Hindawi
Journal of Advanced Transportation
Volume 2022, Article ID 8253175, 15 pages
https://doi.org/10.1155/2022/8253175

Research Article

Metaheuristics for a Large-Scale Vehicle Routing Problem of Same-Day Delivery in E-Commerce Logistics System

Yi Tao,[1] Changhui Lin,[1] and Lijun Wei[2]

[1]*School of Management, Guangdong University of Technology, China*
[2]*Key Laboratory of Computer Integrated Manufacturing System, School of Electromechanical Engineering, Guangdong University of Technology, 510006, China*

Correspondence should be addressed to Lijun Wei; villagerwei@gdut.edu.cn

Received 3 June 2022; Accepted 7 September 2022; Published 29 September 2022

Academic Editor: Yi-Sheng Lv

In this paper, we introduce a new variant of large-scale vehicle routing problem that arises in the goods distribution of city e-commerce logistics, the multi-depot vehicle routing problem with order split and allocation (MD-CVRP-OSA), which incorporates the issue of split and allocation of online orders into traditional VRP. A mathematical formulation is constructed for the MD-CVRP-OSA, and an efficient metaheuristics algorithm based on a variable neighborhood search (VNS) solution framework is designed to solve it. The proposed method is tested on a large family of instances, including real-world data collected from JD.com, and the effectiveness of the VNS algorithm and also the algorithm components are analyzed.

1. Introduction

Global retail e-commerce market has been witnessing a rapid expansion on account of the shifting habits of internet users towards new shopping channels. According to Statista Digital Market Outlook, a market research firm, retail e-commerce sales worldwide in 2021 amounted to 3.78 trillion US dollars, a 16.7% increase over the previous year, and e-retail revenues are projected to grow to 5.73 trillion US dollars in 2025 (https://www.statista.com/outlook/dmo/ecommerce/worldwide). E-retail made up 17.6% of total retail sales worldwide in 2021, up from 15.9% a year prior. This growth in share is largely driven by Asia-Pacific, where e-commerce contributes 14.6% of overall retail spending. Statistics from China E-Commerce Research Center show that the transactions of retail e-commerce market in China reached $2.1 trillion in 2021, an increase of 17.2% year on year, accounting for 25.3% of the total retail sales (https://www.globaldata.com/chinese-e-commerce-market-reach-us3-3-trillion-2025-says-globaldata).

In order to succeed in the highly competitive market, e-retail platforms have been striving to optimize order fulfilment operations and ensure a speedy, flexible, secure, and low-cost delivery service for customers [1]. To meet the increasing customer expectations towards shorter delivery time, many leading e-retailers started in recent years to offer same-day delivery (SDD) service in metropolitan areas that consolidates, dispatches, and delivers orders to customers on the same day the orders are placed [2]. For example, JD.com started its speedy SDD service since 2010 that delivers the orders received prior to 11 AM on the same day and the ones received before 11 PM the next day (before 3 PM) (https://ir.jd.com/news-releases/news-release-details/jdcom-launched-its-speedy-211-program-which-provides-same-day). Amazon's SDD service has now covered major cities in over 20 countries as of May 2022. Moreover, the recently launched Prime Now Program allows customers to receive their orders on selected items within two hours for free and in one hour for $7.99 (https://primenow.amazon.com/).

Management of the SDD system is costly and logistically complicated for e-retailers as it aggravates the inherent issues in urban delivery operations, such as tight delivery deadlines, small-sized orders, and low rate of order consolidations [3–5]. The logistics system of SDD service generally consists of a two-echelon network where the first level are distribution centers/warehouses for massive product storage and the second level are delivery stations or pickup stations for end customer distribution. For a guarantee of fast delivery and high service level, it is common for an SDD service provider to operate multiple warehouses closely located to metropolitan area. For example, JD.com builds four warehouses in urban district of Guangzhou, the third largest city of China. Amazon recently opened a series of so-called mini-fulfillment centers closer to four big U.S. cities as an action to guarantee packages arrive by several set times daily (https://www.reuters.com/article/us-amazon-com-delivery/amazon-adds-warehouse-network-closer-to-cities-to-speed-up-same-day-delivery-idUSKBN20Q0T3). Each warehouse is equipped with a fleet of vehicles. During the course of operating time, vehicles are loaded with parcels and dispatched from warehouses to visit customers' locations [4]. In the operation of SDD, vehicles are usually not dispatched simultaneously in the beginning and may be reused multiple times during a service day [2, 6].

The strategy of maintaining multiple warehouses poses challenges on the operational efficiency of SDD logistics system. Due to the limit of storage space, a single warehouse located in urban area usually cannot cover all stock keeping units (SKUs), but only keeps a certain range of them instead. For example, the four warehouses of JD.com in Guangzhou mainly store 3C products, shoes and clothing, and health care products; snacks, grain and oil, and books; home appliances and 3C products; and clothing and groceries, respectively. Given this, there is high chance that no single warehouse contains all requested SKUs for a customer order. This causes a serious issue that SKUs of an order may need to be split up and separately satisfied by more than one warehouses. Order split issue frequently arises during the order fulfillment process of SDD service under multi-warehouse mode. Even if a warehouse stores all SKUs in an order, inventory of certain SKU of that order may go below the requested amount. Another case is that as a way of promoting sales, tie-in sale strategy is often applied by the marketing department that offers discounts or deals to customers if they buy a specific combination of different SKUs. However, when the sales promotion is planned and executed, the marketing department may not be aware of the availability of the combined SKUs in the same warehouse. Under such circumstances, order split is inevitable. Figure 1 illustrates an order split example where SKUs of order O_1 are split and allocated to warehouses W_1 and W_3 because none of the three warehouses can provide all requested items of O_1 at the same time.

In this work, we address a new goods distribution problem of SDD logistic system with order split issue under multi-warehouse mode. Each warehouse keeps a limited range of SKUs and may not be able to satisfy all order requests. Each customer order contains a number of different SKUs and requests certain amount for each SKU. Due to aforementioned reasons, orders often need to be split up and are allocated to more than one warehouses. In the actual SDD logistics operation, items of the same SKU in an order are usually packaged together as a whole for the convenience of handling, which implies that a single order can only be split on SKU level instead of being split up arbitrarily. That

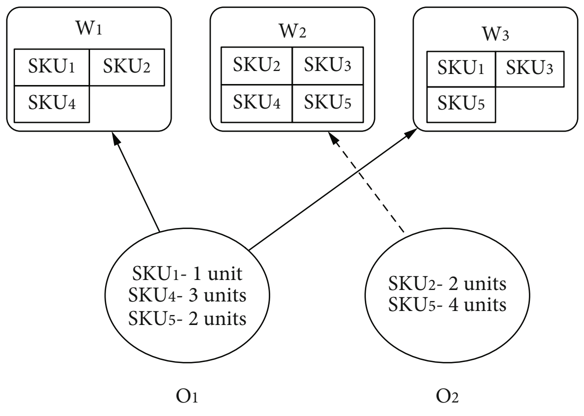


Figure 1: An illustrative example of orders split among different warehouses.

is, items that are of different SKUs in an order is allowed to be separated, but items of the same SKU cannot be further parted. The logistics activities involve two core processes: (1) upon receiving and accumulating online orders, splitting each of them and allocating to different warehouses if necessary and (2) designing a route for each vehicle that is assigned delivery task. The goal is to minimize the total routing cost of all vehicles. It is assumed that all the needed information of the SDD system is known to us and the VRP problem considered in our work is static.

The problem studied in this paper extends the traditional vehicle routing problem (VRP) [7] by incorporating order split and allocation issue under multi-warehouse mode, which makes it essentially a multi-depot capacitated vehicle routing problem with order split and allocation (MD-CVRP-OSA). The review papers (Braekers et al. [8] and Tan & Yeh [9]) survey the most recent works on variants of vehicle routing problem and proposed exact and heuristic methods for the VRP in the literature. Related problem to the MD-CVRP-OSA are the location-routing problem (LRP) and the multi-depot VRP (MDVRP). Reviews about variants and heuristic and exact algorithms for these problems can be found in Drexl and Schneider [10], Prodhon and Prins [11], Montoya-Torres et al. [12], and Chen and Yang [13]. Besides the traditional solution approaches, there is also a recent trend to apply machine learning methods, such as deep (reinforcement) learning as a new type of heuristic to solve VRP. Interested readers may refer to Li et al. [14], Ma et al. [15], and Li et al. [16].

A closely related work to the MD-CVRP-OSA is the multi-depot split delivery vehicle routing problem (MDSDVRP) introduced by Gulczynski et al. [17]. Gulczynski et al. [17] consider that the customers are not necessarily assigned to depots and allow serving customers from multiple depots. Thus, how the split of products and which depots to be assigned need to be decided. The authors developed an integer programming based heuristic for the MDSDVRP. Ray et al. [18] investigated a same problem as Gulczynski et al. [17]. They proposed an integer linear programming model and used an heuristic search method to solve the problem. Izar and Suwilo [19] studied the MDSDVRP with time windows (MDVRPSDTW), whereas

Wang et al. [20] proposed the min-max split delivery multi-depot vehicle routing problem with minimum service time requirement (min-max SDMDVRP-MSTR), which allows the split of service time for each customer. However, we point out that this work differs from the aforementioned ones because SKUs cannot be further split and not all SKUs are available at all depots. Therefore, our problem cannot be considered the SDMDVRP.

