A variable neighborhood search algorithm with constraint relaxation for the two-echelon vehicle routing problem with simultaneous delivery and pickup demands

Ran LIU[a], Shan JIANG[b]

[a.] Department of Industrial Engineering & Management, Shanghai Jiao Tong University, Shanghai, P. R. China, Liuran2009@sjtu.edu.cn

[b.] Department of Industrial and Systems Engineering, Rutgers University, New Jersey, USA, sj576@scarletmail.rutgers.edu

Corresponding author: Ran LIU, http://orcid.org/0000–0002–7922–8969

Abstract. This paper considers a special vehicle routing problem, the two-echelon vehicle routing problem with simultaneous delivery and pickup demands (2E-VRPSDP). The 2E-VRPSDP differs from classic transportation and vehicle routing problems in two ways. First, freight delivery from the depot to the customers is managed by shipping the freight through intermediate satellites. Second, each customer in the 2E-VRPSDP may have simultaneous delivery and pickup demands. The 2E-VRPSDP is an extension of the two-echelon vehicle routing problem (2E-VRP) and the vehicle routing problem with simultaneous delivery and pickup (VRPSDP). A variable neighborhood search algorithm is designed to solve the 2E-VRPSDP in which both feasible and infeasible solutions can be explored. Numerical results show that the proposed algorithm is effective and that the algorithm can provide reasonable solutions within an acceptable computational time.

Keywords:

constraint relaxation; variable neighborhood search; vehicle routing; two-echelon; delivery and pickup

1 Introduction

In the past decade, the *two-echelon vehicle routing problem* (2E-VRP) has become a new interest in the *vehicle routing problems* (VRPs) research field. The logistics network of 2E-VRP is composed of two echelons, i.e., the freight is delivered from the depot to the satellites by large capacity vehicles in the first echelon and then transported from the satellites to the customers by the second echelon vehicles of relatively small capacity. The objective of the 2E-VRP is to minimize the total transportation costs of both echelons. The 2E-VRP problem arises in many practical transportation and distribution contexts, such as city logistics applications (Feliu *et al.* 2007; Jepsen *et al.* 2013).

The 2E-VRP has been studied by many researchers, and some very good studies can be found for this problem (Perboli *et al.* 2011; Jepsen *et al.* 2013; Breunig *et al.* 2016; Liu *et al.* 2017). In the existing research, generally, it is assumed that one customer only has either delivery or pickup demand. However, in some practical applications, the vehicle needs to distribute and collect freight simultaneously. We consider a practical two-echelon logistical problem arising in the home health care (HHC) industry in Shanghai, China. Many HHC companies have been built in Shanghai for patients who require long and regular health care to provide quality health services at their homes. The typical services in the HHC involve various logistic activities, including delivering medicines and medical instruments to patients, picking up biological samples from patients' homes, and collecting medical waste from patients' homes for disposal. Considering the HHC company, the core component in-home health care problems is to find a feasible working schedule for their drivers and vehicles to reduce the operating cost. For a large city, because customers spread in a very large area, one HHC company establishes some subcompanies in different districts, and daily logistic services such as delivering medicines and collecting medical waste are implemented as a two-echelon logistics system. First, in the morning, the goods to be delivered to the customers (e.g., the medicines for the customers) are transported from the company (or center warehouse) to the subcompanies. Next, small subcompany vehicles deliver goods to customers and pick up goods from customers (e.g., collecting medical waste) to take back to the subcompanies. After all the pickup goods are collected in subcompanies, finally, the

vehicles at the HHC company pick up such goods from subcompanies to return to the HHC company and complete one day's operations. Clearly, this is a special two-echelon transportation network in which the first echelon consists of the HHC company and subcompanies, and the second echelon contains subcompanies and customers. We can further observe that, unlike the classical 2E-VRP, the operations by the HHC company consist of three phases. The HHC company must design a set of routes for vehicles in this two-echelon network. According to the HHC company investigations, it is not easy for the company's manager and planner. This is because this routing optimization problem is rather complex, which can be seen as a combination of two challenging problems, i.e., the two-echelon vehicle routing problem (2E-VRP) and the vehicle routing problem with simultaneous delivery and pickup (VRPSDP), because the problem has a two-echelon network structure and the customers have both delivery and pickup demands simultaneously. Since both the VRPSDP and the 2E-VRP are NP-hard, the problem stated above is also NP-hard and even more complex and harder than these two problems.

In this paper, we focus on this special logistical problem, which is named the *two-echelon vehicle routing problem with simultaneous delivery and pickup* (2E-VRPSDP). Although the 2E-VRP and VRPSDP have been studied for more than ten years, as stated above, the 2E-VRPSDP solution structure differs from the classical problems (the detailed differences are shown and analyzed in section 2). To our knowledge, there is no literature about the 2E-VRPSDP proposed in this paper. Because of the difficulties in solving instances of practical interest, we propose a *variable neighborhood search* (VNS) heuristic in which both feasible and infeasible solutions are explored to enhance the algorithmic search ability. Numerical results show that the VNS algorithm can efficiently solve the 2E-VRPSDP.

The rest of this paper is organized as follows. Section 2 defines the problem and discusses the differences with existing problems that have been studied in the literature. Section 3 reviews the relevant literature. Section 4 builds the mathematical model for the 2E-VRPSDP. Section 5 describes the VNS algorithm. Computational results are reported in section 6. Section 7 concludes the paper.

2 Problem definition

The 2E-VRPSDP can be defined formally as follows. Let $G=(V, E)$ be a graph with the node set $V=V_0 \cup V_S \cup V_C$ and edge set $E=E_1 \cup E_2$. In set V, $V_0=\{0\}$ is the depot, set $V_s=\{1, \ldots, |V_S|\}$ represents the set of satellites where $|V_S|$ is the cardinality of set V_S, and set $V_C=\{|V_S|+1, \ldots, |V_S|+|V_C|\}$ is the set of customers. Each customer $i \in V_C$ has a delivery demand and a pickup demand simultaneously, denoted as d_i and p_i, respectively. A fleet of K_1 homogeneous first echelon vehicles with a capacity of Q_1 is located at depot V_0, which can visit the depot and satellites. Additionally, a total of K_2 homogeneous second echelon vehicles with a capacity of Q_2 are located at all satellites, which can only visit the satellite and customers, and $Q_1>Q_2$. Set E is divided into two subsets, E_1 and E_2. Set $E_1=\{(i, j): i, j \in \{V_0\} \cup V_S\}$ corresponds to the edges between the depot and the satellites and between different satellites, which can only be traveled by the first echelon vehicles. Set $E_2=\{(i, j): i, j \in V_S \cup V_C, i, j \notin V_S \times V_S\}$ represents the edges connecting the satellites and the customers and those connecting different customers. Each edge $(i, j) \in E$ has a nonnegative cost (distance), denoted as c_{ij}, and for each pair $(i, j) \in E$, $c_{ij}=c_{ji}$.

Note that a solution of the 2E-VRPSDP consists of three phases. In the first phase, a number of first echelon vehicles start from depot V_0 and deliver goods to a sequence of satellites and then return to the depot. For notational convenience, each of these routes is called the ***first phase delivery route***. Then, in the second phase, second echelon vehicles start from the satellites, each meeting the delivery and pickup demands of one or several customers, and return to the starting satellites. Now, all the pickup demands from the customers are stored at the satellites. Similarly, each route at this echelon is referred to as a ***second phase delivery and pickup route***. Finally, again, the first echelon vehicles depart from the depot and collect all pickup goods from the satellites and return back to the central depot. Such routes are called ***third phase pickup routes***. Note that at each phase, direct vehicle routes between the depot and customers are forbidden. At each phase, some constraints must be respected as follows.

(1) Each ***first phase delivery route*** and ***third phase pickup route*** start and end at the depot;

(2) Each route is assigned to exactly one vehicle;

(3) At any time, the total load of the vehicle cannot exceed the vehicle capacity;

(4) A customer can be served by only one *second phase delivery and pickup route*; a satellite can be served by at most one *first phase delivery route* and at most one-*third phase pickup route*. This constraint means that the demands of both echelons cannot be split.

The objective of 2E-VRPSDP is to determine the vehicle routes of both echelons to the sum of the routing costs, i.e., the total travel distances of all vehicle routes in the above three phases are minimized.

Compared with the 2E-VRPSDP, the 2E-VRP has only two-phase operations: first, the goods to be delivered to the customers are transported from the depot to the satellites by large vehicles; next, small vehicles starting from satellites deliver goods to customers and complete the operations (Feliu *et al.* 2007; Jepsen *et al.* 2013; Hemmelmayr *et al.* 2012). To clearly illustrate the 2E-VRPSDP and compare it with the 2E-VRP, a solution to the 2E-VRP with one depot, three satellites (S_1-S_3), and eight customers (C_1-C_8) is shown in Figure 1, which consists of three parts A-C. Each customer has pickup and delivery demands. First, as shown in part A of Figure 1, the company distributes the goods to the satellites by two first echelon delivery routes (blue lines). Next, in part B, three vehicles start from each satellite and deliver goods to each customer and collect pickup goods to deliver back to the satellites (black dotted line). Finally, part C shows two routes that collect all the goods to deliver to the depot (red lines).

Belgin *et al.* (2018) introduced an interesting two-echelon vehicle routing problem with simultaneous pickup and delivery requests. In their problem, the customers have pickup and delivery demands, and the pickup and delivery activities are performed simultaneously by the same vehicles through satellites to customers in the second echelon. Belgin *et al.* (2018) assumed that in the first echelon, goods are delivered from the central depot to each satellite and collected from satellites to deliver back to the depot by the same vehicles. That is, a solution to the problem of Belgin *et al.* (2018) consists of only two steps: the delivery routes and pickup routes between the depot and satellites (as shown in parts A and C of Figure 1) are combined. Furthermore, we note that many other researchers also state that pickup and

delivery in two-echelon city logistics can improve the efficiency if it is performed simultaneously on the first echelon vehicles. For example, Gianessi *et al.* (2016) solved the related tactical planning problems; for tactical planning problems, please refer to Fontaine *et al.* (2017, 2021). We agree that the operations in these works are reasonable in practical applications. We investigate many logistics and home health care companies and find that they do not adopt a two-step model because, in some scenarios, pickup and delivery operations for a satellite are not easy to perform at the same time. For example, one first echelon vehicle *a* reaches satellite *i* at 8:00 and unloads the goods for this satellite. Next, a second echelon vehicle *b* delivers such goods to customers and returns to this satellite at 10:00, with all pickup demands from its customers. Following the assumption of Belgin *et al.* (2018), vehicle *a* has to wait at satellite *i* until vehicle *b* returns to this satellite, i.e., two hours in this example. In practice, some companies deliver goods from a central depot to satellites in the morning and collect goods from satellites to deliver back to the depot in the afternoon, consistent with the problem definition in this paper.

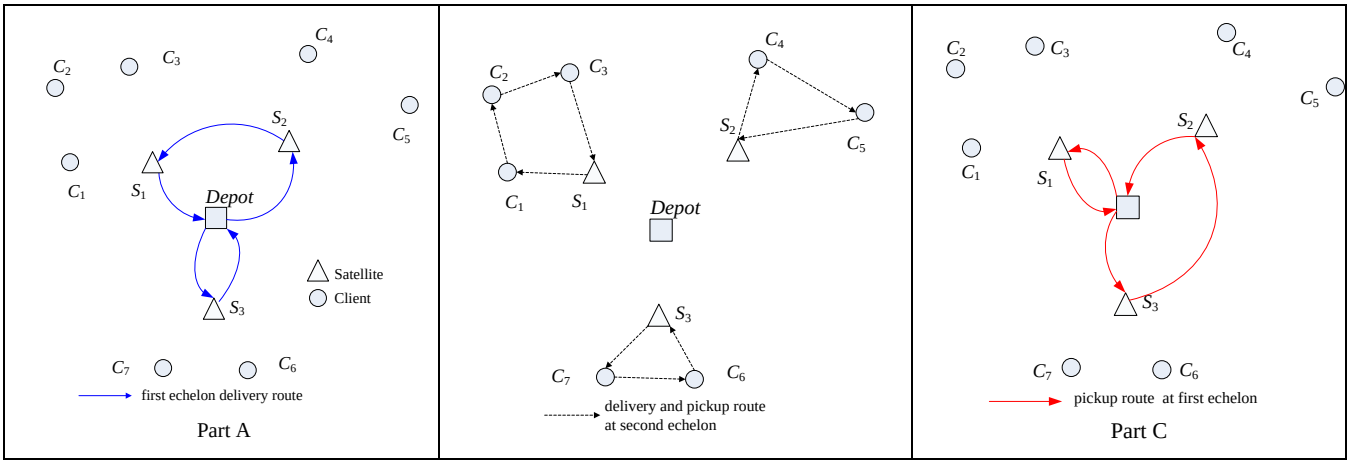

Figure 1. A solution to the 2E-VRPSDP

3 Literature review

Although many variants of the VRP and the 2E-VRP have been studied in the existing literature, little research has been performed on the 2E-VRPSDP. As stated above, two main bodies, the 2E-VRP and the VRPSDP, are relevant to the 2E-VRPSDP. We survey the literature in these two bodies of research.

