RAIRO-Oper. Res. 54 (2020) 1467–1494
https://doi.org/10.1051/ro/2019080

RAIRO Operations Research
www.rairo-ro.org

A VARIABLE NEIGHBORHOOD SEARCH ALGORITHM WITH REINFORCEMENT LEARNING FOR A REAL-LIFE PERIODIC VEHICLE ROUTING PROBLEM WITH TIME WINDOWS AND OPEN ROUTES

Binhui Chen[1,2,*], Rong Qu[2], Ruibin Bai[3] and Wasakorn Laesanklang[4]

Abstract. This paper studies a real-life container transportation problem with a wide planning horizon divided into multiple shifts. The trucks in this problem do not return to depot after every single shift but at the end of every two shifts. The mathematical model of the problem is first established, but it is unrealistic to solve this large scale problem with exact search methods. Thus, a Variable Neighbourhood Search algorithm with Reinforcement Learning (VNS-RLS) is thus developed. An urgency level-based insertion heuristic is proposed to construct the initial solution. Reinforcement learning is then used to guide the search in the local search improvement phase. Our study shows that the Sampling scheme in single solution-based algorithms does not significantly improve the solution quality but can greatly reduce the rate of infeasible solutions explored during the search. Compared to the exact search and the state-of-the-art algorithms, the proposed VNS-RLS produces promising results.

Mathematics Subject Classification. 90B06, 90B40, 90C27.

Received April 6, 2018. Accepted August 14, 2019.

1. Introduction

Research on the Vehicle Routing Problem (VRP) can be tracked back to the *truck dispatching problem* proposed by Dantzig and Ramser [16]. It is defined as, starting from a depot, a number of vehicles with capacity constraints are to be routed to service a set of customers with demands and return to the depot. Each customer is visited only once. In the scheduling network of VRP, all the routes are Hamiltonian Cycles (close routes). The most common objectives in VRPs are minimizing the number of vehicles used (or routes) and minimizing the total travel cost (distance/time). After decades of study, VRP has become one of the mostly investigated combinatorial optimisation problems in operational research, leading to a large number of variants with different features [25], *e.g.* Vehicle Routing Problem with Time Windows (VRPTW), Vehicle Routing Problem with Pickups and Deliveries (VRPPD), Periodic Vehicle Routing Problem (PVRP), Open Vehicle Routing Problem (OVRP) and so on [18, 67].

Keywords. Periodic vehicle routing problem with time windows and open routes, adaptive operator selection, metaheuristics, variable neighbourhood search.

[1] SF Technology Co. Ltd, Shenzhen, P.R. China.

[2] School of Computer Scicence, University of Nottingham, Nottingham, UK.

[3] School of Computer Scicence, The University of Nottingham Ningbo China, Ningbo, P.R. China.

[4] Department of Mathematics, Faculty of Science, Mahidol University, Bangkok, Thailand.

*Corresponding author: psxbc2@nottingham.ac.uk, BinhuiChen@sf-express.com, cbh_fzu@sina.com

Article published by EDP Sciences

1.1. Basic VRP variants

Based on the basic definition of VRP, in VRPTW, customers' demands are associated with time constraints. The time of servicing a customer must be within a specific time interval, and all vehicles must return to the depot before the end of the planning horizon given [59]. VRPTW is the basic model for many other more complicated VRP variants. In this section, we review the variants which are most relevant to our study.

In VRPPD, customers' demands are divided into two categories: pick up shipments from a source and deliver shipments to a destination. Various constraints on pickup and deliver points lead to diverse VRPPD variants [25]. If the depot is the only one pickup point while the customers are delivery points, or all shipments are picked up from customers and delivered to the depot, the problem is classified as a *One-to-Many-to-One* problem. Whilst customers can be both pickup and delivery points, it is a *Many-to-Many* problem. In *One-to-One* problems, one customer's pickup demand is another customer's delivery demand. Furthermore, if consolidation is allowed when picking up, it is called a *Less-than Truckload Transportation* problem, otherwise it is a *Full Truckload Transportation (FTL)* problem [72].

In some real-life problems, the planning horizon is long and divided into multiple periods/shifts, therefore notated as Multi-Period Vehicle Routing Problems (MPVRP) [44]. The vehicles must return to the depot every shift in MPVRP. Especially, if each customer has a specific visiting frequency within the planning horizon, this type of MPVRP is called Periodic Vehicle Routing Problem (PVRP). Each customer can be visited more than once in PVRP, and a scheduling solution is usually represented as a set of visiting day (shift) combinations for customers. PVRP may occur in grocery distribution, soft drink industry, waste collection and so on [29].

The earliest OVRP is proposed by Eppen and Schrage [19], where a fleet collect goods from the central depot and deliver them to a number of geographically scattered customers. The main characteristic of OVRP is that its routes are Hamiltonian paths (open routes) rather than cycles [63]. This characteristic reflects the reality that many companies in real-world do not own a fleet but instead hire external carriers, *e.g.* third party logistic providers and private vehicles, to service customers. Those hired vehicles do not need to return to the depot after servicing all customers assigned. The routes in OVRP terminate at the last customer serviced [38].

1.2. Extended VRP variants and solution methods

After decades of study, both exact and approximate algorithms have been intensively investigated for diverse VRP variants. However, due to the NP-hard feature of VRPs [36], exact methods are more suitable to small or medium size VRPs [67]. On the other hand, in approximate approaches (or heuristics), metaheuristics have shown powerful performance in solving big-scale and complex VRPs.

The first exact method for OVRP is proposed in [37], but only on small and medium size instances (less than 151 customers and 14 vehicles). Tarantilis *et al.* [62,63] propose two local search metaheuristics with annealing based threshold accepting criterion and list based threshold accepting criterion respectively. Both algorithms outperform previous approaches for OVRP. Two years later, a Record-to-Record Travel algorithm is adopted as the acceptance criterion, and the associated algorithm generates better solutions than 11 previous algorithms for OVRP [38]. Two metaheuristic algorithms based on Tabu Search for OVRP can be found in [3,20]. A Static Move Descriptor is proposed in [74] to speed up the evaluation in best improvement search. The associated algorithm produces the best results on benchmarks of OVRP, however, it is not applicable to OVRP with Time Windows (OVRPTW) where time constraints are considered.

The first introduction of the OVRPTW is in [53]. In the proposed solution methodology, an insertion-based construction heuristic employs an improved *IMPACT* criterion [31] to select customer to be inserted in to the routes. Since then, a large number of metaheuristics are developed for OVRPTW. A Variable Neighbourhood Search (VNS) algorithm for OVRPTW can be found in [47], which outperforms the *IMPACT* approach. An Ant Colony Optimization (ACO) algorithm and a Genetic Algorithm are also applied to the problems of relatively small size [26,27].

In a specific case of OVRPTW, the routes start from geographically scattered customers to pick up goods. After visiting the customers assigned, vehicles return to the central depot to unload the goods collected. The

routes in this case are open at the starting point, which is opposite to the standard OVRP. An Adaptive Large Neighbourhood search algorithm is proposed for this Reverse-OVRPTW in [57]. School Bus Routing Problem is another special case of OVRP, where its morning and afternoon routing problems can be addressed as the same problem. Every morning, school buses send students to the school from pickup stops, while in the afternoon students are sent back to the stops in the reverse order of the morning routes [46]. In addition, Liu and Jiang [41] and Brito *et al.* [6] study the VRP with both open and close routes.

The first research on Periodic Vehicle Problem with Time Windows (PVRPTW) is in [12]. Considering travel time, capacity, duration and time windows, a construction heuristic followed by a Tabu Search (TS) is proposed. An improved TS method adopting the *Forward Time Slack* [56] is later proposed to further reduce the route [13]. An ACO model for PVRPTW can be found in [73] where a multiple pheromone information matrix is designed. The flexibility of customer visiting date in PVRPTW further increases the number of parameters in this ACO algorithm and the complexity of this methodology. More solutions for PVRPTW can be found in [10, 48, 52].

An Open Periodic Vehicle Routing Problem (OPVRP) model is introduced in [15]. The vehicles are not obliged to return to the depot at the end of each day in a planning horizon of multiple days. In the proposed solution construction heuristic, a k-means clustering algorithm is used to assign customers to routes, without considering the capacity limit. A feasibility procedure would then fix the infeasible assignments. This approach can quickly generate a feasible solution for small size problems. However, it would be inefficient for larger problems or when the depot is not in the geographic center. Time windows are not considered in this model.

Vidal *et al.* [69] propose a unified hybrid genetic search framework (UHGS) aiming to provide a general-purpose solver for diverse VRP variants. It produces the results better than or close to the state-of-the-art algorithms on benchmarks. However, as a genetic algorithm, it is shown that the computation time sharply increases when facing MPVRP. This is due to the issue of shift and route partition in the chromosome/genotype (solution representation), which is hard to handle for the huge number of possible positions of shift and route delimiters. This solution partition problem is tackled as a shortest path problem in UHGS. The long computation time impedes the application of UHGS to large-scale MPVRP. More metaheuristics for VRPs have been reviewed in [5, 22].

Most research formulate VRPs with connected network, where the connected nodes represent customers, demands, services and so on. The weight of an edge connecting two nodes represents the cost of a vehicle traveling from a node to the other. This method is easy to apply to the problems with simple service activities. However, in some real-life problems, the service activities are complex and hard to be simplified or combined. Bai *et al.* [1] use the loading and unloading nodes pair to represent a transportation task, converting a container transportation problem into a Set-Covering problem. The problem is then solved with an exact algorithm. This method is not suitable to those problems with many container terminals. [71] integrate loading, traveling and unloading activities into a *task node* for a pickup and delivery problem. This task node-based model simplifies the model for real-life complex problems, and is applicable to a large number of classic node-based algorithms, thus has been used in various VRPs research [8, 75].

This paper addresses a real world container transportation problem, which shares some common features with the classic OVRP and PVRP. The mathematical problem model and an illustrative example are presented in Section 2. The proposed solution methodology is introduced in Section 3. In addition to an urgency level-based constructive heuristic, an improvement metaheuristic with reinforcement learning is also developed. Section 4 presents the experiment results and analysis on benchmark instances. The conclusions of this paper are presented in Section 5.

2. Problem definition & mathematical model

The problem studied in this paper is a real-world inter-dock container transportation problem at Ningbo Port, which is the fifth largest port in the world. Everyday a fleet of 100 trucks transport commodities (containers) between nine container terminals (see Fig. 1). Each commodity consists of a number of containers. The containers

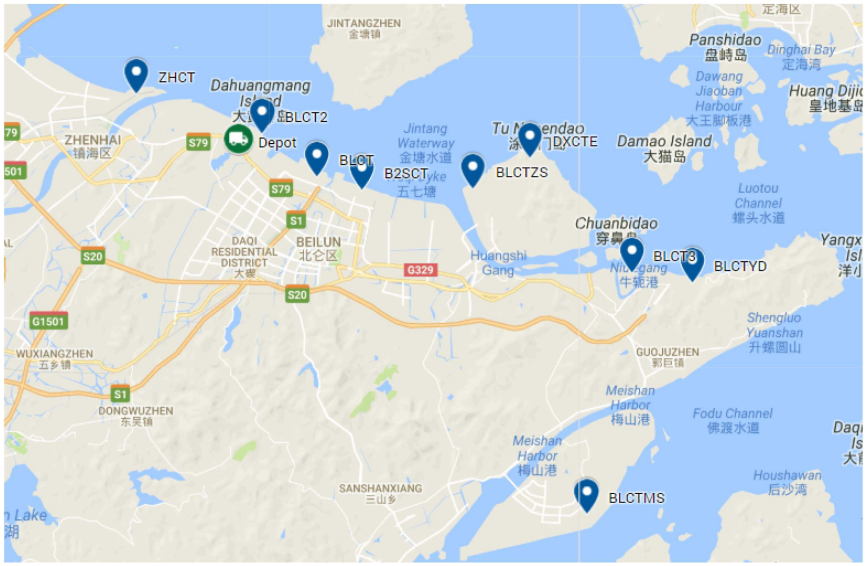

FIGURE 1. Nine container terminals' locations in Ningbo Port, taken from Google Maps [43].

are picked up from source terminals and delivered to destination terminals, satisfying the time constraints for the commodities and drivers. Previous research on the truck scheduling in Ningbo Port can be retrieved *via* https://sites.google.com/nottingham.ac.uk/port-management. Bai *et al.* [1] have proposed a scheduling scheme based on PVRPTW for this problem. Since the size and operational cost of the fleet are fixed at the Ningbo Port, the port manager expects to further decrease the empty-load travel distance of the fleet, optimizing the utility of trucks. To this end, the current scheduling scheme is revised by reducing the empty-load travel from and to the depot, which is introduced below. This problem is a One-to-One VRPPD with FTL. The trucks used are identical and each truck can carry only one container at a time due to its capacity, thus consolidation is not allowed.

