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

Open Periodic Vehicle Routing Problem with Time Windows (OPVRPTW) is a practical transportation routing and scheduling problem arising from a real-world scenario, which shares some common features with some classic VRP variants. It has a large scale and tightly constrained solution space, which requires to well balance the diversification and intensification when searching it. Working at large neighbourhood depth prevents the search from trapping into local optima prematurely, while small depth provides thorough exploitation in the local area. Variable Depth Neighbourhood Search uses variable depth to implement the balance and obtains satisfying results in the literature. Considering the characteristics of 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. In the adaptive large neighbourhood search framework, specially tailored four destroy operators and three repair operators work at variable depth in VD-ALNS. The contribution of each operator is investigated. Comparing to existing methods, VD-ALNS makes a good trade-off between exploration and exploitation and produces promising results on both small and big 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 domain in Operational Research, which rises 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 visit a number of customers to complete their demands satisfying the time constraints (Time Window). The depot and customers visited compose the *route* of a vehicle, and the total demands on the route cannot exceed the capacity of the vehicle. All vehicles have to return to the depot within the planning horizon. This circle trip of each vehicle is 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, the number of vehicles used). Derived from various real-world problems, a large number of extended models are proposed by adding various *Side Constraints* to VRPTW (e.g. driver working hour regulations, demand type, vehicle type and customer preference), while solution methods of both exact approaches and heuristic algorithms are deeply studied (Toth and Vigo 2001).

1.1. Variants of Vehicle Routing Problem

Among the large number of variants of VRP, there are three classical variants available for reference to our study, which are reviewed in this section. Based on VRPTW, when the demands of customers are pickup and delivery of shipments, the problems are called Vehicle Routing Problem with Pickups and Deliveries (VRPPD) (Golden et al. 2008). In this case, each vehicle has to pick up goods from a number of pickup points, then deliver them to the appointed destinations within the associated time windows. If the depot is the only one pickup point while multiple customer locations are delivery points, or contrarily, a number of pickup points (customers) with the only one delivery point depot, the problem would be defined as a *One-to-Many-to-One* problem. Whilst the customers can be both pickup and delivery points, then it is a *Many-to-Many* problem. In *One-to-One* problems, one customer's pickup demand is another customer's delivery demand. Furthermore, if the demands can be consolidated, it is called *Less-than Truckload Transportation* problem, otherwise, it is a *Full Truckload Transportation* problem (Wieberneit 2008).

In Multi-Period Vehicle Routing Problem (MPVRP), the service to a customer could be performed over a multi-period horizon (Mourgaya and Vanderbeck 2007, Hemmelmayr et al. 2012). Especially, in grocery distribution, soft drink industry and waste collection, there would be a specified service frequency for each customer over the multi-period horizon. In this so-called Periodic Vehicle Routing Problem (PVRP) (Eksioglu et al. 2009), the servicing days to customers are more flexible. Reasonably plan the servicing days of customers and routing of vehicles, so as to minimize the total cost of all workdays is the objective of this type of problems.

In practice, for cost reason, many companies hire external carriers, e.g. third party logistic provider and private trucks, instead of establishing their own fleet. Those hired vehicles do not return to the starting collection depot when they complete the tasks assigned, and all routes end at the last customers serviced. The routes are called *open routes* (Hamilton Paths instead of Hamilton Cycles). This type of problems are classified to Open Vehicle Routing Problems (OVRP), which is first proposed by Eppen and Schrage (1981).

1.2. Existing Methods

As a well-known NP-hard problem (Toth and Vigo 2001), a huge number of both exact methods and heuristic algorithms have been developed for VRPs. Exact methods always generate the optimal solutions and solve small and medium size of problems well (Baldacci et al. 2012). However, when facing the larger scale of real-world problems with thousands or more customers and complex constraints, exact methods become time-consuming and unrealistic (El-Sherbeny 2010). Heuristic and Metaheuristic algorithms generate good optimal solutions approximations in acceptable computational time. In the latest three decades, metaheuristic approaches have made great achievements in solving large-scale VRPs (Bräysy and Gendreau 2001).

In Population-Based Metaheuristics, the iterative improvement is conducted in a population of solutions (Talbi 2009). They have shown high-performance in multi-Objective and small to medium size problems (Lourens 2005, Ghoseiri and Ghannadpour 2010). However, when facing the high-dimensional complex structure of solution and large problem size in real-world problems, they would be intractable to population-based metaheuristics. Thus, we focus on the Single Solution-Based Metaheuristics in this paper.