3.1 The 2E-VRP literature

The 2E-VRP has drawn much attention in the past decade. Both exact and heuristic algorithms have been designed to solve this problem. In terms of exact algorithms, first, Feliu *et al.* (2007) proposed a flow-based mathematical model for the 2E-VRP and developed two families of valid inequalities. Feliu *et al.* (2007) used an exact branch-and-cut algorithm to solve instances containing up to 32 customers and 2 satellites. Perboli *et al.* (2011) introduced some new optimality cut classes and improved the algorithm of Feliu *et al.* (2007). Jepsen *et al.* (2013) built a directed three index formulation similar to that of Perboli *et al.* (2011); they derived a relaxation from it to avoid giving incorrect upper bounds when more than two satellites were included in the solution. The branch-and-cut algorithm of Jepsen *et al.* (2013) performed better than that of Perboli *et al.* (2011). A branch-and-cut-and-price algorithm was presented by Santos *et al.* (2014), which overcame symmetry issues and was further strengthened by valid inequalities. Baldacci *et al.* (2013) proposed a new mathematical formulation. Their exact method decomposed the problem into a limited set of multiple-depot vehicle routing problems with side constraints. Computational results on benchmarks proved that their method outperformed previously published exact methods in terms of size, number of problems solved to optimality, and computing time. Considering some variants of the classical 2E-VRP, Liu *et al.* (2018) introduced a 2E-VRP with grouping constraints, in which customers were divided into several disjoint groups, and the grouping constraints ensured that customers from the same group were served by vehicles from the same satellite. They formulated the problem as a mixed-integer program and proposed valid inequalities to strengthen the model. A branch-and-cut algorithm was implemented to solve the problem, which could solve more instances to optimality than CPLEX. Darvish *et al*. (2019) studied *a flexible* 2E-VRP in which a supplier delivered a commodity to its customers through a two-echelon supply network. Two sources of flexibility were analyzed: flexibility in network design and flexibility in due dates. The former was related to the possibility of renting any of the satellites in any period of the planning horizon, whereas the latter was related to the possibility of serving a customer between the period an order was set and a due date. A mathematical programming

formulation to this problem was presented, and an exact method was proposed that was based on the interplay between two branch-and-bound algorithms. Dellaert *et al.* (2019) studied the 2E-VRP with time windows. Different from the classical 2E-VRP, the second echelon consisted of transferring freight from satellites to the final customers within their time windows. They proposed two path-based mathematical formulations for the problem and developed branch-and-price–based algorithms to solve the problem. The algorithms solved instances of up to five satellites and 100 customers to optimality. Breunig *et al.* (2019) formulated an extension of the 2EVRP, called *the electric* 2EVRP, which involved electric vehicles for second echelon deliveries, battery capacity constraints, and possible visits to charging stations, and used it as a prototypical problem for the study of multiechelon battery-powered supply chains. They designed an efficient exact algorithm based on the enumeration of candidate solutions for the first echelon and on bounding functions and route enumeration for the second echelon, along with a problem-tailored large neighborhood search metaheuristic. Marques *et al.* (2020) proposed a branch-cut-and-price algorithm for the 2EVRP. The authors introduced a new route-based formulation for the problems that do not use variables to determine product flows in satellites. They also introduced a new branching strategy that significantly decreased the size of the branch-and-bound tree and introduced a new family of satellite supply inequalities. Because exact algorithms usually cannot effectively solve the large-size 2E-VRP in a reasonable computation time, many researchers have resorted to heuristic methods. Crainic *et al.* (2011) developed a multistart heuristic based on separating the problem by solving customer assignments heuristically and then dealing with the remaining VRPs. In their method, a perturbation mechanism was adopted to iteratively build new solutions, and a feasibility search procedure was used to bring the solution back into the feasible region. In addition to the exact methods, Perboli *et al.* (2011) also presented two math heuristics based on the information obtained by solving the linear relaxation of the mathematical formulation model. Hemmelmayr *et al.* (2012) designed an adaptive large neighborhood search heuristic for the problem. They introduced some large-scale instances with up to 200 customers. Zeng *et al.* (2014) solved the problem using a greedy randomized adaptive search procedure embedded with a route-first

cluster-second procedure and a variable neighborhood descent. Their approach was tested on instances not larger than 50 customers. Breunig *et al.* (2016) designed an effective large neighborhood-based heuristic for solving the 2E-VRP. Their algorithm improved 18 best-known solutions. Grangier *et al.* (2016) addressed a variant of the 2E-VRP that integrated constraints arising in city logistics such as time window constraints, synchronization constraints, and multiple trips at the second echelon. They proposed an adaptive large neighborhood search to solve this problem. Amarouche *et al.* (2018) proposed a new hybrid heuristic method for solving the 2E-VRP that relied on two components. The first component effectively explored the search space to discover a set of interesting routes, and the second recombined the discovered routes into high-quality solutions. Rohmer *et al.* (2019) presented a two-echelon inventory-routing problem for perishable products. Products were delivered from a supplier to an intermediary depot, where storage may occur and from which they were delivered by smaller vehicles to the customer locations. The objective was to minimize the total transportation and holding costs. An adaptive large neighborhood search metaheuristic was designed to address the problem. Jie *et al.* (2019) considered the 2E-VRP with battery swapping stations, which aimed to determine the delivery strategy under battery driving range limitations for city logistics. The electric vehicles operating in the different echelons had different load capacities, battery driving ranges, power consumption rates, and battery swapping costs. The authors proposed a hybrid algorithm that combined column generation and an adaptive large neighborhood search to solve the problem. Mühlbauer *et al.* (2021) studied a variant of the 2E-VRP that used cross-docking from vans to cargo bicycles at so-called satellites. To solve large instances, the authors introduced a new efficient parallelized large neighborhood search algorithm. The algorithm was tested using symmetric 2E-VRP benchmark instances from the literature.

3.2 The VRPSDP literature

In addition to the 2E-VRP, the VRPSDP has also been studied intensively (Koç *et al.* 2020). The VRPSDP was first proposed by Min (1989). Since VRPSDP is NP-hard, exact algorithms have difficulty solving large-scale real-life cases. Only a few exact methods have been developed for the problem. Dell'Amico *et al.* (2006) proposed an exact

branch-and-price approach for the VRPSDP. Their method solved instances containing up to 40 customers. Subramanian *et al.* (2011) designed a branch-and-cut-based method and later Subramanian a branch-and-cut-and-price method for solving the VRPSDP. Hernández-Pérez *et al.* (2021) addressed a generalization of the one-commodity pickup and delivery traveling salesman problem where each customer supplied or demanded a given quantity of a certain product. The authors presented three mathematical models for the problem and designed an exact branch-and-cut algorithm to solve it.

Compared with the exact algorithms, heuristic approaches dominated the solution methodologies of the VRPSDP. Nagy and Salhi (2005) proposed a heuristic that allowed infeasibilities to occur and guided the search toward strong feasibility through search routines. Crispim and Brandao (2005) presented a hybrid algorithm for the VRPSDP that was comprised of the two metaheuristics of tabu search and variable neighborhood descent. Alfredo Tang Montané and Galvão (2006) proposed a tabu search approach that utilized relocation, interchange and crossover movements to obtain interroute adjacent solutions and used a 2-opt procedure to obtain alternative intraroute solutions. Bianchessi and Righini (2007) presented and compared constructive algorithms, local search algorithms and tabu search algorithms on the VRPSDP. Gajpal and Abad (2009) designed an ant colony system (ACS) for solving the VRPSDP, which used a construction rule as well as two multiroute local search schemes. Zachariadis *et al.* (2010) introduced an adaptive memory algorithmic framework to solve the VRPSDP, which combined promising solution features to generate high-quality solutions. Avci and Topaloglu (2015) proposed an adaptive local search solution approach for the VRPSDP, which hybridized a simulated annealing inspired algorithm with variable neighborhood descent. A perturbation-based variable neighborhood search heuristic for solving the VRPSDP was designed by Polat *et al.* (2015). Zachariadis *et al.* (2016) introduced a special VRPSDP with two-dimensional loading constraints, which covered cases where customers posed delivery and pick up requests for transporting nonstackable rectangular items. The authors proposed an optimization framework that employed memorization techniques to accelerate the solution methodology. Computational results were reported on the VRPSPD, the VRP with two-dimensional constraints, and newly constructed

benchmark problems. Qiu *et al.* (2018) studied a variant of VRPSDP, in which customers' demands were discrete in terms of batches (or orders) and each customer could be visited by a variety of vehicles or several times by one vehicle. A tabu search algorithm with specifically designed batch combinations and item creation operations was proposed to solve the problem. Majidi *et al.* (2018) dealt with the pollution-routing version of VRPSDP, where the goal was to minimize fuel consumption and emissions by scheduling and routing customers. A nonlinear mix integer programming model was presented for this problem, and an adaptive large neighborhood search heuristic was proposed for the solution method including new removal and insertion operators. Zhang *et al.* (2019) addressed a multicommodity many-to-many vehicle routing problem with simultaneous pickup and delivery for a fast-fashion retailer in Singapore. To solve large-scale instances, an adaptive memory programming-based algorithm combined with techniques such as the regret insertion method for initializing the solution pool, the segment-based evaluation scheme, and the advanced pool management method was proposed in this work.

Although as stated above, many good exact algorithms and heuristics were proposed for the 2E-VRP and VRPSDP, to our knowledge, only Belgin *et al.* (2018) studied a special two-echelon vehicle routing problem with simultaneous pickup and delivery. However, the basic operation scheme of the problem in Belgin *et al.* (2018) differs from the problem studied in this paper. Belgin *et al.* (2018) assumed that goods of the first echelon are delivered from the central depot to each satellite and collected from satellites back to the depot simultaneously by the same vehicles. As stated in section 2, our 2E-VRPSDP separates the routes in the first echelon into one delivery route and one pickup route. To our knowledge, this is the first work on 2E-VRPSDP.

4 Mathematical formulation

The mathematical formulation of the 2E-VRPSDP (mixed-integer linear program, MILP) is presented below.