To the best of our knowledge, our work is the first to introduce a new and practical logistics problem that combines the joint optimization of vehicle routing and order split and allocation, which is now a big challenge in the logistics system of e-retailers. To solve the problem, we formulate it as an integer programming model and propose an efficient variable neighborhood search (VNS) [21] algorithm. Extensive numerical experiments are conducted on instances derived from classical capacitated VRP instances and on real-world instances.

The rest of the paper is organized as follows. In Section 2, we describe the MD-CVRP-OSA and introduce a mathematical formulation. In Section 3, we apply a VNS technique to solve the MD-CVRP-OSA. Computational results are presented in Section 4, and Section 5 concludes the paper and indicates future research directions.

2. Problem Description and Mathematical Formulation

The MD-CVRP-OSA addressed in this paper can be described as follows. We are given a complete directed graph $G = (V, A)$, where the node set V is further partitioned as $V = V_w \cup V_s$. Set $V_w = \{1, 2, 3, \cdots, K\}$ represents K warehouses, and $V_s = \{K+1, K+2, K+3, \cdots, K+I\}$ represents I delivery stations. The arc set E is defined as $A = \{(k, j), (j, k): k \in V_w, j \in V_s\} \cup \{(i, j), (j, i): i, j \in V_s, i \neq j\}$. We assume that each warehouse $k \in V_w$ has one vehicle to deliver the goods, and all the K vehicles are homogeneous with the same vehicle capacity Q. Each arc $(i, j) \in A$ is associated with a non-negative cost c_{ij} which represents the routing cost between node i and j. Each delivery station $i \in V_s$ serves a set M_i of customers. Figure 2 illustrates an example of the MD-CVRP-OSA.

In the MD-CVRP-OSA, each customer's order is sent to the associated delivery station first and then delivered to its location. The distribution of goods from the delivery station to the individual customers can be regarded as a classical VRP, so the corresponding planning problem is not explicitly considered in the MD-CVRP-OSA.

Let N be the set of all SKUs provided by the e-retailer and let q_n be the weight of n-th SKU. Further, we denote the amount of n-th SKU requested by the m-th customer of station i as s_{imn}. We assume that a customer's order is allowed to be split and allocated to multiple warehouses, each of which may only store a limited number of SKUs, but the amount of a single SKU cannot be further split up. This means that if a warehouse accepts a request on a specific SKU, it must fully satisfy it. Define p_n^k as a binary parameter that takes 1 if the n-th SKU is stored in warehouse k, and 0 otherwise.

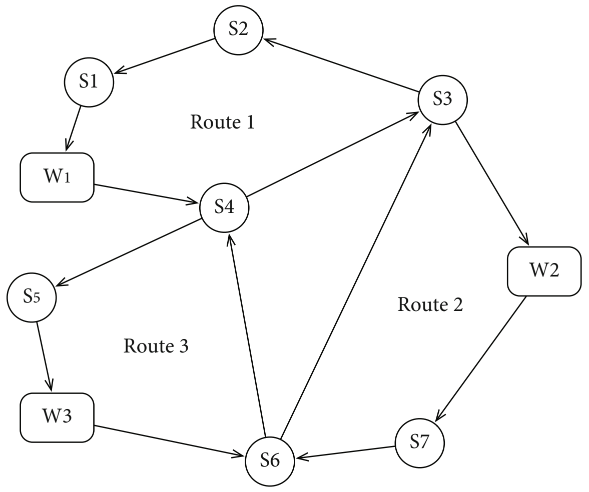


Figure 2: An example of MD-CVRP-OSA vehicle routes.

A vehicle route is a simple cycle $R = \{W_k, S_i, S_j, \cdots, W_k\}$ in graph G passing through a warehouse $W_k \in V_w$ and sequentially visiting stations $\{S_i, S_j, \cdots\} \subseteq V_s$. The cost of a route is equal to the sum of the routing costs of the arcs forming the route. The MD-CVRP-OSA consists of determining a set of at most K vehicle routes of minimum cost such that:

(i) Each warehouse operates at most one vehicle route

(ii) A customer's demand for an SKU must be fully serviced by the same warehouse but customer's requests for different SKUs can be satisfied by different warehouses

(iii) The total demand of each vehicle route does not exceed the vehicle capacity

2.1. Mathematical Formulation. In this section, we describe a mathematical formulation for the MD-CVRP-OSA. The mathematical model defines the following decisions variables:

(i) x_{ij}^k: binary variable equals to 1 if arc $(i,)$ is traversed by warehouse k's vehicle, and 0 otherwise

(ii) z_{imn}^k: binary variable equals to 1 if the n-th SKU requested by the m-th customer of station i is allocated to warehouse k, and 0 otherwise

The MD-CVRP-OSA can be formulated as the following integer programming problem:

$$\min \sum_{k \in V_w} \sum_{(i,j) \in A} c_{ij} x_{ij}^k, \quad \text{s.t.} \tag{1}$$

$$\sum_{k \in V_w} p_n^k z_{imn}^k = 1, \forall i \in V_s, m \in M_i, n \in N, \tag{2}$$

$$\sum_{m\in M_i}\sum_{n\in N} z_{imn}^k \le M \sum_{j\in V, j\neq i} x_{ji}^k, \forall k \in V_w, i \in V_s, \tag{3}$$

$$\sum_{j\in V, j\neq i} x_{ji}^k \le \sum_{m\in M_i}\sum_{n\in N} z_{imn}^k, \forall k \in V_w, i \in V_s, \tag{4}$$

$$\sum_{i\in V, i\neq j} x_{ij}^k = \sum_{i\in V, i\neq j} x_{ji}^k, \forall k \in V_w, j \in V, \tag{5}$$

$$\sum_{i\in V, i\neq j} x_{ij}^k \le 1, \forall k \in V_w, j \in V_{s,} \tag{6}$$

$$\sum_{m\in M_i}\sum_{n\in N} z_{imn}^k \le M \sum_{j\in V_s} x_{kj}^k, \forall k \in V_w, i \in V_s, \tag{7}$$

$$x_{ij}^k = 0, \forall k \in V_w, i \in V_w, j \in V_w, \tag{8}$$

$$\sum_{n\in N} q_n \sum_{i\in V_s}\sum_{m\in M_i} s_{imn} z_{imn}^k \le Q, \forall k \in V_w, \tag{9}$$

$$\sum_{i\in S}\sum_{j\in (V\setminus S\setminus V_w)\cup\{k\}} x_{ij}^k \ge z_{imn}^k, \forall k \in V_w, i \in V_s, m \in M_i, n \in N, \quad S \subseteq V \setminus V_w, i \in S, S\varnothing, \tag{10}$$

$$x_{ij}^k, z_{imn}^k \in \{0,1\}, k \in V_w, i \in V, j \in V, m \in M_i, n \in N. \tag{11}$$

Objective (1) is to minimize the total routing cost of all vehicles. Constraint (2) indicates that the demand of each SKU of a customer order must be satisfied by one and only one warehouse. Constraints (3) and (4) state that the demand of the m-th customer of delivery station i for the n-th SKU can be fulfilled by warehouse k only when the vehicle of warehouse k visits station i, where M is a large constant. Constraint (5) guarantees that any vehicle that enters a node must also depart from that node. Constraint (6) ensures that each vehicle visits any given delivery station at most once. Constraint (7) states that if the vehicle of a warehouse is assigned delivery task, it should start its tour from the warehouse. Constraint (8) ensures that no vehicle is allowed to travel between two warehouses. Constraint (9) limits the capacity of any vehicle. Constraint (10) imposes the connectivity of the route performed by vehicle k by eliminating subtours involving stations only. Constraint (11) defines the domains of decision variables.

The MD-CVRP-OSA is $\mathcal{NP}$-hard. Indeed, its special case where $K = 1$, $|N| = 1$, and $|M_i| = 1$ for all $i \in V_s$ is the classical CVRP, a well-known $\mathcal{NP}$-hard problem.

3. A VNS Algorithm for the MD-CVRP-OSA

To solve the MD-CVRP-OSA efficiently, a VNS algorithm is proposed in this section. The VNS algorithm introduced in Mladenovi´c and Hansen [21] and Hansen et al. [22] makes use of a series of random and improving local searches based on systematically changed neighborhoods and a shaking process that escapes from local optimum by choosing a new starting point for running an improving search. The reason that VNS is chosen to solve the MD-CVRP-OSA is because VNS and its variants are simple and effective. They have been widely applied in solving compelx combinatorial optimization problems, such VRP [23–25], and other related problems [26]. The idea of VNS is generally easy to implement with a limited number of parameters. Moreover, several groups of neighborhood structures are designed with the aim of facilitating the local searches, and the VNS approach has been proven to be very effective in combining different neighborhood structures.

The VNS algorithm proposed for the MD-CVRP-OSA is given in Algorithm 1. The algorithm starts with an initial solution and employs a set of local search operators OPT. Two nested loops are used to improve the solution, where the inner loop uses the local search operators to generate random neighbouring solutions and the outer loop generates different starting points for the inner loop by a diversification mechanism. In the algorithm, variable S^* is used to store the best-found solution, and variable *nonImp* records the number of outer loops during which S^* has not been improved.

At each iteration of the inner loop (lines 7-11), given the current solution S, a random solution S' of S is generated with an operator OPT_k (k is set to equal to 1 initially). If the new solution S' is better than the S, the algorithm accepts S' and replaces S with S'. Then, the loop continues after having set k equal to 1, i.e., a new random solution is generated from S according to operator OPT_1 in the next iteration. Otherwise, S remains unchanged, and a new random solution is generated according to neighborhood N S_{k+1} in the next iteration. This loop repeats until all local search operators are used and no new better solution is found.

