Flexible Services and Manufacturing Journal (2019) 31:331–353
https://doi.org/10.1007/s10696-018-9323-0

A constraint relaxation-based algorithm for the load-dependent vehicle routing problem with time windows

Ran Liu[1] · **Zhibin Jiang**[1]

Published online: 27 August 2018

Abstract
We introduce the load-dependent vehicle routing problem with time windows (LDVRPTW) in this paper. Transportation costs in this new problem, unlike those in the classical vehicle routing problem with time windows (VRPTW), are calculated based on not only the travel distances but also the vehicular loads on travel arcs. To solve this challenging NP-hard problem, we design a new constraint relaxation-based algorithm. In the proposed algorithm, a new constraint relaxation is introduced, i.e., some clients are not visited by a real vehicle and instead are entrusted to an additional virtual vehicle. Based on this relaxation, we present an effective execution scheme of local search procedures. The proposed algorithm is tested on benchmark instances of several special cases of the LDVRPTW, including the VRPTW. Numerical results for different variant problems demonstrate that the algorithm consistently yields impressive results: in particular, for one special variant, namely the fuel consumption rate considered vehicle routing problem (FCR-VRP), the algorithm improves the best-known solutions found by existing state-of-the-art methods.

Keywords Heuristic · Vehicle routing · Time windows · Constraint relaxation

1 Introduction

The vehicle routing problem (VRP) is an important aspect in a diverse range of application systems, including distribution, transportation, healthcare, and supply chains. The VRP costs are traditionally derived from the total distances traveled by

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10696-018-9323-0) contains supplementary material, which is available to authorized users.

✉ Ran Liu
liuran2009@sjtu.edu.cn

[1] Department of Industrial Engineering and Management, Shanghai Jiao Tong University, Shanghai, China

the vehicles or the total travel times consumed by the vehicles (Laporte 2009). However, in numerous practical scenarios, the cost of a vehicle that travels between two nodes depends on not only classical attributes, such as the travel distance, but also the load of the vehicle on the corresponding arc. Two industries exemplify this two-fold computation of the cost, the first of which is the expressway system in China. Most expressways there are owned by for-profit corporations and adopt the toll-by-weight scheme, i.e., tolls are charged in accordance with not only the travel distance of the vehicle but also its weight. In this context, the routing problem solved by carriers should not be modeled as the classical VRP (Zhang et al. 2012). The second is the supply-chain industry, in which companies such as those in cold-chain logistics also adopt both distance and weight as the primary factors in calculating travel costs. In this context, the classical VRP model is not applicable when the companies desire to optimize the vehicular routes. In this paper, we focus on the load-dependent VRP with time windows (LDVRPTW), in which transportation costs are calculated based on the travel distance and vehicular load on the travel arcs and each client must be served within a hard time window. It is evident that the LDVRPTW is an extension of the classical VRP with time windows (VRPTW). It contains many existing routing problems as special cases and can be applied to many practical situations.

Although many researchers consider the vehicular load as a factor in objective functions, the literature remains limited on the VRP with a load-dependent objective and time-window constraints. Zachariadis et al. (2015) examined a load-dependent VRP with the objective of minimizing the total product of the distance traveled and the gross weight carried along this distance. Furthermore, Zachariadis et al. (2015) extended the problem to simultaneous pickup-and-delivery services. However, they did not consider the time window assigned to each service in their problem.

In this paper, we design a new heuristic to address the LDVRPTW. This approach can also effectively solve some LDVRPTW variant problems. The proposed algorithm not only compares favorably with previous algorithms from the literature, but also yields new best solutions on benchmark instances for specific variants of the LDVRPTW, such as the fuel consumption rate considered vehicle routing problem (FCR-VRP).