Decision variables:

$y_{i,j}$ binary variable equal to 1 if a vehicle travels directly from node *i* to node *j* at a first phase delivery route;

$x_{i,j}$ binary variable equal to 1 if a vehicle travels directly from node *i* to node *j* at a second phase delivery and pickup route;

$l_{i,j}$ binary variable equal to 1 if a vehicle travels directly from node *i* to node *j* at a third phase pickup route;

$z_{i,j}$ binary variable equal to 1 if customer *i* is assigned to satellite *j*;

a_i quantity of goods to be delivered and loaded on the second phase vehicle until customer *i* is visited;

b_i quantity of pickup goods loaded on the second phase vehicle immediately after customer *i* is visited;

ξ_i quantity of demands to be delivered at satellite *i*;

∂_i quantity of demands to be picked up at satellite *i*;

A_i quantity of goods to be delivered to satellites loaded on the first phase vehicle until satellite *i* is visited;

B_i quantity of pickup goods loaded on the third phase vehicle immediately after satellite *i* is visited;

Objective function:

$$Min \sum_{i\in V_C\cup V_S}\sum_{j\in V_C\cup V_S} c_{ij}x_{i,j} + \sum_{i\in V_0\cup V_S}\sum_{j\in V_0\cup V_S} c_{ij}\left(y_{i,j}+l_{i,j}\right) \quad \textbf{(1)}$$

Subject to,

$$\sum_{i\in V_C\cup V_S, i\neq j} x_{i,j} = 1 \qquad \forall j \in V_C \quad (2)$$

$$\sum_{i\in V_C\cup V_S, i\neq j} x_{i,j} = \sum_{i\in V_C\cup V_S, i\neq j} x_{j,i} \qquad \forall j \in V_C \quad (3)$$

$$a_j + d_i + Q_2 x_{i,j} \le Q_2 + a_i \qquad \forall i,j \in V_C, i\neq j \quad (4)$$

$$b_i + p_j + Q_2 x_{i,j} \le Q_2 + b_j \qquad \forall i,j \in V_C, i\neq j \quad (5)$$

$$a_i - d_i + b_i \le Q_2 \qquad \forall i \in V_C \quad (6)$$

$$d_i \le a_i \le Q_2 \qquad \forall i \in V_C \quad (7)$$

$$p_i \le b_i \le Q_2 \qquad \forall i \in V_C \quad (8)$$

$$\sum_{j\in V_S} z_{i,j} = 1 \qquad \forall i \in V_C \quad (9)$$

$$x_{i,j} \le z_{i,j} \qquad \forall i \in V_C, j \in V_S \quad (10)$$

$$x_{j,i} \le z_{i,j} \qquad \forall i \in V_C, j \in V_S \quad (11)$$

$$x_{e,m} + z_{e,i} + \sum_{j \in V_S, j \neq i} z_{m,j} \leq 2 \qquad \forall e, m \in V_C, e \neq m, i \in V_S \tag{12}$$

$$\sum_{i \in V_C} \sum_{j \in V_S} x_{i,j} \leq K_2 \tag{13}$$

$$\sum_{i \in V_C} \sum_{j \in V_S} x_{j,i} \leq K_2 \tag{14}$$

$$\partial_j = \sum_{i \in V_C} p_i \cdot z_{i,j} \qquad \forall j \in V_S \tag{15}$$

$$\xi_j = \sum_{i \in V_C} d_i \cdot z_{i,j} \qquad \forall j \in V_S \tag{16}$$

$$\sum_{i \in V_0 \cup V_S, i \neq j} y_{i,j} = 1 \qquad \forall j \in V_S \tag{17}$$

$$\sum_{i \in V_S \cup V_0, i \neq j} y_{i,j} = \sum_{i \in V_S \cup V_0, i \neq j} y_{j,i} \qquad \forall j \in V_S \tag{18}$$

$$A_j + \xi_i + Q_1 y_{i,j} \leq Q_1 + A_i \qquad \forall i, j \in V_S, i \neq j \tag{19}$$

$$A_i \leq \xi_i \leq Q_1 \qquad \forall i \in V_S \tag{20}$$

$$\sum_{j \in V_S} y_{V_0,j} \leq K_1 \tag{21}$$

$$\sum_{j \in V_S} y_{j,V_0} \leq K_1 \tag{22}$$

$$\sum_{i \in V_0 \cup V_S, i \neq j} l_{i,j} = 1 \qquad \forall j \in V_S \tag{23}$$

$$\sum_{i \in V_S \cup V_0, i \neq j} l_{i,j} = \sum_{i \in V_S \cup V_0, i \neq j} l_{j,i} \qquad \forall j \in V_S \tag{24}$$

$$B_i + \partial_j + Q_1 l_{i,j} \leq Q_1 + B_j \qquad \forall i, j \in V_S, i \neq j \tag{25}$$

$$B_i \leq \partial_i \leq Q_1 \qquad \forall i \in V_S \tag{26}$$

$$\sum_{j \in V_S} l_{V_0,j} \leq K_1 \tag{27}$$

$$\sum_{j \in V_S} l_{j,V_0} \leq K_1 \tag{28}$$

$$x_{i,j} \in \{0,1\} \quad \forall i, j \in V_C \cup V_S \tag{29}$$

$$y_{i,j} \in \{0,1\} \quad \forall i, j \in V_0 \cup V_S \tag{30}$$

$$l_{i,j} \in \{0,1\} \quad \forall i, j \in V_0 \cup V_S \tag{31}$$

$$z_{i,j} \in \{0,1\} \quad \forall i \in V_C, j \cup V_S \tag{32}$$

$$a_i \geq 0, \ b_i \geq 0 \qquad \forall i \in V_C \tag{33}$$

$$\xi_i \geq 0, \partial_i \geq 0, A_i \geq 0, B_i \geq 0 \quad \forall i \in V_s \tag{34}$$

The objective function (1) minimizes the total transportation cost. Constraints (2) ensure that each customer is visited exactly once. Constraints (3) guarantee flow equations for customers, i.e., every vehicle that arrives at a customer must leave that customer. Vehicle load constraints in the second phase delivery and pickup route are explained in Constraints (4)-(8). Such constraints also eliminate the subtours in the second echelon routes. Constraint (6) ensures that in any customer's area after the vehicle delivers and picks up goods, the loaded quantity does not exceed the vehicle capacity. Constraint (7) denotes that the demand of customer *i* must not exceed the load quantity on the vehicle before arriving, and this load quantity value must not exceed the capacity of the vehicle. In Constraint (4) if $x_{i,j} = 1$, i.e., customer *j* is next to customer *i*, and $a_i - d_i = a_j$, which indicates the delivery quantity on the second phase vehicle is decreased by the demand quantity of customer *i* after visiting, so that $a_j + d_i + Q_2 \leq Q_2 + a_i$. Otherwise, when $x_{i,j} = 0$, the inequality also holds since $d_i \leq a_i$ and $a_j \leq Q_2$ according to the definition of these notations. Constraints (5) and (8) correspond to the pickup goods case and are of a similar form to Constraints (4) and (7). Constraints (9) force each customer to be assigned to one satellite. Constraints (10)-(11) ensure that there is no arc connecting customer *i* and satellite *j* if customer *i* is not allocated to satellite *j*. Constraint (12) avoids two connected customers belonging to two different satellites. Constraints (13) and (14) ensure the available fleet size for the second echelon transportation. Constraints (15) and (16) equalize the total pickup and delivery demands of the customers and corresponding satellites. Constraints (17)-(22) are related to the first phase delivery route, while Constraints (23)-(28) are about the third phase pickup route; they are similar to some early constraints. Constraints (17)-(18) and (23)-(24) ensure that every satellite is visited by exactly one vehicle from the first phase and the third phase, respectively. Constraints (19) and (20) are vehicle load constraints in the first phase delivery route; Constraints (25)-(26) are corresponding constraints in the third phase pickup route. Constraints (21), (22), (27), and (28) impose fleet size constraints in the first echelon

transportation. Finally, constraints (29)-(34) define the domain of decision variables

5 Variable neighborhood search algorithm

Since the VRPSDP and the 2E-VRP are two difficult NP-hard problems, most research on these two problems resorts to heuristics methods, such as tabu search (TS), variable neighborhood search (VNS), adaptive large neighborhood search heuristic (ALNS), and genetic algorithm (GA). Because the 2E-VRP can be seen as a special case of the 2E-VRPSDP studied in this paper, the 2E-VRPSDP is also an NP-hard problem and even more complex than VRPSDP and 2E-VRP. Thus, exact algorithm or commercial solver behavior is highly unpredictable. For example, we attempt to solve the above 2E-VRPSDP mathematical formulation using the CPLEX 12 solver. We find that even for some moderate-size test instances (e.g., 100 customers and 10-15 satellites), CPLEX cannot find a feasible solution in a reasonable computational time (i.e., 2-3 hours). Thus, in this paper, a variable neighborhood search algorithm (VNS) is designed for the 2E-VRPSDP. The principle of VNS is a systematic change in neighborhoods within a local search procedure. As a local search-based algorithm, the main advantage of VNS is not the computation time. For example, it is slower than some simple *greedy* or *hill-climbing* algorithms. However, it has a great ability to find high-quality solutions, especially for some very complex NP-hard optimization problems. This has been widely proven by solving the VRP as well as its variants (Kytöjoki *et al.* 2007; Fleszar *et al.* 2009; Hemmelmayr *et al.* 2009). For a more thorough description of VNS, refer to Mladenović and Hansen (1997) and Hansen and Mladenović (2001). In this paper, we build a new VNS approach, the outline of which is shown in Algorithm 1. First, the algorithm identifies a set of neighborhood structures N_k (k=1, 2,…,k_{max}) and an initial solution S. Parameter k equals 1. Then, a shaking step is performed by randomly selecting a solution S' of solution S based on the kth neighborhood N_k. Next, the algorithm applies a local search improvement procedure to the current solution S'. This procedure is repeated once a new incumbent solution S'' is found. If this local optimum S'' is better than S, the algorithm sets $S \leftarrow S''$ and searches with k=1. Otherwise, VNS switches to the next neighborhood with k=k+1 to perform another shaking step, followed by local search improvement. Note that our approach extends the search to

infeasible solutions, i.e., both feasible and infeasible solutions are allowed by the VNS. Each solution to the problem corresponds to a set of routes. Infeasibility occurs for a VNS solution if the total demand of a vehicle exceeds the vehicle's specified capacity. The algorithm uses a weighted linear penalty function for violations of this constraint. The details of the implementation of an infeasible solution are presented in the following sections.

Algorithm 1. Variable neighborhood search algorithm

1. Identify neighborhood structures set N_k (k=1,…,k_{max}), penalty parameter values
2. Generate an initial solution S, k=1
3. **while** stop criterion is not reached**, do**
4. **while** $k \neq k_{max}$**, do**
5. generate one random solution S' from S, based on N_k //*shaking*
6. **if** S' is feasible **then**
7. apply local search (restricted in feasible region) to S' and obtain S''
8. **else** // S' is infeasible
9. apply local search, extending to the infeasible region to S' and obtain S''
10. **end if**
11. adjust the penalty parameters
12. **if** S'' is better than S, or the acceptance criterion is satisfied, **then**
13. $S'=S''$, k=1
14. **else**
15. $k = k+1$
16. **end if**
17. **end while**
18. **end while**

5.1 General functioning of the algorithm

Since the 2E-VRPSDP is a rather complicated problem, the VNS approach extends the search space to the infeasible region. It is frequently conjectured that infeasible solutions enable a better transition during the search between structurally different feasible solutions (Cordeau *et al.* 2001). In particular, purposeful management of penalties may enable us to focus the search toward borders of feasibility, a place where high-quality solutions are more likely to be located (Glover and Hao, 2011). Thus, in our VNS, the vehicle capacity constraint is temporarily violated during the search. A weighted and linear penalty function with a penalty weight α is introduced into the cost function for the violation of vehicle

capacity constraints. For a solution S, a general cost function $c(S)$ is defined as $c(S)=d(S)+\alpha\times e(S)$ for solution evaluation, where $d(S)$ denotes the total travel distances of vehicle routes (original objective function), $e(S)$ denotes the total violations of vehicle load of all the routes in three phases, and α is the penalization parameter. Term $e(S)$ is defined as follows:

$$e(S)=\sum_{i\in V_s}\left(load_{i1}-Q_1\right)^+ +\sum_{i\in V_c}\left(load_{i2}-Q_2\right)^+ +\sum_{i\in V_s}\left(load_{i3}-Q_1\right)^+ \tag{35}$$

where $x^+=max(0, x)$, $load_{i1}$ and $load_{i3}$ are the vehicle loads after a *first phase delivery route* and a *third phase pickup route* serves satellite i, respectively; $load_{i2}$ is the load of a vehicle after a *second phase delivery and pickup route* serves customer i. Clearly, if solution S is feasible, i.e., the vehicle capacity is strictly satisfied at each node, $e(S)=0$ and $c(S)=d(S)$; otherwise, $e(S)>0$ and $c(S)>d(S)$. In our VNS penalty parameter, α is adjusted dynamically to facilitate the exploration of the search space. If the solution S'' obtained after shaking and local search is feasible (referred to Algorithm 1), the value of α is divided by a factor of $1+\varphi$ ($\varphi>0$); otherwise, α is multiplied by this factor. This penalization parameter α is initialized with α_0 and is limited between an upper bound α_{max} and a lower bound $\alpha_{min,}$ which limit the maximum and minimum values of this parameter during the search process.

5.2 Construction of an initial solution

Based on the savings algorithm (Clark and Wright 1964), we propose a method to obtain an initial solution to our problem. The initial solution method has three stages.