The commodities to be transshipped are usually issued several days before their shipment deadlines, which brings a long scheduling horizon. There are a few emergent commodities that are also submitted in practice, and they are either inserted to the current schedule or rejected, which is not discussed in this study. Each working day in the scheduling horizon is divided into two shifts (*i.e.* day and night shifts with 12 h per shift) obliging the related regulations on drivers' working hours specified in Labour Law. In the first shift (day shift) everyday, trucks depart the depot with a list of tasks to complete. When all tasks assigned are completed, the trucks would park at the last task destinations, or at the first task source terminals of the second shift (night shift) as long as the trucks can arrive there before the end of the day shift. Then, at the beginning of the night shift, a new group of drivers are assigned their trucks at specified terminals (shift change). In the night shift, after completing all assigned tasks trucks must go back to the depot for maintenance and preparation of the next day. The travel of trucks heading to the next task source is empty-load, with no container transported.

Drivers take shift change in the middle of a working day at different terminals, instead of one specific depot. It can be found that, here the routes in each shift are open as those in the OVRP. More precisely, the scheduled routes of day shifts are open as they do not finish at the depot, while the routes of night shifts are reverse open routes. The difference between this problem and the School Bus Routing Problem is that, in our problem, the routes in night shifts are not the same as the routes in day shifts in a reversed order.

The typical scheduling horizon in Ningbo Port spans from 2 to 4 days (4–8 shifts). The first shift of every working day is denoted as an *odd-indexed* shift, while the second shift is *even-indexed*. In such a multi-shift

problem, the number of containers in each commodity can be seen as the visiting frequency of customers in PVRP. However, the model of PVRP cannot be directly applied due to the distinct practical constraints in this problem. In this paper, the Ningbo Port container transportation problem is modeled as a PVRPTW with Open Routes (PVRPTW-O). To the best of our knowledge, this is the first time PVRPTW-O is introduced in the literature.

In previous VRPPD research, the shipment loading and unloading time is usually ignored or simplified as they are small and/or identical. However, because of the limitation of cranes at terminals, the time of loading and unloading account for a considerable proportion of the total service time in PVRPTW-O. Besides, the service time are different at different terminals in PVRPTW-O, thus cannot be ignored or simplified. In our mathematical PVRPTW-O model, the method of [71] is adopted. The time of loading and unloading activities are combined with the traveling time, defining the *task nodes* with different service time. In the proposed model, a *task node* is composed of the loading at the source terminal, unloading at the destination terminal, loaded travel from the source to the destination, and the associated time window of the task.

Artificial depots (W) are introduced to connect an odd-indexed shift (S_{odd}) to the following even-indexed shift (S_{even}). In S_{odd}, artificial nodes are termination depots, while they become the starting depots in the following S_{even}. When a truck travels from S_{odd} to S_{even}, the first source terminal of a route in S_{even} serves as an artificial depot, if it can be reached before the end of S_{odd}. Otherwise, the last destination terminal of the route in S_{odd} will be the artificial depot.

An example of one day schedule (two consecutive shifts of an odd-indexed and an even-indexed) is presented in Figure 2. The fleet size is five (*i.e.* $K = 5$), corresponding to five routes. Take the first route as an example, two and three tasks are completed in the odd-indexed and the even-indexed shifts, respectively. The lines directly connecting starting depots and termination depots are empty routes, which means no task is completed on them. In this example, each shift has two empty routes. The artificial depots in W are either the last destinations on Shift 1 routes or the first sources on Shift 2 routes. The number of artificial depots visited is decided by the number of terminals at where shift-changes happen. In this example, two trucks change shift at the fourth artificial depot.

The notations used are introduced in Table 1. Every scheduled route starts at a *starting depot*, connects a number of task nodes (tasks to be completed) and ends at a *termination depot*. It is notable that, some of these routes are empty, which means no task is completed on them.

The problem can be formally defined as follows:

$$\text{Minimize} \qquad \text{TD} = \sum_{s\in S}\sum_{i\in N\cup W}\sum_{j\in N\cup W} c_{ij}\cdot x_{ij}^{s}. \tag{2.1}$$

Subject to:

$$\sum_{s\in S}\sum_{i\in W\cup N\setminus\{j\}} x_{ij}^{s} = 1, \qquad \forall j \in N\setminus\{0\} \tag{2.2}$$

$$\sum_{s\in S}\sum_{j\in W\cup N\setminus\{i\}} x_{ij}^{s} = 1, \qquad \forall i \in N\setminus\{0\} \tag{2.3}$$

$$\sum_{i\in W\cup N\setminus\{j\}} x_{ij}^{s} = \sum_{f\in W\cup N\setminus\{j\}} x_{jf}^{s}, \qquad \forall j \in N\setminus\{0\}, s \in S \tag{2.4}$$

$$T_j = \sum_{i\in N\setminus\{0\}} (B_i + l_i + t_{ij})\cdot x_{ij}^{s} + \sum_{i=\{0\}\cup W} (Y_s + t_{ij})\cdot x_{ij}^{s}, \qquad \forall j \in N\setminus\{0\}, s \in S \tag{2.5}$$

$$B_j = T_j + \max\{a_j - T_j, 0\}, \qquad \forall j \in N\setminus\{0\} \tag{2.6}$$

$$x_{ij}^{s}\cdot Y_s \;\le\; x_{ij}^{s}\cdot T_j, \qquad \forall i \in \{0\}\cup W, j \in N\cup W, s \in S \tag{2.7}$$

$$x_{ij}^{s}\cdot (B_i + l_i) \;\le\; x_{ij}^{s}\cdot Z_s, \qquad \forall i \in N\cup W, j \in \{0\}\cup W, s \in S \tag{2.8}$$

$$a_i \le B_i \le b_i - l_i, \qquad \forall i \in N\setminus\{0\} \tag{2.9}$$

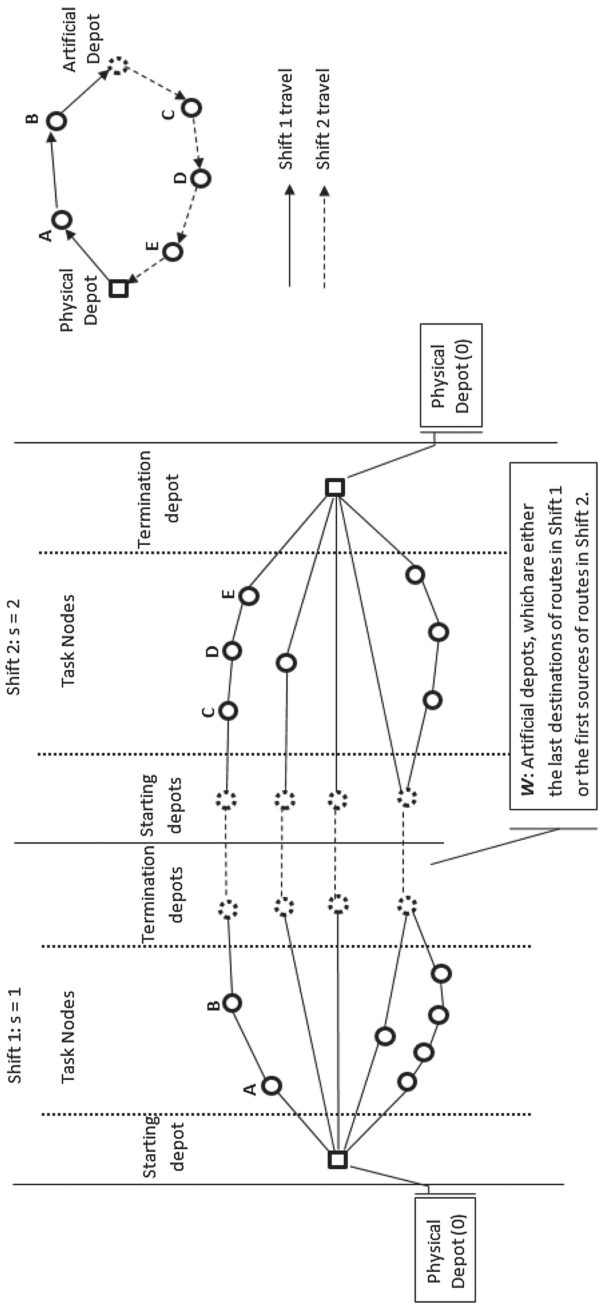

FIGURE 2. A scheduling example of two consequent shifts with five trucks. The solid line circles represent the transportation tasks to be completed. The right sub-figure indicates the first route in the schedule.

TABLE 1. The list of notations.

Input parameters	
K	Fleet size, *i.e.* number of trucks
S	The set of time-continuous working shifts, which can be divided into odd-indexed shifts (S_{odd}) and even-indexed shifts (S_{even})
$[Y_s, Z_s]$	Time window of shift s
$N = \{0, 1, 2, \ldots, n\}$	Set of $n+1$ nodes. Each node represents a task, except node 0 is the physical depot
$[a_i, b_i]$	Time window for node i. The time window for node 0 is zero at the boundary of a shift. If a truck arrives at the source of i early, it has to wait until a_i
A	Set of arcs. Each arc (i, j) represents that node j is immediately serviced/visited after servicing/visiting node i
c_{ij}	The cost of empty load travel from node i to node j. When both i and j are task nodes, c_{ij} is the distance from the destination of i to the source of j. Otherwise, it is the distance between a terminal and the depot
t_{ij}	The travel time from node i to node j. When both i and j are task nodes, t_{ij} is the travel time from the destination of i to the source of j. Otherwise, it is the travel time between a terminal and the depot
l_i	The time for servicing node i, which includes the loading time, transportation time (from pick-up source to delivery destination) and unloading time. The service time of a depot is zero
W	Set of *Artificial Depots*. This set of nodes are introduced to represent the termination depots in S_{odd} or starting depots in S_{even}. W refers to all possible physical terminal locations, but which artificial depots are visited vary in different solutions. Thus, an artificial depot may not be visited or may be visited more than once in W. The use of artificial depots guarantees the constraints on truck driver working time being satisfied
Variables	
T_i	The arrival time at node i
B_i	The time to begin the service of node i
x_{ij}^s	A binary decision variable for nodes $i, j \in N \cup W$. Its value is 1 if arc (i, j) is included in the solution in shift s, otherwise is 0. $i, j \in W$ is not allowed

$$x_{ij}^s \in \{0,1\}, \qquad \forall i, j \in N \cup W, s \in S \tag{2.10}$$

$$x_{vw}^s = 0, \qquad \forall v \in W, w \in W, s \in S. \tag{2.11}$$

In odd-indexed shifts ($\forall s \in S_{\text{odd}}$):

$$\sum_{j \in N \setminus \{0\} \cup W} x_{0j}^s = K, \qquad \forall s \in S_{\text{odd}} \tag{2.12}$$

$$x_{i0}^s = 0, \qquad \forall i \in N \setminus \{0\} \cup W, s \in S_{\text{odd}} \tag{2.13}$$

$$\sum_{i \in N} \sum_{w \in W} x_{iw}^s = K, \qquad \forall s \in S_{\text{odd}}. \tag{2.14}$$

In even-indexed shifts ($\forall s \in S_{\text{even}}$):

$$\sum_{j\in N} x_{jw}^{s-1} = \sum_{e\in N} x_{we}^{s} \qquad \forall w \in W, s \in S_{\text{even}} \tag{2.15}$$

$$x_{0j}^{s} = 0, \qquad \forall j \in N\backslash\{0\} \cup W, s \in S_{\text{even}} \tag{2.16}$$

$$\sum_{w\in W}\sum_{j\in N} x_{wj}^{s} = K, \qquad \forall s \in S_{\text{even}} \tag{2.17}$$

$$\sum_{i\in N\backslash\{0\}\cup W} x_{i0}^{s} = K, \qquad \forall s \in S_{\text{even}}. \tag{2.18}$$

The objective of this problem is to minimize the total travel distance (TD), equation (2.1). In classic VRPs, this objective is the secondary objective. However, the operational cost of the fleet in this real-life problem is fixed, thus, minimizing the travel distance become the only one objective. Since the loaded travel distance is fixed in each instance, the objective actually is equivalent to minimizing the empty-load travel distance.

Constraints (2.2) and (2.3) denote that every task node can be visited exactly once and all the tasks must be visited. Constraint (2.4) means the in-degree of a node equals its out-degree, specifying that a task may only be serviced after the previous task is completed. Constraints (2.2)–(2.4) together make sure arcs of over more than one shift are unacceptable. Constraint (2.5) is the arrival time at a task node. Constraint (2.6) defines the beginning time of servicing a task node. This time is calculated by the arrival time plus the waiting time at the source of a task. Constraints (2.5) and (2.6) enforce the correct temporal successive relationship between consecutive nodes.

Constraints (2.7) and (2.8) are the time window constraints of each shift, while constraint (2.9) represents the time constraint on each task. The domain of the respective decision variable is defined by constraints (2.10) and (2.11). Especially, constraint (2.11) prohibits the travel between two artificial depots.

The starting and termination depots in odd-indexed shifts and even-indexed shifts are different. Constraints (2.12) and (2.14) represent that K trucks leave the physical depot (0) at the beginning of an odd-indexed shift, and they would stop at artificial depots at the end of the shift. Constraint (2.13) represents that no truck returns to the physical depot in odd-indexed shifts. Constraints (2.16)–(2.18) place the reverse restraints in even-indexed shifts. Constraint (2.15) defines the shift change from an odd-indexed shift to the following even-indexed shift on artificial depots, where the in-degree of each artificial terminal in S_{odd} equals its out-degree in the following S_{even}.