In the outer loop, a diversification procedure is introduced to produce a new initial solution based on the best-found solution (line 16). The inner loop resumes the search with the new initial solution. This process is repeated until the best-found solution is not improved during *maxNonI mp* iterations.

The initial solution generation, local search operator, and diversification procedure are described in details as follows.

3.1. Generating an Initial Solution. In this work, a three-stage method is employed to generate initial solutions for the MD-CVRP-OSA. In the first stage, for each delivery station $i \in V_s$, we rank the routing cost between the location of station i and each warehouse $k \in V_w$, i.e., c_{ik}, in an increasing order. Hence, each delivery station i is associated with a sorted list of warehouses. In the second stage, we allocate warehouses for each delivery station. The warehouses in the sorted list of station i are sequentially selected to satisfy the SKUs requested by the customers associated with station i, the demands of which are fulfilled customer by customer, and SKU by SKU for each customer. If there are still customers with unsatisfied demand because the current warehouse does not contain the requested SKUs or the vehicle of the warehouse is fully loaded, the next warehouse in the list will be chosen. Such process continues until all demands

```
Construct an initial solution $S$
Define a set of local search operators $OPT_k(k = 1,\cdots,O)$
$S^* = S$, $nonImp = 0$
while $nonImp \le maxNonImp$
    $k = 1$
    while $k \le |O|$
        Generate a random neighboring solution $S'$ of $S$ using $OPT_k$
        if $S'$ is better than $S$
            $S = S', k = 1$
        else
            $k = k + 1$
    if $S$ is better than $S^*$
        $S^* = S$, $nonImp = 0$
    else
        $S = S^*$, $nonImp = nonImp + 1$
    $S$ = DIVERSIFY($S$)
return $S^*$
```

Algorithm 1: VNS for the MD-CVRP-OSA.

of the customers in station i are fully satisfied. The same process is repeated to assign warehouses to all delivery stations.

As a result of the first two stages, a certain number of warehouses are assigned delivery jobs to serve a given set of stations. In the last stage, we apply the well-known cost saving algorithm [27, 28] to separately determine the sequence of service for each warehouse. Define $\tilde{V}_s^k$ as the set of stations that are to be visited by warehouse k's vehicle. For each station $i \in \tilde{V}_s^k$, a simple route (k, i, k) is constructed in the first step. In the second step, for each pair of routes $(k, \cdots, i, k)$ and $(k, j, \cdots, k)$, we calculate the cost saving $c_{i,k} + c_{k,j} - c_{i,j}$. The pair of routes that have the largest cost saving are merged into a new route $(k, \cdots, i, j, \cdots, k)$ in the third step. The second and third steps are repeated until all stations are included in one single route, and such cost saving algorithm is applied for each warehouse with delivery jobs.

3.2. Description of the Local Search Operators. The effectiveness of a VNS algorithm strongly relies on the structure of the local search operators. In this paper, we designed seven operators to generate random neighboring solutions from the current solution. In the following, we refer to a route R^k that is associated with warehouse k, $R^k = \{r_1^k, r_2^k, \cdots, r_R^k\}$, where $r_1^k = r_R^k \in V_w$ represents a warehouse and $r_i^k(1 \le i \le R-1) \in V_s$ represents the stations visited by the vehicle associated with the route. Besides, for each station $r_i^k(1 \le i \le R-1)$, there is a corresponding SKU number set $\{sku_{in}^k\}$, where $sku_{in}^k(1 \le n \le N)$ is the number of the n-th SKU provided by warehouse r_1^k for station r_i^k. According to our assumption, $sku_{in}^k = \sum_{m \in M_i} x_{mn} s_{imn}$ (x_{mn} is an 0-1 variable that takes 1 if the n-th SKU of the m-th order in station r_i^k is provided by r_1^k and 0 otherwise), which is the combination of the number of n-th SKU requested by the customers in station r_i^k. We define the set of SKUs whose value sku_{in}^k is larger than one as the SKUs provided by r_1^k for station r_i^k.

The seven operators are defined as follows:

(1) Intra-swap: The operator swaps the positions of two selected stations within the same route

(2) Intra-move: The operator shifts a subtour from its position to a different position within the same route

(3) Inter-swap: The operator swaps the positions of two stations from different routes, as depicted in Figure 3

(4) Inter-shift: The operator shifts a station from its position to a different route (see Figure 4)

(5) Order-split: The operator moves the SKUs provided by one route for a station to a different route visiting the same station. Note that this operator is only used to split the SKUs and the travel cost of these route remains unchanged (if a station is removed from one route, all the SKUs are also moved). In Figure 5, the SKU_3 provided by route a is moved to route b

(6) Station-remove: The operator removes a station from one route and shift its provided SKUs for this station to another route that visits the same station. In Figure 6, the station r_4^b is original served by routes a and b (a provides SKU_3 and SKU_5, while b provides SKU_1). After the application of the station-remove operation, the station r_4^b is only served by route b. The difference between station-remove and order-split is that all the SKUs provided by one route for this station are moved to another route

(7) Super-move: The operator tries to move the stations visited by a route to another route. Firstly, it selects two routes a and b. For each station r_i^b, it tries to move r_i^b from b to a step by step. If r_i^b is already in a, it just increases the SKUs number of station r_i^b in

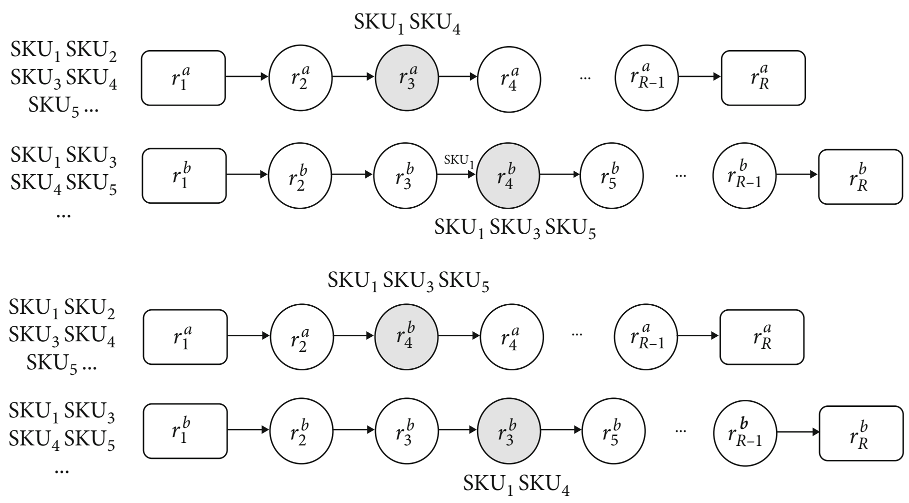
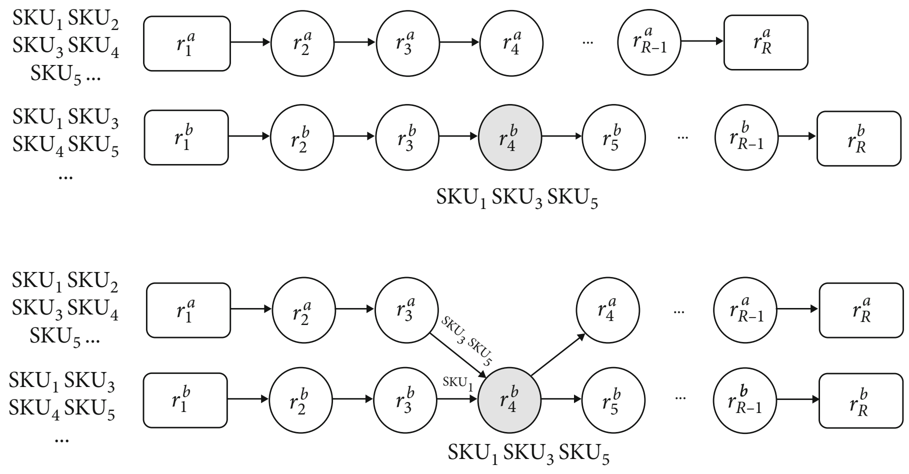
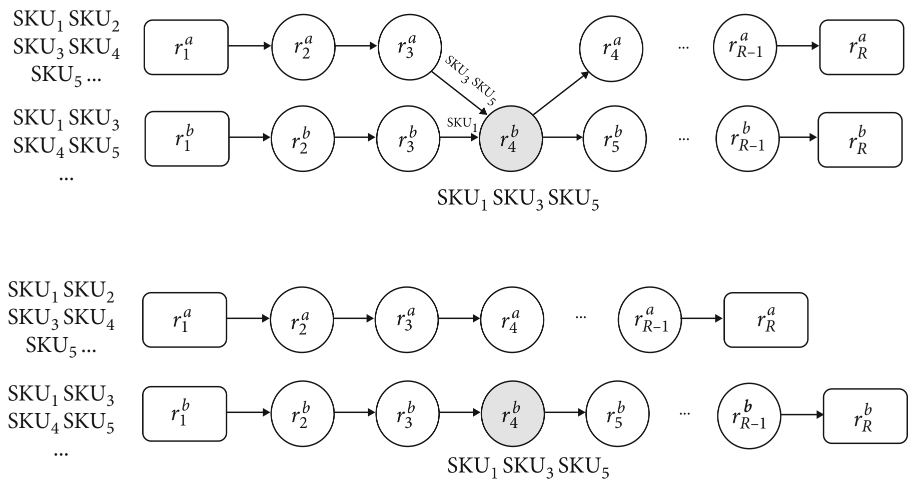


Figure 3: Inter-swap operator.