Stage 1: customer assignment. First, a random sequence of all the customers is generated. Following this sequence, each customer is assigned to its nearest satellite. During this assignment procedure, the algorithm ensures that for each satellite, the total delivery or pickup demands from all associated customers cannot be greater than the first echelon vehicle capacity Q_1. If an assignment of a customer to a satellite violates this constraint, this customer is assigned to its second nearest satellite, and so on until all the customers are assigned to satellites. Because the vehicle capacity is limited, the above procedure may fail to assign each customer to a satellite. In this case, the initial customer sequence is regenerated until all customers have been assigned. Note that in the first step, the random

sequence introduces randomness to the algorithm, similar to the works of Cordeau *et al.* (1997, 2001).

Stage 2: second echelon routing. The algorithm applies the savings algorithm (Clark and Wright 1964) to each satellite to service the customers who have been assigned to this satellite. For satellite *s*, a separate back and forth route for each customer is initially created. For each customer pair *i* and *j* of satellite *s*, the savings is defined as $s_{ij} = c_{si} + c_{js} - c_{ij}$. Starting from the largest and nonnegative saving s_{ij}, two routes (*s*, …, *i*, *s*) and (*s*, *j*, …, *s*) are merged into a single *second phase delivery and pickup route* (*s*, …, *i*, *j*, …, *s*). The above route combinations are executed until the number of *second phase delivery and pickup routes* is no more than K_2 (the total number of second echelon vehicles). Note that the above procedures may construct an infeasible solution that violates the vehicle capacity constraint. For example, part A of Figure 2 shows four routes that start and end at satellites S_1 and S_2. Based on such routes, the saving values of the merging route (S_1, C_1, C_2, S_1) and (S_1, C_3, S_1) and combining (S_2, C_4, S_2) and (S_2, C_5, S_2) are computed. If the former savings value is larger, then the two routes of satellite S_1 are merged, and one larger route (S_1, C_1, C_2, C_3, S_1) is obtained, as shown in part B of Figure 2.

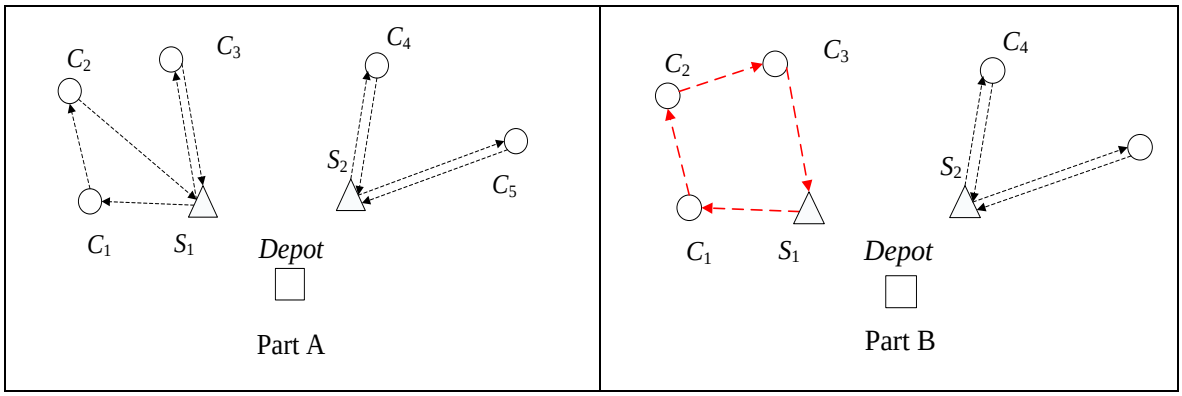

Figure 2. An example of a saving method

Stage 3: first echelon routing. After obtaining the second phase delivery and pickup routes, similar to stage 2, the algorithms apply the savings algorithm to obtain the first phase delivery routes and the third phase pickup routes.

5.3 Shaking

Shaking is the foundation of the VNS, which diversifies the solution and avoids it from being trapped in a local optimum. Each neighborhood should determine a proper balance

between perturbing the incumbent solution and retaining the good parts of the incumbent solution. The algorithm employs ten interroute neighborhood structures (namely, N_1-N_{10}) in VNS shaking. The set of neighborhoods is summarized in Table 1.

Neighborhoods N_1 to N_4 relocate or exchange satellite(s) among two *first phase delivery routes*. Neighborhoods N_5 to N_8 similarly deal with the *third phase pickup routes*. Note that if a satellite is moved, the customers and second phase routes associated with this satellite are moved with this satellite. Figure 3 shows an example of a neighborhood N_1 operator: satellite S_1 is removed from a first phase delivery route (Depot-S_2-S_1-Depot) and inserted into another route (Depot-S_3-Depot).

Table 1. Set of neighborhood structures

Neighborhood	**Operator**
N_1	Randomly choose a satellite from a *first phase delivery route* and relocate it into a random position of another *first phase delivery route*, or construct a new route that only contains this satellite.
N_2	Randomly choose a satellite from a *first phase delivery route* and exchange it with one satellite that is randomly selected from another *first phase delivery route*.
N_3	Randomly choose two sequential satellites from a *first phase delivery route* and relocate them into a random position of another *first echelon delivery route*, or construct a new route that only serves such two satellites.
N_4	Randomly choose two sequential satellites from a *first phase delivery route* and exchange them with two satellites that are randomly chosen from another *first phase delivery route*.
N_5	Same as N_1 but for the *third phase pickup route*.
N_6	Same as N_2 but for the *third phase pickup route*.
N_7	Same as N_3 but for the *third phase pickup route*.
N_8	Same as N_4 but for the *third phase pickup route*.
N_9	Randomly close a satellite. Remove the customers assigned to this satellite one by one; insert each customer into a random position of another *second phase delivery and pickup route*, or construct a new *second phase route* that only contains this customer.
N_{10}	Randomly choose a *second phase delivery and pickup route* and remove a random number of sequential customers (bounded by the customer number of the selected route) and insert them to another *second phase delivery and pickup route*, or construct a new *second phase route* that only contains such customers.

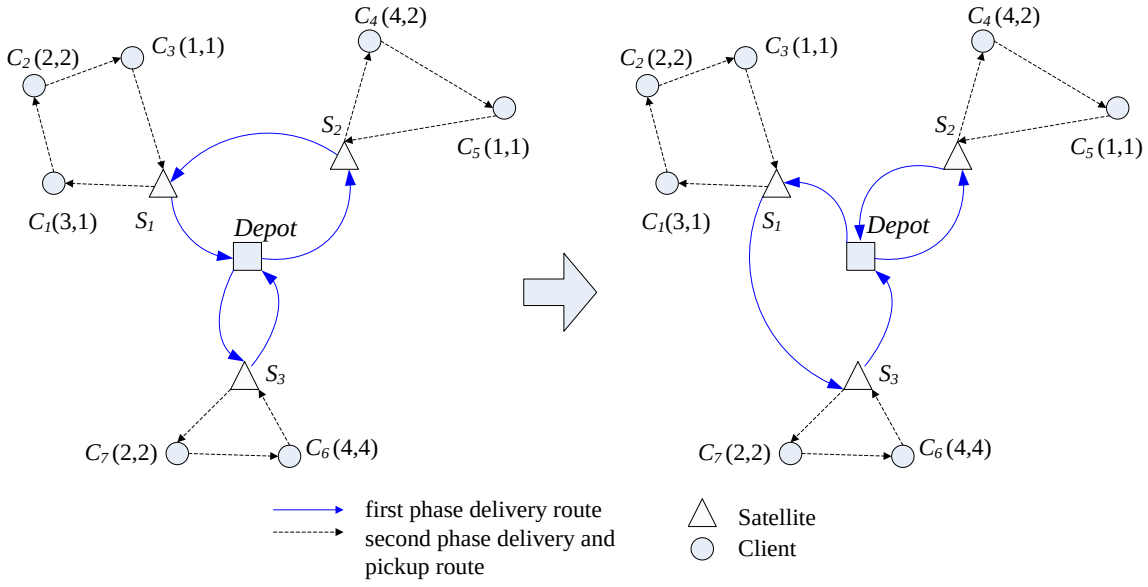

Figure 3. The neighborhood N_1 operator

The last two neighborhoods, N_9 and $N_{10,}$ involve the change in the *second phase delivery and pickup route* and the first echelon routes (pickup route and delivery route), simultaneously. They are more complex than the above neighborhoods. To clarify N_9 and N_{10}, two satellite statuses *close* and *open* are used. In this paper, when one or more customers are assigned to a satellite, the status of this satellite is *open*. For a satellite, when all its customers are assigned to other satellite(s), this satellite becomes *closed*. Neighborhoods N_9 first randomly *close* a satellite. The customers assigned to this satellite are removed from this satellite one by one. The algorithm inserts each removed customer into a random position of another *second phase delivery and pickup route* or constructs a new *second phase delivery and pickup route* that only contains this customer. In terms of constructing a new *second phase route*, it may be built on an *open* satellite or a currently *closed* satellite. Once this new *route is* built on a *closed* satellite, this satellite changes to *open*. Thus, this satellite is simultaneously inserted into a *first phase delivery route* and *third phase pickup route*. For example, in Figure 4, after shaking, satellite S_2 is closed. Customers C_4 and C_5 are assigned to the current open satellite S_1 and the closed satellite S_4, respectively. Thus, satellite S_4 now becomes open and is inserted into a first phase delivery route and a third phase pickup route (Depot-S_3-S_4-Depot).

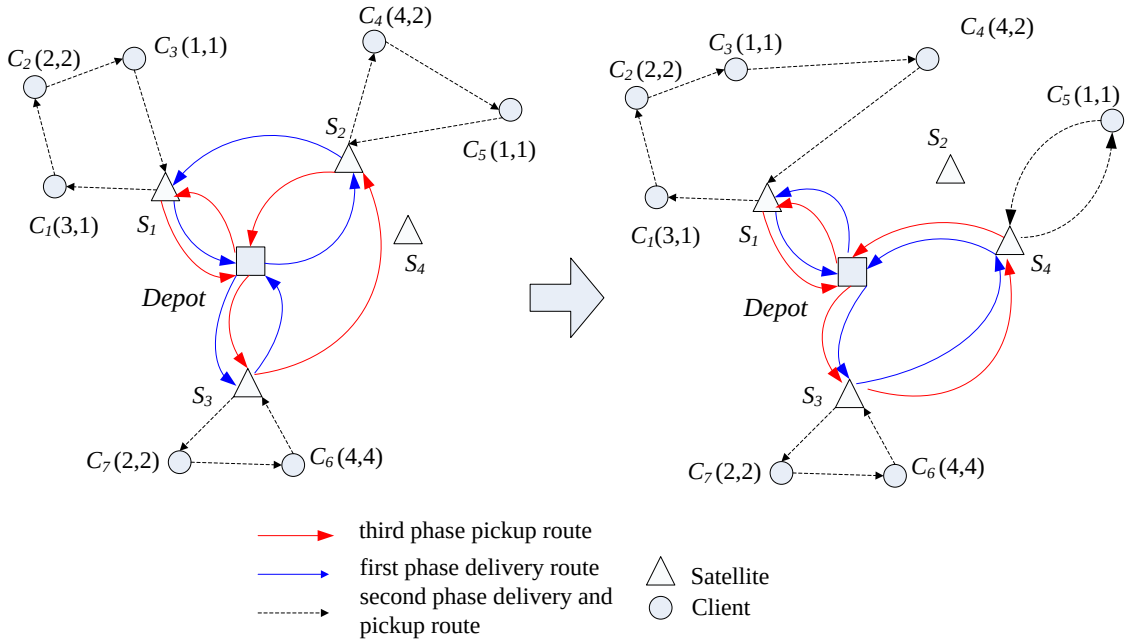

Figure 4. The neighborhood N_9 operator

5.4 Local search

The solution obtained from the shaking procedure needs to be further improved to obtain a local optimum. The local search procedures are successfully hybrid in the metaheuristics. For example, in Prins (2004) and Vidal *et al*. (2012), local search procedures were applied to improve the quality of the offspring solutions in the *genetic algorithms*. Prins (2004) noted that the local search improvement procedure was a key idea of algorithm design. Thus, in our VNS, four kinds of local search methods are used, including interroute and intraroute methods, i.e., interroute 1–0 relocation move, interroute 1–1 exchange move, intraroute 1–0 relocation move and intraroute 1–1 exchange move. Such operators are widely used by VRP studies (Bräysy and Gendreau, 2005; Vidal *et al*. 2012). The interroute 1–0 move refers to relocating a customer from its current position to a position in another route or constructing a new route, while the intraroute 1–0 move denotes relocating a customer from its current position to another position on the same route. Similarly, the interroute 1–1 move involves exchanging two customers' positions of different routes, and the intraroute 1–1 move exchanges two customers' positions on the same route. The four procedures are used for all three vehicle route phases. Note that for the *second phase delivery and pickup route*, the interroute 1–0 move may change the status of satellites. For example, now satellite *j* has only

one customer, and satellite k is closed. When this customer is moved from satellite j to satellite k, satellite j becomes closed, and satellite k is open. During the local search procedure, the algorithm adopts the "first-accept" strategy, i.e., during a local search procedure once a better solution is found, it is adopted as the new seed for repeating this local search. The local search procedure continues until no improvements can be made.

