VARIABLE-DEPTH ADAPTIVE LARGE NEIGHBOURHOOD SEARCH ALGORITHM FOR OPEN PERIODIC VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

Binhui Chen[(a)], Rong Qu[(b)], Hisao Ishibuchi[(c)]

[(a),(b)]School of Computer Science, The University of Nottingham, United Kingdom
[(c)]1) Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. 2) Department of Computer Science and Intelligent Systems, Osaka Prefecture University, Japan

[(a)]Binhui.Chen@nottingham.ac.uk, [(b)]Rong.Qu@nottingham.ac.uk ,[(c)]hisaoi@cs.osakafu-u.ac.jp

ABSTRACT
The Open Periodic Vehicle Routing Problem with Time Windows (OPVRPTW) is a practical transportation routing and scheduling problem arising from real-world scenarios. It shares some common features with some classic VRP variants. The problem has a tightly constrained large-scale solution space and requires well-balanced diversification and intensification in search. In Variable Depth Neighbourhood Search, large neighbourhood depth prevents the search from trapping into local optima prematurely, while small depth provides thorough exploitation in local areas. Considering the multi-dimensional solution structure and tight constraints in OPVRPTW, a Variable-Depth Adaptive Large Neighbourhood Search (VD-ALNS) algorithm is proposed in this paper. Contributions of four tailored destroy operators and three repair operators at variable depths are investigated. Comparing to existing methods, VD-ALNS makes a good trade-off between exploration and exploitation, and produces promising results on both small and large size benchmark instances.

Keywords: adaptive large neighbourhood search, variable depth neighbourhood search, open periodic vehicle routing problem with time windows, metaheuristic

1. INTRODUCTION

Vehicle Routing Problem (VRP) is a well-studied topic in Operational Research, and has a large number of variants. In the classic model of Vehicle Routing Problem with Time Windows (VRPTW) (Solomon 1987) starting from a depot, a fleet of vehicles visits a number of customers satisfying the time constraints. The depot and customers visited compose a *route* of a vehicle. The total demands on the route cannot exceed the vehicle's capacity. All vehicles have to return to the depot within the planning horizon (so called a *close route* (Hamilton Cycle) (Tarantilis et al. 2005)). The objective of VRPTW is to minimize the total cost of all routes (e.g., travel distance, and the number of vehicles used). Derived from various real-world problems, a large number of extended VRP models are proposed with various *Side Constraints* to VRPTW (e.g. driver working hour regulations, demand type, vehicle type and customer preference), or combined with other problems (e.g. inventory routing problem (Coelho, Cordeau and Laporte 2014)), while both exact approaches and heuristic algorithms are heavily studied (Toth and Vigo 2001).

1.1. Variants of Vehicle Routing Problem

The problem model in our study is related to three classical VRP variants. In Vehicle Routing Problem with Pickups and Deliveries (Golden et al. 2008), customers have pickup and delivery demands. Each vehicle picks up goods from a number of pickup points, then delivers them to the appointed destinations within the associated time windows. In *Less-than Truckload Transportation* problem, goods delivered can be consolidated; otherwise, it is a *Full Truckload Transportation* problem (Wieberneit 2008).

In Multi-Period Vehicle Routing Problem, the service to a customer could be performed over a multi-period horizon (Mourgaya and Vanderbeck 2007). Especially in grocery distribution, soft drink industry and waste collection, goods are delivered at a specified service frequency for customers over a multi-period horizon. In this so-called Periodic Vehicle Routing Problem (Eksioglu et al. 2009), the objective is to minimize the total cost of vehicles routing on all workdays servicing all customers.

To reduce cost, in practice many companies hire external carriers via third party logistic providers, instead of having their own fleet. Those hired vehicles do not return to the starting depot after completing the tasks, so all routes end at the last customers serviced. The routes are called *open routes* (Hamilton Paths instead of Hamilton Cycles) in Open VPRs, first proposed by Eppen and Schrage (1981).

1.2. Existing Methods

As a well-known NP-hard problem (Toth and Vigo 2001), VRPs have been investigated by a huge number of exact methods and heuristic algorithms. Exact methods guarantee optimality (Baldacci et al. 2012),

however, become unrealistic when solving larger scale real-world problems with complex constraints (El-Sherbeny 2010). Heuristic and Metaheuristic algorithms generate good approximations of optimal solutions in an acceptable computational time and have made great achievements in solving large-scale VRPs in the last three decades (Bräysy and Gendreau 2001).

Population-Based Metaheuristics evolve improved solutions in populations and have shown high-performance on problems such as multi-objective problems (Lourens 2005, Ghoseiri and Ghannadpour 2010). However, when facing high-dimensional complex solution structures and large problem size in real-world problems, they could be intractable. For the large scale and highly constrained problem in this study, we focus on single solution-based metaheuristics.

Single solution-based metaheuristics, by calling neighbourhood operators, explore only one new solution in each iteration. In Tabu Search (TS), specific solutions in a tabu list are forbidden to avoid cyclic search, and worse solutions within a certain extend are accepted to escape from a local optima trap. A TS is proposed for PVRPTW in (Cordeau et al. 2001), considering travel time, capacity, duration and time windows. TS has been widely applied to many applications in VRPs (Laporte et al. 2000).

Variable Neighbourhood Search (VNS) explores a solution space by changing neighbourhood structures systematically (Mladenović and Hansen 1997). It has obtained good results on various optimization problems (Hansen et al. 2010) including OVRPTW (Redi et al. 2013). In Variable-Depth Neighbourhood Search (VDNS), one operator is used, but at variable neighbourhood depths. It is widely applied in *Very Large Scale Neighbourhood search* (Pisinger and Ropke 2010). Chen et al. (2016) develop a combined VNS and VDNS with compounded neighbourhood operators for VRPTW and obtained a number of new best solutions for benchmark instances.

Large Neighbourhood Search (LNS) (Shaw 1997, 1998) applies *destroy operators* (removal heuristics) and *repair operators* (insertion heuristics) to remove and reinsert a number of customers/demands from the current solution, producing a new solution with a larger difference. Schrimpf et al. (2000) also propose a similar *Ruin & Recreate* scheme. Pisinger and Ropke (2007) introduce the Adaptive Large Neighbourhood Search (ALNS), which employs an LNS strategy with adaptive operator selection, to solve five VRP variants.

When traditional operators of small change (e.g. λ-opt, CROSS-exchange (Bräysy and Gendreau 2005)) are used to explore tightly constrained large neighbourhood, the search can easily stuck into local optima. LNS operators (*destroy & repair*) and ALNS efficiently conquer this weakness by introducing larger changes to the current solution, and produce promising results in a large number of problems compared to existing methods (Pisinger and Popke 2010; Laporte et al. 2010).

In (Azi et al. 2014), the operation depth of neighbourhood operators in the ALNS for VRPs with Multiple Routes changes. E.g., the Random Removal operator can randomly remove workdays, routes or customers from the operated solution. Note that each of the three different depths is used for only once by turn. More ALNS algorithms for practical VRPs can be found in (Ribeiro and Laporte 2012; Schopka and Kopfer 2016).