SKU$_1$ SKU$_2$ SKU$_3$ SKU$_4$ SKU$_5$ ...
SKU$_1$ SKU$_3$ SKU$_4$ SKU$_5$ ...
SKU$_1$ SKU$_3$ SKU$_5$
SKU$_3$ SKU$_5$
SKU$_1$
SKU$_1$ SKU$_3$ SKU$_5$

Figure 4: Inter-shift operator.



SKU$_1$ SKU$_2$ SKU$_3$ SKU$_4$ SKU$_5$ ...
SKU$_3$ SKU$_5$
SKU$_1$
SKU$_1$ SKU$_3$ SKU$_4$ SKU$_5$ ...
SKU$_1$ SKU$_3$ SKU$_5$
SKU$_1$ SKU$_3$ SKU$_5$

Figure 5: Order-split operator.

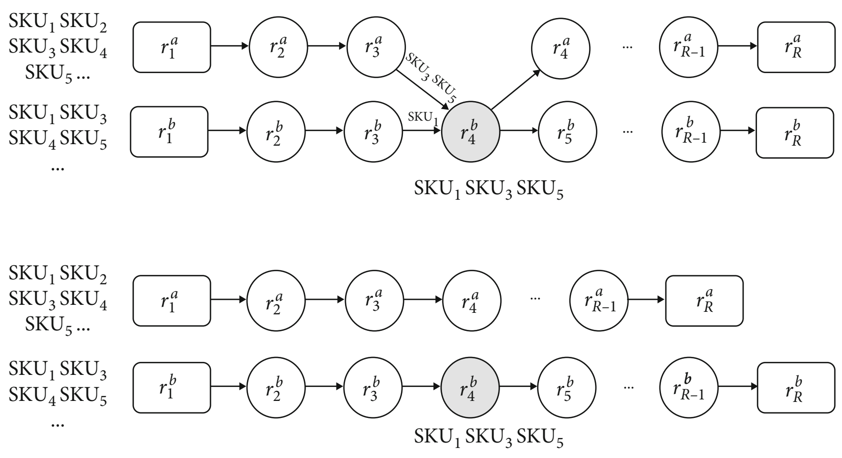


Figure 6: Station-remove operator.

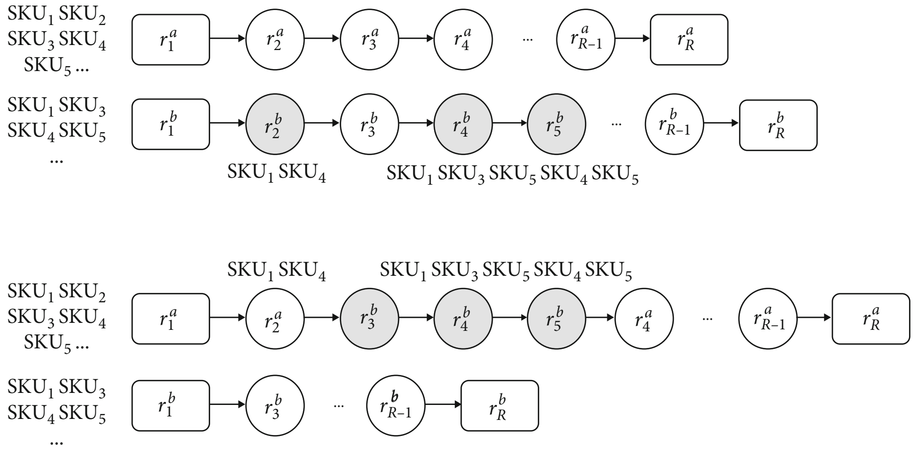


Figure 7: Super-move operator.

a. Otherwise, it inserts r_i^b into a randomly selected position in a. In Figure 7, the stations r_2^b, r_4^b and r_5^b are originally served by route b. After the super-move operation, these stations are moved from the route b to that of warehouse a

Note that in the first four operators, only the position of the station is changed, while the corresponding SKUs number set remains unchanged. The resulting route is feasible only if the total demand is not larger than the vehicle capacity Q and all the SKUs provided by this route for each station must be available in the warehouse.

3.3. Diversification Mechanism. Diversification helps to prevent the search from trapping in local optima, and the mechanism adopted in our algorithm works as follows. Firstly, it selects two routes a and b. Then, a subsequence $r_s{}^b$ is selected from route b, and an insert position k is selected from route a. For each station $r_i^b \in r_{s'}^b$, it tries to move r_i^b from route b to route a step by step. If r_i^b is already in a, it just increases the SKU number of station r_i^b in a. Otherwise, it inserts r_i^b behind the last inserted items (the first item is inserted into position k). In Figure 8, the stations r_4^b and r_5^b are moved from route b to that of warehouse a.

4. Numerical Experiments

This section reports on the computational results of the VNS algorithm described in Section 3. The algorithm was implemented in C++ and tested on a computer equipped with an Intel(R) Core(TM) CPU i7-7920HQ @3.10Ghz with 32GB RAM. The general purpose mixed-integer programming solver CPLEX was used to solve the integer model described in Section 2.1, where the subtour elimination constraints (10) have been handled in a cutting-plane fashion using the CPLEX callable library. A time limit of 7200 seconds was imposed on the CPLEX solver. The number of iterations iterCount of the VNS algorithm was set to 300000, and a time limit of 7200 seconds was also imposed to its execution. Besides, to show the performance of our VNS algorithm, we also compare it with another well-known metaheuristic, tabu search. Similarly, the number of iterations of tabu search is 300000, and a time limit of 7200 second is imposed. Each of the local search operators in the proposed VNS algorithm is randomly selected with the same

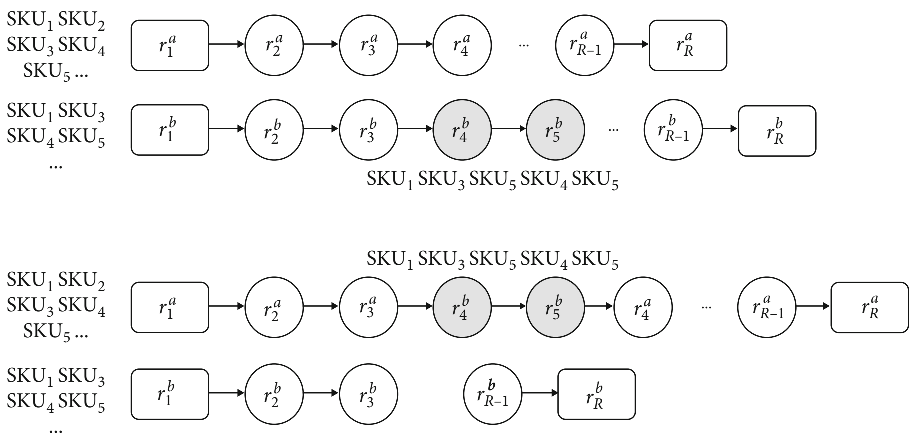

SKU$_1$ SKU$_2$ SKU$_3$ SKU$_4$ SKU$_5$ ...
r_1^a r_2^a r_3^a r_4^a ... r_{R-1}^a r_R^a
SKU$_1$ SKU$_3$ SKU$_4$ SKU$_5$ ...
r_1^b r_2^b r_3^b r_4^b r_5^b ... r_{R-1}^b r_R^b
SKU$_1$ SKU$_3$ SKU$_5$ SKU$_4$ SKU$_5$
SKU$_1$ SKU$_3$ SKU$_5$ SKU$_4$ SKU$_5$
SKU$_1$ SKU$_2$ SKU$_3$ SKU$_4$ SKU$_5$ ...
r_1^a r_2^a r_3^a r_4^b r_5^b r_4^a ... r_{R-1}^a r_R^a
SKU$_1$ SKU$_3$ SKU$_4$ SKU$_5$ ...
r_1^b r_2^b r_3^b r_{R-1}^b r_R^b

Figure 8: The diversification mechanism.

possibility during the execution. The length of tabu list is set to 50, and the number of candidate solutions during the search is 300.

In our experiments, we considered both random and real-life instances.

4.1. Random Instances. We randomly generated eight classes or sets of instances by varying the number of warehouses and of stations. In the first 5 sets, the number of warehouses is set equal to 2, 4, 6, 8, and 10, respectively, and the corresponding number of stations is set equal to 5, 10, 15, 20, and 25, respectively. The number of warehouses in the last three sets is set equal to 12, and the number of stations is set equal to 30, 60 and 120, respectively. For each set, 10 instances were generated, thus providing a total of 80 instances.

In each instance, the number of SKUs is set equal to 20, and the weight of each SKU is set to an integer random number between 1 and 3. The coordinates x and y of the warehouses and the stations are randomly sampled in the range of $[1, 1000]$. The availability of the n-th SKUs at warehouse k (i.e., value p_n^k) is also randomly defined. The number of orders for each station and the number of SKUs for each order are randomly generated in the ranges $[1, 10]$ and $[1, 3]$, respectively. For each order, the SKUs are randomly selected from the 20 types until the requested number is reached. The number of each selected SKU is randomly selected from the range $[1, 3]$. Let the total weight of the SKUs required by all customers is Q_{sum}; then, the vehicle capacity Q is set equal to $q \times Q_{sum}/K$, where q is a constant factor which is set equal to 1.25.