The main contributions of this paper is twofold. First, this paper introduces a new, practical and important problem, the load-dependent vehicle routing problem with time windows. In the problem the load-dependant transportation cost and time window constraints are considered simultaneously. Since the basic VRP and VRPTW are well-known NP-hard problems, obviously, the proposed problem is also NP-hard, i.e., we cannot find a polynomial time algorithm o solve the problem unless P = NP. Therefore, we design a new constraint relaxation-based heuristic algorithm for the LDVRPTW and its variant problems. Although it is built on a general tabu search framework, it uses a new constraint relaxation that disregards the one-visit-per-client requirement.

In the literature, some relaxation schemes have been used in the algorithms for the VRPs. For example, Gendreau et al. (1994) designed a tabu search heuristic for solving the VRPTW, in which a solution might violate the maximum load and duration constraints associated with the routes, and the time-window constraints associated with the customers and the depot. Each solution of the tabu search was then

evaluated using a cost function consisting of the original cost and penalty cost which was proportional to the violation of the constraints. Vidal et al. (2012) proposed a Genetic Algorithm for the multi-depot and the periodic VRPs, in which the vehicular capacity and the route maximum duration were relaxed. Similarly, the penalties for exceeding such constrains were computed as part of the route costs. In our algorithm, in addition to the classical relaxations, we relax the one-visit-per-client requirement in the algorithmic search process. Therefore, it is probable that only a subset of clients are visited and served in a solution. Correspondingly, we assume that an additional *virtual* vehicle is available to serve all un-assigned clients. This relaxation enables the algorithm to explore a broader solution space and enhance the search capability. Furthermore, based on this relaxation, we design the corresponding inter-route neighborhood structures and introduce a new local search (LS)-based post-optimization execution scheme. We refer to this algorithm as the *Virtual Vehicle-based Tabu Search* (VVTS).

The remainder of the paper is organized as follows. Section 2 defines the LDVRPTW. Section 3 provides a review of the relevant literature. Section 4 summarizes the framework of the proposed algorithm, while Sects. 5–8 detail its main components. Finally, Sect. 9 provides the conclusions.

2 Problem statement

Let $G=(V, A)$ be a directed graph, where $V=\{0\}\cup\{1,\ldots,n\}\cup\{n+1\}$ is the node set and $A=\{\{(i, j)\colon i, j\in V, i\neq j\}$ is the arc set. $C=\{1,\ldots,n\}$ denotes the set of clients. Nodes 0 and $n+1$ represent the depot for start and finish of the route, respectively. Let K be the set of all vehicles. Each vehicle $k\in K$ has a capacity Q, and a no-load weight w. Each vehicular route corresponds to a path that starts at node 0 and ends at $n+1$. Each client $i\in C$ has a demand q_i to be delivered from the depot. For convenience of notation, nodes 0 and $n+1$ are assigned to demands $q_0=q_{n+1}=0$. A time window $[e_i, l_i]$ is associated with each node $i\in V$ to ensure that the visit to node i occurs only between e_i and l_i, i.e., a vehicle is allowed to reach i before e_i, but must wait until e_i before the service can be performed. Each arc $(i, j)\in A$ is associated with a travel distance d_{ij} and a travel time τ_{ij}. The service duration for a client is included in the travel time from this node. Given a vehicular route r, let (r_0, r_1, …, $r_{n(r)}$, $r_{n(r)+1}$) be the sequence of the nodes on this route, where r_i denotes the ith nodes in r, r_0 and $r_{n(r)+1}$ both represent the depot, and $n(r)$ refers to the number of clients on this route. The vehicular starting weight on this route is $w+\sum_{i=1}^{n(r)} q_{r_i}$, and vehicular load is progressively reduced by q_{r_i} with a visit to client i along route r. The travel cost along an arc (i, j) is calculated as $d_{ij}\times f(w_{ij})$, where $f(w_{ij})=c_d+c_w w_{ij}$, w_{ij} denotes the vehicular total weight (no-load vehicular weight and its load) during this travel, c_d is a non-negative constant that represents the vehicular travel cost per unit distance, and c_w is the constant of delivering the product per unit weight and unit distance. The LDVRPTW is to determine a set of routes with the minimum total cost, such that the following are attained: (1) each route is assigned to exactly one vehicle; (2) the total demand of all clients served on a route cannot exceed the

vehicular capacity Q; and (3) each client is visited once within the corresponding time window.