In this paper, we propose a Variable-Depth Adaptive Large Neighbourhood Search algorithm (VD-ALNS) for the Open Periodic Vehicle Routing Problem with Time Windows (OPVRPTW) (Chen et al. 2017). Inspired by the idea of systematically adjusting neighbourhood operators during the search in VNS and VDNS, the operation depth of LNS operators in our algorithm is variable.

2. PROBLEM MODEL

Based on a practical *Full Truckload Transportation* problem at the Ningbo Port, the second biggest port in China, Chen et al. (2017) propose an OPVRPTW model. A fleet of 100 identical trucks is available in the depot to complete container transportation tasks among nine terminals. The objective of this problem is to minimize the total unloaded travel distance of the fleet.

The problem is a Periodic VRP with a planning horizon of two to four days, each day has two shifts. One shipment request may contain a number of containers. At the beginning of a working day, the trucks leave the depot to complete a number of tasks of container pickup and delivery between terminals and return to the depot at the end of the day. In the middle of a workday, due to regulations of working hours on Labour Law, drivers working on the first (*Odd-Indexed*) shift of a day handover a truck to a driver working on the second (*Even-Indexed*) shift at a terminal. The terminal can be the first pickup point (source terminal) to the even-indexed shift driver or the last delivery point (destination terminal) to the odd-indexed shift driver. The routes in this problem are *open*, i.e. routes in odd-indexed shifts do not have to end at the depot, and routes in even-indexed shifts do not need to start from the depot.

We use the same problem model as (Chen et al. 2017). All tasks of transporting a container are represented as one *task node* including: loading the container into a truck at the source terminal, travelling from the source to the destination terminal, and unloading at the destination terminal. Therefore, the travel between two nodes is always unloaded travel, because the loaded travel has been packaged into the task nodes. In this *Open Periodic VRP with Time Windows*, one truck can carry only one container at a time for its capacity.

To connect the route of a truck from an odd-indexed shift to the following even-indexed shift, *Artificial Depots* are used in between on each workday. In one shift, every route starts from a *starting depot* and ends at a *termination depot*. The main notations used in this model are summarized in Table 1.

In Figure 1, a small example of one workday schedule (of two consecutive shifts) is presented. A fleet

of five trucks completes 14 transportation tasks. The physical move of the truck in the top route is demonstrated on the right side, with a handover at the artificial depot from Shift 1 (odd-indexed) to the driver in Shift 2 (even-indexed). It is worth to note that, the second and third routes in Shift 1 and the third and fourth routes in Shift 2 are *empty routes*, which directly connect artificial depots and the physical depot. This means no task is completed on these routes. Notice that the cost of an empty route is not always zero, e.g. the cost of the fourth route in Shift 2 could be non-zero, due to the unloaded travel distance from the last destination of the fourth route in Shift 1 to the physical depot is not zero. The cost of empty route will be zero only if the connected artificial node actually represents the physical depot.

Table 1: The List of Notations

Input Parameters:	
K	Fleet size.
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.
$N = \{0,1,2,\cdots,n\}$	Set of $n+1$ nodes. Each node represents a task except node 0 is the *physical depot*.
$[a_i, b_i]$	The time window for node i. The time window for a depot 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.
W	Set of *Artificial Depots*. This set of nodes are introduced to represent the destination terminals in S_{odd} or source terminals in S_{even} on each day, which is decided by if the associated trucks in S_{odd} can arrive at their terminals before the end of the shift. This set varies in different solutions, i.e. a physical terminal may not appear or may appear more than once in W.
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 (distance) of unloaded travel from node i to node j. If the destination terminal of task i and the source terminal of task j is the same, $c_{ij} = 0$.
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 from or to a depot.
T_i	The arrival time at node i.
B_i	The time to begin the service of node i.
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.
Decision Variable:	
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 and $j \in W$ at the same time is not allowed

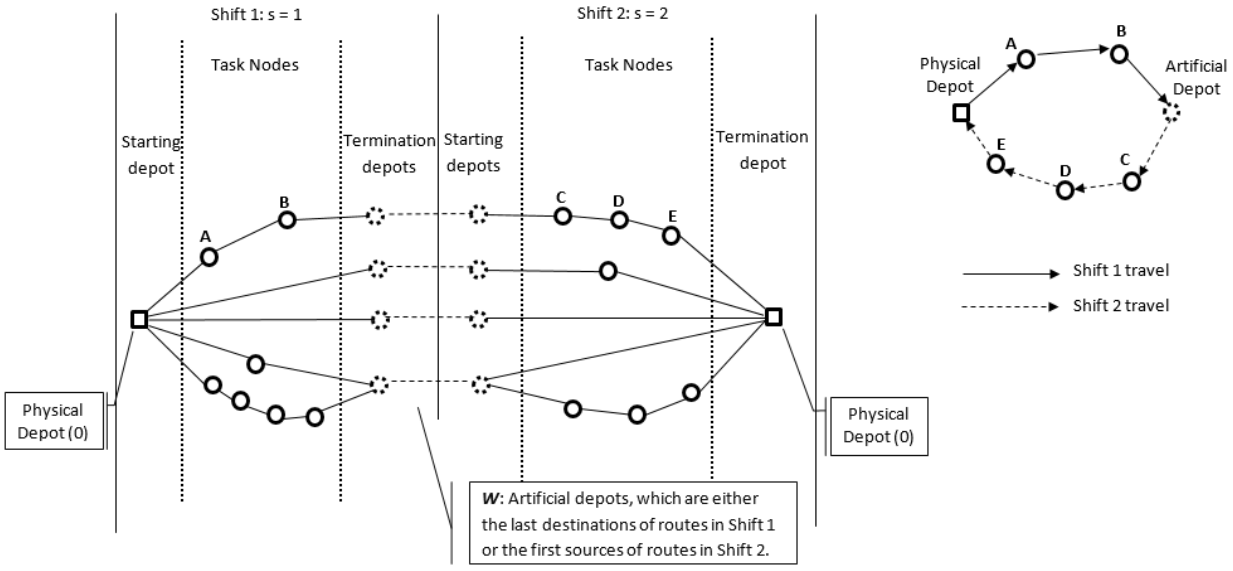

Figure 1: A scheduling example of two consequent shifts with five trucks.

ISBN 978-88-97999-87-4; Bottani, Bruzzone, Longo, Merkuryev, and Piera Eds.