4.1.1. Results on Random Instances. Tables 1 and 2 give the results obtained for small-sized and large-sized instances, respectively. The tables show the percentage of improvement of the solution cost computed by the VNS and tabu search with respect to the solution cost provided by the CPLEX solver ("I_{VNS}%" and "I_{TS}%"), computed as $100 \times (1 - \text{cost}_{VNS}/\text{cost}_{cplex})$ and $100 \times (1 - \text{cost}_{TS}/\text{cost}_{cplex})$, where cost_{VNS},cost_{TS}, and cost_{cplex} are the solution costs found by the VNS algorithm, tabu search algorithm and by CPLEX, respectively. In addition, for each method, the tables show the computing time in seconds ("$t(s)$").

On the small-sized instances, Table 1 shows that both CPLEX and VNS computed the optimal solutions for all the instances of set 1. Tabu search can find eight optimal solutions out of ten. On the remaining sets, the table shows that the VNS algorithm and tabu search algorithm greatly improved the solutions provided by CPLEX. The improvement increases as the size of instances becomes larger. Meanwhile, our proposed VNS algorithm outperformed tabu search algorithm on almost all instances. Further, the computing time spent by the VNS and tabu search is on average limited, being equal to less than 800 seconds.

On the large-sized instances, the VNS algorithm and tabu search algorithm found better solutions than CPLEX for all the instances. None of the instances was solved to optimality by CPLEX within the imposed time limit, and the VNS and tabu search were capable of greatly improving the solutions provided by CPLEX, especially for the instances of sets 7 and 8. Similarly, as the case of small-sized instances, our proposed VNS algorithm outperformed tabu search algorithm on almost all instances. On the larger instances, the VNS and tabu search algorithm reached the time limit imposed to its execution.

In summary, the VNS approach improves the results significantly compared to CPLEX, especially for the large-sized instances.

To investigate the convergence behavior of the VNS algorithm on difficult instances of sets 7 and 8, we ran the VNS algorithm without any time limit, and we recorded the cost of the best solution found after every 10000 iterations. The results obtained for each instances of the two sets are given in Figures 9(a) and 9(b), respectively, where the x axis reports the number of iterations (the unit is 10^4) and the y axis reports the percentage of the improvement of the solution cost with respect to the previous best solution. The results show that VNS converges quite quickly for instances of set 7, whereas a slower converge rate can be observed for instances of set 8.

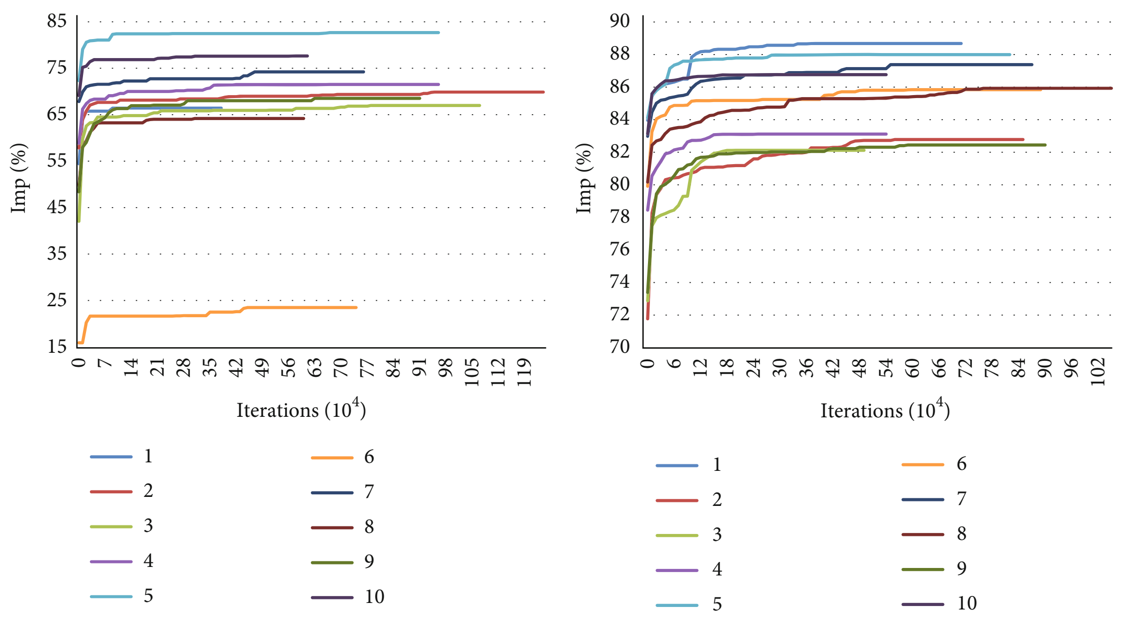
4.2. Real-World Instances. This work was motivated by a real-world problem. In this section, a case study from

Table 1: Results on randomly generated small-sized instances.

Set	Id	CPLEX		TS			VNS		
		$cost_{cplex}$	t (s)	$cost_{TS}$	I_{TS} (%)	t (s)	$cost_{VNS}$	I_{VNS} (%)	t (s)
$1(K=2\,;I=5)$	1	4401.6	0	4401.6	0.0	53	4401.6	0.0	49
	2	1955.5	0	1955.5	0.0	48	1955.5	0.0	43
	3	2674.5	0	2674.5	0.0	46	2674.5	0.0	50
	4	2292.5	0	2292.5	0.0	58	2292.5	0.0	49
	5	3317.7	1	3321.8	-0.1	47	3317.7	0.0	51
	6	2494.5	0	2494.5	0.0	55	2494.5	0.0	48
	7	2061.9	0	2061.9	0.0	51	2061.9	0.0	46
	8	3595.5	23	3595.5	0.0	62	3595.5	0.0	57
	9	2380.5	0	2380.5	0.0	58	2380.5	0.0	47
	10	3313.4	38	3316.2	-0.1	54	3313.4	0.0	45
	Avg	2848.8	6	2849.5	-0.02	47	2848.8	0.0	49
$2(K=4\,;I=10)$	1	6838.5	7202	6833.3	0.1	158	6831.4	0.1	167
	2	7697.2	7201	7428.5	3.5	135	7044.2	8.5	127
	3	6383.4	7200	6383.4	0.0	198	6383.4	0.0	225
	4	6400.6	7202	6387.0	0.2	201	6387.0	0.2	212
	5	5419.1	7201	5302.4	2.2	151	5216.3	3.7	106
	6	7011.8	7201	6725.1	4.1	153	6613.3	5.7	142
	7	5198.2	7201	5198.2	0.0	205	5198.2	0.0	231
	8	6734.6	7201	6684.2	0.7	147	6684.2	0.7	135
	9	5141.0	7200	5141.0	0.0	156	5141.0	0.0	124
	10	5664.9	7201	5664.9	0.0	169	5664.9	0.0	172
	Avg	6248.9	7201	6174.8	1.1	167	6116.4	1.7	164
$3(K=6\,;I=15)$	1	7784.3	7202	7251.8	6.8	293	7050.0	9.4	271
	2	8703.2	7202	8488.5	2.5	295	8461.0	2.8	264
	3	6100.3	7209	5896.6	3.3	315	5865.7	3.8	227
	4	6710.0	7206	6712.1	0.0	284	6705.2	0.1	397
	5	7059.7	7202	6995.0	0.9	497	6995.0	0.9	525
	6	8312.8	7212	7205.3	13.3	258	7193.0	13.5	240
	7	6279.5	7201	5915.3	5.8	218	5859.1	6.7	327
	8	7079.5	7201	6431.8	9.1	263	6488.6	8.3	257
	9	6977.8	7207	6912.4	0.9	264	6905.1	1.0	247
	10	8633.3	7209	8125.6	5.9	428	8091.4	6.3	440
	Avg	7364.0	7206	6993.4	4.9	328	6961.4	5.3	336
$4(K=8\,;I=20)$	1	8192.4	7211	8015.3	2.2	620	7941.0	3.1	568
	2	9335.7	7215	7938.6	15.0	512	7526.4	19.4	490
	3	10744.0	7215	8125.3	24.4	488	7876.5	26.7	453
	4	9506.2	7201	9410.3	1.0	468	9321.0	1.9	403
	5	8530.4	7218	8284.6	2.9	472	8218.8	3.7	510
	6	8276.8	7202	7648.9	7.6	415	7599.7	8.2	476
	7	7201.3	7212	7015.3	2.6	561	6904.4	4.1	575
	8	8201.4	7218	7529.8	8.2	368	7363.2	10.2	397
	9	10527.7	7216	8564.2	18.7	810	8298.8	21.2	774
	10	8235.2	7209	7771.3	5.6	623	7574.2	8.0	559
	Avg	8875.1	7212	8030.4	8.8	554	7862.4	10.6	554

Table 2: Results on randomly generated large-sized instances.

