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, Ghoesiri 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 extent 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 stick 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 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

<table>
<thead>
<tr>
<th>Input Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( K )</td>
<td>Fleet size.</td>
</tr>
<tr>
<td>( S )</td>
<td>The set of time-continuous working shifts, which can be divided into odd-indexed shifts (( S_{\text{odd}} )) and even-indexed shifts (( S_{\text{even}} )).</td>
</tr>
<tr>
<td>([a_i, b_i])</td>
<td>Time window of node ( i ). The time window for a depot is zero at the boundary of a shift.</td>
</tr>
<tr>
<td>( W )</td>
<td>Set of Artificial Depots. This set of nodes are introduced to represent the destination terminals in ( S_{\text{odd}} ) or source terminals in ( S_{\text{even}} ) on each day, which is decided by if the associated trucks in ( S_{\text{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 ).</td>
</tr>
<tr>
<td>( A )</td>
<td>Set of arcs. Each arc ((i, j)) represents that node ( j ) is immediately serviced/visited after servicing/visiting node ( i ).</td>
</tr>
<tr>
<td>( c_{ij} )</td>
<td>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 ).</td>
</tr>
<tr>
<td>( T_i )</td>
<td>The arrival time at node ( i ).</td>
</tr>
<tr>
<td>( B_i )</td>
<td>The time to begin the service of node ( i ).</td>
</tr>
<tr>
<td>( l_i )</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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.
This OPVRPTW problem can be formally defined as follows.

Minimise $\sum_{s \in S} \sum_{i \in N \setminus \{0\}} \sum_{j \in N \setminus \{0\}} c_{ij} \cdot x_{ij}^s \quad (1)$

Subject to

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

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

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

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

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

$$x_{ij}^s \cdot (B_i + l_i) \leq x_{ij}^s \cdot Z_s, \quad \forall i \in N \cup W, j \in [0] \cup W, s \in S \quad (7)$$

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

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

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

$$x_{iw}^s = 0, \quad \forall w \in W, w \in W, s \in S \quad (11)$$

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

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

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

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

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

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

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

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

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

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 ($|S| \cdot K^n$). It has been proved 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* records the contributions of operator $i$ in solution improvement within a fixed number of iterations (so called a Segment). *Score* is used to update the value of *Weight*, 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 $UNIMPROVEM$ iterations, or the improvement is less than 1% in the last $INC_{MAX}$ 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.**

\[ \begin{align*}
\text{Weight} & \leftarrow \{1, \ldots, 1\}, \\
\text{Score} & \leftarrow \{0, \ldots, 0\}, \\
S_{\text{current}} & \leftarrow S, \\
\text{Depth} & \leftarrow \text{HORIZON}.
\end{align*} \]

**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

\[ \text{Depth} \leftarrow \text{SHIFT}. \]

else

\[ \text{Depth} \leftarrow \text{HORIZON}. \]

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: \[ S' \leftarrow R_j(D_i(S_{\text{current}})). \]

**Step 2.3 Accept or Reject.**

A Record-to-Record Travel algorithm is employed to determine if the newly generated solution is accepted (S_{\text{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 \( n \) is the number of candidate operators.

\[ \Pr_i = \frac{\text{Weight}_i}{\sum_{k=1}^{n} \text{Weight}_k} \quad \text{(19)} \]

Step 2.3 decides if S' is accepted as new \( S_{\text{current}} \) 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
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. $\text{COST}(S') < \text{COST}(S)$), $S'$ will be accepted as the current solution ($S_{\text{current}}$). A new solution worse than $S_{\text{current}}$ is still acceptable as long as the gap between their $\text{COST}$ is less than a $\text{DEVIATION}$ threshold (i.e. $0.01 \cdot \text{COST}(S)$).