A three-index mathematical formulation model to the problem is presented as follows. Three types of decision variables are used. Let the binary variable x_{ijk} be the number of times traveled by vehicle k through arc (i, j), y_{ijk} be the total weight on arc (i, j) of vehicle k, and t_{ik} be the time at which vehicle k begins to serve at node i.

$$\min \sum_{i\in C\cup\{0\}} \sum_{j\in C\cup\{n+1\}} \sum_{k\in K} x_{ijk} d_{ij}(c_d + c_w y_{ijk}) \tag{1}$$

Subject to:

$$\sum_{j\in C\cup\{n+1\}} x_{ijk} = \sum_{j\in C\cup\{0\}} x_{jik} \quad \forall i \in C,\ \forall k \in K \tag{2}$$

$$\sum_{k\in K} \sum_{j\in C\cup\{n+1\}} x_{ijk} = 1 \quad \forall i \in C \tag{3}$$

$$\sum_{j\in C\cup\{n+1\}} x_{0,j,k} = \sum_{j\in C\cup\{0\}} x_{j,n+1,k} = 1 \quad \forall k \in K \tag{4}$$

$$\sum_{i\in C\cup\{0\}} \sum_{j\in C\cup\{n+1\}} q_i x_{ijk} \le Q \quad \forall k \in K \tag{5}$$

$$\sum_{j\in C\cup\{0\}} \sum_{k\in K} y_{jik} - \sum_{j\in C\cup\{n+1\}} \sum_{k\in K} y_{ijk} = q_i \quad \forall i \in C \tag{6}$$

$$\sum_{j\in C\cup\{0\}} y_{j,n+1,k} \ge w \quad \forall k \in K \tag{7}$$

$$t_{ik} + \tau_{ij} - \left(1 - x_{ijk}\right)M \le t_{jk} \quad \forall i \in C \cup \{0\}, \forall j \in C \cup \{n+1\}, k \in K \tag{8}$$

$$e_i \le t_{ik} \le l_i \quad \forall i \in V, k \in K \tag{9}$$

$$x_{ijk} \in \{0, 1\} \quad \forall i \in C \cup \{0\},\ j \in C \cup \{n+1\},\ k \in K \tag{10}$$

The objective function (1) serves to minimize the total vehicular costs, which depends on the travel distance along an arc and the flow on the arc. Constraints (2) to (4) specify that each client must be visited by exactly one vehicle. Constraints (5)

to (7) ensure the feasibility of the vehicular capacity. Constraints (8) and (9) impose the time windows.

3 Literature review

The LDVRPTW is a variant of the VRPTW. There have been numerous works on the VRPTW in the past three decades, including both heuristic approaches and exact methods. For example, Bard et al. (2002) designed a branch-and-cut algorithm to the VRPTW; Desaulniers et al. (2008) presented an efficiency branch-and-price-and-cut approaches to the VRPTW. The most recent and best exact approaches have been designed by Desaulniers et al. (2008), Baldacci et al. (2011) and Røpke (2012), whereas the most recent and highly-competitive heuristic approaches have been presented by Muter et al. (2010), Demir et al. (2012), Vidal et al. (2013), and Kramer et al. (2015). Readers are referred to the reviews by Bräysy and Gendreau (2005a, b) and Desaulniers et al. (2014) for greater insights into the literature. The main objective of much of the literature on the VRPTW is to minimize either the number of vehicles or the total distance traveled (Braysy and Gendreau 2005a). In this regard, the objective of the VRPTW can be subsumed into that of the LDVRPTW, e.g., the LDVRPTW can be reduced to the VRPTW with an objective of minimizing travel distances when $c_d = 1$ and $c_w = 0$. Compared with the classical VRPTW, there are fewer works on the LDVRPTW. We review the load-dependent VRP both with and without time-window constraints, and discuss the relationships between the LDVRPTW and these problems.