Set	Id	CPLEX		TS			VNS		
		$cost_{cplex}$	t (s)	$cost_{TS}$	I_{TS} (%)	t (s)	$cost_{VNS}$	I_{VNS} (%)	t (s)
$5(K = 10\,; I = 25)$	1	8157.0	7213	7548.6	7.5	957	7399.8	9.3	1086
	2	9635.0	7200	8842.3	8.2	655	8073.9	16.2	630
	3	10847.7	7222	9899.2	8.7	931	9408.7	13.3	949
	4	9072.3	7228	8321.4	8.3	1206	8146.6	10.2	1127
	5	11067.3	7207	8902.5	19.6	630	8305.8	25.0	642
	6	9904.4	7221	9328.6	5.8	602	9167.1	7.4	635
	7	8639.7	7225	7425.3	14.1	684	7478.1	13.4	630
	8	8904.4	7221	8518.6	4.3	1124	8420.1	5.4	1198
	9	8928.3	7229	8215.3	8.0	798	7813.6	12.5	776
	10	9697.9	7201	8962.0	7.6	658	8623.7	11.1	609
	Avg	9485.4	7217	8596.4	9.2	880	8283.7	12.4	884
$6(K = 12\,; I = 30)$	1	16404.4	7215	11053.2	32.6	1423	10135.0	38.2	1342
	2	14227.2	7202	8925.3	37.3	1511	8745.8	38.5	1438
	3	16787.8	7201	9563.2	43.0	1102	9030.1	46.2	1061
	4	16888.4	7219	9847.3	41.7	984	8700.8	48.5	1012
	5	11812.1	7219	10642.5	9.9	2106	10708.5	9.3	2008
	6	14900.4	7214	11254.9	24.5	2658	10661.4	28.4	2836
	7	27326.9	7204	13478.6	50.7	1028	10213.9	62.6	1097
	8	8421.8	7210	8210.6	2.5	1987	8149.2	3.2	1819
	9	15415.0	7200	9102.5	41.0	1089	8965.4	41.8	1165
	10	13331.3	7218	10894.5	18.3	1384	10649.3	20.1	1464
	Avg	15551.5	7214	10297.3	30.1	1527	9595.9	33.7	1524
$7(K = 12\,; I = 60)$	1	33768.6	7223	11673.6	65.4	7167	11324.8	66.5	7200
	2	41853.6	7214	12868.4	69.3	7200	12897.3	69.2	7200
	3	38638.8	7218	13546.0	64.9	7200	12988.8	66.4	7200
	4	39574.1	7212	12023.8	69.6	7200	11266.7	71.5	7200
	5	64875.9	7222	11675.4	82.0	7200	11367.0	82.5	7200
	6	17687.5	7259	13245.2	25.1	7200	13515.5	23.6	7200
	7	48120.1	7230	13845.2	71.2	7200	12389.0	74.3	7200
	8	39105.5	7216	14265.3	63.5	7157	13987.1	64.2	7077
	9	36150.2	7205	11547.8	68.1	7200	11363.4	68.6	7200
	10	56085.3	7225	12875.6	77.0	7200	12530.7	77.7	7200
	Avg	41586.0	7215	12756.6	65.6	7192	12363.0	66.4	7188
$8(K = 12\,; I = 120)$	1	166196.7	7202	20144.3	87.8	7200	19794.0	88.1	7200
	2	100338.8	7202	18125.4	81.9	7200	17859.7	82.2	7200
	3	118469.2	7203	21425.3	81.9	7200	21192.2	82.1	7200
	4	115770.5	7202	20015.6	82.7	7200	19568.6	83.1	7200
	5	166901.8	7201	21546.8	87.1	7200	20921.2	87.5	7200
	6	137609.3	7201	20571.3	85.1	7200	20250.3	85.3	7200
	7	124766.9	7201	17785.6	85.7	7200	17334.3	86.1	7200
	8	137447.7	7211	21445.7	84.4	7200	21093.2	84.7	7200
	9	119388.4	7203	22863.4	80.8	7200	22197.9	81.4	7200
	10	130576.4	7203	17557.6	86.6	7200	17425.9	86.7	7200
	Avg	131746.6	7206	20148.1	84.4	7199	19763.7	84.7	7199



(a) Instances in set 7

(b) Instances in set 8

Figure 9: Convergence of the VNS algorithm on sets of instances 7 and 8.

JD.com, one of China's biggest e-commerce companies, is analyzed and solved with the proposed VNS approach.

4.2.1. A Case Study from JD.com: Problem Description. To serve online customer demands in Guangzhou, the third largest city of China, JD.com maintains four warehouses located within or very close to the urban area. The locations of the fours warehouses are depicted in Figure 10: Asia No. 1 warehouse, South China Logistics Center Foshan Park (FS warehouse), the 2nd South China Logistics Center Machong Park (MC warehouse), and South China Logistics Center Luogang Park (LG warehouse). In the figure, the black circles represent the warehouses, and the empty circles are the delivery stations. The following categories of SKUs are stored in the warehouses: 3C products, shoes and clothing, and health care products; snacks, grain and oil, and books; home appliances and 3C products; and clothing and groceries, respectively.

Parcel deliveries are handled by means of a fleet of vehicles from the warehouses to the delivery stations that are scattered over the urban area. At each station, orders placed by nearby residents are accumulated. To fulfill the demands in these delivery stations, logistics manager should simultaneously optimize the order allocation and routing with the aim of minimizing the total travelling distance. As one of the largest e-retailers of China, JD.com faces massive amount of online orders everyday. Upon receiving orders, the handling system needs to split and allocate it immediately and delivers it within very short time once the vehicle is available.

In this case study, we consider five real cases in Guangzhou city. The following assumptions were made in considering the data provided by JD.com:

(i) The travelling distances between each pair of points of the distribution network were approximated with the Euclidean measure

(ii) Time windows constrains were ignored, and also all returned orders were not considered

(iii) The vehicle fleet is assumed to be homogeneous

The number of the delivery stations with delivery demands ranges between 13 and 33, and the total number of orders from all delivery stations that need to be fulfilled in each case ranges from 157 to 396. The vehicle capacity of the vehicles is set equal to 6,800 KG.

4.2.2. Results Obtained. The results obtained by applying the VNS algorithm are summarized in Table 1. The first five columns of the table give the instance name, the number of warehouses K, the number of delivery stations I, the total number of orders $O_T (O_T = \sum_{i \in V_s} M_i)$, and the vehicle capacity$Q$.

Due to the huge amount of daily delivery orders and the latest service options, such as "deliver within one hour" and "deliver within three hours," launched in many big cities by JD.com, the decision on the order split, allocation, and routing needs to be made within a very limited computing time. For this reason, we record the values of the solutions generated by the VNS algorithm after 15, 30, and 60 minutes, and we compare the obtained values with respect to the cost of the initial solution.

For each instance, Table 3 reports the total distance (in kilometers) corresponding to the cost of the initial solution computed by the VNS ("z_0"), and the solution costs after 15, 30, and 60 minutes ("z_1, z_2, z_3"), respectively. The table also shows the percentage gap of solutions z_1, z_2, and z_3 with respect to solution "z_0".

The results shown in Table 3 can be commented as follows. The initial heuristic finds feasible solutions for all the instances with an average value of 124.96 kilometers.

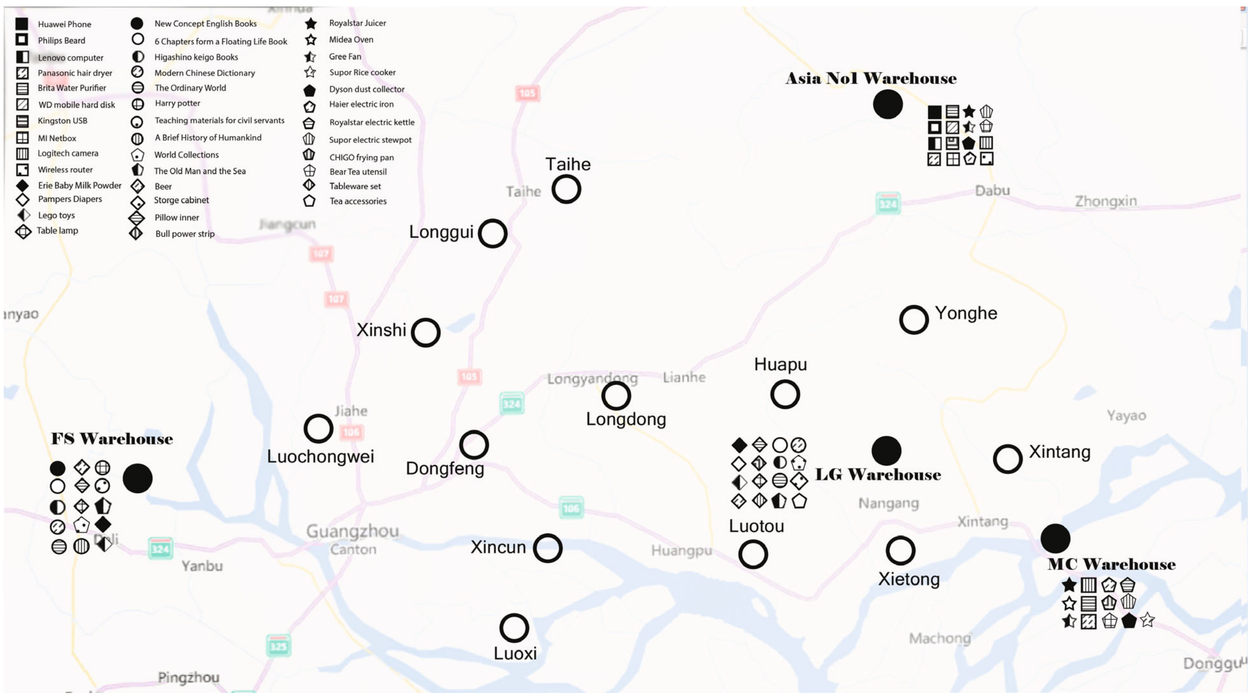


Figure 10: Real-world instance: locations of warehouses and stations points.