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 $\sigma$ is further studied in Section 4.2.

$$\sigma = \begin{cases} 
\sigma_1 & S' \text{ is accepted and } \text{COST}(S') < \text{COST}(S) \\
\sigma_2 & S' \text{ is accepted AND } \text{COST}(S') < \text{COST}(S_{\text{current}}) \\
\sigma_3 & S' \text{ is rejected and } \text{COST}(S_{\text{current}}) \leq \text{COST}(S') \\
\sigma_4 & S' \text{ is rejected and } \text{COST}(S_{\text{current}}) > \text{COST}(S') 
\end{cases} \quad (26)
$$

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$_i$. Score$_i$ will be reset to zero to start the accumulation of reward in Segment $t$.

$$\text{Weight}_i^t = r \cdot \text{Weight}_i^{t-1} + (1 - r) \frac{\text{Score}_i}{u_i} \quad (27)$$

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 $UNIMPRO_MAX$ 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>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>30</th>
<th>15</th>
<th>5</th>
<th>0</th>
<th>150</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_1$</td>
<td>2</td>
<td>0.3</td>
<td>0.2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>$\sigma_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$UNIMPRO_MAX$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$INCRE_MAX$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ITE_MAX$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{No. of shifts}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In adaptive weight adjustment, the values of rewards represent the contributions in solution improvement. To obtain the best setting of reward values, $\sigma_4$ is set to zero, which indicates $Score_j$ stays the same when $S^j$ is rejected. Besides, $\sigma_3$ is set to 5 as a base unit. Different $\sigma_1$ and $\sigma_2$ are tested in parameter tuning experiments to find the setting generating the best solutions. It is observed that a too large $\sigma_1$ would cause premature search. The best solutions are obtained when the reward to producing a new best solution ($\sigma_1$) is two times of that of generating an acceptable solution better than $S_{current}$ ($\sigma_2$), and six times of that of obtaining an acceptable solution worse than $S_{current}$ ($\sigma_3$).

When tuning the definition of Relatedness ($\sigma_2$), all the five components are firstly assigned equal weights ($\alpha = \beta = \gamma = \delta = \epsilon = 0.2$). 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.), $R_{ij}^{ST}$ can hardly represent the relatedness of two tasks and shows low contribution in tuning tests. A lower coefficient is given to $R_{ij}^{ST}$.

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.

$$HLDR = \frac{\text{Loaded Distance}}{\text{Loaded Distance+Unloaded Distance}}$$