This OPVRPTW problem can be formally defined as follows.

$$Minimise \quad \sum_{s\in S}\sum_{i\in N\cup W}\sum_{j\in N\cup W} c_{ij}\cdot x_{ij}^{s} \qquad (1)$$

Subject to

$$\sum_{s\in S}\sum_{i\in N\setminus\{0\}} x_{ij}^{s} = 1, \qquad \forall\, j \in N\setminus\{0\} \qquad (2)$$

$$\sum_{s\in S}\sum_{j\in N\setminus\{0\}} x_{ij}^{s} = 1, \qquad \forall\, i \in N\setminus\{0\} \qquad (3)$$

$$\sum_{i\in N\cup W} x_{ij}^{s} = \sum_{f\in N\cup W} x_{jf}^{s}\,, \quad \forall j \in N\setminus\{0\}, s \in S \qquad (4)$$

$$T_j = \sum_{i\in N\setminus\{0\}} \left(B_i + l_i + t_{ij}\right)\cdot x_{ij}^{s} + \sum_{i=\{0\}\cup W} \left(Y_s + t_{ij}\right)\cdot x_{ij}^{s}\,, \quad \forall j \in N\setminus\{0\}, s \in S \qquad (5)$$

$$B_j = T_j + max\{a_j - T_j, 0\}\,, \qquad \forall j \in N\setminus\{0\} \qquad (6)$$

$$x_{ij}^{s}\cdot Y_s \le x_{ij}^{s}\cdot T_j\,, \quad \forall i \in \{0\}\cup W, j \in N\cup W, s \in S \qquad (7)$$

$$x_{ij}^{s}\cdot (B_i + l_i) \le x_{ij}^{s}\cdot Z_s\,, \quad \forall i \in N\cup W, j \in \{0\}\cup W, s \in S \qquad (8)$$

$$a_i \le B_i \le b_i - l_i\,, \qquad \forall i \in N\setminus\{0\} \qquad (9)$$

$$x_{ij}^{s} \in \{0,1\}\,, \qquad \forall i,j \in N\cup W, s \in S \qquad (10)$$

$$x_{vw}^{s} = 0\,, \qquad \forall v \in W, w \in W, s \in S \qquad (11)$$

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

$$\sum_{j\in N\setminus\{0\}\cup W} x_{0j}^{s} = K\,, \qquad \forall\, s \in S_{odd} \qquad (12)$$

$$x_{i0}^{s} = 0\,, \qquad \forall i \in N\setminus\{0\}\cup W, s \in S_{odd} \qquad (13)$$

$$\sum_{i\in N}\sum_{w\in W} x_{iw}^{s} = K\,, \qquad \forall s \in S_{odd} \qquad (14)$$

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

$$\sum_{j\in N} x_{jw}^{s-1} = \sum_{e\in N} x_{we}^{s}\,, \quad \forall w \in W, s \in S_{even} \qquad (15)$$

$$x_{0j}^{s} = 0\,, \qquad \forall j \in N\setminus\{0\}\cup \mathrm{W}, s \in S_{even} \qquad (16)$$

$$\sum_{w\in W}\sum_{j\in N} x_{wj}^{s} = K\,, \qquad \forall s \in S_{even} \qquad (17)$$

$$\sum_{i\in N\setminus\{0\}\cup W} x_{i0}^{s} = K\,, \qquad \forall s \in S_{even} \qquad (18)$$

The objective of this problem (1) is to minimize the total unloaded travel distance. Constraints (2) and (3) denote that every task node can be visited exactly once, and all tasks are visited. Constraint (4) specifies that a task may only be serviced after the previous task is completed. Constraints (2) - (4) together make sure arcs of over more than one shift are unacceptable. Constraint (5) is the arrival time at a task node. Constraint (6) defines the beginning time of servicing a task node, calculated by the arrival time plus the waiting time at the source of the task. Constraints (5) and (6) enforce the correct successive relationship between consecutive nodes.

Constraints (7) and (8) are the time window constraints of each shift, while constraint (9) represents the time constraint on each task. The domain of the respective decision variable is defined by constraints (10) and (11). Constraint (11) prohibits the travel between two artificial depots.

In odd-indexed shifts and even-indexed shifts, the constraints for the start and termination depots are different. Constraints (12) and (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 (13) represents that no truck returns to the physical depot in odd-indexed shifts. Constraints (16) - (18) place the reverse restraints in even-indexed shifts. Constraint (15) defines the shift change from an odd-indexed shift to the following even-indexed shift on artificial depots, where the incoming of each artificial terminal in S_{odd} equals its outgoing in the following S_{even}.

It is easy to see that, this problem is highly constrained with an exponential growing search space ($|S|\cdot K\cdot n!$). It has been proofed that exact methods are not suitable to solve this problem due to the exorbitant computing requirement (Chen et al. 2017). To address the tightly constrained problems with large neighbourhood, a Variable-Depth ALNS algorithm (VD-ALNS) is proposed.

3. VARIABLE-DEPTH ADAPTIVE LARGE NEIGHBOURHOOD SEARCH

3.1. Framework of VD-ALNS

The framework of VD-ALNS is shown in ALGORITHM 1. An emergency-based construction heuristic (Chen et al. 2017) is firstly used to generate an initial solution by considering shifts chronologically, and assigning the tasks with higher emergency first. According to their time windows, those tasks that must be completed before the next shift will be assigned first. Starting from the initial solution, four destroy operators and three repair operators are then used to produce new solutions by modifying the current solution ($S_{current}$), pursuing solutions with higher quality.

$Weight$ and $Score$ in Step 1 are two scalars used to guide the subsequent search. Specifically, $Score_i$ records the contributions of operator i in solution improvement within a fixed number of iterations (so called a *Segment*). $Score_i$ is used to update the value of $Weight_i$, which determines the probability of operator i being adopted during search, in the next *Segment*. Their values are set as the same for all operators at the beginning, and then updated during the search. The algorithm iteratively explores the solution space until the Stopping Criterion is met, i.e. the quality of the best found solution (S) has not been improved in the last $UNIMPR_{MAX}$ iterations, or the improvement is less than 1% in the last $INCRE_{MAX}$ iterations.

In Step 2.1, $Depth$ is the range the operators work upon. It is systematically switched between the whole planning horizon ($HORIZON$) and a specified shift

($SHIFT$) to balance the exploration and exploitation. In Step 2.2, a pair of a destroy operator (D_i) and a repair operator (R_j) are used to generate a new solution (S').

Every single operator in ALNS has its own weight ($Weight_i$). However, a research issue here is whether an operator should be assigned two different weights for two different depths to separately record its contribution to improvement at depths $HORIZON$ and $SHIFT$, or only one weight is sufficient to record all previous contribution. In the literature, this question has been addressed in VNS and VDNS (Pisinger and Ropke 2010). Using two independent weights separately records knowledge collected during the search employing two independent operators at different depths, thus would prevent the knowledge collected at the other depth from being used. However, in our preliminary experiments, it is found that search experience at different operation depths can contribute and promote each other. In VD-ALNS, thus, we adopt one operator in both scenarios and record an operator's information with only one scalar.

ALGORITHM 1: Framework of VD-ALNS

Input: An initial feasible solution (S) generated by the construction heuristic in (Chen et al. 2017), Stopping Criterion, ITE_{MAX} and LEN_SEGMENT.

Step 1. Set up the initial parameters.
$Weight \leftarrow \{1, \cdots, 1\}$.
$Score \leftarrow \{0, \cdots, 0\}$.
$S_{current} \leftarrow S,\ Depth \leftarrow HORIZON$.

Step 2.
while Stopping Criterion is not met **do**
Step 2.1 Variable-Depth Setting.
if S is not improved in the last ITE_{MAX} iterations
if $Depth = HORIZON$ **then**
$Depth \leftarrow SHIFT$.
else
$Depth \leftarrow HORIZON$.
end
end
Step 2.2 Operators Selection and Execution.
Select a Destroy Operator (D_i) and a Repair Operator (R_j) based on $Weight$.
Execute D_i and R_j at Depth, and obtain a new solution: $S' \leftarrow R_j(D_i(S_{current}))$.
Step 2.3 Accept or Reject.
A Record-to-Record Travel algorithm is employed to determine if the newly generated solution is accepted ($S_{current} \leftarrow S'$) or rejected. If the quality of S' is better than S, update the best-found solution $S \leftarrow S'$.
Step 2.4 Weight Adjustment.
The Scores of D_i and R_j ($Score_i$ and $Score_j$) are updated at every iteration according to the quality of S'.
At every LEN_SEGMENT iteration, $Weight$ is updated based on the accumulated *Score*, *Score* is then reset.
end

Output: An improved solution S.