From this integer programming model, we can find that this problem is nonlinearly constrained and with a huge solution space. It is no doubt that the nonlinear constraints can be converted to linear ones, but leads to a more complex model, which is discussed in Section 4.2. The size of the solution space of PVRPTW-O is decided by the length of the scheduling horizon ($|S|$), the fleet size (K) and the number of tasks (n). Since the number of total routes in a solution could be up to $|S| \cdot K$, while the number of permutations of tasks is $n!$, the size of the search space is $|S| \cdot K \cdot n!$. In real-life scenarios, n in the whole planning horizon could be larger than 1000. This model is a basic framework for PVRPTW-O, in future extended research more specific constraints (*e.g.* subtour elimination constraints) can be added to further improve the problem formulation or expressing it in different forms. It is worth noting that, empty routes in S_{even} do not mean their travel distances are zero. The cost of empty route can be zero, only when the connected artificial node represents the physical depot.

3. Solution methodology

3.1. Motivations & algorithm framework

3.1.1. Motivitions

We use the CPLEX solver on both real-life and artificial benchmark instances, and the results (see Sect. 4.2) show that it is unrealistic to address the PVRPTW-O with exact methods. Heuristics and metaheuristics are

thus investigated in this study for PVRPTW-O. A large number of metaheuristic algorithms have obtained promising results in VRPs, including Population-Based and Single Solution-Based metaheuristics, *e.g.* Evolutionary Algorithms (EA) [2], Population-Based Incremental Learning [42], Simulated Annealing [34], TS [51] and VNS [9].

Population-based approaches evolve a population of solutions during the search, showing a powerful performance for highly constrained and multi-objective VRPs [33]. However, PVRPTW-O is challenging to population-based approaches due to its high dimensional solution structure and large problem scale. The tasks in PVRPTW-O must be indexed from three dimensions in the solution (indexes of the shift, the route and the position in the route). This structure increases the difficulty of operations between solution individuals, *e.g.* the above-mentioned solution partition problem in Genetic Algorithms. In addition, when the problem size and population size are big, the computation time for the large population would be very long. Thus, population-based methods are not suitable to this large scale high-dimensional problem.

Single solution-based metaheuristics (or Local Search) use different strategies and neighbourhood operators to explore the solution space iteratively, while only one solution is updated in each iteration. Apart from straightforward operators, *e.g.* Swap and Insertion, many delicately designed neighbourhood operators are developed and applied in various metaheuristics, such as λ-opt [39], or-opt [45], Cyclic Transfers [65], 2-opt* [50], CROSS-exchange [61] and so on. More successful single solution-based approaches can be found in [4].

VNS systematically changes neighbourhood operators to explore search space and shows excellent performance in VRPs [28]. However, in real-life large-scale and highly constrained instances, the basic VNS structure often shows not powerful enough and promoted by being combined with other mechanisms [14]. To this end, *Reinforcement Learning* is introduced to VNS in our study. Reinforcement Learning (RL) is an adaptive learning and decision-making scheme, based on a probability distribution sampled over the candidate set. The probability distribution is continuously updated according to the reinforcement feedback from the learning environment [64].

RL scheme can be used in *Adaptive Operator Selection* (AOS), guiding the search process [68]. Credit Assignment mechanism and Selection Rule are the two essential components in AOS. According to the historical performance of operators, credit assignment mechanism updates the invoked probability (or weight) of candidate operators during the search, *e.g.* increasing or reducing the probability with elaborately designed reward function [49]. Then, the operators to be executed are chosen with a Selection Rule. Roulette Wheel scheme and Pursuit Algorithm [66] are the two commonly used selection rules. The former chooses operators on the basis of their probability distribution and the latter always selects the operator with the largest probability/weight.

AOS has been widely used in metaheuristics. In population-based metaheuristics, credit assignment mechanism calculates the feedback values based on the quality of solution population [40,58]. To single solution-based approaches, RL and AOS has also shown outstanding performance in VRPPD and other VRP variants, *e.g.* in Adaptive Large Neighbourhood Search [30,57]. Some associated methodologies produce the best results on the benchmarks of classic VRP variants [54,55]. More applications of adaptive learning in single solution-based heuristics can be found in [7,68].

In metaheuristics, Sampling can be used to measure the fitness landscape surrounding a solution, providing guidance for the search. It shows powerful performance in Evolutionary Computation [32]. Soria Alcaraz *et al.* [60] propose an Evolvability Metric of Sampling, where two scalars are proposed as the indicators in the credit assignment mechanism during evolution. One indicator is the probability of the offspring having higher or equal fitness comparing to their parents, while the other indicator is the mean fitness of all sampling offspring. The proposed EA approach with Sampling produces promising results in three classic optimization problems. Whether Sampling also works in single solution-based heuristics and PVRPTW-O is worthy to be investigated further.

3.1.2. Algorithm framework

A VNS algorithm with Reinforcement Learning and Sampling (VNS-RLS) is proposed in this paper, see the framework in Algorithm 1. The initial feasible solution *Sol* is constructed with a heuristic in *Step 1*. Then, at

TABLE 2. Operations of the eight operators within NS in VNS-RLS.

Operator	Description
Intra-Shift 2-opt*	Execute 2-opt* exchange [50], where the two chosen routes are randomly selected from the same shift
Inter-Shift 2-opt*	Execute 2-opt* exchange, where the two chosen routes are randomly selected from different shifts
Intra-Route Or-opt	A string of task nodes is repositioned in the original route [45]
Intra-Shift Or-opt	A string of task nodes is repositioned from one route to another route in the same shift
Inter-Shift Or-opt	A string of task nodes is repositioned from one route to another route in a different shift
Intra-Route CROSS	Swap two strings of task nodes in the same route [61]
Intra-Shift CROSS	Swap two strings of task nodes from different routes in the same shift
Inter-Shift CROSS	Swap two strings of task nodes from two routes which are from different shifts

each VNS iteration, *Shaking* generates a new feasible solution Sol' as the starting solution by perturbing Sol. In *Step 2.2*, with Sol', *Sampling* initializes a set of weights (WS) for all the operators in the neighbourhood operator set (NS). With the invoked probability distribution of operators, which is generated according to WS, *Local Search* explores better solutions iteratively (*Step 2.3*). These steps are repeated until the objective value has not been improved by 0.01% after a predefined number of iterations.

Algorithm 1. The VNS-RLS framework.

Input: The set of tasks to be assigned and the scheduling horizon.
Start
  Step 1: Produce an initial feasible solution Sol with a constructive heuristic. // Sect. 3.2.
  Step 2:
  while Terminate condition is not met **do**
    Step 2.1: $Sol' \leftarrow$ **Shaking**(Sol). // Sect. 3.3.
    Step 2.2: WS $\leftarrow$ **Sampling**(Sol', NS). // Sect. 3.4.
    Step 2.3: $Sol \leftarrow$ **Local Search**(Sol, Sol', WS). // Sect. 3.5.
  end while
Stop
Output: Sol.

Eight improved neighbourhood operators, which require relatively less computation time, are adopted in VNS-RLS. These operators are devised considering the specific solution structure of the PVRPTW-O. The solutions of classic VRPs are 2-dimensional. However, as mentioned before, the solution of PVRPTW-O has one more dimension of shift. Previous research has shown that properly using operators with diverse degrees of perturbation can significantly improve search performance [9]. To systematically increase the diversification of the search in this tightly constrained problem, we specify the level of perturbation for each operator. The eight improved operators work at different levels (intra-route, intra-shift and inter-shift) with diverse levels of perturbations, details explained in Table 2. Considering the time constraints in PVRPTW-O, the directions of the operated strings of tasks in the routes are kept in all the eight neighbourhood structures to reduce the constraint violation.

3.2. Initial solution construction

A large number of constructive heuristics have been developed for VRPs. Saving Algorithm [11] always selects the operation which brings the largest travel distance saving. However, it shows worse performance

on asymmetrical Capacitated Vehicle Routing Problem and cannot control the number of vehicles used [70]. The later constructive algorithms, *e.g.* Cluster First-Route Second [23], Route First-Cluster Second, Sweep Algorithm [24] and so on, perform better on instances where customers are geographically clustered around the depot. More constructive heuristics for VRPs can be found in [35].

Based on the above-mentioned methods, Solomon [59] proposes four constructive heuristics for VRPTW, while the Insertion-Based heuristic outperforms the other three. Insertion-based heuristics can be easily integrated with diverse insertion selection strategies. In addition, by adjusting the insertion strategy applied during the construction procedure, the number of vehicles used can be better controlled. At Ningbo Port, the nine container terminals are neither clustered nor uniformly located around the depot, and the triangle rules do not apply in its transportation network due to the real traffics, which make the other classic construction heuristics inapplicable or poor-performance under the circumstances. Considering the number of trucks in this problem is limited (100), an Urgency Level-based Insertion constructive Heuristic (ULIH) is thus developed for PVRPTW-O.

In ULIH, the solution construction is done shift-by-shift. Because the number of containers to be transshipped is large every day in Ningbo Port, to complete all the tasks before their deadlines, the tasks are classified into two categories to specific shifts: *mandatory* and *optional*. For each shift, the mandatory tasks will be assigned first. The urgency level of a task varies along with the change of the operating shift. In brief, to a feasible task i of shift s, if it must be completed no later than s, then it is *mandatory* to shift s; otherwise, it is *optional*. This classification is executed before the solution construction. Tasks are inserted into the scheduled routes, which means being assigned to trucks. When no feasible insertion available to existing routes, a new route will be created, meaning the number of trucks used (k) is increased by one. Considering the *big* tasks (with long service time) are harder to assign in scheduling, the biggest task is selected first in the newly created route.

Another strategy often used in practice is that tasks should be assigned as early as possible to avoid leaving too many unassigned tasks to later shifts. In real-life, there may be new tasks submitted to the scheduling system in real-time, so leaving more free time slots and trucks for later shifts can also improve the system reliability. Thus, after all mandatory tasks being assigned, optional tasks will be inserted until no feasible insertion or free truck is available ($k \leq K$).

The insertion selection tactic determines the solution quality and constructive speed of insertion-based heuristics. Some criteria have been developed to select the best candidate and insertion position for routing problems [31, 53], however, the greedy selection strategies dramatically increase the computation time in large problems. To reduce the computation time, some researchers randomly select the next insertion or narrow the selection range in each iteration [54]. Three insertion selection tactics are proposed in ULIH, which are introduced below:

- *Greedy tactic*. All unassigned tasks and insertion positions on all routes are evaluated, and the insertion which brings the lowest empty-load travel distance increase would be executed. This greedy tactic generates *tighter* routes but longer evaluation time is required.
- *First-Feasible tactic*. In this case, all tasks are sorted according to their closeness to their *deadlines*. This tactic always inserts the first task in the unassigned task set (closest to deadline) into its first feasible position. This strategy can construct routes more quickly, but sacrificing the solution quality as more trucks are required, and some mandatory tasks may not be assigned due to the lack of trucks. When K is small, this tactic should be used cautiously.
- *One-Route tactic*. Considering all the unassigned tasks, the insertion with the lowest empty-load travel distance increase on the latest created route execute with this tactic. This is a greedy insertion tactic on the newest route. It may cost more computation time than *First-Feasible* tactic, but produce tighter routes accordingly. This tactic trades off construction speed and solution quality.

These three tactics are tested and compared on benchmark instances, and the comparison results are presented in Section 4.3.

3.3. Shaking

Shaking is often used in VNS algorithms to escape from the local optimum and diversify the search. The commonly used Shaking schemes include: randomly invoking neighbourhood operators, designing specific Shaking Operators and so on. To jump to farther areas from the current search region, in VNS-RLS, four neighbourhood operators which bring larger changes are employed in Shaking. Inter-Shift 2-opt*, Intra-Shift 2-opt*, Inter-Shift Or-opt and Inter-Shift CROSS are sequentially used to generate the starting solution of *Local Search*. Each operator is executed once, and the solution generated is passed to the following operator as the seed solution. Note that the new solution generated will be accepted in Shaking as long as it is feasible, even if its quality is worse. The aim of Shaking is to increase diversification and avoid search cycle, rather than myopic solution improvement.

3.4. Sampling of neighbourhood

Sampling of surrounding environment can provide guidance for the search and has improved the search performance in many population-based algorithms. The impact of sampling on the search trajectory of single solution-based algorithms is investigated in our study. A sampling scheme is applied before the *Local Search* phase in VNS-RLS.

Based on the perturbed solution Sol' generated by *Shaking*, every candidate neighbourhood operator NS_i is sampled, producing d sampling solutions ($Sol_{ij}^{\mathrm{spl}} \in Sol_i^{\mathrm{spl}}, j \in \{1, \ldots, d\}$). In VNS-RLS, search with more sampling iterations explores the neighbourhood environment more thoroughly, however at the cost of a higher computation time. Our preliminary experiment results show that, when the number of samples (d) for each operator is higher, the performance of the algorithm is more stable with a lower standard deviation, while the computation time increases drastically. However, when $d > 50$, further improvement is not significant. In the experiments, no significant change on solution quality is observed along with the increment of d. To balance the evaluation time and the sampling approximation accuracy, each operator is thus sampled 50 times ($d = 50$). The obtained sampling solutions are then measured, while the feedback will be used to generate the initial weights of operators *WS*, providing an initial search direction to *Local Search*.