Single Solution-Based Metaheuristics, by calling neighbourhood operators, explore only one new solution in each iteration, e.g. Tabu Search (TS), Variable Neighbourhood Search (VNS), and Large Neighbourhood Search (LNS). TS rejects a number of specific solutions (tabu list) to avoid search cycle, and worse solutions within a certain range are accepted so as to escape a local optima trap. A construction heuristic followed by TS is proposed for PVRPTW in (Cordeau et al. 2001), considering travel time, capacity, duration and time windows. An improved TS approach using the *Forward Time Slack* (Savelsbergh 1992) is later proposed to further optimize the route (Cordeau et al. 2004). TS has been applied widely, and many other applications in VRPs can be found in (Laporte et al. 2000).

Mladenoviċ and Hansen (1997) propose VNS, which systematically changes neighbourhood structures during exploring solution space. VNS has been applied to many optimization problem domains and obtains good results (Hansen et al. 2010). Redi et al. (2013) propose a VNS algorithm for OVRPTW, which outperforms almost all previous algorithms. Differently, Variable-Depth Neighbourhood Search (VDNS) uses one type of operator but at variable neighbourhood depth. It is widely applied in *Very Large Scale Neighbourhood search* (Pisinger and Ropke 2010). Chen et al. (2016) develop an algorithm which proposes compounded neighbourhood operators combining VNS and VDNS for VRPTW, and a number of newly found best solutions of the benchmark are produced.

LNS (Shaw 1997, 1998) applies *destroy operators* (removal heuristics) and *repair operators* (insertion heuristics) to remove a number of customers/demands from the current solution and reinsert them into the destroyed solution, producing a new solution with a great change from the previous one. Schrimpf et al. (2000) propose a similar scheme which is called *Ruin & Recreate*. Pisinger and Ropke (2007) introduce a general heuristic named Adaptive Large Neighbourhood Search (ALNS), which employs an LNS strategy with adaptive operator selection, solves five different variants of VRPs.

When traditional operators of small change (e.g. λ-opt, CROSS-exchange (Bräysy and Gendreau 2005)) are used to explore the tightly constrained large neighbourhood, the search is easy to trap into local optima. LNS operators (*destroy & repair*) efficiently conquer this weakness by bringing significant change to the current solution. ALNS produces promising results in a large number of benchmarks and outperforms most existing methods (Pisinger and Popke 2010; Laporte et al. 2010).

In the subsequent research, ALNS is applied to various practical VRPs and shows powerful ability to solve real-world large scale VRPs with tight constraints. An ALNS algorithm for Pickup and Delivery Problem with Time Windows is proposed by Ropke and Pisinger (2006). In (Prescott-Gagnon et al. 2009), ALNS is used in both stages of the proposed algorithm, which respectively optimize the two objectives of VRPTW. ALNS is also used to tackle the Two-Echelon Vehicle Routing Problem and Location Routing Problem, whose solutions contain two-level planning (i.e. depots selection and satellite facilities scheduling), in (Hemmelmayr et al. 2012). An ALNS algorithm for Reverse OVRPTW is proposed by Schopka and Kopfer (2016). In this problem, the routes start from scattered customers and end at a common depot. To enhance the intensity of search, local search is employed after Destroy & Repair in (Dayarian et al. 2013), which addresses MPVRP with up to 200 customers.

In the ALNS for Vehicle Routing Problems with Multiple Routes proposed by Azi et al. (2014), the operation depths of neighbourhood operators are different when the algorithm is running at different levels. E.g. when the algorithm is respectively running at the levels of *workday*, *route* and *customer*, the operator of Random Removal would randomly remove workdays, routes and customers from the operated solution, correspondingly. However, in that paper, the different depths are used only once at each level. Wen et al. (2011) use LNS operators like traditional neighbourhood operators in a VNS framework. More ALNS algorithms for practical VRPs can be found in (Ribeiro and Laporte 2012; Masson et al. 2013).

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). Inpired by the idea of systematically adjusting neighbourhood during the search from VNS and VDNS, the operation depth of LNS operators in our algorithm is variable. Comparing to the existing methods, the proposed algorithm produces promising results.