In terms of the classic local search methods, if a local search starts from a feasible solution seed, the algorithm limits the search in a feasible solution area. Otherwise, if it starts from an infeasible seed, the whole feasible and infeasible solution areas are allowed to be searched. In this paper, the algorithm adopts a new local search strategy. If the solution obtained from shaking (the origin of the whole local search) is feasible, then the local search is restricted to the feasible region, which means that the vehicle capacity cannot be exceeded at any move of the local search. Otherwise, if the shaking result is infeasible, the local search is extended to the infeasible area where the generalized cost, including penalties, is used to evaluate each move.

5.5 Acceptance criterion

After the shaking and the local search procedures have been performed, the solution obtained at the current iteration is compared to the incumbent solution to decide whether it is accepted. To prevent the VNS from quickly becoming stuck in a local optimum, inspired by simulated annealing (SA) (Kindervater and Savelsbergh 1997), the scheme of accepting a new but worse solution is adopted. If, after shaking and local search procedures, a new solution S'' that is better than the incumbent solution S is identified, S'' is accepted directly to replace S. Otherwise, this new solution is accepted with a probability of $\exp-\left(c(s'')-c(s)\right)\Big/T$, where $c(S'')$ and $c(S)$ are the generalized costs of two solutions. The annealing temperature T decreases linearly from an initial value T_0, i.e., after each VNS iteration, T is reduced by $T\times(1/ite)$, where ite is the VNS maximum iteration time.

6 Computational results

In this section, computational experiments are conducted to assess the performance of the proposed approach. All algorithms in this paper were coded in C++. The computational

experiments were conducted on an Intel E5–2670 processors clocked at 2.6 GHz with 2 GB of memory running Linux. All the algorithms are run 10 times for each test instance. The best and the average solution costs and the average algorithmic running time are obtained from 10 runs for each test instance.

6.1 Test instances from the 2E-VRPSDP benchmarks

Since there is no benchmark instance for our problem, the 2E-VRP benchmark instances from the literature are modified. Thanks to Breunig *et al*. (2016), the existing 2E-VRP instances were summarized and are available at https://www.univie.ac.at/prolog/research/TwoEVRP. The 2E-VRPSDP test instances are derived from existing 2E-VRP benchmark instances as follows. We select 46 basic 2E-VRP instances in which twenty 2E-VRP instances contain 50 customers, twenty have 100 customers, and the remaining six instances have 200 customers. For each basic 2E-VRP instance, we derive a new instance as follows. We randomly select half of the customers and set 1/3 of the original demand as the delivery demands and 2/3 of the original demand as the number of pickups. For the left half of the customers, we set 1/3 and 2/3 of the original demand as the pickup and delivery demands, respectively. Other parameters, including locations of all points, vehicle number limits, and vehicle capacity, are the same as those of the original 2E-VRP instance. Since there are only six 2E-VRP instances of 200 customers available in the literature, we change the vehicle capacity and the vehicle number of an echelon and derive another fourteen 2E-VRPSDP instances with 200 customers (with signs of c-g in Table 8). In total, we have sixty 2E-VRPSDP instances for computational experiments. The original 2E-VRP benchmark set, which each of our new 2E-VRPSDP instances comes from, is marked at the beginning of the instance name, and "SDP" is added at the end. For example, the new instance generated based on "E-n51-k5-s2-17" (2E-VRP benchmark, Set 2b) is named "Set2b_E-n51-k5-s2-17_SDP".

6.2 Parameter tuning

Based on the above test data, we select 18 2E-VRPSDP test instances to tune the parameters in our algorithm, consisting of six small instances with 50 customers, six medium instances with 100 customers, and six large instances with 200 customers. Each instance is solved by the algorithm with a one-parameter setting ten times. We obtain ten computational

results for each test instance, including the ***best solution cost*** and ***mean solution cost*** among the 10 running results. Then, we calculate the average of all ***best solution costs*** and the average of the ***mean solution cost*** across all 18 test instances. We use these two statistical results to assess the solution quality of such parameter settings. Table 2 summarizes the final parameter settings of our VNS algorithm used in the experiments. In the following subsections, some parameters are tuned, and a sensitivity analysis is executed to elucidate the effects of the components of the proposed algorithm.

Table 2. Parameter setting in the experiment.

Symbol	Explanation	Value
T_0	The initial temperature in the acceptance criterion	100
α_0, α_{min}, α_{max}	Dynamic penalization parameter in $c(s)$	α_0: 1 α_{max}: 10,000 α_{min}: 0.001
φ	Parameter used to adjust penalization parameter α	1.05
ite	The maximum number of VNS iterations	10,000

6.2.1 Parameter T_0

Parameter T_0 is the initial temperature of the simulated annealing-based acceptance criterion (section 4.5). Clearly, this parameter has an important effect on the VNS, which controls the probability of accepting a relatively worse solution in the algorithm iterations. We test parameter T_0 on the set of {50, 75, 100, 125, 150, 200} while using the values in Table 2 for other algorithmic parameters. The results are summarized in Table 3. In this table, row "*Avg-Best*" shows the average value of all *best solution costs* to a total of 18 test instances, and row "*Avg-Mean*" gives the average of all *mean solution costs*. We find that the solution quality improves with the increase in parameters from a minimum value of 50 and peaks at a value of 100, and then the accuracy of the algorithm decreases with the value of this parameter.

Table 3. Sensitivity of the algorithm performances to T_0

T_0	50	75	100	125	150	200
Avg-Best	1,089.98	1,091.59	**1,088.00**	1,088.52	1,090.65	1,090.56
Avg-Mean	1,118.27	1,117.41	**1,116.44**	1,117.68	1,123.37	1,119.24

6.2.2 Parameter φ

In the VNS, for each solution S, an extended cost $c(S)$ is defined as $c(S)=d(S)+\alpha \times e(S)$ for solution evaluation (see section 5.1). At each VNS iteration, when a solution S'' is obtained after shaking and local search, the algorithm judges its feasibility. If this solution is feasible, the value of α is divided by $1+\varphi$; otherwise, α is multiplied by $1+\varphi$. Clearly, this parameter φ controls the change frequency of parameter α and the frequency at which the algorithm jumps between feasible and infeasible solution areas. We test φ in the set of {1.01, 1.05, 1.1, 1.2, 1.3, 1.5}. The results are shown in Table 4. The best algorithm performance is yielded at φ equal to 1.05.

Table 4. Sensitivity of the algorithm performances to φ

φ	1.01	1.05	1.1	1.2	1.3	1.5
Avg-Best	1,088.94	**1,088.00**	1,110.57	1,099.20	1,101.57	1,101.52
Avg-Mean	1,112.68	**1,116.44**	1,136.89	1,125.63	1,130.40	1,137.72

6.3 Sensitivity analysis of algorithmic components

Furthermore, a set of experiments is designed to elucidate the effects of VNS algorithm components. We attempt to remove one or a set of components from the algorithm, and the remaining algorithm's performance is explored. Based on these experiments, the roles of the algorithm components are tested and verified.

The first is the ***shaking*** component (see section 4.3). Ten interroute neighborhood structures (namely, N_1-N_{10}) are adopted in VNS as *shaking*. In this part of the experiments, such neighborhood structures are classified into four sets. The first set contains N_1, N_3, N_5 and N_7, which ***relocate*** a satellite to another position; the second set consists of N_2, N_4, N_6, and N_8, which ***exchange*** the positions of two satellites. The third is N_9, which ***closes*** a satellite, and the final fourth is N_{10}, which ***removes*** a number of sequential customers and inserts them into another route. We sequentially delete one set of neighborhood structures from the whole algorithm and yield four versions of algorithms, i.e., "***no-relocation***", "***no-exchange***", "***no-close***", and "***no-remove***". We also aim to test the component of ***the local search*** (section 4.4). The solution obtained from the shaking procedure is further improved by a set of ***local search methods***. The first set of local search methods is **1–0 *relocation,*** and the second is **1–1 *exchange*** (details are referenced in section 4.4). Again,

each time one set of local search methods is shielded, the remaining algorithms "***no*-1–0 *LS***" and "***no*-1–1 *LS***" are tested. Each version of the modified algorithm is executed 10 times over each instance. The results are shown in Table 5. The results show that all these algorithmic components enhance the performance of the algorithm because "***Whole VNS***" yields the best results. Therefore, it is concluded that the algorithm performance will be compromised if one component is deleted from the algorithm.

Table 5. Sensitivity analysis of algorithmic components

	No-relocation	***No-exchange***	***No-close***	***No-remove***	***No-1–0 LS***	***No-1–1 LS***	***Whole VNS***
Avg-Best	1,092.99	1,089.23	1,094.77	1,115.30	1,158.40	1,100.32	**1,088.00**
Avg-Mean	1,122.07	1,118.60	1,137.93	1,148.26	1,216.62	1,144.40	**1,116.44**

Furthermore, in terms of each of the above operators, we also want to know how many times it is used, or whether it truly improves the solution during the VNS iteration process. For example, considering four types of *Shaking* operators (***relocation***, ***exchange***, ***close***, and ***remove***), we record the mean execution times of each operator during VNS iterations. They contribute 12.7%, 10.7%, 37.2%, and 39.4% of shaking executions, respectively. This statistical result shows that each shaking operator is executed and has a considerable positive effect on the algorithm performance. We also record the success rate of two local search methods **1–0 *relocation*** and **1–1 *exchange*** (i.e., the number of times that one *LS* operator improves the solution during VNS whole iterations, over the total improvement times of all *LS* operations). It is found that 1–0 *relocation* contributes approximately 53% local search improvement, whereas 1–1 *exchang*e contributes 47% improvement. This result also indicates that two local search methods are necessary in the algorithm. Furthermore, we find that the most running time of VNS is used by local search procedures, and they account for approximately 90% of the running time of the whole algorithm. The remaining time is mainly used by shaking operators. Furthermore, the computational time of each shaking operator accounts for a similar percentage of the total shaking time as their executions (i.e., approximately 10%, 10%, 35%, and 45%). Similarly, each local search operator's running times also account for a similar percentage of the total consumption time as their contributions stated above.

6.4 Computational results for 2E-VRPSDP instances

We cannot find other algorithms for the 2E-VRPSDP, so in this paper, we first develop a simplified version of our approach, named VNS1, to test the design and performance of the proposed VNS. In the VNS1 approach, the basic framework of the algorithm is the same as that of the above approach, including the construction of an initial solution, shaking, local search and the acceptance criterion. However, the search space of VNS1 is always restricted to the feasible region, which means that in the shaking and local search procedure, the vehicle capacity constraints are always obeyed. Thus, the solution obtained at any iteration of VNS1 is always feasible, and its penalty cost equals zero. Such a version of VNS is more classic and standard. The VNS approach proposed above (extending the algorithm to an infeasible solution area) is denoted as VNS2. To give a fair comparison between two approaches, VNS1 and VNS2, for each test instance, we change the stop criterion of algorithm VNS1, i.e., we do not use maximum iterations to terminate VNS1; it is stopped after a special total running time that equals the computational time of VNS2. That is, the two approaches have the same running times for solving each test instance.

We tested a total of sixty 2E-VRPSDP instances on VNS1 and VNS2. Tables 6 – 8 show the computational results for test instances of different sizes. Columns "$|V_c|$" and "$|V_s|$" represent the numbers of customers and satellites, respectively. "K_1" and "K_2" are the numbers of available first echelon and second echelon vehicles. Columns "*Avg*" and "*Best*" give the average solution cost and best solution cost over 10 runs of the corresponding algorithm, respectively. The average running time over 10 runs in seconds is given in column "*t*". Two columns, "Gap_1" and "Gap_2", present the percentage gaps of average solution costs and best solution costs between VNS1 and VNS2.