Three scalars (Eqs. (3.1)–(3.3)) are defined to measure the sampling solutions. To operator NS_i, the first scalar E_i^a indicates the probability of finding a solution which is not worse than the current solution. E_i^b presents the average solution quality of the solutions generated. ρ is a parameter for balancing the impact of E_i^a and E_i^b, which is set to 10 empirically. In the preliminary experiments, it is observed that more than half of the neighbourhood moves produce infeasible solutions. To reduce the infeasible moves in the search, a third indicator E_i^c is proposed in VNS-RLS, to indicate the probability of obtaining a feasible solution with operator NS_i.

$$E_i^a = \frac{\{|Sol_i^{\mathrm{spl}}| \mid \mathrm{TD}(Sol_{ij}^{\mathrm{spl}}) \leq \mathrm{TD}(Sol')\}}{d} \tag{3.1}$$

$$E_i^b = \rho \cdot \left(\frac{\mathrm{TD}(Sol)}{\sum_{j=1}^{d} \mathrm{TD}(Sol_{ij}^{\mathrm{spl}}) \;/\; d} - 1 \right) \tag{3.2}$$

$$E_i^c = \frac{\{|Sol_i^{\mathrm{spl}}| \mid Sol_{ij}^{\mathrm{spl}} \text{ is feasible}\}}{d}. \tag{3.3}$$

For each operator NS_i, the overall evaluation value (E_i) is a combination of the above three metrics. Based on the preliminary experiment results, the weights of indicators are set as $\alpha = 0.5$, $\beta = 0.2$ and $\gamma = 0.3$.

$$E_i = \alpha \cdot E_i^a + \beta \cdot E_i^b + \gamma \cdot E_i^c \qquad (\text{s.t. } \alpha + \beta + \gamma = 1). \tag{3.4}$$

The weight (W_i) determines the probability of a neighbourhood operator NS_i being invoked in the Local Search, stored in WS. To each operator, the initial W_i is calculated according to the feedback of Sampling (E_i), see equation (3.5). The sum of the initial W_i of the P operators is 1. Let $E_{\min}$ be the minimal E_i among all operators, an operator with larger E_i would have a larger initial W_i. To avoid some of the weights being too small and the associated operators never being invoked, a minimal initial weight $W_{\min}$ is adopted, which is empirically set to 5% in VNS-RLS.

$$W_i = W_{\min} + (1 - P \cdot W_{\min})(E_i - E_{\min}) / \sum_{i=1}^{P} (E_i - E_{\min}) \tag{3.5}$$

3.5. Local search with reinforcement learning

Algorithm 2. Local search (Sol, Sol', WS).

Input: Current best solution Sol, starting solution Sol' and the initial weight set WS (initial W_i).
Start
  Set $L \leftarrow 0$.
  while $(L < L_{\max})$ **do**
    Step 1: Generate the invoked probability of each operator based on its weight (W_i).
      $Pr_i \leftarrow W_i / \sum_{i=1}^{P} W_i$
    Step 2: Neighbourhood Search
      Execute a neighbourhood operator NS_i selected with *Roulette Wheel Scheme* based on Pr_i, generating a new solution (Sol''): $Sol'' \leftarrow NS_i(Sol')$, $L \leftarrow L + 1$.
    Step 3: Move or Not
    if Sol'' is infeasible **then** //Feasibility Indicator (FI)
      $W_i \leftarrow W_i * 0.9$
    else if $Evaluation(Sol'') < Evaluation(Sol)$ **then** //Solution Quality Indicator: TD
      $Sol \leftarrow Sol'', Sol' \leftarrow Sol'', L \leftarrow 0$.
      $W_i \leftarrow W_i * 1.1$
    else if $Evaluation(Sol'') - Evaluation(Sol') < DEVIATION$ **then**
      $Sol' \leftarrow Sol''$.
    else
      $W_i \leftarrow W_i * 0.9$
    end if
  end while
Stop
Output: The best found solution Sol.

In VNS-RLS, the *Local Search* is an iterative search with reinforcement learning, see Algorithm 2. Firstly, the initial invoked probabilities are distributed to operators according to W_i. Using the *Roulette Wheel scheme* as the selection rule, in Step 2, one neighbourhood operator is selected and applied once to the current solution (Sol'), generating a new solution (Sol''). In *Move or Not*, Sol'' is evaluated to decide whether it is accepted as the new current solution. W_i is then updated according to the evaluation feedback. These steps are repeated until no improvement is obtained after a predefined number ($L_{\max}$) of evaluations. Based on the preliminary experiments, $L_{\max}$ is set to 150.

In the *Move or Not* phase, a *Record-to-Record Travel* scheme [17] is used as the acceptance criterion. When Sol'' has better solution quality than Sol, or the difference of the solution quality between Sol'' and Sol' is less than a predefined value ($DEVIATION$), Sol'' will be accepted as new Sol'. In our algorithm, $DEVIATION$ is set to 2 based on preliminary experiments. In practice, less than 2 km increase in total travel distance is acceptable.

The weight of NS_i is updated during the search. In the literature, most learning indicators focus on the solution quality [54], while the feasibility of search attracts less attention. When updating the weight, a feasibility indicator (FI) is employed in VNS-RLS, which is just the feasibility of the newly generated solution. To reduce the computing time in calculating the updated weights, a simple Credit Assignment Mechanism is employed. If Sol'' is infeasible or cannot be accepted due to its low solution quality, a penalty of 10% reduction of weight

TABLE 3. The list of real-life instances of the Ningbo Port.

Instance	No. of shifts	No. of tasks
NP4-1	4	465
NP4-2	4	405
NP4-3	4	526
NP4-4	4	565
NP4-5	4	765
NP6-1	6	1073
NP6-2	6	920
NP6-3	6	384
NP6-4	6	746
NP6-5	6	557
NP8-1	8	913
NP8-2	8	827
NP8-3	8	786
NP8-4	8	1008
NP8-5	8	798

will be applied to NS_i. Otherwise, if the quality of Sol'' is better than the current best solution Sol, the weight will be increased 10% as a reward. The penalty and reward rates are set based on the preliminary experiments. More discussion on adaptive weight adjustment can be found in [64].

In our study, the contribution of the feasibility indicator and different components in VNS-RLS are analyzed and discussed, results presented in Section 4.5.

4. Benchmark & computational experiments

4.1. Benchmark instances

Bai *et al.* [1] extract 15 instances from the real-life Ningbo Port dataset. In different instances, the planning horizons are 4, 6 or 8 shifts, respectively, of 384 up to 1073 tasks. An artificial instance set including 17 instances is created as well, with different feature combinations on time window tightness (Tight/Loose), workload balance at terminals (Balanced/Unbalanced) and planning horizons of 4 and 8 shifts. Especially, a very large instance with more than 2000 tasks is created as well. The detailed features of instances are presented in Tables 3 and 4. The time window of a task, defined by the time it becomes available and the deadline it must be delivered to its destination terminal, varies from 1 to 2 h, up to 6 shifts.

Comparing to classic VRP benchmarks [21, 59], it is easy to notice that the number of tasks in the Ningbo Port benchmark is quite big. To systematically test the performance of VNS-RLS on diverse instances, we have generated two more datasets scaled down to 25% and 50% respectively by extracting tasks from the complete dataset, keeping the original features (available at http://www.cs.nott.ac.uk/~pszrq/benchmarks.htm).

4.2. Exact approach

In Section 2, the new mathematical model of PVRPTW-O is established. CPLEX is employed to solve it, adopting the default parameter setting. Because CPLEX is a MILP/MIQP solver, some modification should be made to the original model in Section 2 to adapt to the solver. In the original model, the constraints (2.6)–(2.8) are nonlinear, which are reformulated to the linear form as follows,

$$\text{wait}_j \geq a_j - T_j, \qquad \forall j \in N \tag{4.1}$$

$$\text{wait}_j \geq 0, \qquad \forall j \in N \tag{4.2}$$

TABLE 4. The list of artificial instances.

Instance	Configuration	No. of shifts	No. of tasks
LB4-1	Loose, Balanced	4	484
LB4-2	Loose, Balanced	4	396
TB4-3	Tight, Balanced	4	282
TB4-4	Tight, Balanced	4	368
LU4-5	Tight, Unbalanced	4	448
LU4-6	Tight, Unbalanced	4	479
TU4-7	Loose, Unbalanced	4	217
TU4-8	Loose, Unbalanced	4	354
LB8-1	Loose, Balanced	8	592
LB8-2	Loose, Balanced	8	657
TB8-3	Tight, Balanced	8	497
TB8-4	Tight, Balanced	8	621
LU8-5	Tight, Unbalanced	8	551
LU8-6	Tight, Unbalanced	8	559
TU8-7	Loose, Unbalanced	8	607
TU8-8	Loose, Unbalanced	8	525
Large	Mixed, Unbalanced	8	2614

$$B_j = T_j + \text{wait}_j, \qquad \forall j \in N\backslash\{0\} \tag{4.3}$$

$$B_i + l_i + t_{ij} - B_j \leq M \cdot (1 - x_{ij}^s), \qquad \forall i, j \in N\backslash\{0\}, s \in S \tag{4.4}$$

$$\text{slack}_{ij} \geq B_i + l_i + t_{ij} - M \cdot (1 - x_{ij}^s), \qquad \forall i, j \in N \tag{4.5}$$

$$\text{slack}_{ij} \geq 0, \qquad \forall i, j \in N \tag{4.6}$$

$$T_j \geq \sum_{i \in W \cup N\backslash\{j\}} \text{slack}_{ij}, \qquad \forall j \in N \tag{4.7}$$

$$Y_s - T_j \geq M \cdot (1 - x_{ij}^s), \qquad \forall i, j \in N \tag{4.8}$$

$$B_i + l_i - Z_s \leq M \cdot (1 - x_{ij}^s), \qquad \forall i, j \in N. \tag{4.9}$$

The original constraint (2.6) is linearized to three constraints (4.1)–(4.3), correspondingly, the constraint (2.5) is reformulated to (4.4). Here wait_j is a variable that has been introduced to linearize the function max, and M is a parameter with very large value. In constraints (4.5)–(4.7), the slack_{ij} are variables to force T_j to be larger than the assigning B_i. In constraint (4.5), only the assigned B_i is positive value from this formulation. The non-assigning B_i will be deducted by M so that they are negative. Correspondingly, T_j is required to be larger than the summation of slack_{ij} in constraint (4.7). Note that there is only one slack_{ij} in i that is not zero (according to constraints (2.2) and (4.5)), so $\sum_i \text{slack}_{ij}$ equals $\sum_i \max(\text{slack}_{ij})$. Instead of intuitively formulating constraint (4.7) to $T_j \geq \text{slack}_{ij}$ for all $i, j \in N$, the current formulation reduces the number of constraints in the model from $|N| \cdot |N|$ to $|N|$. Constraints (4.8) and (4.9) replace the original constraints (2.7) and (2.8). Adding extra cutting plane constraints to the problem model (*e.g.* subtour elimination constraints) might reduce the scope of search, however, we do not add them in order to avoid increasing the memory required for loading the model in CPLEX.

In the preliminary experiments, the solver runs on a machine of 16 cores (2.6 GHz), 16 GB memory and 24 h running time limit. The solver found feasible solutions for 7 out of 16 the 25% scaled down artificial instances and 2 out of 15 real-life instances, while on the other instances ran out of memory. Therefore, we increase the memory resource to 100 GB while the same running time limit is kept in the later experiments. The results are reported in Tables 5 and 6. In these tables, the objective value of a solution is converted into *Heavy-Loaded Distance Rate* (HLDR), which is widely used by logistic companies in practice, see equation (4.10). Note that

TABLE 5. CPLEX solutions on the 25% scaled down real-life instances.

	NP4-1	NP4-2	NP4-3	NP4-4	NP4-5
HLDR	78.36%	65.14%	64.83%	54.39%	NF
Bound	92.36%	97.04%	100%	97.72%	100%
Time (h)	24	24	24	24	24
	NP6-1	NP6-2	NP6-3	NP6-4	NP6-5
HLDR	NF	NF	54.30%	NF	66.11%
Bound	NF	NF	95.20%	NF	98.39%
Time (h)	24	24	24	+	24
	NP8-1	NP8-2	NP8-3	NP8-4	NP8-5
HLDR	NF	NF	NF	NF	NF
Bound	98.98%	100%	100%	NF	100%
Time (h)	24	24	24	+	24

TABLE 6. CPLEX solutions on the 25% scaled down artificial instances.

	LB4-1	LB4-2	TB4-3	TB4-4	LU4-5	LU4-6	TU4-7	TU4-8
HLDR	66.62%	76.41%	69.91%	69.30%	NF	58.65%	50.37%	55.36%
Bound	100%	94.87%	86.31%	83.51%	79.94%	73.90%	52.17%	66.38%
Time (h)	24	24	13*	19*	24	24	1*	24
	LB8-1	LB8-2	TB8-3	TB8-4	LU8-5	LU8-6	TU8-7	TU8-8
HLDR	NF	NF	56.85%	52.40%	57.42%	NF	47.65%	50.74%
Bound	100%	100%	82.33%	88.75%	78.33%	86.84%	71.59%	70.43%
Time (h)	24	24	24	24	24	24	24	24

objective (2.1) is equivalent to (4.10).

$$\text{HLDR} = \text{Loaded Distance} \,/\, (\text{Loaded Distance} + \text{Unloaded Distance}). \tag{4.10}$$

By using HLDR, the problem becomes a maximization problem. In Tables 5 and 6, NF means no solution is found. Bound is the upper bound of HLDR obtained by CPLEX, while Time is the actual runtime. + indicates that the memory is insufficient to generate a feasible solution, and * represents that the search reached the memory limit while a feasible solution is obtained.