2. PROBLEM MODEL

Based on a practical *one-to-one Full Truckload Transportation* problem happens at the Ningbo Port, which is 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 minimizing the total unloaded travel distance of the fleet.

First of all, this problem is a Periodic Vehicle Routing Problem whose planning horizon consists of two to four days, while each day is split into two shifts. One shipment request may contain a number of containers (service frequency). At the beginning of a working day, the trucks leave the depot to complete a number of assigned tasks of container pickup and delivery between terminals and then return to the depot at the end of the day. In the middle of a workday, a shift handover takes place for each truck to satisfy the associated regulations about driver working hours in Labour Law. The driver working in the *Odd-Indexed* shift (the first shift of a day) handovers a truck to the driver working in the *Even-Indexed* shift (the second shift of a day) 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, since routes in odd-indexed shifts do not end at the depot, and routes in even-indexed shifts do not start from the depot.

Every container to be transported (i) has a time window $[a_i, b_i]$, which is defined by the available time (a_i) to pick up i at the source terminal and the deadline (b_i) of delivering i to the destination terminal. In this *Open Periodic Vehicle Routing Problem with Time Windows*, one truck can carry only one container every single time for its capacity. We use the same problem model as (Chen et al. 2017), where all the actions of transporting a container are packaged into 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.

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

In Figure 1, a small example of one workday schedule (two consecutive shifts) is presented. As shown in the figure, a fleet of five trucks ($K = 5$) completes 14 transportation tasks. In the top route, a truck leaves the physical depot and completes two tasks in the Shift 1 (odd-indexed). Then, the truck is handover to the driver of the Shift 2 (even-indexed) at the artificial depot. Eventually, the truck returns to the physical depot after completing three tasks. The physical move of this truck is demonstrated on the right side of Figure 1. 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. That 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. The cost of empty route will be zero, only if the connected artificial node actually represents the physical depot.

This problem can be formally defined as follows.

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

Subject to

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

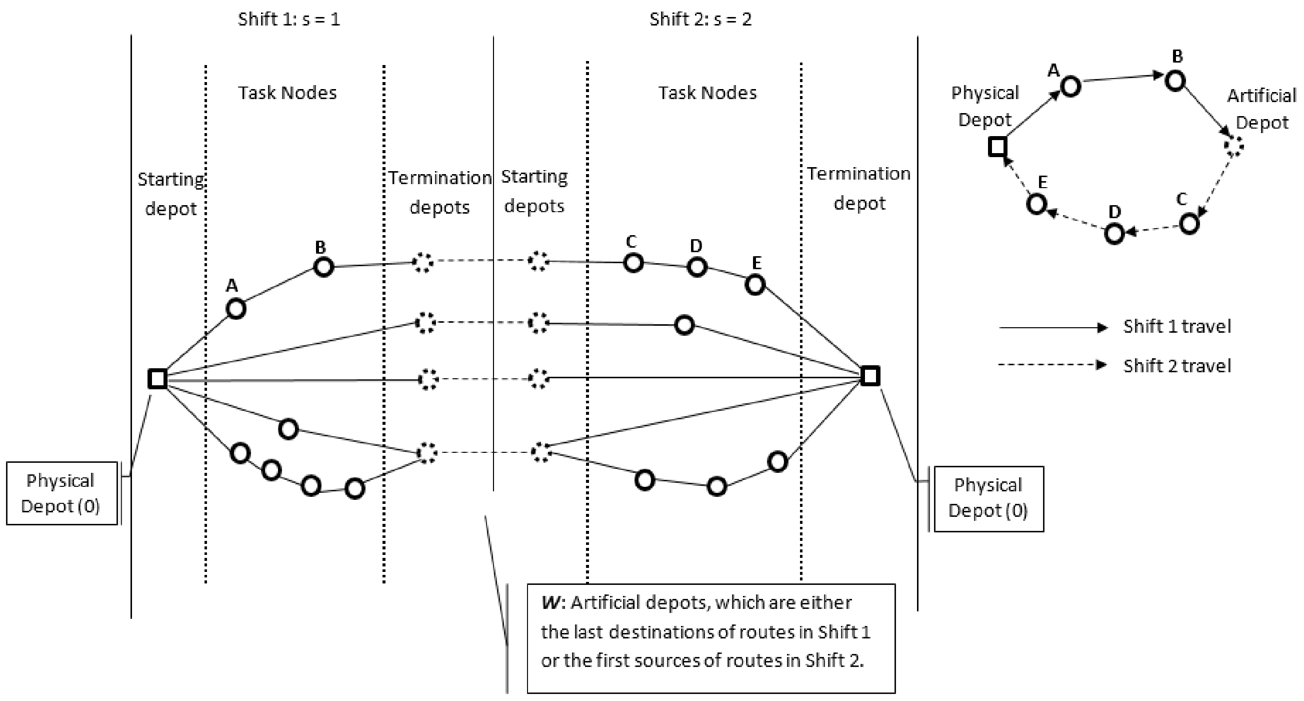

Figure 1：A scheduling example of two consequent shifts with five trucks.