As shown in Tables 6–8, for all instances, VNS2 yields better solutions from the perspective of solution cost (quality). For all test instances, VNS2 can find a better solution than VNS1. The deviations between the *average solution costs* of VNS1 and VNS2 are 17.76%, 8.17% and 5.05% for 50-customer, 100-customer, and 200-customer instances, respectively. Concerning the *best solution costs*, VNS2 deviates from VNS1 by 16.09%, 6.30% and 6.45%, respectively.

In addition to such apparent statistical data, we also use the ***t-test*** method to check and compare two algorithms, VNS1 and VNS2. The *t-test* is well known and commonly used to

determine if the means of two sets of data are significantly different from each other. First, we compare the mean of "*average solution costs*" and "*best solution costs*" for all test instances between the two algorithms. Clearly, VNS2 is much better than VNS1; for example, the means of the "*best solution costs*" of VNS1 and VNS2 are 1,171.53 and 1,050.98, respectively. Next, we compute the *p*-value on the basis of Tables 6–8. For both the "*average solution costs*" and "*best solution costs*", the *p*-values are 0.000, which indicates that the performances of the two algorithms are significantly different. Therefore, it can be asserted that compared with the classic VNS approach, the extension to the infeasible solution area increases the algorithmic ability to find high-quality solutions.

Table 6. Computational results of 2E-VRPSDP instances with 50 customers

Instance	$\|V_c\|$	$\|V_s\|$	K_1	K_2	VNS1		VNS2			Gap_1	Gap_2
					Avg.	*Best*	*Avg.*	*Best*	*t*		
Set 2b_E-n51-k5-s2-17_SDP	50	2	3	5	610.87	610.87	599.63	590.76	84.5	1.84%	3.29%
Set2b_E-n51-k5-s11-19_SDP	50	2	3	5	641.23	640.68	626.07	621.40	99.7	2.36%	3.01%
Set2b_E-n51-k5-s27-47_SDP	50	2	3	5	540.04	538.61	522.49	515.14	87.7	3.25%	4.36%
Set3_E-n51-k5-s12-18_SDP	50	2	3	5	743.08	743.08	715.51	712.25	73.6	3.71%	4.15%
Set3_E-n51-k5-s12-43_SDP	50	2	3	5	844.47	844.47	828.35	814.41	99.4	1.91%	3.56%
Set3_E-n51-k5-s39-41_SDP	50	2	3	5	788.61	784.46	753.66	751.52	89.9	4.43%	4.20%
Set4b_Instance50-22_SDP	50	3	3	6	1,482.96	1,482.96	1,211.88	1,089.85	21.8	18.28%	26.51%
Set4b_Instance50-24_SDP	50	3	3	6	1,410.54	1,396.31	1,096.61	1,080.84	28.8	22.26%	22.59%
Set4b_Instance50-26_SDP	50	3	3	6	1,382.81	1,338.97	1,030.33	1,030.30	34.3	25.49%	23.05%
Set4b_Instance50-28_SDP	50	3	3	6	1,394.47	1,394.47	978.51	978.18	32.6	29.83%	29.85%
Set4b_Instance50-31_SDP	50	3	3	6	1,594.25	1,525.97	1,454.63	1,450.94	55.7	8.76%	4.92%
Set4b_Instance50-33_SDP	50	3	3	6	1,658.05	1,574.37	1,529.63	1,529.49	61.3	7.75%	2.85%
Set4b_Instance50-38_SDP	50	5	3	6	1,422.85	1,353.08	1,013.03	1,007.91	35.7	28.80%	25.51%
Set4b_Instance50-40_SDP	50	5	3	6	1,437.06	1,412.89	969.74	958.43	34.8	32.52%	32.17%
Set4b_Instance50-42_SDP	50	5	3	6	1,424.63	1,310.02	1,096.80	1,054.65	34.8	23.01%	19.49%
Set4b_Instance50-44_SDP	50	5	3	6	1,285.33	1,073.38	916.00	915.36	37.9	28.73%	14.72%
Set4b_Instance50-46_SDP	50	5	3	6	1,335.16	1,270.52	891.85	887.46	39.1	33.20%	30.15%
Set4b_Instance50-48_SDP	50	5	3	6	1,316.42	1,149.74	966.92	965.11	36.6	26.55%	16.06%
Set4b_Instance50-50_SDP	50	5	3	6	1,284.08	1,284.08	965.91	962.51	36.9	24.78%	25.04%
Set4b_Instance50-54_SDP	50	5	3	6	1,405.54	1,375.91	1,014.31	1,013.87	36.1	27.83%	26.31%
Average					1,200.12	1,155.24	959.09	946.52	53.1	17.76%	16.09%

Table 7. Computational results of 2E-VRPSDP instances with 100 customers

Instance	$\|V_c\|$	$\|V_s\|$	K_1	K_2	VNS1		VNS2				
					Avg.	*Best*	*Avg.*	*Best*	*t*	Gap_1	Gap_2
Set5_100-5-1_SDP	100	5	5	32	1,388.98	1,362.28	1,346.64	1,331.34	313.0	3.05%	2.27%
Set5_100-5-1b_SDP	100	5	5	15	1,162.59	1,122.32	1,071.38	1,064.86	630.5	7.85%	5.12%
Set5_100-5-2_SDP	100	5	5	32	1,031.69	1,012.28	999.33	989.80	217.7	3.14%	2.22%
Set5_100-5-2b_SDP	100	5	5	15	906.62	890.01	855.11	840.19	438.9	5.68%	5.60%
Set5_100-5-3_SDP	100	5	5	30	1,115.22	1,084.87	1,058.53	1,040.96	247.0	5.08%	4.05%
Set5_100-5-3b_SDP	100	5	5	16	967.23	911.45	885.12	877.96	473.1	8.49%	3.67%
Set5_100-10-1_SDP	100	10	5	35	1,174.91	1,135.57	1,112.02	1,091.35	161.6	5.35%	3.89%
Set5_100-10-1b_SDP	100	10	5	18	975.81	904.16	859.48	856.34	298.5	11.92%	5.29%
Set5_100-10-2_SDP	100	10	5	33	1,016.03	995.01	996.10	985.22	221.1	1.96%	0.98%
Set5_100-10-2b_SDP	100	10	5	18	870.18	845.98	823.62	818.41	345.8	5.35%	3.26%
Set5_100-10-3_SDP	100	10	5	32	1,127.46	1,110.11	1,100.96	1,083.48	186.2	2.35%	2.40%
Set5_100-10-3b_SDP	100	10	5	17	965.24	932.42	904.91	893.04	336.2	6.25%	4.22%
Set6a_A-n101-4_SDP	100	4	4	100	1,225.32	1,224.96	1,100.60	1,088.05	553.8	10.18%	11.18%
Set6a_A-n101-5_SDP	100	5	4	100	1,252.28	1,230.04	1,122.67	1,088.09	380.9	10.35%	11.54%
Set6a_A-n101-6_SDP	100	6	4	100	1,237.39	1,218.93	1,048.57	1,031.64	340.9	15.26%	15.37%
Set6a_B-n101-4_SDP	100	4	4	100	968.77	965.38	870.47	863.88	386.0	10.15%	10.51%
Set6a_B-n101-5_SDP	100	5	4	100	986.94	939.25	866.54	858.00	519.6	12.20%	8.65%
Set6a_B-n101-6_SDP	100	6	4	100	984.35	927.81	901.47	892.01	342.4	8.42%	3.86%
Set6a_C-n101-4_SDP	100	4	4	100	1,363.23	1,359.61	1,128.70	1,123.01	422.0	17.20%	17.40%
Set6a_C-n101-5_SDP	100	5	4	100	1,301.93	1,174.62	1,129.99	1,121.21	847.0	13.21%	4.55%
Average					1,101.11	1,067.35	1,009.11	996.94	383.1	8.17%	6.30%

Table 8. Computational results of 2E-VRPSDP instances with 200 customers

Instance	$\|V_c\|$	$\|V_s\|$	K_1	K_2	VNS1		VNS2				
					Avg.	*Best*	*Avg.*	*Best*	*t*	Gap_1	Gap_2
Set5_200-10-1_SDP	200	10	5	62	1,485.71	1,464.67	1,470.23	1,447.40	1,060.1	1.04%	1.18%
Set5_200-10-1b_SDP	200	10	5	30	1,304.35	1,293.35	1,241.60	1,180.06	1,945.9	4.81%	8.76%
Set5_200-10-1c_SDP	200	10	5	45	1,389.84	1,372.55	1,346.91	1,318.51	1,377.2	3.09%	3.94%
Set5_200-10-1d_SDP	200	10	5	25	1,279.55	1,247.54	1,154.58	1,099.05	3,112.4	9.77%	11.90%
Set5_200-10-1e_SDP	200	10	5	21	1,248.33	1,224.87	1,089.97	1,050.84	3,803.4	12.69%	14.21%
Set5_200-10-1f_SDP	200	10	4	62	1,363.02	1,348.50	1,328.53	1,305.29	1,202.9	2.53%	3.20%
Set5_200-10-1g_SDP	200	10	8	62	1,670.99	1,643.20	1,608.19	1,522.89	1,152.4	3.76%	7.32%
Set5_200-10-2_SDP	200	10	5	63	1,310.52	1,308.99	1,276.79	1,261.41	1,291.2	2.57%	3.63%
Set5_200-10-2b_SDP	200	10	5	30	1,077.04	1,060.85	1,037.78	1,015.25	1,907.4	3.65%	4.30%
Set5_200-10-2c_SDP	200	10	5	45	1,359.60	1,327.35	1,100.86	1,078.91	1,286.0	19.03%	18.72%
Set5_200-10-2d_SDP	200	10	5	25	1,060.35	1,058.78	1,013.19	986.16	2,043.2	4.45%	6.86%
Set5_200-10-2e_SDP	200	10	5	21	1,020.28	1,012.84	973.54	935.60	2,767.0	4.58%	7.63%
Set5_200-10-2f_SDP	200	10	4	63	1,160.55	1,160.50	1,150.25	1,139.96	1,330.6	0.89%	1.77%
Set5_200-10-2g_SDP	200	10	8	63	1,482.33	1,475.15	1,451.03	1,378.51	1,480.6	2.11%	6.55%
Set5_200-10-3_SDP	200	10	5	63	1,603.36	1,599.81	1,550.60	1,523.34	1,099.5	3.29%	4.78%
Set5_200-10-3b_SDP	200	10	5	30	1,223.05	1,198.71	1,174.16	1,145.79	2,585.6	4.00%	4.41%
Set5_200-10-3c_SDP	200	10	5	45	1,356.74	1,335.98	1,303.40	1,270.18	1,670.0	3.93%	4.93%
Set5_200-10-3d_SDP	200	10	5	26	1,192.60	1,164.66	1,126.08	1,090.43	2,871.6	5.58%	6.37%
Set5_200-10-3e_SDP	200	10	5	21	1,148.98	1,109.38	1,071.24	1,038.90	3,678.1	6.77%	6.35%
Set5_200-10-3f_SDP	200	10	4	63	1,454.36	1,432.03	1,419.56	1,401.23	1,102.9	2.39%	2.15%
Average					1,309.58	1,291.99	1,244.42	1,209.49	1,938.4	5.05%	6.45%

6.5 Computational results for 2E-VRP benchmarks

To further assess the proposed VNS approach, we tested it on existing 2E-VRP benchmark instances from the literature. The 2E-VRP can be regarded as a special 2E-VRPSDP when the pickup demand of each customer of the 2E-VRPSDP is zero. In such cases, the second phase delivery and pickup route is reduced to the second phase delivery route; the third phase pickup routes do not exist. Note that the classic 2E-VRP assumes that each satellite can be visited by more than one first phase delivery route; however, in the 2E-VRPSDP, a satellite is limited to service from no more than one vehicle. Thus, we select twenty 2E-VRP benchmark instances in which the *best-known solutions* (*BKS*) of the 2E-VRP from the literature (Breunig *et al.* 2016, and Mühlbauer *et al.* 2021) do not assign more than one vehicle to visit a satellite. To fully compare with other 2E-VRP methods, the results of the following existing state-of-the-art algorithms are presented, i.e., the large neighborhood search-based heuristic (LNS-2E) of Breunig *et al*. (2016), the adaptive large neighborhood search heuristic (ALNS) of Hemmelmayr *et al*. (2012), the hybrid heuristic (GRASP+VND) from Zeng *et al*. (2014), and the parallelized large neighborhood search heuristic (PLNS) from Mühlbauer *et al.* (2021) when applicable. The LNS-2E of Breunig *et al*. (2016) is insensitive to its parameters, which are presented in Table 2 of Breunig *et al*. (2016). The numerical experiments of Breunig *et al*. (2016) are executed on an Intel E5–2670v2 CPU at 2.5 GHz with 3 GB RAM. The parameters of ALNS in Hemmelmayr *et al*. (2012) are detailed in their section 6.2, and the ALNS algorithm is carried out on a 2.2 GHz AMD Opteron 275 Processor. The algorithm of Zeng *et al*. (2014) is carried out using a single core of an Intel Pentium Dual-Core E5500 processor 2.8 GHz and 2 GB of memory. The PLNS in Mühlbauer *et al*. (2021) is performed on a computer with 8G RAM and an Intel i5–6200 processor with 2.4 GHz, two cores and four threads. The PLNS is a parallelized algorithm, which is executed on the four threads in parallel. In addition to the results of the above algorithms, we also compare our results with the current *BKS* of each test instance from the literature.