It can be found that, given a large amount of computing resources, it is still difficult to obtain optimal solutions on all the 25% scaled down instances. Except TU4-7, on other instances, the gaps between the obtained solutions and the upper bounds are quite large. On NP6-4 and NP8-4 no feasible solution or upper bound were found by CPLEX, due to memory overflow. It is no doubt that there must be other exact methods may find better solution than CPLEX on this benchmark. However, it still can be concluded that, exact search would be time and resource-consuming in this large scale and nonlinearly constrained problem. The results obtained on the 25% instances provide the upper bounds of the optimal solutions, which can be used to estimate the performance of our proposed heuristic algorithms.

4.3. Construction scheme selection

Completing all tasks on time with the limited vehicle resource is the primary job to logistic companies. Currently, considering the fleet owned by the company and outsourced vehicles, the truck resource in the Ningbo Port is relatively sufficient. But when the number of tasks is large, efficiently using the limited trucks becomes critical. In Section 3.2, three insertion selection tactics are proposed for solution construction. Nine

TABLE 7. Comparison of nine different combinations of insertion selection tactics.

Constructive Heuristic	Mandatory Task tactic	Optional Task tactic	Average HLDR	Average Time (s)
1	Greedy	Greedy	59.16%	3576
2	Greedy	First-Feasible	56.40%	445
3	Greedy	One-Route	58.54%	710
4	First-Feasible	Greedy	58.60%	2302
5	First-Feasible	First-Feasible	56.16%	705
6	First-Feasible	One-Route	57.99%	753
7	One-Route	Greedy	59.16%	3402
8	One-Route	First-Feasible	56.64%	599
9	One-Route	One-Route	58.54%	620

different combinations of tactics are tested on the 100% benchmark to choose the recommended heuristics. Comparison results are presented in Table 7.

Among these nine heuristics in Table 7, the three tactics are applied to mandatory tasks and optional tasks, respectively. The runtime is obtained on a PC with i7 CPU 3.2 GHz and 6 GB memory. It can be found that, heuristics 1 and 7 obtain the best solution quality (the highest average HLDR), while the runtime are significantly higher than the other heuristics. Besides, when Greedy is used on mandatory tasks and First-Feasible is applied to optional tasks (heuristic 2), the fastest constructive speed is obtained, while the obtained average HLDR is not the worst. Applying First-Feasible to both mandatory and optional tasks (heuristic 5) does not bring the fastest heuristic.

The details of constructed solution quality is presented in Tables 8 and 9. It can be found that applying the Greedy heuristics to optional tasks (heuristics 1, 4 and 7) obtains significantly better objective value than those of First-Feasible (heuristics 2, 5 and 8). The HLDR of tactic One-Route group (heuristics 3, 6 and 9) is between the above-mentioned two groups. However, to mandatory tasks, no tactic shows obvious better performance than the others. It can be concluded that, the tactic used on optional tasks is the key issue in devising a constructive heuristic. This is because, in practice, optional tasks always account for a major proportion of total unassigned tasks in one shift, and also mandatory tasks have less available insertion options due to their closeness to their deadlines. On real-life instances, 72% tasks in a shift are optional on average, while the proportion on artificial instances is affected by instance type. On loose instances, the optional task rate can be 100% in every shift, while there may be around 50% mandatory tasks on tight instances. Therefore, the tactic for optional tasks has a greater influence on solution construction.

These constructive heuristics have not shown bias on specific features of instances. Overall, the obtained HLDR values span from 46.79% to 69.93%, higher than 50% on most instances. It should be noted that, two pairs of heuristics (1, 7) and (3, 9) have constructed the same solutions while heuristics 7 and 9 spend less computation time. It is because of that, when the candidate tasks are few, if the existing routes are too tight to insert other tasks, Greedy tactic can only assign a task to the newest route. The same thing occurs to tactic One-Route in the same case, but its computation time is less since it only evaluate one route. As there are few mandatory tasks which are always assigned first in each shift and no randomization in this heuristic, applying tactic Greedy or One-Route to mandatory tasks may generate the same partial routes. In this case, if the tactics applied to optional tasks are the same, their final routes (solutions) generated would be the same as well.

In metaheuristics, the quality of initial solutions does not always directly determine the quality of final improved solutions. Our study also finds that better initial solutions cannot guarantee better final solutions in VNS-RLS. Considering all the nine heuristics can produce feasible solutions on the benchmark instances, in normal cases, the construction heuristic 2 is recommended, as it has in general the fastest construction speed.

TABLE 8. HLDR of initial solutions constructed with the nine construction heuristics (on 100% real-life instances).

Constructive Heuristic	1	2	3	4	5	6	7	8	9
NP4-1	60.14%	**64.10%**	58.64%	60.11%	61.69%	57.63%	60.14%	61.88%	58.64%
NP4-2	**58.42%**	54.13%	58.36%	**58.42%**	54.60%	58.36%	**58.42%**	54.60%	58.36%
NP4-3	**60.51%**	55.88%	59.22%	59.48%	54.73%	57.96%	**60.51%**	55.46%	59.22%
NP4-4	**56.37%**	50.86%	**56.37%**	**56.37%**	50.29%	**56.37%**	**56.37%**	50.29%	**56.37%**
NP4-5	**69.93%**	57.45%	69.68%	**69.93%**	59.98%	69.68%	**69.93%**	59.98%	69.68%
NP6-1	**62.28%**	54.90%	62.18%	62.03%	56.64%	62.06%	**62.28%**	56.74%	62.18%
NP6-2	60.10%	54.94%	**60.18%**	60.06%	54.55%	60.14%	60.10%	54.58%	**60.18%**
NP6-3	**49.66%**	46.79%	49.60%	**49.66%**	47.17%	49.60%	**49.66%**	47.17%	49.60%
NP6-4	**66.99%**	59.16%	63.20%	64.59%	56.92%	62.78%	**66.99%**	57.52%	63.20%
NP6-5	**56.90%**	54.82%	56.48%	56.39%	55.58%	55.52%	**56.90%**	56.22%	56.48%
NP8-1	64.05%	**64.56%**	63.19%	61.07%	62.08%	60.99%	64.05%	64.38%	63.19%
NP8-2	**64.83%**	60.59%	64.74%	63.42%	59.10%	63.21%	**64.83%**	61.01%	64.74%
NP8-3	68.56%	58.45%	68.22%	**68.98%**	58.39%	68.63%	68.56%	58.22%	68.22%
NP8-4	**57.27%**	55.53%	55.74%	57.16%	**57.27%**	55.88%	**57.27%**	56.83%	55.74%
NP8-5	55.42%	55.57%	55.35%	54.88%	54.96%	54.90%	55.42%	**55.58%**	55.35%

Notes. Best obtained HLDR are in bold.

TABLE 9. HLDR of initial solutions constructed with the nine construction heuristics (on 100% artificial instances).

Constructive Heuristic	1	2	3	4	5	6	7	8	9
LB4-1	**59.79%**	54.41%	58.76%	59.42%	53.78%	58.75%	**59.79%**	54.39%	58.76%
LB4-2	**67.30%**	65.07%	66.29%	67.28%	66.92%	66.83%	**67.30%**	65.93%	66.29%
TB4-3	63.14%	63.06%	60.57%	**63.32%**	62.13%	58.95%	63.14%	63.06%	60.57%
TB4-4	**58.19%**	57.67%	57.95%	57.89%	57.55%	57.74%	**58.19%**	57.84%	57.95%
LU4-5	53.84%	52.66%	**53.89%**	51.88%	51.36%	52.68%	53.84%	52.69%	**53.89%**
LU4-6	**61.35%**	59.67%	60.69%	53.79%	52.95%	53.57%	**61.35%**	60.66%	60.69%
TU4-7	48.12%	**48.14%**	47.93%	48.10%	48.11%	47.66%	48.12%	**48.14%**	47.93%
TU4-8	**53.20%**	**53.20%**	52.97%	52.79%	52.66%	52.49%	**53.20%**	**53.20%**	52.97%
LB8-1	**58.32%**	54.64%	**58.32%**	**58.32%**	54.81%	**58.32%**	**58.32%**	54.81%	**58.32%**
LB8-2	65.88%	61.10%	**65.88%**	**65.88%**	61.08%	**65.88%**	65.88%	61.08%	**65.88%**
TB8-3	**59.93%**	59.67%	59.69%	58.74%	58.35%	58.43%	**59.93%**	59.67%	59.69%
TB8-4	54.73%	54.48%	53.59%	54.83%	**55.42%**	54.11%	54.73%	55.07%	53.59%
LU8-5	59.67%	59.70%	57.81%	**62.90%**	62.89%	60.59%	59.67%	59.70%	57.81%
LU8-6	49.63%	49.39%	48.87%	50.77%	**51.00%**	48.66%	49.63%	50.10%	48.87%
TU8-7	56.64%	56.28%	**57.65%**	56.59%	56.29%	57.63%	56.64%	56.28%	**57.65%**
TU8-8	**52.65%**	**52.65%**	**52.65%**	51.70%	51.70%	51.56%	**52.65%**	**52.65%**	**52.65%**

Notes. Best obtained HLDR are in bold.

The two main constraints of PVRPTW-O are the time constraints and the limited trucks. To evaluate and identify effective heuristics in tight constrained scenario, we compare the performances of the nine constructive heuristics on the largest instance of 2614 tasks, results presented in Table 10. It can be found that, when tactic First-Feasible is applied to optional tasks (heuristics 2, 5 and 8), the quality of solutions is poor (lower HLDR and more violations) with around 23 h computation time. Heuristics 1 and 7 produce the same best solutions, while heuristic 7 takes less computation time (7 h less). Heuristic 7 is thus chosen for this kind of very large

TABLE 10. Quality of solutions for the largest instance.

Constructive Heuristic	HLDR	Time (h)	Additional trucks requirement	No. of failed tasks
1	73.38%	35.7	0	56
2	65.49%	23.1	116	98
3	73.24%	30.0	4	59
4	73.18%	30.3	0	67
5	65.01%	22.1	100	118
6	73.31%	22.8	1	57
7	73.38%	28.5	0	56
8	65.00%	22.7	98	118
9	73.24%	23.2	4	59

instances. Other solutions with acceptable quality are generated when tactic Greedy or One-Route is applied to optional tasks (heuristics 3, 4, 6 and 9).

4.4. Impact of the sampling

To evaluate the contribution of *Sampling* introduced in Section 3.4, a variant of VNS-RLS without the *Sampling* scheme and with equal initial weights for all candidate operators is implemented. This variant of VNS with Reinforcement Learning (VNS-RL) is tested on benchmark instances of different sizes. The categorized results of 30 runs on 25% scaled down instances are presented in Tables A.1 and A.2. It can be summarized that VNS-RLS performs better than VNS-RL in both the best found and average HLDR on 19/31 of all instances, but more evaluations were conducted. On 26/31 instances, both VNS-RLS and VNS-RL obtain better results (higher HLDR) than CPLEX with previously mentioned configuration, obtaining feasible solutions on all instances while CPLEX fails on 13 instances.

Results of large 100% instances are presented in Tables A.3 and A.4. In 16/31 instances, solutions with better best-found and average HLDR are found by VNS-RLS. In [1], the HLDR for the relaxed problem removing the unloaded travels from and to the depot are calculated as the upper bounds of the benchmark. Since PVRPTW-O can be relaxed to the same form, those upper bounds are also adopted in this study. In the experiments, the gaps between the obtained solutions and upper bounds are less than 5% on 14 instances. Especially, on the instances of NP4-2, NP4-5, NP6-4, NP8-4 and TU8-8, the objective values of solutions generated are very close to the upper bounds.

However, the contribution of *Sampling* on improving the objective value is not significant. The t-test results on VNS-RLS and VNS-RL (see Tabs. 11 and 12) show that the improvement is significant on only 25% instances. This observation indicates that, by sampling on the search trajectory of single solution-based algorithm, the information collected may not sufficiently reflect the neighbourhood environment and provide valuable guidance for the search. In other words, limited *Sampling* points in single solution-based approaches is not so helpful in improving objective value comparing to its impact in evolutionary approaches. The further discussion on the impact of Sampling indicators is presented in Section 4.5.

The same observation is found on the medium size 50% scaled down instances (see Tabs. A.5 and A.6). VNS-RLS outperforms VNS-RL on 16 out of 31 instances and higher stability (smaller standard deviation) is obtained on 17 instances, which is not significant better. In this single solution-based algorithm, no significant solution quality improvement is achieved by adopting *Sampling*.

TABLE 11. *T*-test between VNS-RLS and VNS-RL on 25% scaled down and 100% real-life instances.

	NP4-1		NP4-2		NP4-3		NP4-4		NP4-5	
Instance scale	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%
T-test result	N	Y	N	N	N	Y	N	N	N	N
	NP6-1		NP6-2		NP6-3		NP6-4		NP6-5	
Instance scale	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%
T-test result	N	N	N	N	N	Y	Y	N	N	N
	NP8-1		NP8-2		NP8-3		NP8-4		NP8-5	
Instance scale	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%
T-test result	Y	Y	N	N	N	N	N	N	Y	N

Notes. Y means significantly different, while N means not.