A pair of operators is selected by *Roulette Wheel* based on the weights of operators in Step 2.2. The probability of an operator i being selected is calculated with Eq. (19), where h is the number of candidate operators.

$$Pr_i = \frac{Weight_i}{\sum_{k=1}^{h} Weight_k} \quad (19)$$

Step 2.3 decides if S' is accepted as new $S_{current}$ and S is updated, while Step 2.4 adjusts the scores and weights of operators according to the quality of S'. These adaptive weights guide the search to promising solution regions. More details are introduced in Sections 3.2 – 3.5.

3.2. Variable-Depth Setting

Variable search depth endows a balanced search performance. When $Depth$ is $SHIFT$, the destroy operators remove a number of nodes from one specified shift, while the repair operators reinsert them back into that shift. All the shifts are specified and checked sequentially. When $Depth$ is $HORIZON$, the removal and reinsertion happen within the whole planning horizon. Obviously, $HORIZON$ is a greater depth than $SHIFT$, and lead to a greater change in a solution, thus improves the diversification of search. Contrarily, using a $Depth$ of $SHIFT$ modifies routes in a single shift by locally optimizing the solution, thus increases the intensification of search.

$Depth$ is systematically switched to seek a trade-off between exploration and exploitation. Searching with smaller depth exploits a relatively smaller solution area intensively, while larger search depth avoids search trapping into local optima. In the proposed algorithm, $Depth$ would be switched to the other value when S is not improved in ITE_{MAX} iterations, to keep both the diversification and intensification in searching the large scale tightly constrained solution space.

3.3. Operators of Destroy and Repair

Four destroy operators and three repair operators are developed in our proposed VD-ALNS.

3.3.1. Destroy Operators

In each iteration, q nodes are removed by a destroy operator (Removal Heuristic). The value of q increases by 5 when the solution is not improved in the last iteration. As a too small q will hardly bring change to a solution, while a too large q will significantly increase repair operation time and the algorithm becomes a random search, a lower bound of $\max\{0.1n, 10\}$ and an upper bound of $\min\{0.5n, 60\}$ are set for q, where n is the total number of nodes.

1. *Random Removal*: The q nodes to be removed are randomly selected.
2. *Worst Removal*: This is a greedy heuristic, where the top q nodes causing the greatest cost will be removed. In other words, removing the q task nodes brings the greatest cost reduction to the solution.
3. *Worst Edge Removal*: This is also a greedy heuristic, which deletes q nodes adjacent to arcs of the highest cost.
4. *Related Removal*: Shaw (1997) proposes this operator based on the observation that, if nodes relate to one another are removed together, there would be an opportunity to interchange them in the later repaired solution. In VD-ALNS, we define the *Relatedness* of two task nodes (i and

j) from five aspects: Service Time (R_{ij}^{ST}), Time window (R_{ij}^{TW}), Service Starting Time (R_{ij}^{SST}), Vehicle used (R_{ij}^{V}) and Source and Destination (R_{ij}^{SD}) as follows.

$$R_{ij}^{ST} = \frac{|l_i - l_j|}{(l_i + l_j) \cdot 0.5} \quad (20)$$

$$R_{ij}^{TW} = \frac{0.5 \cdot (|a_i - a_j| + |b_i - b_j|)}{max\{b_i, b_j\} - min\{a_i, a_j\}} \quad (21)$$

$$R_{ij}^{SST} = \frac{|B_i - B_j|}{Length\ of\ Planning\ Horizon} \quad (22)$$

$$R_{ij}^{V} = \begin{cases} 0 & i\ and\ j\ are\ serviced\ by\ a\ same\ vehicle \\ 0.5 & i\ and\ j\ are\ servieced\ by\ different\ vehicles\ in\ the\ same\ shift \\ 1 & otherwise. \end{cases} \quad (23)$$

$$R_{ij}^{SD} = \begin{cases} 0 & i\ and\ j\ have\ the\ same\ source\ AND\ destination \\ 0.5 & i\ and\ j\ have\ the\ same\ source\ OR\ destination \\ 1 & otherwise. \end{cases} \quad (24)$$

Correspondingly, the relatedness of two task nodes (R_{ij}) is a linear combination of the five components above-mentioned (25). The values of the five linear coefficients are discussed in Section 4.2. In *Related Removal*, the first node to be removed is randomly selected, then the other nodes are sorted in ascending order of their relatedness R_{ij} to the first node.

$$R_{ij} = \alpha \cdot R_{ij}^{ST} + \beta \cdot R_{ij}^{TW} + \gamma \cdot R_{ij}^{SST} + \delta \cdot R_{ij}^{V} + \varepsilon \cdot R_{ij}^{SD} \quad (s.t.\ \alpha + \beta + \gamma + \delta + \varepsilon = 1) \quad (25)$$

The rest $q - 1$ nodes to be removed are selected with a preference of smaller R_{ij}, where the nodes with the index of $\lceil N\rho^D \rceil$ will be removed. Here, N is the number of the current candidate nodes, ρ is a random number between 0 and 1, and D is a constant greater or equal to 1. The greater D is, the stronger the preference would be, while D is set to 3 in VD-ALNS. This selection scheme with a preference has been widely used in ALNS methods (Ropke and Pisinger 2006; Prescott-Gagnon 2009; Azi et al. 2014).

3.3.2. Repair Operators

The nodes removed in the Destroy phase will be reinserted back into the solution following the below specific rules of each repair operator (Insertion Heuristic).

1. *Random Insertion*: The removed nodes are randomly inserted into feasible positions.
2. *Greedy Insertion*: The removed nodes are inserted into their best feasible positions causing the least cost increase.
3. *Regret2 Insertion*: This greedy insertion heuristic is proposed by Pisinger and Ropke (2007), which always inserts firstly the node of the largest *REGRET* value into its best feasible position. The *REGRET* of a node is the cost difference between inserting the node to its best and second best feasible positions.

3.4. Acceptance Criterion

Record-to-Record Travel acceptance criterion (Dueck 1993) is used to determine if the newly generated solution (S') is acceptable in the search. If S' is better than the best-found solution S (i.e. $COST(S') < COST(S)$), S' will be accepted as the current solution ($S_{current}$). A new solution worse than $S_{current}$ is still acceptable as long as the gap between their *COST* is less than a *DEVIATION* threshold (i.e. $0.01 \cdot COST(S)$).