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

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

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

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

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

The objective of this problem (equation 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 the 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. This time is calculated by the arrival time plus the waiting time at the source of a 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). Especially, constraint (11) prohibits the travel between two artificial depots.

In odd-indexed shifts and even-indexed shifts, the constraints for starting 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 find that, this problem is tightly 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 for the exorbitant computing requirement (Chen et al. 2017). To take advantage of ALNS in addressing the tightly constrained problems with large neighbourhood, a Variable-Depth ALNS algorithm (VD-ALNS) is proposed in the next section.

3. VARIABLE-DEPTH ADAPTIVE LARGE NEIGHBOURHOOD SEARCH

3.1. Framework of VD-ALNS

The framework of VD-ALNS is shown in ALGORITHM 1. At the beginning, we use an emergency-based construction heuristic (Chen et al. 2017) to generate the initial solution. In this construction heuristic, shifts are considered chronologically, and the tasks with higher emergency will be first assigned. Namely, 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 used to produce new solutions by modifying the current solution ($S_{current}$), pursuing the solution with higher quality.

ALGORITHM 1: Framework of VD-ALNS

```
Input: An initial feasible solution ($S$) generated by
    construction heuristic from (Chen et al. 2017),
    Stopping Criterion, $ITE_{MAX}$ and
    LEN_SEGMENT.
Step 1. Set up the initial weights and scores of operators.
    $Weight \leftarrow \{1,\cdots,1\}$.
    $Score \leftarrow \{0,\cdots,0\}$.
Step 2. Set up the other initial parameters.
    $S_{current} \leftarrow S,\ Depth \leftarrow HORIZON$.
Step 3.
  while Stopping Criterion is not met do
    Step 3.1 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 3.2 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 3.3 Accept or Reject.
      A Record-to-Record Travel algorithm is
      employed to determine should the newly
      generated solution be 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 3.4 Weight Adjustment.
      The Scores of $D_i$ and $R_j$ ($Score_i$ and $Score_j$)
      will be updated every iteration, according to
      the quality o$S'$f.
      Every LEN_SEGMENT iteration, the $Weight$
      will be updated once, while Score will be reset.
  end
Output: An improved solution $S$.
```

The $Weight$ and $Score$ in Step 1 are two scalars which record the contributions of operators in solution improvement, and their values are set to equal for all operators at the beginning. After setting the initial parameters in Steps 1 and 2, the algorithm iteratively explores the solution space until the Stopping Criterion is met in Step 3. Here we define the stopping conditions as follows: 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 3.1, $Depth$ is the range where the operators work at, which can be the whole planning horizon ($HORIZON$) or a specified shift ($SHIFT$). $Depth$ is systematically shifted between $HORIZON$ and $SHIFT$ to balance the exploration and exploitation. In Step 3.2, we select a pair of a destroy operator (D_i) and a repair operator (R_j) to generate a new solution (S').

Every single operator should have its own weight and score ($Weight_i$ and $Score_i$). However, it is controversial whether we should give an operator two different weights when it uses two different depths. For example, should an operator have two weights to separately record its improvement contribution at depths of $HORIZON$ and $SHIFT$, or record all previous contribution with only one weight? Different answers to this question represent the different view between VDNS and VNS (Pisinger and Ropke 2010). Using two independent weights would prevent using the guidance based on the search experience at the other depth, because the searching performance history at different depths is separately recorded as employing two independent operators. But, in our preliminary experiments, it is found that search experience at different operation depths can promote each other. In VD-ALNS, thus, we adopt the VDNS idea, which considers this issue as one operator working in two scenarios, and records an operator's information with only one scalar.