TABLE 12. T-test between VNS-RLS and VNS-RL on 25% scaled down and 100% artificial instances.

	LB4-1		LB4-2		TB4-3		TB4-4		LU4-5		LU4-6		TU4-7		TU4-8	
Scale	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%
T-test	N	N	N	N	Y	N	N	Y	N	N	Y	N	N	N	N	N
	LB8-1		LB8-2		TB8-3		TB8-4		LU8-5		LU8-6		TU8-7		TU8-8	
Scale	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%
T-test	Y	N	Y	Y	N	N	N	N	Y	N	N	N	N	N	N	Y

4.5. Feasibility indicators & feasible solution space exploration

In *Sampling* and *Local Search*, feasibility measure is employed aiming to improve the search efficiency. The hypothesis is that, by using the feasibility indicators, the infeasible solutions explored during the search will decrease. To validate this hypothesis, the feasibility indicators in *Sampling* (E^c) and *Local Search* (FI) are removed respectively leading to three new algorithm variants (VNS-RLS-NoEc, VNS-RLS-NoFI, VNS-RL-NoFI). The detailed experiment results are also presented in Tables A.2–A.5. Comparing the best and average solutions, VNS-RLS-NoFI produces the most best results on medium and large instances, while VNS-RLS-NoEc outperforms the other variants on small instances.

We record the amount of infeasible solutions explored over every 10 000 evaluations during the search and present the trend of infeasible solutions explored by the five algorithm variants. Figure 3 presents the typical results of two instances. It is easy to see that, without *Sampling*, both VNS-RL and VNS-RL-NoFI explore much more infeasible solutions than the other three variants throughout of the search. The infeasible rates of the three VNS-RLS variants decrease at the beginning of search and stay at a low level afterwards, showing the effects of Sampling on reducing infeasible neighbourhood moves. At each VNS iteration, by estimating the surrounding environment of the starting solution, Sampling leads Local Search to the search regions with high feasibility. Using feasibility indicators in both Sampling and Local Search, VNS-RLS achieves the lowest infeasible rate dring the search. In addition, in Figure 3, the two lines of VNS-RL and VNS-RL-NoFI are shorter than the other three VNS-RLS variants, indicating the two variants converge earlier.

It can be concluded that, both *Sampling* and *Feasibility indicator* can reduce the infeasible rate of neighbourhood moves, while *Sampling* brings a greater impact, obtaining better solutions within the same computational time. *Sampling*, even without the feasibility indicator E^c, can significantly decrease the infeasible rate. This indicates that the other two indicators (E^a and E^b) apt to guide the search to feasible solution regions, meaning the high quality neighbourhood has higher feasibility. The use of feasibility indicator in *Sampling* and *Local Search* contributes to a higher probability of exploring feasible neighbourhood moves during the search.

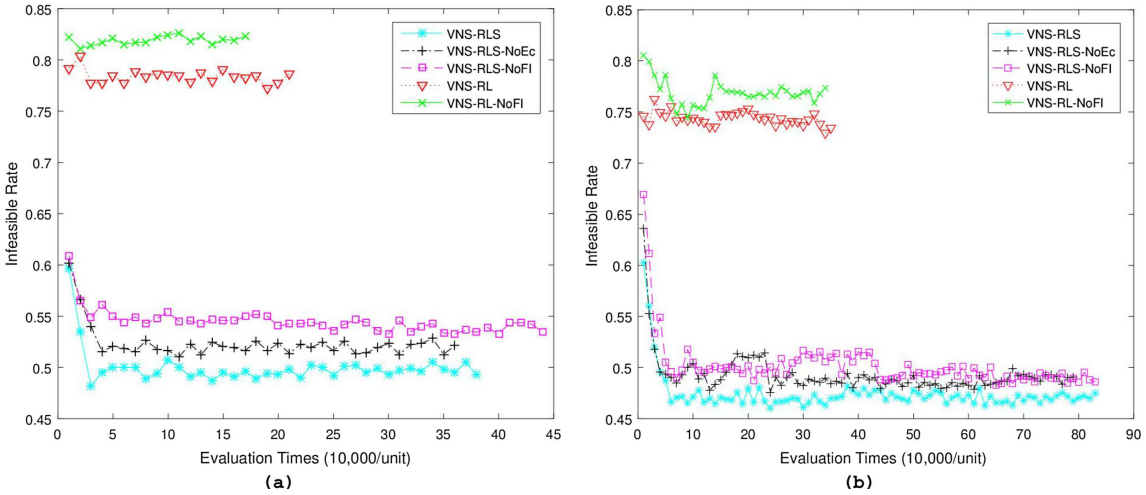

FIGURE 3. Comparison of infeasible rates on the 25% NP4-3 (a) and 100% LB4-1 (b).

It should be noted that, the lowest infeasible rate does not necessarily contribute to the best objective function value. The removal of feasibility indicators may also increase the flexibility and diversification of the search, thus improves the search performance, *e.g.* on the LB8-1 100% instance, VNS-RLS-NoFI is the only one variant which finds the solution with objective value 87.49%. It is almost 2% higher than the other four variants. Overall, VNS-RLS-NoEc and VNS-RLS-NoFI produce the most best-found solutions and average solutions in all the five variants.

4.6. Comparison with the general VNS and state-of-the-art algorithms

PVRPTW-O is a new model in the literature, thus there is not many existing algorithms applied to it directly. A general VNS algorithm (VND) [28] with the same operators as VNS-RLS is implemented to be compared. Besides, considering the similarity between PVRP and PVRPTW-O, two state-of-the-art methodologies for PVRP, namely RVNS [48] and FVNS [29], are also modified (*e.g.* initial solution construction, side constraints consideration and problem specific parameter setting) and applied to PVRPTW-O for comparison in our research. Both RVNS and FVNS use the VNS framework as well. Apart from the different operators used (*i.e.* different shaking operators and neighbourhood structures), the main difference between them is that the shaking operators are randomly selected in RVNS but applied in a fixed sequence in FVNS. In addition, due to the time constraints and multiple periods in PVRPTW-O, algorithms for Open-VRP and Open-VRPTW are either inapplicable or not suitable for PVRPTW-O, thus they are not compared in this study. Chen *et al.* [10] develop an adaptive large neighbourhood search algorithm (VD-ALNS) which has large perturbation in the search. This algorithm is also applied to the Ningbo Port problem to investigate the improvement from VNS-RLS.

The comparison results are presented in Table 13, which gives the deterioration comparing to the results of VNS-RLS-NoEc. Generally, our algorithm outperforms the other four algorithms on most categories of instances on both the solution quality and stability. VND produces significantly better results than RVNS and FVNS, while RVNS and FVNS virtually tie. It indicates that the customized neighbourhood operators in VNS-RLS contribute much to the improvement of algorithm performance in this problem, since RVNS and FVNS use different neighbourhood operators. The operators in VNS-RLS bring systematical exploration and perturbation at different levels, cooperating with the proposed reinforcement learning scheme and the effective indicators, thus exploration and exploitation are better balanced in the search. The solution deterioration is smaller on

TABLE 13. Result deterioration comparing with VNS-RLS.

		NP4	NP6	NP8	LB4	LB8	TB4	TB8	LU4	LU8	TU4	TU8
VND	Best	−0.05%	−0.22%	−0.28%	−0.24%	−0.33%	−0.38%	−0.66%	−0.71%	−0.48%	−0.14%	−0.31%
	Average	−0.31%	−0.29%	−0.36%	−0.46%	−1.36%	−0.48%	−1.57%	−0.62%	−0.47%	−0.62%	−0.55%
	S.D.	−0.03%	−0.10%	0%	−0.15%	−0.19%	−0.13%	−0.34%	−0.18%	−0.04%	−0.06%	−0.04%
RVNS	Best	−1.77%	−2.36%	−1.73%	−3.42%	−5.66%	−1.00%	−2.58%	−1.58%	−1.39%	−0.47%	−0.55%
	Average	−2.23%	−3.43%	−2.11%	−4.21%	−6.62%	−0.74%	−2.19%	−2.52%	−2.02%	−0.42%	−0.80%
	S.D.	−0.33%	−0.50%	−0.34%	−0.49%	−0.51%	0.06%	0.20%	−0.20%	−0.21%	0.15%	0.04%
FVNS	Best	−1.86%	−2.26%	−1.64%	−2.67%	−5.59%	−1.55%	−3.16%	−2.07%	−1.99%	−0.47%	−0.73%
	Average	−2.53%	−3.94%	−2.58%	−4.65%	−7.21%	−0.94%	−2.27%	−2.83%	−2.50%	−0.48%	−0.83%
	S.D.	−0.35%	−0.60%	−0.50%	−0.71%	−0.71%	0.20%	0.24%	−0.15%	−0.11%	0.16%	0.07%
VD-ALNS	Best	−0.38%	0.49%	−1.60%	0.80%	0.57%	−3.21%	−0.24%	−0.53%	−0.98%	−3.46%	−0.40%
	Average	−0.48%	−0.33%	−1.82%	0.67%	0.67%	−3.35%	−0.34%	−2.16%	−1.22%	−4.30%	−0.78%
	S.D.	−0.19%	−0.15%	−0.04%	−0.07%	−0.06%	−0.16%	−0.13%	−1.10%	−0.08%	−0.37%	−0.14%

TABLE 14. The growth of HLDR brought by the PVRPTW-O scheduling scheme.

Instances	NP4	NP6	NP8	LB4	TB4	LU4	TU4	LB8	TB8	LU8	TU8
Growth	0.59%	1.44%	0.29%	0.80%	4.48%	3.39%	4.31%	1.98%	4.51%	5.11%	6.26%

tight and unbalance instances (TU), which shows the stronger search ability of VNS-RLS on the larger search space caused by the loose time windows and balanced workload. However, RVN and FVNS show better stability on tight instances (TB4, TB8, TU4 and TU8) than VNS-RLS. In terms of VD-ALNS, it shows lower stability than VNS-RLS. Except LB4 and LB8 instances, VNS-RLS outperforms VD-ALNS on vast majority of other instances. The larger perturbation in VD-ALNS achieves more solution improvement on loose and balance instances.

4.7. Comparison with the previous scheduling scheme

Comparing the PVRPTW-O model with the study of [1, 8], the biggest difference is that the shift-change between day-shift and night-shift can occur at any terminal in PVRPTW-O. The open routes in each shift may bring the growth of HLDR to the fleet. Table 14 presents the increase of HDLR obtained by applying PVRPTW-O, where the values are the differences of HLDR between the best-found results on both models. It can be found that the use of PVRPTW-O increases the utility of vehicle to varying degrees, while the improvement on artificial instances is larger. This scheduling scheme can further save operating cost and promote efficiency for the company.

5. CONCLUSIONS

A Periodic Vehicle Routing Problem with Time Windows and Open Routes (PVRPTW-O) derived from a real-life container transportation problem is investigated in this paper. A node-based mathematical model is established by combining service activities into task nodes. This problem model has a huge solution space with nonlinear constraints. The study shows that, comparing to the previous scheduling schemes, the utility of vehicles is increased in the PVRPTW-O model. The experiment results on both real-life and artificial benchmarks indicate that it is unrealistic to address this problem with exact methods even with a large amount of computation resources, thus a Variable Neighbourhood Search algorithm with Reinforcement Learning and Sampling (VNS-RLS) for PVRPTW-O is developed accordingly.

In VNS-RLS, an urgency level-based insertion heuristic is devised to construct initial feasible solutions. After classifying tasks according to their urgency levels (mandatory and optional), nine different insertion selection

tactic combinations are investigated and compared. Experiment results show that the tactic applied to optional tasks mainly determines the performance and effectiveness of the heuristic. When the resource of trucks is sufficient, applying the *First-Feasible* tactic to optional tasks is of the fastest speed. In the case of extremely large amount of tasks, *One-Route* tactic on mandatory tasks and *Greedy* tactic on optional tasks is recommended to produce better solutions within a less computation time.

In the improvement phase, a *Sampling* scheme is proposed to investigate the neighbourhood of the starting solution, providing an initial search direction for the local search with reinforcement learning. Experiment results show that the contribution of Sampling to solution improvement is not significant. This indicates that, Sampling is not as powerful as it is in evolutionary algorithms due to search region changes significantly in single solution-based metaheuristics. In addition, to reduce the number of infeasible solutions generated in the search, a feasibility indicator is used in both phases of Sampling and Local Search. It is found that Sampling greatly reduces the infeasible solutions generated, while the feasibility indicators make further contribution as well. Compared to the results of exact method and solution upper bounds, the proposed algorithms generate promising results with strong search ability within large search spaces.

On several instances, the gaps between the obtained objective values and upper bounds are still large. Considering that the operators used in VNS-RLS bring relatively small changes to solutions, advanced techniques with balanced exploration and exploitation are worthy of further investigation in future work. In addition, more practical objectives and constraints in real-life problems, such as the balance of workload and carbon emission reduction, may also be introduced to the PVRPTW-O model.

Appendix A. Detailed results of experiments

Table A.1. Comparison on 25% scaled down artificial instances.