<table>
<thead>
<tr>
<th>Instance</th>
<th>NP4-1</th>
<th>NP4-2</th>
<th>NP4-3</th>
<th>NP4-4</th>
<th>NP4-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>VNS-RLS</td>
<td>78.36%</td>
<td>65.14%</td>
<td>64.83%</td>
<td>54.39%</td>
<td>NF</td>
</tr>
<tr>
<td>Aluminum</td>
<td>92.36%</td>
<td>97.04%</td>
<td>100%</td>
<td>97.72%</td>
<td>100%</td>
</tr>
<tr>
<td>Best</td>
<td>75.6%</td>
<td>73.9%</td>
<td>62.32%</td>
<td>80.58%</td>
<td>82.48%</td>
</tr>
<tr>
<td>NP8</td>
<td>76.24%</td>
<td>73.9%</td>
<td>62.32%</td>
<td>80.58%</td>
<td>82.48%</td>
</tr>
<tr>
<td>NP6</td>
<td>74.99%</td>
<td>72.83%</td>
<td>62.06%</td>
<td>79.84%</td>
<td>80.53%</td>
</tr>
<tr>
<td>Ave</td>
<td>696.514</td>
<td>624.078</td>
<td>253.037</td>
<td>541.548</td>
<td>565.435</td>
</tr>
<tr>
<td>S.D.</td>
<td>0.96%</td>
<td>0.41%</td>
<td>0.20%</td>
<td>0.41%</td>
<td>1.72%</td>
</tr>
<tr>
<td>Best</td>
<td>79.92%</td>
<td>70.28%</td>
<td>65.00%</td>
<td>78.43%</td>
<td>82.15%</td>
</tr>
<tr>
<td>NP8</td>
<td>78.03%</td>
<td>69.42%</td>
<td>64.26%</td>
<td>77.07%</td>
<td>80.58%</td>
</tr>
<tr>
<td>NP6</td>
<td>642.49</td>
<td>412.42</td>
<td>426.45</td>
<td>450.7</td>
<td>450.7</td>
</tr>
<tr>
<td>Ave</td>
<td>420.49</td>
<td>412.42</td>
<td>426.45</td>
<td>450.7</td>
<td>450.7</td>
</tr>
<tr>
<td>S.D.</td>
<td>0.69%</td>
<td>0.49%</td>
<td>0.42%</td>
<td>0.80%</td>
<td>0.69%</td>
</tr>
<tr>
<td>Best</td>
<td>79.98%</td>
<td>70.55%</td>
<td>65.31%</td>
<td>78.26%</td>
<td>82.75%</td>
</tr>
<tr>
<td>NP8</td>
<td>78.33%</td>
<td>69.85%</td>
<td>64.40%</td>
<td>77.07%</td>
<td>80.54%</td>
</tr>
<tr>
<td>NP6</td>
<td>549.53</td>
<td>515.55</td>
<td>515.49</td>
<td>469.7</td>
<td>469.7</td>
</tr>
<tr>
<td>Ave</td>
<td>549.53</td>
<td>515.55</td>
<td>515.49</td>
<td>469.7</td>
<td>469.7</td>
</tr>
<tr>
<td>S.D.</td>
<td>0.92%</td>
<td>0.49%</td>
<td>0.47%</td>
<td>0.76%</td>
<td>1.19%</td>
</tr>
<tr>
<td>Best</td>
<td>79.07%</td>
<td>70.35%</td>
<td>65.00%</td>
<td>78.43%</td>
<td>82.15%</td>
</tr>
<tr>
<td>NP8</td>
<td>78.03%</td>
<td>69.42%</td>
<td>64.26%</td>
<td>77.07%</td>
<td>80.58%</td>
</tr>
<tr>
<td>NP6</td>
<td>642.49</td>
<td>412.42</td>
<td>426.45</td>
<td>450.7</td>
<td>450.7</td>
</tr>
<tr>
<td>Ave</td>
<td>420.49</td>
<td>412.42</td>
<td>426.45</td>
<td>450.7</td>
<td>450.7</td>
</tr>
<tr>
<td>S.D.</td>
<td>0.69%</td>
<td>0.49%</td>
<td>0.42%</td>
<td>0.80%</td>
<td>0.69%</td>
</tr>
<tr>
<td>Best</td>
<td>79.98%</td>
<td>70.55%</td>
<td>65.31%</td>
<td>78.26%</td>
<td>82.75%</td>
</tr>
<tr>
<td>NP8</td>
<td>78.33%</td>
<td>69.85%</td>
<td>64.40%</td>
<td>77.07%</td>
<td>80.54%</td>
</tr>
<tr>
<td>NP6</td>
<td>549.53</td>
<td>515.55</td>
<td>515.49</td>
<td>469.7</td>
<td>469.7</td>
</tr>
<tr>
<td>Ave</td>
<td>549.53</td>
<td>515.55</td>
<td>515.49</td>
<td>469.7</td>
<td>469.7</td>
</tr>
<tr>
<td>S.D.</td>
<td>0.92%</td>
<td>0.49%</td>
<td>0.47%</td>
<td>0.76%</td>
<td>1.19%</td>
</tr>
<tr>
<td>Best</td>
<td>79.07%</td>
<td>70.35%</td>
<td>65.00%</td>
<td>78.43%</td>
<td>82.15%</td>
</tr>
<tr>
<td>NP8</td>
<td>78.03%</td>
<td>69.42%</td>
<td>64.26%</td>
<td>77.07%</td>
<td>80.58%</td>
</tr>
<tr>
<td>NP6</td>
<td>642.49</td>
<td>412.42</td>
<td>426.45</td>
<td>450.7</td>
<td>450.7</td>
</tr>
<tr>
<td>Ave</td>
<td>420.49</td>
<td>412.42</td>
<td>426.45</td>
<td>450.7</td>
<td>450.7</td>
</tr>
<tr>
<td>S.D.</td>
<td>0.69%</td>
<td>0.49%</td>
<td>0.42%</td>
<td>0.80%</td>
<td>0.69%</td>
</tr>
</tbody>
</table>
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: HLRD on the original full real-life dataset. (Best-found HLRD in bold.)