A pair of operators is selected by *Roulette Wheel* based on the weights of operators in Step 3.2. The probability of an operator i being selected is calculated with Eq. (19) where h is the number of candidate operators.

$$Pr_i = \frac{Weight_i}{\sum_{k=1}^{h} Weight_k} \tag{19}$$

Step 3.3 decides if accept S' as $S_{current}$ and update S, while Step 3.4 adjusts the scores and weights of operators according to the quality of S'. These adaptive weights guide the search to the promising solution region. 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 the removed nodes into that shift then. All the shifts would be specified sequentially. On the other hand, when $Depth$ is $HORIZON$, the removal and reinsertion happen in the whole planning horizon. Obviously, $HORIZON$ is a greater depth than $SHIFT$, and it may cause a greater change in a solution, which improves the diversification of search. Contrarily, using the $Depth$ of $SHIFT$ modifies routes in a single shift in each iteration. It locally optimizes the solution which increases the intensification of search.

We regularly switch $Depth$ 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, so as 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 this paper, which use diverse heuristics to remove and reinsert nodes to an existing solution.

3.3.1. Destroy Operators

q nodes would be removed by an destroy operator (Removal Heuristic) in each iteration. The value of q increases by 5 when the solution is not improved in the last iteration. Because too small q is hard to bring change to the solution, while too large q will significantly increase the repair operation time and cause the algorithm degenerating to random search, a lower bound of $\max\{0.1n, 10\}$ and an upper bound of $\min\{0.5n, 60\}$ are given to q, here 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 increase will be removed. In other words, removing the q task nodes brings the greatest reduction of the cost in the solution.
3. *Worst Edge Removal*: This is also a greedy heuristic, which deletes q nodes connected by the arcs with the highest cost.
4. *Related Removal*: Shaw (1997) proposes this destroy operator which claims that, if nodes close to one another are removed together, there would be an opportunity for interchanging them in the latter 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}).

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

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

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

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

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

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

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

The rest $q - 1$ nodes to be removed are randomly selected with the preference of smaller R_{ij}, where the nodes with the index of $\lceil N\rho^D \rceil$ in P will be removed. Here, N is the number of 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 random selection scheme with preference is also used in some other 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 into the destroyed solution following the specific rules of each repair operator (Insertion Heuristic).

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

3.4. Acceptance Criterion

We use Record-to-Record Travel acceptance criterion (Dueck1993) to determine if the newly generated solution (S') is acceptable as the new starting point of exploration. Comparing the quality of solutions (i.e. *COST* (Eq. 1)), if S' is better than the best-found solution S (i.e. $COST(S') < COST(S)$), S' will be accepted as the current solution ($S_{current}$). Besides, a new solution worse than $S_{current}$ is still acceptable as long as the gap between their $COST$ is less than the $DEVIATION$ (i.e. $0.01 \cdot COST(S)$).

3.5. Weight Adjustment

To obtain the weights, the scores of operators should be calculated first. In each iteration, a reward ($\sigma \geq 0$) will be added to $Score_i$ which is the score of the employed operator. The value of σ is decided by the quality of S' (see Eq. 26), and we will discuss it further in Section 4.2.

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

The weights are adjusted according to the operators' performance (improvement contribution) shown in the last *Segment* which is a single $LEN\_SEGMENT$ iteration in Step 3.4 of Algorithm 1 in Section 3. At the beginning of the current Segment t, the weight of each operator ($Weight_i$) is updated according to its score obtained from the previous Segment $t - 1$ (see Eq. (27)). In the equation, r is a reaction factor, which controls how quickly the adjustment scheme reacts. u_i is the number of usage of operator i in Segment $t - 1$.

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

After updating the $Weight$s, the accumulated score of each operator will be reset to zero to start the calculation of the new rewards in Segment t.

4. EXPERIMENTS AND ANALYSIS

4.1. Benchmark

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

The sizes of these 31 instances are large comparing to the classical VRP datasets (Solomon1987; Gehring and Homberger 1999). To test the efficiency of the proposed algorithms in small size instances, Chen et al. (2017) extract a 25% scaled down dataset from the Ningbo Port dataset, while the features of instances are kept. We test VD-ALNS on both datasets.