Instance		LB4	TB4	LU4	TU4	LB8	TU8	LU8	TU8
VNS-RLS	Best	79.68%	67.66%	60.54%	**51.86%**	92.44%	64.08%	66.07%	**54.53%**
	Average	77.95%	**66.38%**	59.87%	51.54%	90.74%	62.40%	65.09%	**53.98%**
	Eval	296 501	273 258	286 733	166 087	523 395	364 296	448 189	292 300
	S.D.	**0.86%**	**0.70%**	0.37%	0.31%	1.03%	0.88%	0.53%	0.29%
VNS-RL	Best	79.45%	67.49%	60.61%	51.86%	92.18%	63.38%	65.83%	54.33%
	Average	77.86%	66.17%	59.89%	51.52%	90.02%	62.14%	64.83%	53.92%
	Eval	132 169	125 310	129 135	78 463	225 891	154 865	183 406	124 411
	S.D.	0.91%	0.79%	0.41%	0.31%	1.28%	**0.71%**	0.54%	0.24%
VNS-RLS No Ec	Best	**80.17%**	67.75%	**60.90%**	**51.86%**	92.41%	**64.68%**	**66.17%**	54.38%
	Average	78.21%	66.37%	**59.96%**	**51.61%**	**90.93%**	**62.52%**	**65.22%**	53.86%
	Eval	297 278	149 469	305 959	179 892	520 241	407 346	439 039	275 322
	S.D.	1.02%	**0.70%**	0.47%	0.32%	0.91%	0.85%	0.47%	**0.24%**
VNS-RLS No FI	Best	79.51%	**67.91%**	60.63%	**51.86%**	**92.82%**	64.22%	65.86%	54.40%
	Average	77.99%	66.27%	59.93%	51.55%	90.74%	62.38%	65.13%	53.95%
	Eval	297 269	306 646	304 078	178 385	468 562	356 700	387 061	282 661
	S.D.	0.81%	0.81%	0.44%	0.33%	1.14%	0.75%	0.41%	0.24%
VNS-RL No FI	Best	80.00%	67.39%	60.31%	**51.86%**	92.08%	63.02%	65.85%	54.29%
	Average	**78.22%**	66.15%	59.92%	51.54%	90.81%	62.07%	65.07%	**53.98%**
	Eval	120 067	137 449	147 877	76 068	223 702	159 162	197 951	133 322
	S.D.	1.09%	0.80%	0.31%	**0.24%**	**0.71%**	0.77%	**0.43%**	0.25%
CPLEX		71.52%	69.61%	NF	52.87%	NF	54.63%	57.42%	49.20%
Upper bound		97.44%	84.91%	76.92%	59.28%	100%	85.54%	82.59%	71.01%

Notes. Eval: average evaluation times; **S.D.**: standard deviation of obtained HLDR. Best heuristic solutions are in bold.

TABLE A.2. Comparison on 25% scaled down real-life instances.

Instance		NP4	NP6	NP8
VNS-RLS	Best	72.16%	75.14%	73.03%
	Average	70.99%	73.95%	**72.21%**
	Eval	335 233	482 232	512 479
	S.D.	0.66%	0.70%	0.45%
VNS-RL	Best	72.14%	75.23%	72.96%
	Average	70.87%	73.95%	71.97%
	Eval	157 251	230 364	230 822
	S.D.	0.69%	0.75%	0.47%
VNS-RLS No Ec	Best	71.97%	74.98%	72.64%
	Average	71.10%	74.05%	71.79%
	Eval	354 882	496 22	504 675
	S.D.	**0.63%**	0.74%	0.51%
VNS-RLS No FI	Best	**72.49%**	**75.39%**	**73.04%**
	Average	**71.11%**	**74.31%**	72.13%
	Eval	361 076	471 321	498 858
	S.D.	**0.63%**	0.69%	0.48%
VNS-RL No FI	Best	71.72%	75.07%	72.95%
	Average	70.75%	74.11%	72.14%
	Eval	155 767	219 621	226 467
	S.D.	0.65%	**0.65%**	**0.44%**
CPLEX		65.68%	NF	NF
Upper bound		97.42%	NF	99.75%

Notes. Eval: average evaluation times; **S.D.**: standard deviation of obtained HLDR. Best heuristic solutions are in bold.

TABLE A.3. Comparison on 100% artificial instances.

Instance		LB4	TB4	LU4	TU4	LB8	TU8	LU8	TU8
VNS-RLS	Best	75.80%	**71.22%**	66.27%	53.43%	90.40%	67.75%	68.18%	57.04%
	Average	75.16%	**70.44%**	65.86%	53.24%	88.35%	67.13%	67.56%	**56.77%**
	Eval	629 229	702 774	747 125	275 158	142 2140	1 021 055	939 376	737 121
	S.D.	0.44%	0.52%	**0.25%**	0.14%	1.14%	0.55%	0.41%	0.20%
VNS-RL	Best	76.03%	71.05%	66.29%	53.43%	89.18%	68.14%	68.16%	56.94%
	Average	74.92%	69.94%	65.88%	53.23%	87.35%	66.97%	67.40%	56.64%
	Eval	321 166	327 571	346 889	166 356	635 497	487 487	467 465	336 660
	S.D.	0.58%	0.59%	0.36%	0.16%	1.13%	0.81%	0.46%	0.18%
VNS-RLS No Ec	Best	75.80%	70.78%	66.40%	**53.51%**	89.76%	68.22%	68.11%	**57.05%**
	Average	75.32%	69.98%	**65.95%**	**53.32%**	88.47%	**67.37%**	67.64%	56.72%
	Eval	630 226	625 684	753 381	345 285	1 289 204	1 074 057	1 070 581	715 918
	S.D.	**0.32%**	**0.46%**	0.30%	**0.12%**	1.00%	0.57%	**0.28%**	0.19%
VNS-RLS No FI	Best	**76.26%**	70.98%	66.29%	53.43%	**90.81%**	**68.28%**	**68.36%**	56.89%
	Average	**75.39%**	70.40%	65.86%	53.16%	**88.94%**	67.28%	**67.77%**	56.69%
	Eval	697 442	880 954	771 560	313 368	1 270 641	1 118 612	1 050 605	671 000
	S.D.	0.50%	0.46%	0.34%	0.19%	0.53%	0.61%	0.33%	0.14%
VNS-RL No FI	Best	75.91%	70.88%	**66.43%**	53.45%	89.88%	68.02%	67.98%	56.95%
	Average	75.12%	69.80%	65.93%	53.18%	88.30%	**67.37%**	67.43%	56.69%
	Eval	296 656	158 444	342 112	172 752	634 962	517 818	464 796	370 338
	S.D.	0.43%	0.77%	0.31%	0.24%	**0.93%**	**0.43%**	0.33%	**0.16%**
Upper bound		82.90%	86.40%	74.29%	63.78%	98.12%	89.75%	72.82%	63.42%

Notes. Eval: average evaluation times; **S.D.**: standard deviation of obtained HLDR. Best heuristic solutions are in bold.

TABLE A.4. Comparison on 100% real-life instances.

Instance		NP4	NP6	NP8
VNS-RLS	Best	74.70%	74.58%	71.66%
	Average	73.95%	73.77%	71.22%
	Eval	562 726	836 272	906 966
	S.D.	0.37%	0.41%	0.25%
VNS-RL	Best	74.34%	74.53%	71.71%
	Average	73.77%	73.83%	71.31%
	Eval	368 425	522 249	485 562
	S.D.	0.41%	0.46%	0.23%
VNS-RLS No Ec	Best	74.66%	74.45%	71.77%
	Average	74.01%	73.77%	71.49%
	Eval	599 634	949 295	1 045 616
	S.D.	**0.35%**	0.49%	**0.21%**
VNS-RLS No FI	Best	**74.74%**	**74.58%**	71.92%
	Average	**74.03%**	**74.09%**	**71.53%**
	Eval	707 591	1 011 850	1 008 835
	S.D.	0.39%	**0.30%**	0.25%
VNS-RL No FI	Best	74.32%	74.49%	**71.93%**
	Average	73.75%	73.99%	71.39%
	Eval	369 997	492 044	510 486
	S.D.	0.41%	0.33%	0.32%
Upper bound		79.03%	78.55%	74.41%

Notes. Eval: average evaluation times; **S.D.**: standard deviation of obtained HLDR. Best heuristic solutions are in bold.

TABLE A.5. Comparison on 50% scaled down artificial instances.

Instance		LB4	TB4	LU4	TU4	LB8	TU8	LU8	TU8
VNS-RLS	Best	75.72%	**68.95%**	64.16%	55.48%	91.41%	66.90%	67.78%	57.40%
	Average	74.87%	67.67%	63.33%	55.27%	89.33%	65.90%	67.15%	57.12%
	Eval	382 057	369 850	487 188	241 151	769 643	635 862	652 194	352 426
	S.D.	0.52%	0.71%	0.43%	**0.16%**	1.57%	0.63%	0.43%	0.14%
VNS-RL	Best	75.36%	68.48%	64.00%	**55.72%**	91.09%	66.52%	67.44%	57.38%
	Average	74.60%	67.52%	63.48%	55.26%	89.24%	65.94%	66.78%	57.12%
	Eval	196 713	181 361	237 730	122 769	408 976	300 379	250 690	156 436
	S.D.	0.67%	**0.58%**	0.39%	0.22%	1.25%	0.47%	**0.37%**	0.18%
VNS-RLS No Ec	Best	75.43%	68.76%	**64.57%**	55.48%	91.37%	66.70%	**67.87%**	57.36%
	Average	74.83%	67.67%	63.66%	55.21%	89.93%	65.90%	67.17%	57.14%
	Eval	371 437	403 755	559 215	215 794	839 269	551 066	600 618	359 232
	S.D.	**0.41%**	0.69%	0.49%	0.17%	**0.79%**	**0.45%**	0.49%	0.16%
VNS-RLS No FI	Best	75.40%	68.60%	64.18%	55.49%	**92.21%**	**67.29%**	67.81%	**57.44%**
	Average	**74.90%**	**67.70%**	63.70%	**55.28%**	**90.26%**	**66.10%**	**67.18%**	57.09%
	Eval	394 147	434 742	500 203	265 208	944 019	611 504	624 228	319 457
	S.D.	0.43%	0.54%	0.34%	0.13%	1.48%	0.76%	0.53%	0.21%
VNS-RL No FI	Best	**75.73%**	68.50%	64.23%	55.61%	91.57%	66.86%	67.65%	57.37%
	Average	74.73%	67.58%	**63.80%**	55.22%	89.44%	65.95%	66.86%	**57.15%**
	Eval	184 352	208 175	236 189	118 093	381 723	281 159	269 336	199 711
	S.D.	0.59%	0.71%	**0.31%**	0.23%	1.04%	0.57%	0.52%	**0.12%**

Notes. Eval: average evaluation times; **S.D.**: standard deviation of obtained HLDR. Best heuristic solutions are in bold.

TABLE A.6. Comparison on 50% scaled down real-life instances.

Instance		NP4	NP6	NP8
VNS-RLS	Best	**71.31%**	72.66%	71.68%
	Average	70.17%	71.87%	70.99%
	Eval	482 326	714 533	753 081
	S.D.	0.66%	0.54%	0.40%
VNS-RL	Best	71.09%	72.50%	71.44%
	Average	70.07%	71.84%	70.93%
	Eval	236 451	334 156	375 684
	S.D.	0.69%	0.51%	0.34%
VNS-RLS No Ec	Best	71.18%	72.76%	71.55%
	Average	**70.34%**	72.06%	71.03%
	Eval	499 798	594 148	551 098
	S.D.	0.49%	**0.51%**	**0.32%**
VNS-RLS No FI	Best	71.00%	**72.98%**	71.54%
	Average	70.33%	72.02%	71.04%
	Eval	514 525	692 252	779 342
	S.D.	**0.43%**	0.51%	0.35%
VNS-RL No FI	Best	71.22%	72.97%	**71.75%**
	Average	70.26%	**72.15%**	**71.10%**
	Eval	244 092	345 314	357 288
	S.D.	0.58%	0.58%	0.43%

Notes. Eval: average evaluation times; **S.D.**: standard deviation of obtained HLDR. Best heuristic solutions are in bold.

Acknowledgements. This research is supported by the National Natural Science Foundation of China (Grant No. 71471092), Zhejiang Natural Science Foundation (Grant No. LR17G010001), Ningbo Science & Technology Bureau (Grant No. 2014A35006), School of Computer Science, the University of Nottingham and SF Technology Co. Ltd. We are grateful for access to the University of Nottingham High Performance Computing Facility.

