transportation. *Eur. J. Oper. Res.* **2013**, *230*, 666–680. [CrossRef]
40. Reinhardt, L.B.; Pisinger, D.; Spoorendonk, S.; Sigurd, M.M. Optimization of the drayage problem using exact methods. *INFOR Inf. Syst. Oper. Res.* **2016**, *54*, 33–51. [CrossRef]
41. Braekers, K.; Caris, A.; Janssens, G.K. Bi-objective optimization of drayage operations in the service area of intermodal terminals. *Transp. Res. Part E Logist. Transp. Rev.* **2014**, *65*, 50–69. [CrossRef]
42. Shiri, S. Development of Models and Solution Methods for Different Drayage Applications. Ph.D. Thesis, University of South Carolina, Columbia, SC, USA, 2018.
43. Jia, S.; Cui, H.P.; Chen, R.; Meng, Q. Dynamic container drayage with uncertain request arrival times and service time windows. *Transp. Res. Part B Methodol.* **2022**, *166*, 237–258. [CrossRef]
44. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1992; pp. 1–211.
45. Reeves, C.R. Feature Article—Genetic Algorithms for the Operations Researcher. *Inf. J. Comput.* **1997**, *9*, 231–250. [CrossRef]
46. Jaszkiewicz, A. Genetic local search for multi-objective combinatorial optimization. *Eur. J. Oper. Res.* **2002**, *137*, 50–71. [CrossRef]
47. Zhang, H.; Ishikawa, M. A solution to combinatorial optimization with time-varying parameters by a hybrid genetic algorithm. *Int. Congr. Ser.* **2004**, *1269*, 149–152. [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.