4.2. Parameter Tuning and Sensitivity

The tuning is conducted on one parameter at one time, while the other parameters are fixed. It is easy to understand that, higher $UNIMPR_{MAX}$ and $INCRE_{MAX}$ mean more iterations in search, which might bring better solutions but demand more running time. ITE_{MAX} represents the times of the same $Depth$ value would be used in one cycle. The trade-off between the solution quality and running time as well as the balance between search thoroughness and efficiency is what all metaheuristics have to consider. 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 the adaptive weight adjustment, four levels of rewards ($\sigma_1 > \sigma_2 > \sigma_3 > \sigma_4 \geq 0$) are given according to the quality of the newly generated solution S'. Firstly, σ_4 is set to zero to reward zero to the employed operator if S' is rejected. Then, σ_3 is set to 5 as a base unit, and σ_1 and σ_2 are adjusted to find the best setting. It is observed that too large σ_1 would cause premature search. When the reward to the operator producing the new best solution is too large, the algorithm would degenerate to a Hill Climbing method.

When tuning the linear coefficients in the definition of *Relatedness* (Eq. (25)), it starts with giving all the five components equal weights ($\alpha = \beta = \gamma = \delta = \varepsilon = 0.2$). Then, each coefficient is gradually increased to test 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 properly high, the quality of generated solutions is better. This observation indicates that reassigning two tasks with a higher similarity of Service Time has the higher possibility to produce a new 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, a precedent task is reassigned, etc.), R_{ij}^{SST} is hard to represent the relatedness of two tasks and shows low contribution in tuning tests. A lower coefficient is given to it.

Too small *LEN_SEGMENT* may lead to the weights of operators changing frequently and search converging prematurely, while large *LEN_SEGMENT* cannot provide guidance 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. Experiment Results

To verify the contribution of variable depth, a standard ALNS variant for OPVRPTW is also implemented to be compared, which uses the Destroy and Repair operators only at the depth of *HORIZON* for globally searching. Comparing to other metaheuristics using small change operators, both VD-ALNS and ALNS have the 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 tables, we convert the objective value into Heavy-Loaded Distance Rate (HLDR) (Eq. (28)), which is widely used by logistic companies in practice. This objective also pursues the lowest unloaded travel distance like 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. The NF in tables means no feasible solution is generated.

$$HLDR = \frac{Loaded\ Distance}{Loaded\ Distance + Unloaded\ Distance} \tag{28}$$

Table 3: HLDR comparison on 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 comparison 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, which indicates that the variable neighbourhood depth scheme does improve the search performance of ALNS. This scheme enhances the exploit ability in local area leading to the growth of 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 equal solutions, which shows no significantly difference. However, VD-ALNS takes remarkably less evaluation times and 90% running time of VNS-RLS to obtain that results. All the three methods have close stability where their difference on S.D. is lower than 1%.

Table 5: HLDR comparison on original 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%	%	66.63%	78.16%
	Ave	80.21%	68.62%	%	66.11%	77.78%
	Times	212	281		271	267
	S.D.	0.99%	0.36%	%	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	%	69.16%	**65.27%**	%	77.43%
	Ave	%	64.76%	64.79%	%	76.64%
	Times		44	251		274
	S.D.	%	3.04%	0.35%	%	0.56%
VD-ALNS	Best	**81.74%**	%	65.16%	%	77.39%
	Ave	77.04%	%	**64.84%**	%	76.52%
	Times	483		303		387
	S.D.	1.20%	%	0.24%	%	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	%	%	%	%	%
	Ave	%	%	%	%	%
	Times					
	S.D.	%	%	%	%	%
VD-ALNS	Best	%	%	%	%	73.07%
	Ave	%	%	%	%	72.59%
	Times					365
	S.D.	%	%	%	%	0.34%
Upper Bound		77.04%	77.55%	78.82%	62.53%	76.09%

Table 6: HLDR comparison on original 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	%	%	%	%	%	%	%	%
	Ave	%	%	%	%	%	%	%	%
	Times								
	S.D.	%	%	%	%	%	%	%	%
VD-ALNS	Best	%	77.15%	69.03%	**73.66%**	61.04%	65.33%	%	%
	Ave	%	76.83%	68.51%	**72.78%**	60.40%	64.80%	%	%
	Times		253	309	315	400	255		
	S.D.	%	0.18%	0.38%	0.64%	0.43%	0.49%	%	%
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	%	%	%	%	%	%	%	%
	Ave	%	%	%	%	%	%	%	%
	Times								
	S.D.	%	%	%	%	%	%	%	%
VD-ALNS	Best	**88.71%**	89.74%	%	%	%	62.30%	%	%
	Ave	**85.96%**	86.67%	%	%	%	61.29%	%	%
	Times	339	347				343		
	S.D.	2.43%	1.77%	%	%	%	0.76%	%	%
Upper Bound		98.26%	97.97%	87.06%	92.44%	74.27%	71.36%	70.29%	56.54%