Table 3: Results on real-world instances: the JD.com case.

Index	K	I	O_T	Q	z_0	z_1	gap_1	z_2	gap_2	z_3	gap_3
Case1	4	13	157	6,800	65.74	54.37	20.91%	54.37	20.91%	54.37	20.91%
Case2	4	18	218	6,800	114.86	91.97	24.89%	91.09	26.10%	91.09	26.10%
Case3	4	23	276	6,800	126.01	103.76	21.44%	102.39	23.07%	100.63	25.22%
Case4	4	28	336	6,800	144.69	119.21	21.37%	116.79	23.89%	114.18	26.72%
Case5	4	33	396	6,800	173.52	143.55	20.88%	140.21	23.76%	132.04	31.41%
Average					124.96	102.57	21.83%	100.97	23.76%	98.46	26.92%

Starting from the initial solutions, the VNS algorithm improves this value by 21.83%, 23.76%, and 26.92% on average for the 15, 30, and 60 minutes runs, respectively. The results have shown that the proposed VNS algorithm can be an effective tool to deal with the fast decisions needed by the order fulfillment process in JD.com.

As an example of the solutions obtained, Figure 11 gives the solution obtained after 60 minutes by the VNS regarding instance Case1 depicted in Figure 10. In instance *Case*1, four warehouses are used to serve customers in Guangzhou city, and 13 delivery stations are associated with the customer orders that need to be fulfilled at that time. The SKUs stored in each warehouse and the SKUs requested by each station are also shown in Figure 10. Note that for each warehouse, we only show the SKUs that may be needed in this case. Figure 11 displays the four routes designed by the algorithm to deliver parcels to the delivery stations. Among these stations, Xinshi, Longgui, Taihe, Longdong, Huapu, Yonghe, Xincun, Luotou, Xintang, Xietong, and Luoxi need to be visited twice which means that the customer orders associated with these stations are spilt up and allocated to more than one warehouses.

4.3. Impact of the Main VNS Components. We conducted additional experiments to measure the contribution of each operator of the VNS. More precisely, from each of the 5, 6, 7, and 8 instance sets, we selected two instances and ran the VNS algorithm by selectively disabling one of the following three operators: order-split, station-remove and supper-move.

Table 4 gives the results obtained. In the table, column head "VNS" reports the best results obtained by the VNS algorithm, whereas column heads "NoOrderSplit," "NoStationRemove," and "NoSupperMove" give the results obtained by the VNS without the order-split, station-remove, and supper-move operators, respectively. Column $Imp(\%)$ reports the percentage of improvement of the different VNS variants considered; a negative value means that the corresponding solution is worse than the solution computed by the VNS with all operators.

The table clearly shows that all operators are affective and that in only two cases improved solutions have been computed.

4.4. Sensitivity Analysis-Vehicle Capacity and SKUs Availability. In this section, we perform a second set of

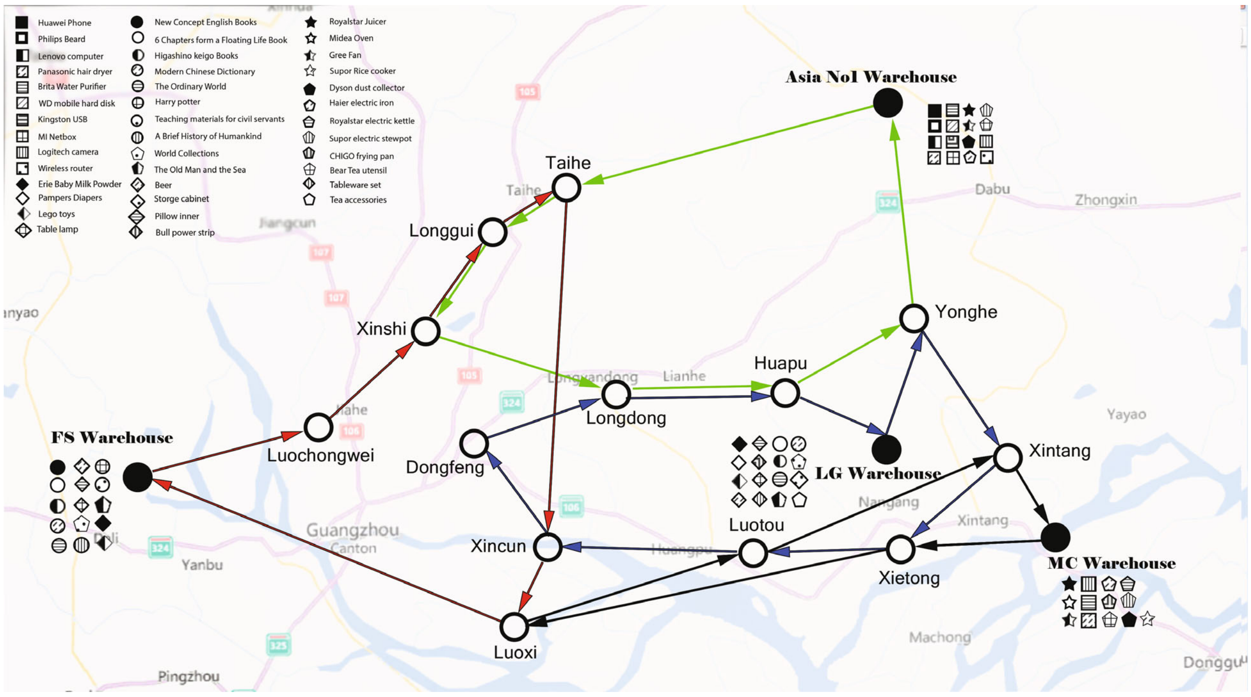


Figure 11: Vehicle routes for real-world instance Case1.

Table 4: Impact of the main VNS operators.

Set	Id	VNS		NoOrderSplit			NoStationRemove			NoSupperMove		
		Cost	t (s)	Cost	$Imp(\%)$	t (s)	Cost	$Imp(\%)$	t (s)	Cost	$Imp(\%)$	t (s)
5	1	9408.7	949	10210.6	− 8.5	831	10775.6	− 14.5	558	9827.7	− 4.5	579
	10	8146.6	1127	8225.5	− 1.0	1172	8316.4	− 2.1	966	8357.0	− 2.6	936
6	1	9030.1	1061	9078.7	− 0.5	1074	9648.4	− 6.8	1655	9255.0	− 2.5	1260
	10	8700.8	1882	9284.8	− 6.7	954	8873.6	− 2.0	1002	8955.6	− 2.9	1887
7	1	12988.8	7200	13357.3	− 2.8	6548	15115.6	− 16.4	2988	**12794.3**	**1.5**	5905
	10	11266.7	7200	11931.7	− 5.9	7200	12905.6	− 14.5	4944	11449.0	− 1.6	7200
8	1	21192.2	7200	21749.6	− 2.6	7200	21532.7	− 1.6	7200	23694.8	− 11.8	7200
	10	19568.6	7200	20638.5	− 5.5	7200	20283.7	− 3.7	7200	**19491.1**	**0.4**	7200

experiments focused on the impact of the vehicle capacity and the SKUs availability. Also for these experiments, we considered two instances from each of the 5, 6, 7, and 8 instance sets.

4.4.1. Vehicle Capacity. We recall that the vehicle capacity Q is set equal to $q \times Q_{sum}/K$, where q is a parameter, and Q_{sum} is the total weight of all the required SKUs. In our experiments, we set the value of q equal to 1.1, 1.15, 1.2, 1.25, 1.3, 1.35, and 1.4, respectively. The results obtained are summarized in Table 5 for the different values of q, where column $Imp(\%)$ gives the percentage of the improvement of the solution cost with respect to the solution cost computed with $q = 1.1$.

The table shows that, as expected, the smaller the value q, the more expensive the delivery cost and hence implies that the right trade off between the distribution cost and the vehicle capacity needs to be done.

4.4.2. SKUs Availability. We recall that when the (0-1) coefficient p_n^k is equal to 1, it means that the n-th SKUs is stored or available at warehouse k. In the results reported in Section 4.1, the probability of stockout (i.e., $p_n^k = 0$) was set equal to $1/K$. In the new experiments reported in this section, we compare the case where all SKUs are available to all warehouses ("NoStockOut" case), i.e., $p_n^k = 1 \forall i \in N$, $k \in V_w$, with the case where the probability of stockout is set equal to $1/K$ (as in Section 4.1 ("StockOut($1/K$)" case)) and the case where the probability is set equal to 0.5 ("StockOut(0.5)" case)).

The results obtained are summarized in Table 6, where column $Imp(\%)$ gives the percentage of improvement computed with respect to the solution cost obtained for case "NoStockOut."

The results clearly show that the probability of stock out has a significant effect on the total distribution cost. When the probability increases to 0.5, the cost increases up to

Table 5: Impact of the vehicle capacity Q.