Table 9 reports and compares detailed computational results on the existing

2E-VRP benchmark instances. In addition to the columns labeled the same as those in Tables 6-8, column "*BKS*" refers to the best-known solution of the instance from the existing literature. We highlight a BKS with an asterisk if the best-known solution of the instance is known to be optimal from previous literature. For other approaches (LNS-2E, ALNS, GRASP+VND, PLNS), subcolumn "*Best*" gives the best objective value found within their five runs; column "*Avg*" shows the average objective value of these five runs. Solutions are in bold when they are equal to the *BKS* and underlined when improving the best-known solution. Note that Zeng *et al*. (2014) only provided the average running time (in seconds) to find the best solutions (not the whole algorithm running time) in their paper. Therefore, for their results, Table 9 gives such times, whereas for other approaches, the whole algorithm running times (average run one time, in seconds) for solving each instance are shown.

First, we compare the solution quality of different algorithms. As shown in Table 9, for the first ten instances that contain no more than 50 customers, the first four approaches can find BKS. The VNS algorithm of this paper and the LNS-2E of Breunig *et al*. (2016) have better performances since they can obtain BKS for every algorithm running, i.e., the average solution (column *Avg*) equals the best solution (column *Best*). Algorithm PLNS in Mühlbauer *et al*. (2021) fails to find the BKS for three test instances (Instance50–2, Instance50–37, and Instance50–49).

Considering 10 larger-scale test instances (no less than 100 customers), the VNS of this paper finds 5 existing BKS; LNS-2E and PLNS both obtain 3 existing BKS. Compared with LNS-2E, VNS can find 6 better solutions, and LNS-2E can only find one better solution than VNS. Compared with PLNS, VNS finds 3 better solutions (100–5-1, 100–5-1b, and B-n101–5), and for 3 test instances, PLNS obtains better solutions (200–10–1, 200–10–1b, and 200–10–2). These results indicate that for such large test instances, VNS is better than LNS-2E, and the two algorithms VNS and PLNS are similar in solution quality.

Additionally, Table 10 shows the best objective value found by each approach during all experiments, including those used for parameter calibration. We observe

that for 15 out of a total of 20 test instances, the VNS can find the *BKS* from the literature. The BKS found by each other algorithm is much less than VNS. Such numerical results indicate that the VNS can obtain high-quality solutions for the two-echelon vehicle routing problem with simultaneous delivery and pickup.

In addition to such data, we further use the *t-test* method to check the differences among VNS and other algorithms. In terms of the "*average solution costs*" and "*best solution costs*", the *p*-values between VNS and every other algorithm are calculated, as shown in Table 11. We find that the *p*-values between VNS and *LNS-2E*, *ALNS*, and *GRASP+VND* are small, whereas the *p*-values between VNS and *PLNS* are relatively larger (0.715 and 0.442). Such results prove that our VNS is superior to the *LNS-2E*, *ALNS*, and *GRASP+VND* algorithms. Comparing VNS with *PLNS*, the difference in solution quality is not clear.

In addition to the solution quality, in terms of runtime for such test instances, as shown in Table 9, the mean running time of our VNS is shorter than that of *LNS-2E* and *ALNS*. For example, the mean computational times of *VNS* and *LNS-2E* over all test instances are 284 s and 480 s, respectively. Note that the total runtime of *GRASP+VND* is not given in the literature; its running time shown in Table 9 is only the running time to find the best solutions (not the whole algorithm running time). We also found that the running time of *PLNS* is shorter than that of our VNS, especially for large-scale test instances with 100+ customers. However, please note that the *PLNS* is a parallelized metaheuristic, and our VNS is not designed as a parallel algorithm. The parallelized technique can greatly reduce the running time of *PLNS*. In summary, compared with existing state-of-art approaches, the VNS in this paper does not have an obvious superiority in computation time. However, its running time is acceptable and reasonable for all these test instances.

Table 9. Computational results of instances from 2E-VRP benchmarks

Instance			*VNS*			LNS-2E			ALNS			GRASP+VND			*PLNS*			*BKS*
	$\|V_c\|$	$\|V_s\|$	*Avg*	*Best*	*t*	*Avg*	*Best*	*t*	*Avg*	*Best*	*t*	*Avg*	*Best*	*t*	*Avg*	*Best*	*t*	
E-n51-k5-s6-12-32-37	50	4	531.92	**531.92**	44	531.92	**531.92**	60	531.92	**531.92**	150	531.92	**531.92**	1	531.92	**531.92**	1.9	531.92*
E-n51-k5-s11-19-27-47	50	4	527.63	**527.63**	46	527.63	**527.63**	60	527.63	**527.63**	147	527.63	**527.63**	1	527.63	**527.63**	2.0	527.63*
Instance50-2	50	2	1,438.32	**1,438.32**	69	1,438.32	**1,438.32**	60	1,441.02	**1,438.33**	155	1,438.33	**1,438.33**	40	1,452.30	1,445.05	4.9	1,438.33*
Instance50-4	50	2	1,424.04	**1,424.04**	75	1,424.04	**1,424.04**	60	1,424.04	**1,424.04**	130	1,429.04	**1,424.04**	98	1,443.83	**1,424.04**	3.9	1,424.04*
Instance50-37	50	5	1,528.73	**1,528.73**	36	1,528.73	**1,528.73**	60	1,528.81	**1,528.73**	143	1,528.98	**1,528.73**	58	1,548.94	1,530.65	5.1	1,528.73*
Instance50-38	50	5	1,163.07	**1,163.07**	62	1,163.07	**1,163.07**	60	1,163.07	**1,163.07**	88	1,163.07	**1,163.07**	2	1,164.87	**1,163.07**	3.6	1,163.07*
Instance50-39	50	5	1,520.92	**1,520.92**	63	1,520.92	**1,520.92**	60	1,520.92	**1,520.92**	158	1,520.92	**1,520.92**	21	1,521.78	**1,520.92**	4.4	1,520.92*
Instance50-42	50	5	1,190.17	**1,190.17**	37	1,190.17	**1,190.17**	60	1,190.17	**1,190.17**	95	1,190.17	**1,190.17**	95	1,191.66	**1,190.17**	3.6	1,190.17*
Instance50-46	50	5	1,058.10	**1,058.10**	60	1,058.10	**1,058.10**	60	1,058.97	**1,058.10**	74	1,058.10	**1,058.10**	9	1,058.97	**1,058.10**	3.6	1,058.10*
Instance50-49	50	5	1,434.88	**1,434.88**	36	1,434.88	**1,434.88**	60	1,435.28	**1,434.88**	140	1,434.88	**1,434.88**	51	1,450.94	1,450.94	5.0	1,434.88*
100-5-1	100	5	1,567.46	**1,564.46**	362	1,566.87	**1,564.46**	900	1,588.73	**1,564.46**	353	-	-	-	1,572.15	1,568.42	63	1,564.46*
100-5-1b	100	5	1,110.90	1,099.84	416	1,111.93	1,108.62	900	1,126.93	1,111.34	397	-	-	-	1,112.27	1,108.77	73	1,099.35*
200-10-1	200	10	1,593.77	1,563.41	991	1,598.46	1,580.34	900	1,626.83	1,574.12	888	-	-	-	1,538.26	1,537.95	227	1,537.52*
200-10-1b	200	10	1,218.46	1,187.62	1,114	1,217.23	1,191.59	900	1,239.79	1,201.75	692	-	-	-	1,174.88	1,174.64	215	1,173.07*
200-10-2	200	10	1,399.75	1,367.17	958	1,406.16	1,366.36	900	1,416.87	1,374.74	1072	-	-	-	1,354.55	1,353.22	306	1,352.87*
A-n101-5	100	5	1,215.87	**1,211.38**	323	1,215.89	1,211.40	900	-	-	-	-	-	-	1,216.27	1,211.35	76	1,211.38*
A-n101-6	100	6	1,164.00	1,155.96	331	1,161.91	1,158.97	900	-	-	-	-	-	-	1,156.83	1,155.96	97	1,155.89*
B-n101-4	100	4	939.58	**939.21**	212	939.79	**939.21**	900	-	-	-	-	-	-	939.42	**939.21**	58	939.21*
B-n101-5	100	5	971.65	**967.82**	200	971.27	969.13	900	-	-	-	-	-	-	970.72	968.38	72	967.82*
B-n101-6	100	6	961.97	**960.29**	241	961.91	**960.29**	900	-	-	-	-	-	-	960.48	**960.29**	77	960.29*

Table 10. Best found solutions for the 2E-VRP

Instance	$\|V_c\|$	$\|V_s\|$	*VNS*	*LNS-2E*	*ALNS*	*GRASP+VND*	*PLNS*	*BKS*
E-n51-k5-s6-12-32-37	50	4	**531.92**	**531.92**	**531.92**	**531.92**	**531.92**	531.92*
E-n51-k5-s11-19-27-47	50	4	**527.63**	**527.63**	**527.63**	**527.63**	**527.63**	527.63*
Instance50-2	50	2	**1,438.32**	**1,438.32**	**1,438.32**	**1,438.33**	1,445.05	1,438.33*
Instance50-4	50	2	**1,424.04**	**1,424.04**	**1,424.04**	**1,424.04**	**1,424.04**	1,424.04*
Instance50-37	50	5	**1,528.73**	**1,528.73**	**1,528.73**	**1,528.73**	1,530.65	1,528.73*
Instance50-38	50	5	**1,163.07**	**1,163.07**	**1,163.07**	**1,163.07**	**1,163.07**	1,163.07*
Instance50-39	50	5	**1,520.92**	**1,520.92**	**1,520.92**	**1,520.92**	**1,520.92**	1,520.92*
Instance50-42	50	5	**1,190.17**	**1,190.17**	**1,190.17**	**1,190.17**	**1,190.17**	1,190.17*
Instance50-46	50	5	**1,058.11**	**1,058.11**	**1,058.11**	**1,058.10**	**1,058.10**	1,058.10*
Instance50-49	50	5	**1,434.88**	**1,434.88**	**1,434.88**	**1,434.88**	1,450.94	1,434.88*
100-5-1	100	5	**1,564.46**	**1,564.46**	**1,564.46**	-	1,568.42	1,564.46*
100-5-1b	100	5	1,099.84	1,108.62	1,111.34	-	1,108.77	1,099.35*
200-10-1	200	10	1,556.79	1,556.79	1,574.12	-	1,537.95	1,537.52*
200-10-1b	200	10	1,187.62	1,187.62	1,201.75	-	1,174.64	1,173.07*
200-10-2	200	10	1,365.74	1,365.74	1,374.74	-	1,353.22	1,352.87*
A-n101-5	100	5	**1,211.38**	1,211.40	-	-	1,211.35	1,211.38*
A-n101-6	100	6	1,155.96	1,155.96	-	-	1,155.96	1,155.89*
B-n101-4	100	4	**939.21**	**939.21**	-	-	**939.21**	939.21*
B-n101-5	100	5	**967.82**	969.13	-	-	968.38	967.82*
B-n101-6	100	6	**960.29**	**960.29**	-	-	**960.29**	960.29*

Table 11. *p*-values between VNS and the other algorithms

Algorithms	*P*-value of the "*best solution costs*"	*P*-value of the "*average solution costs*"
VNS *VS.* **LNS-2E**	0.093	0.353
VNS *VS.* **ALNS**	0.041	0.020
VNS *VS.* **GRASP+VND**	0.343	0.318
VNS *VS.* **PLNS**	0.715	0.442

7 Conclusions and future work

In this paper, a new two-echelon transportation problem was investigated, the *two-echelon vehicle* routing *problem with simultaneous deliveries and pickups*

(2E-VRPSDP). The 2E-VRPSDP considers the distribution and collection of freight on two echelons. It is more complex than the classic *two-echelon vehicle routing problem* (2E-VRP) and the *vehicle routing problem with simultaneous deliveries and pickups* (VRPSDP). To address this challenging problem, a VNS algorithm was developed, which extends the search space into the infeasible region and strikes a balance between infeasible solutions and feasible solutions by adding a linear penalty cost function. To prevent the algorithm from becoming stuck in a local optimum, an acceptance criterion inspired by simulated annealing was embedded into the proposed method. To examine the algorithmic performances, new 2E-VRPSDP test instances were generated based on the existing 2E-VRP benchmarks. Our approach was compared with its simplified version, which restricts the search in the feasible region. The results show the priority of the proposed method in obtaining higher-quality solutions. The proposed approach was also compared with existing state-of-the-art algorithms on the 2E-VRP benchmarks from the literature. The numerical results prove the efficiency and effectiveness of the proposed approach.