3.5. Weight Adjustment

In each iteration, the employed operator *i* is rewarded a value $\sigma \geq 0$ based on the quality of the generated solution S' (see Eq. 26). The effect of σ is further studied in Section 4.2.

$$\sigma = \begin{cases} \sigma_1 & S'\ is\ accepted\ and\ COST(S') < COST(S) \\ \sigma_2 & S'\ is\ accepted\ AND\ COST(S) < COST(S') < COST(S_{current}) \\ \sigma_3 & S'\ is\ accepted\ AND\ COST(S_{current}) < COST(S') \\ \sigma_4 & S'\ is\ rejected \end{cases} \quad s.t.\ \ \sigma_1 > \sigma_2 > \sigma_3 > \sigma_4 \geq 0 \quad (26)$$

After a fixed number ($LEN\_SEGMENT$) of iterations (a *Segment*), the total accumulated reward saved in $Score_i$ in the current *Segment t-1* is used to update the weight of operator *i* for the next *Segment t* (see Eq. (27)). In Eq. (27), the reaction factor r controls how quickly the adjustment scheme reacts. u_i is the number times operator i is used in *Segment* $t-1$. After updating $Weight_i^t$, $Score_i$ will be reset to zero to start the accumulation of reward in Segment t.

$$Weight_i^t = r \cdot Weight_i^{t-1} + (1 - r) \cdot \frac{Score_i}{u_i} \quad (27)$$

4. EXPERIMENTS AND ANALYSIS

4.1. Benchmark

Bai et al. (2015) generate a dataset including 15 real-life instances extracted from the container transportation historical data at Ningbo Port, and 16 artificial instances with diverse features. The planning horizons are four, six and eight shifts in the real-life instances, and four or eight shifts in artificial instances, respectively. The artificial instances are classified and named by the tightness of the time windows (Tight/Loose) and workload balance at terminals (Balanced/Unbalanced). For example, the instance named NP4-1 is the first real-life instance with four shifts, and instance TU8-7 is the seventh artificial instance with eight shifts, tight time window and unbalanced workload at terminals.

The sizes of these 31 instances are large comparing to the classical VRP datasets (Solomon1987; Gehring and Homberger 1999). To test the effectiveness and efficiency of the proposed algorithms on small size instances, the Ningbo Port dataset is scaled down by

25%, while keeping the same features in Chen et al. (2017). We test our proposed VD-ALNS on both the original and scaled down datasets.

4.2. Parameter Sensitivity Analysis

Parameters in VD-ALNS are studied one at a time, fixing the other parameters. It is easy to understand that, higher $UNIMPR_{MAX}$ and $INCRE_{MAX}$ lead to more iterations in search, so might bring better solutions but at the cost of longer time. ITE_{MAX} represents the times of one $Depth$ value would be continuously used. The trade-off between the solution quality and running time needs to be considered to strike a balance between effectiveness and efficiency of the search. The values of parameters used in VD-ALNS are presented in Table 2.

Table 2: Parameters in VD-ALNS.

Parameter	σ_1	σ_2	σ_3	σ_4	$UNIMPR_{MAX}$	$INCRE_{MAX}$	ITE_{MAX}
Value	30	15	5	0	150	200	4*No. of shifts
Parameter	α	β	γ	δ	ε	r	*LEN_SEGMENT*
Value	0.3	0.2	0.1	0.2	0.2	70	0.4

In adaptive weight adjustment, the values of rewards represent the contributions in solution improvement. To obtain the best setting of reward values, σ_4 is set to zero, which indicates $Score_i$ stays the same when S' is rejected. Besides, σ_3 is set to 5 as a base unit. Different σ_1 and σ_2 are tested in parameter tuning experiments to find the setting generating the best solutions. It is observed that a too large σ_1 would cause premature search. The best solutions are obtained when the reward to producing a new best solution (σ_1) is two times of that of generating an acceptable solution better than $S_{current}$ (σ_2), and six times of that of obtaining an acceptable solution worse than $S_{current}$ (σ_3).

When tuning the definition of *Relatedness* (Eq. (25)), all the five components are firstly assigned equal weights ($\alpha = \beta = \gamma = \delta = \varepsilon = 0.2$). Then, each coefficient is gradually increased to reflect the contribution of the associated component to the total relatedness. It is found that when the weight of Service Time Relatedness (R_{ij}^{ST}) is high, the quality of solutions is higher. This indicates that reassigning two tasks with a higher similarity of Service Time leads to a higher possibility to produce a better solution. Since the Service Staring Time of a task may change for various reasons (e.g., a task is assigned to a new truck, and a precedent task is reassigned, etc.), R_{ij}^{SST} can hardly represent the relatedness of two tasks and shows low contribution in tuning tests. A lower coefficient is given to R_{ij}^{SST}.

A too small *LEN_SEGMENT* will change the weights of operators frequently and thus the search may converge prematurely. On the other hand, a large *LEN_SEGMENT* cannot update the guidance information in time. Our preliminary experiments show that the best performance is found when *LEN_SEGMENT* is between 50 and 80. In Eq. (27), the higher r is, the slower the algorithm reacts to the latest guidance information. VD-ALNS performs the best when r is between 0.4 and 0.6.

4.3. Comparison of Solution Algorithms

To demonstrate the contribution of variable depth, a standard ALNS for OPVRPTW is also implemented, where the Destroy and Repair operators are only used at the depth of *HORIZON* in global searching. Comparing to other metaheuristics using small change operators, both VD-ALNS and ALNS have a stronger ability to escape from local optima in a tightly constrained solution space. They are compared to VNS-RLS (Chen et al. 2017), which uses neighbourhood operators with small changes.

The comparison results on the 25% scaled down instances are presented in Tables 3 and 4. The three algorithms are compared from four aspects: best-found solution (Best), average solution (Ave), evaluation times (Times) and standard deviation (S.D.). All the results are obtained from 30 runs. In these results, we convert the objective value into Heavy-Loaded Distance Rate (HLDR) (Eq. (28)), which is widely used by logistic companies in practice. This objective is equivalent to the lowest unloaded travel distance in Eq. (1), but it converts the problem into a maximization problem. The lower and upper bounds of optimal solutions, which are obtained by CPLEX (Chen et al. 2017), are also given. NF in the tables means no feasible solution can be found.

$$HLDR = \frac{Loaded\ Distance}{Loaded\ Distance + Unloaded\ Distance} \quad (28)$$

Table 3: HLDR on the 25% scaled down real-life instances. (Best-found HLDR in bold.)