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

4.4. Evaluating the Contribution of Operators

Table 7 provides a statistic on the Destroy and Repair operators. On the scaled down dataset, we ran VD-ALNS while a single operator is excluded and the others are kept. When each operator is excluded by turn, the degeneration caused by missing that specific operator is recorded. The second and third columns show the average degeneration on the best-found solution and average solution, while the last two columns give the maximum degeneration on the dataset.

Table 7: Evaluation of contribution 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 usefulness of each operator in VD-ALNS. It can be found that, Worst Edge Removal is the most efficient destroy operator, followed by the Worst Removal. The Related Removal contributes the least in this case. Among repair operators, Greedy Insertion is the most useful one, followed by the Regret2 Insertion. Overall, greedy heuristics provide effective complement on search intensification and outperform the other heuristics. It proofs that the using of exact method is a crucial factor to the performance of ALNS.

4.5. Analysis of Runtime

The Destroy and Repair operators in ALNS bring greater change than traditional neighbourhood operators by operating more nodes and greater perturbation in each usage of the operators. Therefore, the calculation time spent on choosing removal nodes and insertion places is considerable. To obtain the results presented in the tables, the evaluation times of ALNS and VD-ALNS are significantly less than VNS-RLS, but the running time of VD-ALNS is around 43% more than VNS-RLS on the original instances, while it is slightly less than VNS-RLS on small instances. This observation indicates that the runtime of VD-ALNS increases faster than VNS-RLS with the growth of instance size.

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

5. CONCLUSIONS

Open Periodic Vehicle Routing Problem with Time Windows (OPVRPTW) has a large scale search space with tight side constraints, which arises from a real-world container transportation problem. This paper proposed a Variable-Depth Adaptive Large Neighbourhood Search algorithm (VD-ALNS) for OPVRPTW, using specially tailored four destroy operators and three repair operators at variable neighbourhood depth. In this vehicle routing problem with high-dimensional solution structure, the variable depth scheme significantly promotes the performance of the proposed algorithm.

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 efficiently find good solutions. Comparing to the existing solution metaheuristic with small change operators, a number of new best-found solutions were produced by VD-ALNS. In the future research, multi-objective feature and other trade-off strategies between solution quality and search speed will be able to cooperate with ALNS. It will be also possible to apply advanced customized exact methods to both destroy and repair operators.

ACKNOWLEDGMENTS

This research was supported by Ningbo Science & Technology Bureau (2014A35006), Royal Society International Exchange Scheme, 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.

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.

Cordeau J.F., Laporte G., and Mercier A., 2004. Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. Journal of the Operational Research Society 55, 5 (2004), 542–546.

Dayarian I., Crainic T.G., Gendreau M., and Rei W., 2013. An adaptive large neighborhood search heuristic for a multi-period vehicle routing problem. Report. Technical Report CIRRELT-2013-60, Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT).

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., Mladenoviċ N., and Pėrez J.A.M., 2010. Variable neighbourhood search: methods and applications. Annals of Operations Research 175, 1 (2010), 367–407.

Hemmelmayr V.C., Cordeau J.F., and Crainic T.G., 2012. An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics. Computers & Operations Research 39, 12 (2012), 3215–3228.

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.

Masson R., Lehuėdė F., and Pėton O., 2013. An adaptive large neighborhood search for the pickup and delivery problem with transfers. Transportation Science 47, 3 (2013), 344–355.

Mladenoviċ 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.

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.

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.

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.

Savelsbergh M.W.P., 1992. The vehicle routing problem with time windows: Minimizing route duration. ORSA journal on computing 4, 2 (1992), 146–154.

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.

Talbi E.G., 2009. Metaheuristics: from design to implementation. Vol. 74. John Wiley & Sons.

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.

Wen M., Krapper E., Larsen J., and Stidsen T.K., 2011. A multilevel variable neighborhood search heuristic for a practical vehicle routing and driver scheduling problem. Networks 58, 4 (2011), 311–322.

Wieberneit N., 2008. Service network design for freight transportation: a review. OR spectrum 30, 1 (2008), 77–112.