The work presented in this paper can be extended in two directions. The first is that uncertain elements can be added to the current framework, such as customers with random delivery and pickup demands. Stochastic demands may make the problem more similar to real-life applications. Additionally, this paper focuses on a heuristic algorithm, and the optimal solution cannot be guaranteed. It is interesting to consider exact decomposition algorithms such as bender decomposition or branch-and-price algorithms to solve this problem optimally in future work.

Acknowledgments

This work is supported by Research Grants from the National Natural Science Foundation of China (71672112 and 71972133).

Conflicts of interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

References

Alfredo Tang Montané, F., and Galvão, R. D. 2006. "A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service." *Computers & Operations Research* 33: 595–619.

Amarouche, Y., Guibadj, R. N., and Moukrim, A. 2018. "A neighborhood search and set cover hybrid heuristic for the two-echelon vehicle routing problem." In 18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Avci, M., and Topaloglu, S. 2015. "An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries." *Computers & Industrial Engineering* 83: 15–29.

Baldacci, R., Mingozzi, A., Roberti, R., and Calvo, R. W. 2013. "An exact algorithm for the two-echelon capacitated vehicle routing problem." *Operations Research* 61(2): 298–314.

Belgin, O., Karaoglan, I., and Altiparmak, F. 2018. "Two-echelon vehicle routing problem with simultaneous pickup and delivery: Mathematical model and heuristic approach." Computers & Industrial Engineering 115: 1-16.

Bianchessi, N., and Righini, G. 2007. "Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery." *Computers & Operations Research* 34: 578–594.

Bräysy, O., and Gendreau, M. 2005. "Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms." Transportation Science 39(1): 104-118.

Breunig, U., Schmid, V., Hartl, R. F., and Vidal, T. 2016. "A large neighbourhood based heuristic for two-echelon routing problems." *Computers & Operations Research* 76: 208-225.

Breunig, U., Baldacci, R., Hartl, R. F., and Vidal, T. 2019. "The electric two-echelon vehicle routing problem." *Computers & Operations Research* 103: 198-210.

Clark, G., and Wright, J. W. 1964. "Scheduling of vehicles from a central depot to a number of delivery points." *Operation Research* 12: 568–581.

Cordeau, J. F., Gendreau, M., and Laporte, G. (1997). "A tabu search heuristic for periodic and multi-depot vehicle routing problems." Networks 30(2): 105-119.

Cordeau, J. F., Laporte, G., and Mercier, A. (2001). "A unified tabu search heuristic for vehicle routing problems with time windows." Journal of the Operational Research Society 52(8): 928-936.

Crainic, T. G., Mancini, S., Perboli, G., and Tadei, R. 2011. "Multi-start Heuristics for the Two-echelon Vehicle Routing Problem." In *Evolutionary Computation in Combinatorial Optimization: Lecture Notes in Computer Science 6622*: 179-190, edited by P. Merz, and J.-K Hao. Torino, Italy: Springer.

Crispim, J., and Brandao, J. 2005. "Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls." *Journal of the Operational Research Society* 56: 1296–1302.

Darvish, M., Archetti, C., Coelho, L. C., and Speranza, M. G. 2019. "Flexible two-echelon location routing problem." *European Journal of Operational Research* 277(3): 1124-1136.

Dellaert, N., Saridarq, F. D., Woensel, T. V., and Crainic, T. G. 2019. "Branch-and-Price–Based Algorithms for the Two-Echelon Vehicle Routing Problem with Time Windows." *Transportation Science* 53(2): 463-479.

Dell'Amico, M., Righini, G., and Salani, M. 2006. "A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection." *Transportation Science* 40(2): 235–247.

Feliu, J. G., Perboli, G., Tadei, R., and Vigo, D. 2007. "The two-echelon capacitated vehicle routing problem." Technical report DEIS OR.INGCE 2007/2(R), Department of Electronics, Computer Science, and Systems, University of Bologna, Bologna, Italy.

Fleszar, K., Osman, I. H., and Hindi, K. S. 2009. "A variable neighbourhood search algorithm for the open vehicle routing problem." *European Journal of Operational Research* 195(3): 803–809.

Fontaine, P., Crainic, T. G., Jabali, O., and Rei,, W. 2017. "The Impact of Combining Inbound and Outbound Demand in City Logistics Systems." 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC).

Fontaine, P., Crainic, T. G., Jabali, O., and Rei,, W. 2021. "Scheduled service network design with resource management for two-tier multimodal city logistics." European Journal of Operational Research 294(2): 558-570.

Gajpal, Y., and Abad, P. 2009. "An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup." *Computers & Operations Research* 36(2009): 3215-3223.

Gianessi, P., Alfandari, L., Létocart, L., and Calvo, R. W. 2016. "The Multicommodity-Ring Location Routing Problem." *Transportation Science* 50(2): 541-558.

Glover, F., and Hao, J.K. 2011. "The case for strategic oscillation." *Annals of Operations Research* 183(1): 163-173.

Grangier, P., M. Gendreau, F. Lehuédé and L.-M. Rousseau (2016). "An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization." *European Journal of Operational Research* 254(1): 80-91.

Hansen, P., and Mladenović, N. 2001. "Variable neighborhood search: Principles and applications." European Journal of Operational Research. 130(3):449-467.

Hemmelmayr, V. C., Cordeau, J. F., and Crainic, T.G. 2012. "An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics." *Computers & Operations Research* 39: 3215–3228.

Hemmelmayr, V. C., Doerner, K. F., and Hartl, R. F. 2009. "A variable neighborhood search heuristic for periodic routing problems." *European Journal of Operational Research* 195(3): 791–802.

Hernández-Pérez, H., Landete, M., and Rodríguez-Martín, I. 2021. "The single-vehicle two-echelon one-commodity pickup and delivery problem." Computers & Operations Research 127: 105152.

Jepsen, M., Spoorendonk, S., and Ropke, S. 2013. "A branch-and-cut algorithm for the symmetric two-echelon capacitated vehicle routing problem." *Transportation Science* 47(1): 23–37.

Jie, W., Yang, J., Zhang, M., and Huang, Y. 2019. "The two-echelon capacitated electric vehicle routing problem with battery swapping stations: Formulation and efficient methodology." *European Journal of Operational Research* 272(3): 879-904.

Kindervater, G.A.P., and Savelsbergh, M.W. P. 1997. *Vehicle routing: Handling edges exchanges windows*. In: Aarts, E., Lenstra, J.K. (Eds.)

Koç, Ç., Laporte G., and Tükenmez İ. 2020. "A review of vehicle routing with simultaneous pickup and delivery." Computers & Operations Research 122: 104987.

Kytöjoki, J., Nuortio, T., Bräysy, O., and Gendreau, M. 2007. "An efficient variable neighborhood search heuristic for very large scale vehicle routing problems." *Computers & Operations Research* 34(9): 2743–2757.

Liu, R., Tao, Y. Y., Hu, Q.Y., and Xie, X.X. 2017. "Simulation-based optimisation approach for the stochastic two-echelon logistics problem." *International Journal of Production Research* 55(1): 187–201.

Liu, T., Luo, Z., Qin, H., and Lim, A. 2018. "A branch-and-cut algorithm for the

two-echelon capacitated vehicle routing problem with grouping constraints." *European Journal of Operational Research* 266(2): 487-497.

Majidi, S., Hosseini-Motlagh, S. M., and Ignatius, J. 2018. "Adaptive large neighborhood search heuristic for pollution-routing problem with simultaneous pickup and delivery." *Soft Computing* 22(9): 2851-2865.

Marques, G., Sadykov, R., Deschamps, J. C., and Dupas, R. 2020. "An improved branch-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem." *Computers & Operations Research* 114: 104833.

Min, H. 1989. "The multiple vehicle routing problem with simultaneous delivery and pick-up points." *Transportation Research Part A* 23: 377–386.

Mladenović, N., and Hansen, P. 1997. "Variable Neighborhood Search." *Computers & Operations Research* 24: 1097–1100.

Mühlbauer, F., and Fontaine, P. (2021). "A parallelised large neighbourhood search heuristic for the asymmetric two-echelon vehicle routing problem with swap containers for cargo-bicycles." European Journal of Operational Research 289(2): 742-757.

Nagy, G., and Salhi, S. 2005. "Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries." *European Journal of Operational Research* 162: 126–141.

Perboli, G., Tadei, R., and Vigo, D. 2011. "The two-echelon capacitated vehicle routing problem: Models and math-based heuristics." *Transportation Science* 45(3):364–380.

Polat, O., Kalayci, C. B., Kulak, O., and Günther, H. O. 2015. "A perturbation based variable neighborhood search heuristic for solving the vehicle routing problem with simultaneous pickup and delivery with time limit." *European Journal of Operational Research* 242: 369–382.

Prins, C. 2004. "A simple and effective evolutionary algorithm for the vehicle routing problem." *Computers & Operations Research* 31(12): 1985-2002.

Qiu, M., Fu, Z., Egles, R., and Tang, Q. 2018. "A Tabu Search algorithm for the vehicle routing problem with discrete split deliveries and pickups." *Computers & Operations Research* 100: 102-116.

Rohmer, S. U. K., Claassen, G. D. H., and Laporte, G. 2019. "A two-echelon inventory routing problem for perishable products." *Computers & Operations Research* 107: 156-172.

Santos, F. A., Mateus, G. R., Salles da Cunha, A. 2014. "A branch-and-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem." Transportation Science 49(2): 355–368.

Subramanian, A., Uchoa, E., Pessoa, A. A., and Ochi, L. S. 2011. "Branch-and-cut with lazy separation for the vehicle routing problem with simultaneous pickup and delivery." *Operations Research Letters* 39: 338–341.

Subramanian, A., Uchoa, E., Pessoa, A. A., and Ochi, L. S. 2013. "Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery." *Optimization Letters* 7: 1569–1581.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. 2012. "A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems." *Operations Research* 60(3): 611-624.

Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T. 2010. "An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries." *European Journal of Operational Research* 202: 401–411.

Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T.. 2016. "The Vehicle Routing Problem with Simultaneous Pick-ups and Deliveries and Two-Dimensional Loading Constraints." *European Journal of Operational Research* 251(2): 369-386.

Zeng, Z.Y., Xu, W. S., Xu, Z. Y., and Shao, W. H. 2014. "A hybrid GRASP+VND heuristic for the two echelon vehicle routing problem arising in city logistics." *Mathematical Problems in Engineering* 2014:1–11.

Zhang, Z., Cheang, B., Li C., and Lim, A. 2019. "Multi-commodity demand fulfillment via simultaneous pickup and delivery for a fast fashion retailer." *Computers & Operations Research* 103: 816.