Instance		NP4-1	NP4-2	NP4-3	NP4-4	NP4-5
VNS-RLS	Best	**82.89%**	62.32%	**75.64%**	59.76%	**79.24%**
	Ave	**81.51%**	61.42%	**74.92%**	59.18%	**78.48%**
	Times	469,233	311,885	319,202	347,134	326,956
	S.D.	1.16%	0.60%	0.62%	0.35%	0.42%
ALNS	Best	81.15%	65.51%	75.17%	61.86%	77.14%
	Ave	79.80%	65.08%	73.60%	61.47%	76.15%
	Times	385	500	458	499	395
	S.D.	0.72%	0.33%	0.80%	0.27%	0.57%
VD-ALNS	Best	81.74%	**65.45%**	75.54%	**62.53%**	77.67%
	Ave	79.61%	**65.16%**	74.15%	**61.75%**	77.03%
	Times	483	529	503	549	573
	S.D.	1.20%	0.25%	0.82%	0.27%	0.53%
Lower Bound		78.36%	65.14%	64.83%	54.39%	NF
Upper Bound		92.36%	97.04%	100%	97.72%	100%
Instance		NP6-1	NP6-2	NP6-3	NP6-4	NP6-5
VNS-RLS	Best	76.24%	**73.39%**	62.32%	**80.50%**	82.44%
	Aver	74.99%	**72.83%**	62.06%	**79.84%**	80.53%
	Times	698.514	624,078	253,037	541,548	365,435
	S.D.	0.96%	0.41%	0.20%	0.41%	1.72%
ALNS	Best	79.07%	70.28%	65.00%	78.43%	82.15%
	Ave	78.03%	69.42%	64.26%	77.07%	**80.58%**
	Times	420	449	412	426	450
	S.D.	0.69%	0.49%	0.42%	0.80%	0.69%
VD-ALNS	Best	**79.95%**	70.75%	**65.31%**	78.26%	**82.75%**
	Ave	**78.33%**	69.85%	**64.40%**	77.07%	80.34%
	Times	549	537	553	515	496
	S.D.	0.92%	0.49%	0.47%	0.76%	1.19%
Lower Bound		NF	NF	54.30%	NF	66.11%
Upper Bound		NF	NF	95.20%	NF	98.39%
Instance		NP8-1	NP8-2	NP8-3	NP8-4	NP8-5
VNS-RLS	Best	**76.91%**	**77.76%**	**75.35%**	60.90%	**72.27%**
	Ave	**74.72%**	**77.16%**	**74.93%**	60.47%	**71.68%**
	Times	607,961	525,479	442,103	430,962	516,872
	S.D.	1.20%	0.37%	0.31%	0.32%	0.36%
ALNS	Best	74.74%	74.32%	75.08%	61.85%	71.60%
	Ave	73.90%	73.07%	74.29%	61.66%	71.05%
	Times	445	444	442	421	439
	S.D.	0.54%	0.49%	0.59%	0.14%	0.29%

VD-ALNS	Best	75.50%	74.76%	75.09%	**61.92%**	71.58%
	Ave	74.22%	73.53%	74.53%	**61.70%**	71.10%
	Times	579	524	528	456	527
	S.D.	0.57%	0.58%	0.36%	0.14%	0.31%
Lower Bound		NF	NF	NF	NF	NF
Upper Bound		98.98%	100%	100%	NF	100%

Table 4: HLDR on 25% scaled down artificial instances. (Best-found HLDR in bold.)

Instance		LB4-1	LB4-2	TB4-3	TB4-4	LU4-5	LU4-6	TU4-7	TU4-8
VNS-RLS	Best	76.92%	**83.42%**	**69.08%**	66.41%	**60.71%**	61.08%	48.75%	54.97%
	Ave	74.80%	**81.61%**	**67.78%**	64.95%	**59.29%**	**60.62%**	48.54%	**54.68%**
	Times	313,707	280,849	286,059	298,651	321,835	290,082	166,248	193,536
	S.D.	0.95%	1.09%	0.65%	0.75%	0.64%	0.29%	0.30%	0.33%
ALNS	Best	78.85%	81.85%	68.41%	66.94%	58.87%	59.35%	**49.42%**	54.12%
	Ave	77.84%	80.08%	67.36%	66.06%	57.84%	58.60%	48.87%	53.35%
	Times	438	421	426	410	396	287	371	287
	S.D.	0.67%	1.01%	0.51%	0.39%	0.52%	0.37%	0.39%	0.43%
VD-ALNS	Best	**79.16%**	**83.42%**	68.92%	**67.01%**	59.84%	**60.16%**	**49.42%**	**55.31%**
	Ave	**77.98%**	80.92%	67.45%	**66.22%**	58.74%	59.37%	**49.05%**	54.19%
	Times	445	448	457	443	472	477	411	448
	S.D.	0.75%	0.95%	0.65%	0.36%	0.47%	0.46%	0.38%	0.48%
Lower Bound		66.62%	76.41%	69.91%	69.30%	NF	58.65%	50.37%	55.36%
Upper Bound		100%	94.87%	86.31%	83.51%	79.94%	73.90%	52.17%	66.38%

Instance		LB8-1	LB8-2	TB8-3	TB8-4	LU8-5	LU8-6	TU8-7	TU8-8
VNS-RLS	Best	**91.25%**	**93.56%**	63.05%	66.31%	**65.76%**	**66.58%**	**56.46%**	52.29%
	Ave	**89.76%**	**92.09%**	61.78%	63.25%	**64.86%**	**65.58%**	**55.79%**	**51.93%**
	Times	492,628	547,853	296,837	517,855	438,295	439,782	269,164	281,479
	S.D.	0.95%	0.87%	0.54%	1.16%	0.44%	0.49%	0.29%	0.18%
ALNS	Best	87.37%	87.87%	63.61%	66.12%	64.84%	60.34%	55.37%	51.89%
	Ave	83.02%	84.41%	62.75%	64.89%	63.61%	58.13%	54.69%	51.28%
	Times	398	396	403	461	437	318	334	385
	S.D.	2.40%	1.40%	0.59%	0.74%	0.54%	0.73%	0.23%	0.42%
VD-ALNS	Best	88.71%	89.62%	**64.37%**	**67.01%**	65.30%	63.08%	55.52%	**52.41%**
	Ave	84.32%	84.35%	**62.99%**	**65.26%**	63.93%	59.95%	54.78%	51.81%
	Times	515	499	549	535	598	590	482	577
	S.D.	1.87%	1.95%	0.59%	0.54%	0.57%	1.29%	0.14%	0.39%
Lower Bound		NF	NF	56.85%	52.40%	57.42%	NF	47.65%	50.74%
Upper Bound		100%	100%	82.33%	88.75%	78.33%	86.84%	71.59%	70.43%

From the experiment results, we can find that VD-ALNS beats ALNS in almost all instances, indicating that the variable depth scheme does improve the performance of ALNS. This scheme enhances the exploitation in local areas, leading to increased total evaluation times in ALNS. Comparing to VNS-RLS, on 6 of 15 real-life instances and half of artificial instances, VD-ALNS finds better or equally good solutions, showing no significant difference. However, VD-ALNS takes remarkably fewer evaluation times and 90% running time of VNS-RLS to obtain those results. All the three methods have the similar stability of a difference on S.D. lower than 1%.

Table 5: HLDR on the original full real-life dataset. (Best-found HLDR in bold.)

Instance		NP4-1	NP4-2	NP4-3	NP4-4	NP4-5
VNS-RLS	Best	**83.29%**	**69.85%**	72.90%	66.61%	**80.65%**
	Ave	**81.88%**	**69.56%**	72.20%	65.91%	**80.48%**
	Times	779,504	575,894	661,384	923,891	718,219
	S.D.	0.55%	0.16%	0.38%	0.47%	0.17%
ALNS	Best	81.68%	69.08%	**74.72%**	66.63%	78.16%
	Ave	80.21%	68.62%	**74.06%**	66.11%	77.78%
	Times	212	281	288	271	267
	S.D.	0.99%	0.36%	0.49%	0.29%	0.22%
VD-ALNS	Best	82.30%	69.13%	73.94%	**67.05%**	78.96%
	Ave	81.42%	68.83%	73.01%	**66.28%**	78.11%
	Times	313	501	243	345	297
	S.D.	0.58%	0.21%	0.86%	0.56%	0.49%
Upper Bound		90.43%	70.23%	79.58%	73.72%	81.20%