Set	Id	$q=1:1$	$q=1:15$		$q=1:2$		$q=1:25$		$q=1:3$		$q=1:35$		$q=1:4$	
		Cost	Cost	*Imp*(%)	Cost	*Imp*(%)	Cost	*Imp*(%)	*Cost*	*Imp*(%)	Cost	*Imp*(%)	Cost	*Imp*(%)
5	1	9450.1	8461.8	10.5	7853.1	16.9	7478.1	20.9	7471.2	20.9	7397.3	21.7	7355.0	22.2
	10	9529.6	9284.4	2.6	9209.9	3.4	8420.1	11.6	8227.9	13.7	7925.3	16.8	7746.6	18.7
6	1	12142.7	11012.2	9.3	10896.9	10.3	10135.0	16.5	10085.2	16.9	9789.3	19.4	9728.4	19.9
	10	10198.6	9966.0	2.3	9363.6	8.2	8745.8	14.2	8519.6	16.5	8378.0	17.9	8193.4	19.7
7	1	10198.6	9966.0	2.3	9363.6	8.2	8745.8	14.2	8519.6	16.5	8378.0	17.9	8193.4	19.7
	10	13886.9	13588.1	2.2	12997.4	6.4	12897.3	7.1	12535.8	9.7	12349.8	11.1	12094.9	12.9
8	1	23603.0	23603.0	0.0	21144.1	10.4	19794.0	16.1	19573.6	17.1	19178.7	18.7	18392.8	22.1
	10	21297.7	19905.6	6.5	18067.4	15.2	17859.7	16.1	17617.8	17.3	17606.2	17.3	16245.4	23.7

Table 6: Sensitivity analysis on probability of stock out.

Set	Id	NoStockOut	StockOut (1/K)		StockOut (0.5)	
		Cost	Cost	*Imp*(%)	Cost	*Imp*(%)
5	1	5748.8	7399.8	– 28.7	16796.3	– 192.2
	10	6156.2	9408.7	– 52.8	15638.5	– 154.0
6	1	8760.1	10135.0	– 15.7	22336.9	– 155.0
	10	6847.6	8745.8	– 27.7	20300.6	– 196.5
7	1	8122.1	11324.8	– 39.4	31633.1	– 289.5
	10	9320.3	12897.3	– 38.4	33695.7	– 261.5
8	1	13687.3	19794.0	– 44.6	49911.2	– 264.7
	10	14308.0	17859.7	– 24.8	50973.9	– 256.3

almost 4 times the cost of no "NoStockOut," and the right trade off between the distribution cost and SKUs availability at the different warehouses needs to be done.

5. Conclusions and Future Research

In this paper, we proposed a new variant of vehicle routing problem motivated by real-world applications arising in the e-commerce logistics. The problem, called the multi-depot vehicle routing problem with order split and allocation (MD-CVRP-OSA) considers the split and allocation issue of online customer orders in traditional vehicle routing problem. We formally introduced the new problem and formulated it as an integer programming problem. To solve the MD-CVRP-OSA, we proposed an effective variable neighborhood search (VNS) algorithm that makes use of seven elaborately designed operators and a diversification mechanism.

The VNS algorithm was extensively tested on both randomly generated and real-life instances. In particular, a case study from JD.com, one of China's biggest e-commerce companies, was analyzed and solved with the proposed VNS approach. The computational results obtained showed the effectiveness of the VNS approach and also different components of the algorithm.

The future research perspectives are multiple. Firstly, we recommend to pursue the study of exact methods for the MD-CVRP-OSA. Secondly, our problem definition and solution method could be extended to incorporate additional operational constraints, such as time window constraints, the use of an heterogeneous vehicle fleet, and order returns.

Data Availability

The numerical experiment data used to support the findings of this study were generated by the authors. Requests for access to these data should be made to Yi Tao (oryitao@foxmail.com).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Dr. Yi Tao was partially supported by National Natural Science Foundation of China under grant number 72071050 and Guangdong Basic and Applied Basic Research Foundation under grant number 2022A1515010541. Dr. Lijun Wei was partially supported by National Natural Science Foundation of China under grant number 71871070.

References

[1] I. Dayarian, M. Savelsbergh, and J.-P. Clarke, "Same-day delivery with drone resupply," *Transportation Science*, vol. 54, no. 1, pp. 229–249, 2020.

[2] M. A. Klapp, A. L. Erera, and A. Toriello, "The dynamic dispatch waves problem for same-day delivery," *European Journal of Operational Research*, vol. 271, no. 2, pp. 519–534, 2018.

[3] I. Dayarian and M. Savelsbergh, "Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders," *Production and Operations Management*, vol. 29, no. 9, pp. 2153–2174, 2020.

[4] A. M. Stroh, A. L. Erera, and A. Toriello, "Tactical Design of Same-day Delivery Systems," *Management Science*, vol. 68, no. 5, pp. 3444–3463, 2022.

[5] S. A. Voccia, A. M. Campbell, and B. W. Thomas, "The same-day delivery problem for online purchases," *Transportation Science*, vol. 53, no. 1, pp. 167–184, 2019.

[6] W. J. A. van Heeswijk, M. R. K. Mes, and J. M. J. Schutten, "The delivery dispatching problem with time windows for urban consolidation centers," *Transportation Science*, vol. 53, no. 1, pp. 203–221, 2019.

[7] P. Toth and D. Vigo, *Vehicle Routing: Problems, Methods, and Applications, Second Edition. MOS-SIAM Series on Optimization*, Society for Industrial and Applied Mathematics, Philadelphia, 2014.

[8] K. Braekers, K. Ramaekers, and I. V. Nieuwenhuyse, “The vehicle routing problem: state of the art classification and review,” *Computers & Industrial Engineering*, vol. 99, pp. 300–313, 2016.

[9] S. Y. Tan and W. C. Yeh, “The vehicle routing problem: state-of-the-art classification and review,” *Applied Sciences*, vol. 11, no. 21, p. 10295, 2021.

[10] M. Drexl and M. Schneider, “A survey of variants and extensions of the location-routing problem,” *European Journal of Operational Research*, vol. 241, no. 2, pp. 283–308, 2015.

[11] C. Prodhon and C. Prins, “A survey of recent research on location-routing problems,” *European Journal of Operational Research*, vol. 238, no. 1, pp. 1–17, 2014.

[12] J. R. Montoya-Torres, J. López Franco, S. Nieto Isaza, H. Felizzola Jiménez, and N. Herazo-Padilla, “A literature review on the vehicle routing problem with multiple depots,” *Computers & Industrial Engineering*, vol. 79, pp. 115–129, 2015.

[13] D. Chen and Z. Yang, “Multiple depots vehicle routing problem in the context of total urban traffic equilibrium,” *Journal of Advanced Transportation*, vol. 2017, Article ID 8524960, 14 pages, 2017.

[14] J. Li, Y. Ma, R. Gao et al., “Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem,” *IEEE Transactions on Cybernetics*, pp. 1–14, 2021, https://ieeexplore.ieee.org/abstract/document/9547060.

[15] Y. Ma, J. Li, Z. Cao et al., “Efficient neural neighborhood search for pickup and delivery problems,” 2022, https://arxiv.org/abs/2204.11399.

[16] J. Li, L. Xin, Z. Cao, A. Lim, W. Song, and J. Zhang, “Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 2306–2315, 2022.

[17] D. Gulczynski, B. Golden, and E. Wasil, “The multi-depot split delivery vehicle routing problem : an integer programming-based heuristic , new test problems , and computational results,” *Computers & Industrial Engineering*, vol. 61, no. 3, pp. 794–804, 2011.

[18] S. Ray, A. Soeanu, J. Berger, and M. Debbabi, “The multi-depot split-delivery vehicle routing problem : model and solution algorithm,” *Knowledge-Based Systems*, vol. 71, pp. 238–265, 2014.

[19] S. Izar and S. Suwilo, “Multi-depot vehicle routing problem with split delivery and time windows,” *Bulletin of Mathematics*, vol. 8, pp. 169–177, 2016.

[20] X. Wang, B. Golden, E. Wasil, and R. Zhang, “The min-max split delivery multi-depot vehicle routing problem with minimum service time requirement,” *Computers and Operations Research*, vol. 71, pp. 110–126, 2016.

[21] N. Mladenovic and P. Hansen, “Variable neighborhood search,” *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[22] P. Hansen, N. Mladenovié, J. Brimberg, and J. A. M. Pérez, “Variable neighborhood search,” in *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Eds., pp. 57–97, Springer International Publishing, Cham, 2019.

[23] J. Fong, S. Salhi, and N. Wassan, “The cumulative capacitated vehicle routing problem with min-sum and minmax objectives : an effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search,” *Transportation Research Part B: Methodological*, vol. 101, pp. 162–184, 2017.

[24] A. Stenger, D. Vigo, S. Enz, and M. Schwind, “An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping,” *Transportation Science*, vol. 47, no. 1, 2013.

[25] L. Wei, Z. Zhang, D. Zhang, and A. Lim, “A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints,” *European Journal of Operational Research*, vol. 243, no. 3, p. 798, 2015.

[26] P. Olender and W. Ogryczak, “A revised variable neighborhood search for the discrete ordered median problem,” *European Journal of Operational Research*, vol. 274, no. 2, pp. 445–465, 2019.

[27] G. Clarke and J. Wright, “Scheduling of vehicles from a central depot to a number of delivery points,” *Operations Research*, vol. 12, no. 4, pp. 568–581, 1964.

[28] R. Özsen and U. W. Thonemann, *An optimal expediting policy for periodic-review inventory systems*, 2012, Working paper http://ssrn.com/abstract=2182036.