REFERENCES

[1] R. Bai, N. Xue, J. Chen and G.W. Roberts, A set-covering model for a bidirectional multi-shift full truckload vehicle routing problem. *Transp. Res. Part B: Methodol.* **79** (2015) 134–148.
[2] B.M. Baker and M.A. Ayechew, A genetic algorithm for the vehicle routing problem. *Comput. Oper. Res.* **30** (2003) 787–800.
[3] J. Brandão, A tabu search algorithm for the open vehicle routing problem. *Eur. J. Oper. Res.* **157** (2004) 552–564.
[4] O. Bräysy and M. Gendreau, Metaheuristics for the vehicle routing problem with time windows. Report STF42 A1025 (2001).
[5] O. Bräysy and M. Gendreau, Vehicle routing problem with time windows, Part II: Metaheuristics. *Transp. Sci.* **39** (2005) 119–139.
[6] J. Brito, F.J. Martínez, J. Moreno and J.L. Verdegay, An aco hybrid metaheuristic for close–open vehicle routing problems with time windows and fuzzy constraints. *Appl. Soft Comput.* **32** (2015) 154–163.
[7] E.K. Burke, M. Gendreau, G. Ochoa and J.D. Walker, Adaptive iterated local search for cross-domain optimisation. In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. ACM (2011) 1987–1994.
[8] J. Chen, R. Bai, R. Qu and G. Kendall, A task based approach for a real-world commodity routing problem. In: *2013 IEEE Workshop on Computational Intelligence in Production And Logistics Systems (CIPLS)*. IEEE (2013) 1–8.
[9] B. Chen, R. Qu, R. Bai and H. Ishibuchi, A variable neighbourhood search algorithm with compound neighbourhoods for VRPTW. In: *Proceedings of the 5th International Conference on Operations Research and Enterprise Systems (ICORES 2016), Rome, Italy*. SCITEPRESS (2016) 25–35.
[10] B. Chen, R. Qu and H. Ishibuchi, Variable-depth adaptive large meighbourhood search algorithm for open periodic vehicle routing problem with time windows. In: *Proceedings of the International Conference on Harbor, Maritime and Multimodal Logistic Modelling and Simulation (HMS 2017), Barcelona, Spain* (2017) 25–34.
[11] G. Clarke and J.W. Wright, Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.* **12** (1964) 568–581.

[12] J.-F. Cordeau, G. Laporte and A. Mercier, A unified tabu search heuristic for vehicle routing problems with time windows. *J. Oper. Res. Soc.* **52** (2001) 928–936.

[13] J.-F. Cordeau, G. Laporte and A. Mercier, Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *J. Oper. Res. Soc.* **55** (2004) 542–546.

[14] J.-F. Cordeau, G. Laporte, M.W. Savelsbergh and D. Vigo, Vehicle routing. *Handbooks Oper. Res. Manage. Sci.* **14** (2007) 367–428.

[15] A. Danandeh, M. Ghazanfari, R. Tavakoli-Moghaddam and M. Alinaghian, A swift heuristic algorithm based on capacitated clustering for the open periodic vehicle routing problem. In: *Proceedings of the 9th WSEAS International Conference on Artificial intelligence, Knowledge Engineering and Data Bases, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA* (2010) 208–214.

[16] G.B. Dantzig and J.H. Ramser, The truck dispatching problem. *Manage. Sci.* **6** (1959) 80–91.

[17] G. Dueck, New optimization heuristics: the great deluge algorithm and the record-to-record travel. *J. Comput. Phys.* **104** (1993) 86–92.

[18] B. Eksioglu, A.V. Vural and A. Reisman, The vehicle routing problem: a taxonomic review. *Comput. Ind. Eng.* **57** (2009) 1472–1483.

[19] G. Eppen and L. Schrage, Centralized ordering policies in a multi-warehouse system with lead times and random demand. Multi-Level Prod./Inventory Control Syst.: Theory Pract.. In Vol. 16. North-Holland (1981) 51–67.

[20] Z. Fu, R. Eglese and L.Y. Li, A new tabu search heuristic for the open vehicle routing problem. *J. Oper. Res. Soc.* **56** (2005) 267–274.

[21] H. Gehring and J. Homberger, A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: Vol. 2 of *Proceedings of EUROGEN99*. Citeseer (1999) 57–64.

[22] M. Gendreau, J.-Y. Potvin, O. Bräumlaysy, G. Hasle and A. Løkketangen, Metaheuristics for the vehicle routing problem and its extensions: a categorized bibliography. In: The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, Boston, MA (2008) 143–169.

[23] B.E. Gillett and L.R. Miller, A heuristic algorithm for the vehicle-dispatch problem. *Oper. Res.* **22** (1974) 340–349.

[24] B. Golden, A. Assad, L. Levy and F. Gheysens, The fleet size and mix vehicle routing problem. *Comput. Oper. Res.* **11** (1984) 49–66.

[25] B.L. Golden, S. Raghavan and E.A. Wasil, The Vehicle Routing Problem: Latest Advances and New Challenges. In Vol. 43. Springer Science & Business Media (2008).

[26] L. Guiyun, An improved ant colony algorithm for open vehicle routing problem with time windows. In: Vol. 2 of *2009 International Conference on Information Management, Innovation Management and Industrial Engineering*. IEEE (2009) 616–619.

[27] L. Guiyun, Research on open vehicle routing problem with time windows based on improved genetic algorithm. In: *International Conference on Computational Intelligence and Software Engineering, 2009. CiSE 2009*. IEEE (2009) 1–5.

[28] P. Hansen, N. Mladenović and J.A.M. Pérez, Variable neighbourhood search: methods and applications. *Ann. Oper. Res.* **175** (2010) 367–407.

[29] V.C. Hemmelmayr, K.F. Doerner and R.F. Hartl, A variable neighborhood search heuristic for periodic routing problems. *Eur. J. Oper. Res.* **195** (2009) 791–802.

[30] V.C. Hemmelmayr, J.-F. Cordeau and T.G. Crainic, An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Comput. Oper. Res.* **39** (2012) 3215–3228.

[31] G. Ioannou, M. Kritikos and G. Prastacos, A greedy look-ahead heuristic for the vehicle routing problem with time windows. *J. Oper. Res. Soc.* **52** (2001) 523–537.

[32] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation. *Soft Comput. Fusion Found. Methodol. App.* **9** (2005) 3–12.

[33] N. Jozefowiez, F. Semet and E.-G. Talbi, Multi-objective vehicle routing problems. *Eur. J. Oper. Res.* **189** (2008) 293–309.

[34] T. Kaji and A. Ohuchi, A simulated annealing algorithm with the random compound move for the sequential partitioning problem of directed acyclic graphs. *Eur. J. Oper. Res.* **112** (1999) 147–157.

[35] G. Laporte, M. Gendreau, J.-Y. Potvin and F. Semet, Classical and modern heuristics for the vehicle routing problem. *Int. Trans. Oper. Res.* **7** (2000) 285–300.

[36] J.K. Lenstra and A. Kan, Complexity of vehicle routing and scheduling problems. *Networks* **11** (1981) 221–227.

[37] A.N. Letchford, J. Lysgaard and R.W. Eglese, A branch-and-cut algorithm for the capacitated open vehicle routing problem. *J. Oper. Res. Soc.* **58** (2007) 1642–1651.

[38] F. Li, B. Golden and E. Wasil, The open vehicle routing problem: algorithms, large-scale test problems, and computational results. *Comput. Oper. Res.* **34** (2007) 2918–2930.

[39] S. Lin, Computer solutions of the traveling salesman problem. *Bell Syst. Tech. J.* **44** (1965) 2245–2269.

[40] Q. Lin, Z. Liu, Q. Yan, Z. Du, C.A.C. Coello, Z. Liang, W. Wang and J. Chen, Adaptive composite operator selection and parameter control for multiobjective evolutionary algorithm. *Inf. Sci.* **339** (2016) 332–352.

[41] R. Liu and Z. Jiang, The close–open mixed vehicle routing problem. *Eur. J. Oper. Res.* **220** (2012) 349–360.

[42] T. Lourens, *Using population-based incremental learning to optimize feasible distribution logistic solutions*. Thesis, University of Stellenbosch, Stellenbosch (2005).

[43] G. Maps, Google maps. Accessed: 2018-05-11. https://www.google.co.uk/maps/@29.8715435,121.8372319,12z/data=!3m1!4b1!4m2!6m1!1s1IPQurvRAx3x96-V7XEUw6h9kmFs (2018).

[44] M. Mourgaya and F. Vanderbeck, Column generation based heuristic for tactical planning in multi-period vehicle routing. *Eur. J. Oper. Res.* **183** (2007) 1028–1041.
[45] I. Or, Traveling Salesman-type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking. Xerox University Microfilms (1976).
[46] J. Park and B.-I. Kim, The school bus routing problem: a review. *Eur. J. Oper. Res.* **202** (2010) 311–319.
[47] A. Perwira Redi, M.F. Maghfiroh and V.F. Yu, An improved variable neighborhood search for the open vehicle routing problem with time windows. In: *2013 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. IEEE (2013) 1641–1645.
[48] S. Pirkwieser and G.R. Raidl, A variable neighborhood search for the periodic vehicle routing problem with time windows. In: *Proceedings of the 9th EU/meeting on Metaheuristics for Logistics and Vehicle Routing, Troyes, France* (2008) 23–24.
[49] D. Pisinger and S. Ropke, A general heuristic for vehicle routing problems. *Comput. Oper. Res.* **34** (2007) 2403–2435.
[50] J.-Y. Potvin and J.-M. Rousseau, An exchange heuristic for routeing problems with time windows. *J. Oper. Res. Soc.* **46** (1995) 1433–1446.
[51] J.-Y. Potvin, T. Kervahut, B.-L. Garcia and J.-M. Rousseau, The vehicle routing problem with time windows Part I: tabu search. *INFORMS J. Comput.* **8** (1996) 158–164.
[52] A. Rahimi-Vahed, T. Gabriel Crainic, M. Gendreau and W. Rei, Fleet-sizing for multi-depot and periodic vehicle routing problems using a modular heuristic algorithm. *Comput. Oper. Res.* **53** (2015) 9–23.
[53] P.P. Repoussis, C.D. Tarantilis and G. Ioannou, The open vehicle routing problem with time windows. *J. Oper. Res. Soc.* **58** (2007) 355–367.
[54] S. Ropke and D. Pisinger, An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **40** (2006) 455–472.
[55] S. Ropke and D. Pisinger, A unified heuristic for a large class of vehicle routing problems with backhauls. *Eur. J. Oper. Res.* **171** (2006) 750–775.
[56] M.W. Savelsbergh, The vehicle routing problem with time windows: minimizing route duration. *ORSA J. Comput.* **4** (1992) 146–154.
[57] K. Schopka and H. Kopfer, An Adaptive Large Neighborhood Search for the Reverse Open Vehicle Routing Problem with Time Windows. Springer (2016) 243–257.
[58] J.E. Smith and T.C. Fogarty, Operator and parameter adaptation in genetic algorithms. *Soft Comput.* **1** (1997) 81–87.
[59] M.M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* **35** (1987) 254–265.
[60] J.A. Soria Alcaraz, G. Ochoa, M. Carpio and H. Puga, Evolvability metrics in adaptive operator selection. In: *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*. ACM (2014) 1327–1334.
[61] É. Taillard, P. Badeau, M. Gendreau, F. Guertin and J.-Y. Potvin, A tabu search heuristic for the vehicle routing problem with soft time windows. *Transp. Sci.* **31** (1997) 170–186.
[62] C.D. Tarantilis, G. Ioannou, C.T. Kiranoudis and G.P. Prastacos, A threshold accepting approach to the open vehicle routing problem. *RAIRO: OR* **38** (2004) 345–360.
[63] C.D. Tarantilis, G. Ioannou, C.T. Kiranoudis and G.P. Prastacos, Solving the open vehicle routeing problem via a single parameter metaheuristic algorithm. *J. Oper. Res. Soc.* **56** (2005) 588–596.
[64] M. Thathachar and P.S. Sastry, Varieties of learning automata: an overview. *IEEE Trans. Syst. Man Cybern. Part B: Cybern.* **32** (2002) 711–722.
[65] P.M. Thompson and J.B. Orlin, The theory of cyclic transfers (1989).
[66] D. Thierens, An adaptive pursuit strategy for allocating operator probabilities. In: *Proceedings of the 7th Annual Conference on Genetic and evolutionary Computation*. ACM (2005) 1539–1546.
[67] P. Toth and D. Vigo, The Vehicle Routing Problem. SIAM (2001).
[68] N. Veerapen, J. Maturana and F. Saubion, An exploration-exploitation compromise-based adaptive operator selection for local search. In: *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. ACM (2012) 1277–1284.
[69] T. Vidal, T.G. Crainic, M. Gendreau and C. Prins, A unified solution framework for multi-attribute vehicle routing problems. *Eur. J. Oper. Res.* **234** (2014) 658–673.
[70] D. Vigo, A heuristic algorithm for the asymmetric capacitated vehicle routing problem. *Eur. J. Oper. Res.* **89** (1996) 108–126.
[71] X. Wang and A.C. Regan, Local truckload pickup and delivery with hard time window constraints. *Transp. Res. Part B: Methodol.* **36** (2002) 97–112.
[72] N. Wieberneit, Service network design for freight transportation: a review. *OR Spect.* **30** (2008) 77–112.
[73] B. Yu and Z.Z. Yang, An ant colony optimization model: the period vehicle routing problem with time windows. *Transp. Res. Part E: Logistics Transp. Rev.* **47** (2011) 166–181.
[74] E.E. Zachariadis and C.T. Kiranoudis, An open vehicle routing problem metaheuristic for examining wide solution neighborhoods. *Comput. Oper. Res.* **37** (2010) 712–723.
[75] R. Zhang, W.Y. Yun and H. Kopfer, Heuristic-based truck scheduling for inland container transportation. *OR Spect.* **32** (2010) 787–808.