Instance		NP6-1	NP6-2	NP6-3	NP6-4	NP6-5
VNS-RLS	Best	79.64%	**74.14%**	58.94%	**79.52%**	**79.99%**
	Aver	**79.07%**	**73.72%**	58.62%	**79.10%**	**78.36%**
	Times	1.03×10^6	1.16×10^6	513,974	1.05×10^6	984,987
	S.D.	0.47%	0.21%	0.23%	0.53%	0.99%
ALNS	Best	76.73%	69.16%	**65.27%**	77.99%	77.43%
	Ave	76.27%	64.76%	64.79%	77.11%	76.64%
	Times	265	44	251	236	274
	S.D.	0.29%	3.04%	0.35%	0.49%	0.56%
VD-ALNS	Best	**81.74%**	71.73%	65.16%	78.67%	77.39%
	Ave	77.04%	70.95%	**64.84%**	77.86%	76.52%
	Times	483	300	303	381	387
	S.D.	1.20%	0.69%	0.24%	0.50%	0.54%
Upper Bound		83.93%	76.67%	66.90%	80.97%	84.30%

Instance		NP8-1	NP8-2	NP8-3	NP8-4	NP8-5
VNS-RLS	Best	**73.80%**	**75.27%**	**74.20%**	**61.97%**	**73.62%**
	Ave	**73.48%**	**74.86%**	**73.96%**	**61.91%**	**73.26%**
	Times	1.49×10^6	978,695	867,663	693,779	1.18×10^6
	S.D.	0.15%	0.28%	0.22%	0.06%	0.35%
ALNS	Best	69.53%	71.88%	74.02%	61.13%	72.63%
	Ave	68.58%	71.56%	73.22%	61.00%	72.05%
	Times	113	253	227	322	290
	S.D.	0.45%	0.23%	0.40%	0.09%	0.45%
VD-ALNS	Best	70.13%	72.48%	74.02%	61.17%	73.07%
	Ave	69.72%	71.39%	73.67%	60.98%	72.59%
	Times	303	284	338	306	365
	S.D.	0.31%	0.28%	0.23%	0.09%	0.34%
Upper Bound		77.04%	77.55%	78.82%	62.53%	76.09%

Table 6: HLDR on the original full artificial dataset. (Best-found HLDR in bold.)

Instance		LB4-1	LB4-2	TB4-3	TB4-4	LU4-5	LU4-6	TU4-7	TU4-8
VNS-RLS	Best	73.52%	**78.08%**	**69.32%**	72.24%	**64.67%**	**68.12%**	**53.21%**	**53.80%**
	Ave	72.93%	**77.70%**	**68.54%**	71.42%	**64.38%**	**67.52%**	**53.03%**	**53.61%**
	Times	642,796	617,656	616,237	635,130	724,154	782,608	399,970	290,599
	S.D.	0.32%	0.32%	0.42%	0.49%	0.20%	0.40%	0.16%	0.08%
ALNS	Best	75.98%	77.28%	68.68%	73.03%	61.11%	64.45%	52.75%	53.39%
	Ave	75.41%	76.68%	68.05%	71.52%	60.59%	63.85%	52.01%	53.39%
	Times	328	193	222	257	316	202	242	106
	S.D.	0.48%	0.35%	0.43%	1.26%	0.35%	0.30%	0.43%	0.00%
VD-ALNS	Best	**76.05%**	77.15%	69.03%	**73.66%**	61.04%	65.33%	52.88%	53.66%
	Ave	**75.14%**	76.83%	68.51%	**72.78%**	60.40%	64.80%	52.49%	53.47%
	Times	379	253	309	315	400	255	294	151
	S.D.	0.60%	0.18%	0.38%	0.64%	0.43%	0.49%	0.39%	0.10%
Upper Bound		79.47%	86.33%	84.05%	88.74%	74.11%	74.47%	64.05%	63.50%

Instance		LB8-1	LB8-2	TB8-3	TB8-4	LU8-5	LU8-6	TU8-7	TU8-8
VNS-RLS	Best	85.49%	**94.03%**	**69.59%**	**66.85%**	**67.81%**	**68.41%**	**59.60%**	**54.50%**
	Ave	84.11%	**92.83%**	**69.04%**	**65.70%**	**67.20%**	**68.07%**	**59.21%**	**54.23%**
	Times	1.44×10^6	1.13×10^6	669,136	1.47×10^6	$1,11\times10^6$	1.03×10^6	572,065	859,770
	S.D.	0.95%	1.05%	0.38%	0.76%	0.34%	0.21%	0.21%	0.16%
ALNS	Best	**91.22%**	92.98%	68.60%	63.76%	66.95%	61.68%	59.26%	53.78%
	Ave	83.01%	84.98%	67.80%	63.33%	65.28%	60.12%	58.86%	53.18%
	Times	231	212	236	232	275	225	242	210
	S.D.	3.44%	3.35%	0.49%	0.28%	0.34%	0.57%	0.15%	0.32%
VD-ALNS	Best	88.71%	89.74%	69.53%	64.95%	67.01%	62.30%	58.99%	54.31%
	Ave	**85.96%**	86.67%	68.52%	63.78%	65.38%	61.29%	58.77%	53.10%
	Times	339	347	427	336	280	343	251	175
	S.D.	2.43%	1.77%	0.55%	0.75%	0.53%	0.76%	0.15%	0.50%
Upper Bound		98.26%	97.97%	87.06%	92.44%	74.27%	71.36%	70.29%	56.54%

Tables 5 and 6 present results on the original Ningbo Port instances. The upper bounds are obtained with relaxing the travels of leaving and returning to the depot (Bai et al. 2015). It can be found that, with the variable depth scheme, VD-ALNS outperforms ALNS again from the aspects of both the average and best found solution. New best solutions are generated by VD-ALNS on 7 out of 31 benchmark instances.

4.4. Contributions of Operators

Table 7 provides statistics on the Destroy and Repair operators. On the scaled down dataset, one single operator is excluded at a time in VD-ALNS to record the

resulting solution quality deterioration. The second and third columns show the average deterioration on the best found solution and average solution, while the last two columns give the maximum deterioration on the dataset.

Table 7: Contributions of each operator

Operator	Best sol. deg.	Avg. deg.	Max best sol. deg.	Max avg. deg.
Random Removal	0.15%	0.23%	1.08%	0.13%
Worst Removal	0.33%	0.60%	2.18%	2.14%
Related Removal	0.09%	0.08%	1.32%	0.68%
Worst Edge Removal	0.55%	0.56%	2.87%	2.14%
Random Insertion	0.21%	0.12%	1.80%	1.09%
Greedy Insertion	4.84%	5.34%	9.64%	7.69%
Regret2 Insertion	0.54%	0.25%	4.07%	1.31%

The results indicate the contributions of each operator in VD-ALNS. It can be found that Worst Edge Removal is the most efficient destroy operator, followed by Worst Removal. Related Removal contributes the least. Among all repair operators, Greedy Insertion is the most useful, followed by Regret2 Insertion. Overall, greedy heuristics provide effective complement on search intensification and outperform the others in VD-ALNS.