<table>
<thead>
<tr>
<th>Instance</th>
<th>LB-4</th>
<th>LB-2</th>
<th>LB-3</th>
<th>LB-4</th>
<th>LB-5</th>
<th>TB-3</th>
<th>TB-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>79.64%</td>
<td>87.00%</td>
<td>67.89%</td>
<td>92.46%</td>
<td>95.12%</td>
<td>80.46%</td>
<td>86.75%</td>
</tr>
<tr>
<td>Ave</td>
<td>70.16%</td>
<td>80.45%</td>
<td>76.24%</td>
<td>87.86%</td>
<td>88.98%</td>
<td>74.16%</td>
<td>82.58%</td>
</tr>
<tr>
<td>Best</td>
<td>79.64%</td>
<td>87.00%</td>
<td>67.89%</td>
<td>92.46%</td>
<td>95.12%</td>
<td>80.46%</td>
<td>86.75%</td>
</tr>
<tr>
<td>Ave</td>
<td>70.16%</td>
<td>80.45%</td>
<td>76.24%</td>
<td>87.86%</td>
<td>88.98%</td>
<td>74.16%</td>
<td>82.58%</td>
</tr>
<tr>
<td>S.D.</td>
<td>0.47%</td>
<td>0.21%</td>
<td>0.23%</td>
<td>0.53%</td>
<td>0.99%</td>
<td>0.35%</td>
<td>0.56%</td>
</tr>
<tr>
<td>Lower Bound</td>
<td>NF</td>
<td>NF</td>
<td>NF</td>
<td>NF</td>
<td>NF</td>
<td>NF</td>
<td>NF</td>
</tr>
<tr>
<td>Upper Bound</td>
<td>99.98%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Instance</th>
<th>LB-4A</th>
<th>LB-2B</th>
<th>LB-3B</th>
<th>LB-4A</th>
<th>LB-5A</th>
<th>TB-3B</th>
<th>TB-4A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>75.92%</td>
<td>78.90%</td>
<td>68.49%</td>
<td>80.14%</td>
<td>82.17%</td>
<td>73.52%</td>
<td>75.39%</td>
</tr>
<tr>
<td>Ave</td>
<td>67.56%</td>
<td>73.59%</td>
<td>63.75%</td>
<td>74.78%</td>
<td>76.91%</td>
<td>68.75%</td>
<td>71.59%</td>
</tr>
<tr>
<td>Best</td>
<td>75.92%</td>
<td>78.90%</td>
<td>68.49%</td>
<td>80.14%</td>
<td>82.17%</td>
<td>73.52%</td>
<td>75.39%</td>
</tr>
<tr>
<td>Ave</td>
<td>67.56%</td>
<td>73.59%</td>
<td>63.75%</td>
<td>74.78%</td>
<td>76.91%</td>
<td>68.75%</td>
<td>71.59%</td>
</tr>
<tr>
<td>S.D.</td>
<td>0.36%</td>
<td>0.20%</td>
<td>0.23%</td>
<td>0.55%</td>
<td>0.99%</td>
<td>0.35%</td>
<td>0.56%</td>
</tr>
<tr>
<td>Lower Bound</td>
<td>NF</td>
<td>NF</td>
<td>NF</td>
<td>NF</td>
<td>NF</td>
<td>NF</td>
<td>NF</td>
</tr>
<tr>
<td>Upper Bound</td>
<td>99.98%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 5: Table 6 presents 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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Removal</td>
<td>0.15%</td>
<td>0.23%</td>
<td>1.08%</td>
<td>0.13%</td>
</tr>
<tr>
<td>Worst Removal</td>
<td>0.33%</td>
<td>0.60%</td>
<td>2.18%</td>
<td>2.14%</td>
</tr>
<tr>
<td>Related Removal</td>
<td>0.09%</td>
<td>0.08%</td>
<td>1.32%</td>
<td>0.68%</td>
</tr>
<tr>
<td>Worst Edge Removal</td>
<td>0.55%</td>
<td>0.56%</td>
<td>2.87%</td>
<td>2.14%</td>
</tr>
<tr>
<td>Random Insertion</td>
<td>0.21%</td>
<td>0.12%</td>
<td>1.80%</td>
<td>1.09%</td>
</tr>
<tr>
<td>Greedy Insertion</td>
<td>4.84%</td>
<td>5.34%</td>
<td>9.64%</td>
<td>7.69%</td>
</tr>
<tr>
<td>Regret2 Insertion</td>
<td>0.54%</td>
<td>0.25%</td>
<td>4.07%</td>
<td>1.31%</td>
</tr>
</tbody>
</table>

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