4.5. Analysis of Runtime

The Destroy and Repair operators in ALNS bring greater changes than the traditional neighbourhood operators by operating on more nodes and making greater perturbation. Therefore, the computation time spent on choosing removal nodes and insertion positions is considerable. The evaluation times of ALNS and VD-ALNS to obtain these results are significantly less than that of VNS-RLS, but the running time of VD-ALNS compared to VNS-RLS is around 17% more on the original instances, and slightly less on small instances. This observation indicates that scalability of the runtime of VD-ALNS is worse (increases faster) than VNS-RLS along with the instance size.

Choosing the insertion position is time-consuming. Actually, the computational time of the repair operators accounts for a larger proportion of the overall time, around 3.5 times of the destroy operators' on scaled down instances. What's more, on the original dataset, the repair operation may spend more than 95% of the total computing time.

5. CONCLUSIONS

This paper investigates an open Periodic Vehicle Routing Problem with Time Windows (OPVRPTW) from a real-world container transportation problem. To address this OPVRPTW of large scale search space with tight side constraints, a Variable-Depth Adaptive Large Neighbourhood Search algorithm (VD-ALNS) is proposed, using four destroy operators and three repair operators at variable neighbourhood depth. In this OPVRPTW with high-dimensional solution structure, the variable depth scheme shows to significantly improve the performance of the proposed algorithm on benchmark instances.

On both small and big size benchmarks, it was demonstrated that the proposed variable depth scheme can handle the trade-off between exploration and exploitation and find good solutions efficiently, significantly promoting the performance of the classic Adaptive Large Neighbourhood Search algorithm. Comparing to an existing solution metaheuristic with small change operators, a number of new best-found solutions are obtained by VD-ALNS.

In our future research, the multi-objective feature will be considered, and other effective trade-off strategies between solution quality and search speed will be adapted within ALNS. It will be interesting to also integrate advanced customized exact methods into both the destroy and repair operators.

ACKNOWLEDGMENTS

This research was supported by Ningbo Science & Technology Bureau (2014A35006) and School of Computer Science, the University of Nottingham.

REFERENCES

Azi N., Gendreau M., and Potvin J.Y., 2014. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. Computers & Operations Research 41 (2014), 167–173.

Bai R., Xue N., Chen J., and Roberts G.W., 2015. A set-covering model for a bidirectional multi-shift full truckload vehicle routing problem. Transportation Research Part B: Methodological 79 (2015), 134–148.

Baldacci R., Mingozzi A., and Roberti R., 2012. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. European Journal of Operational Research 218, 1 (2012), 1–6.

Bräysy O. and Gendreau M., 2001. Metaheuristics for the vehicle routing problem with time windows. Report STF42 A 1025 (2001).

Bräysy O. and Gendreau M., 2005. Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. Transportation science 39, 1 (2005), 104–118.

Chen B., Qu R., Bai R., and Ishibuchi H., 2016. A variable neighbourhood search algorithm with compound neighbourhoods for VRPTW. Springer, 25–35.

Chen B., Qu R., Bai R., and Laesanklang W., 2017. A Reinforcement Learning Based Variable Neighborhood Search Algorithm for Open Periodic Vehicle Routing Problem with Time Windows. Submitted to the Special Issue of the Journal "Networks" on Vehicle Routing and Logistic, 2017.

Coelho, L.C., Cordeau, J.F. and Laporte, G., 2013. Thirty years of inventory routing. Transportation Science, 48(1), pp.1-19.

Cordeau J.F., Laporte G., and Mercier A., 2001. A unified tabu search heuristic for vehicle routing

problems with time windows. Journal of the Operational research society 52, 8 (2001), 928–936.
Dueck G., 1993. New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel. J. Comput. Phys. 104, 1 (1993), 86–92.
Eksioglu B., Vural A.V., and Reisman A., 2009. The vehicle routing problem: A taxonomic review. Computers & Industrial Engineering 57, 4 (2009), 1472–1483.
El-Sherbeny N.A., 2010. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. Journal of King Saud University-Science 22, 3 (2010), 123–131.
Eppen G. and Schrage L., 1981. Centralized ordering policies in a multi-warehouse system with lead times and random demand. Multi-level production/inventory control systems: Theory and practice 16 (1981), 51–67.
Gehring H. and Homberger J., 1999. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In Proceedings of EUROGEN99, Vol. 2. Citeseer, 57–64.
Ghoseiri K., and Ghannadpour S.F., 2010. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. Applied Soft Computing 10, 4 (2010), 1096–1107.
Golden B.L., Raghavan S., and Wasil E.A., 2008. The Vehicle Routing Problem: Latest Advances and New Challenges: latest advances and new challenges. Vol. 43. Springer Science & Business Media.
Hansen P., Mladenović N., and Pėrez J.A.M., 2010. Variable neighbourhood search: methods and applications. Annals of Operations Research 175, 1 (2010), 367–407.
Laporte G., Gendreau M., Potvin J.Y., and Semet F., 2000. Classical and modern heuristics for the vehicle routing problem. International transactions in operational research 7, 45 (2000), 285–300.
Laporte G., Musmanno R., and Vocaturo F., 2010. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. Transportation Science 44, 1 (2010), 125–135.
Lourens T., 2005. Using population-based incremental learning to optimize feasible distribution logistic solutions. Thesis.
Mladenović N. and Hansen P., 1997. Variable neighborhood search. Computers & Operations Research 24, 11 (1997), 1097–1100.
Mourgaya M. and Vanderbeck F., 2007. Column generation based heuristic for tactical planning in multi-period vehicle routing. European Journal of Operational Research 183, 3 (2007), 1028–1041.
Pisinger D. and Ropke S., 2007. A general heuristic for vehicle routing problems. Computers & operations research 34, 8 (2007), 2403–2435.
Pisinger D. and Ropke S., 2010. Large neighborhood search. Springer, 399–419.
Prescott-Gagnon E., Desaulniers G, and Rousseau L.M., 2009. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. Networks 54, 4 (2009), 190–204.
Redi A.A.N.P., Maghfiroh M.F.N., and Yu V.F., 2013. An improved variable neighborhood search for the open vehicle routing problem with time windows. In Industrial Engineering and Engineering Management (IEEM), 2013 IEEE International Conference on. IEEE, 1641–1645.
Ribeiro G.M. and Laporte G., 2012. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. Computers & Operations Research 39, 3 (2012), 728–735.
Ropke S. and Pisinger D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation science 40, 4 (2006), 455–472.
Schopka K. and Kopfer H., 2016. An Adaptive Large Neighborhood Search for the Reverse Open Vehicle Routing Problem with Time Windows. Springer, 243–257.
Schrimpf G., Schneider J., Stamm-Wilbrandt H., and Dueck G., 2000. Record breaking optimization results using the ruin and recreate principle. J. Comput. Phys. 159, 2 (2000), 139–171.
Shaw P., 1997. A new local search algorithm providing high quality solutions to vehicle routing problems. APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK (1997).
Shaw P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. Springer, 417–431.
Solomon M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations research 35, 2 (1987), 254–265.
Tarantilis C.D., Ioannou G., Kiranoudis C.T., and Prastacos G.P., 2005. Solving the open vehicle routeing problem via a single parameter metaheuristic algorithm. Journal of the Operational Research Society 56, 5 (2005), 588–596.
Toth P. and Vigo D., 2001. The vehicle routing problem. Siam.
Wieberneit N., 2008. Service network design for freight transportation: a review. OR spectrum 30, 1 (2008), 77–112.