The first related problem is the Energy Minimizing VRP introduced by Kara et al. (2007), in which the linear weighted distance objective concerned the vehicular energy requirements. The authors formulated a mathematical model and solved the problem using the Cplex solver. Motivated by the toll-by-weight schemes implemented on expressways in over twenty provinces in China, Zhang et al. (2010) introduced the toll-by-weight VRP (TBW-VRP), in which the transportation cost per unit distance monotonically increased with the vehicular total weight. The authors designed a branch-and-bound algorithm to solve a simplified problem with only one vehicle. The toll-by-weight VRP is a special case of the LDVRPTW when tolls are charged proportionally to the vehicular weight. For example, as shown by Zhang et al. (2012), in the Gansu Province of China, the expressway toll on road (i, j) was calculated as $d_{ij} \times 0.08 w_{ij}$ where w_{ij} represents the vehicular weight on this road. In this case, the toll-by-weight VRP can be converted into the LDVRPTW with $c_d = 0$, $c_w = 0.08$ and the relaxation of the time-window constraints. Xiao et al. (2012) demonstrated that the fuel consumed by a vehicle per unit travel distance (referred to as fuel consumption rate, *FCR*) was proportional to the vehicular load. They defined a linear relationship between the *FCR* and the vehicular load. Based on this definition, Xiao et al. (2012) built a mathematical model for the FCR-VRP and developed a simulated annealing method to solve the problem. Gaur et al. (2013) established a constant-factor approximation algorithm for the FCR-VRP. Zachariadis et al. (2015) extended

the basic load-dependent VRP by allowing clients to simultaneously require pick-up and delivery services. This extension aimed to minimize the total product between the distance traveled and the gross weight carried along this distance. To address large-scale instances of the examined problems, the authors proposed an effective local-search algorithm. They further demonstrated that the FCR-CVRP of Xiao et al. (2012) could also be viewed as a special case of their problem and solved the FCR-CVRP benchmark instances used by Xiao et al. (2012). Kopfer et al. (2014) introduced a special VRP, aiming at minimizing fuel consumption instead of driving distances. The classical distance-minimizing objective function was replaced by a vehicle-specific affine linear fuel-consumption function. Additional constraints to determine the actual payload along a vehicular route have been established. According to their experimental results, the authors found that the quantity of fuel needed to serve a given request portfolio could be reduced tremendously by using an inhomogeneous fleet with vehicles of different sizes. Besides the transportation distance and the loading weight, some researchers have considered the vehicular travel speed in models for calculating fuel consumption and integrated it into the problem objective. For example, Kuo (2010) integrated the transportation distance, transportation speed and loading weight into a model and designed a simulated annealing heuristic to optimize the VRP with the objective of minimizing fuel consumption. The author performed computational experiments comparing fuel consumption, transportation time and travel distance for the time-dependent VRP. Kuo and Wang (2011) re-considered the problem of Kuo (2010) and solved it with a tabu search algorithm. Figliozzi (2010) focused on a problem in which the minimization of emissions and fuel consumption was the primary objective or was part of a generalized cost function, and departure times and travel speeds became decision variables. A formulation and solution approaches were presented. Results obtained with the proposed solution approaches for different levels of congestion were compared and analyzed.

In addition to the above problems, a relatively new problem, the Green VRP, is also relevant to the LDVRPTW, in which the influences to the environment such as carbon dioxide-equivalent (CO_2e) emissions are considered. The Pollution Routing Problem (PRP) (Bektaş and Laporte 2011; Grabenschweiger et al. 2017) is a typical Green VRP, which seeks to minimize a more comprehensive objective that accounts for not only operational costs, but also environmental costs, such as the amount of greenhouse emissions (Bektaş and Laporte 2011). The total travel distance, load carried per distance unit and vehicular speed have been simultaneously adopted as the components of the objective. Bektaş and Laporte (2011) first built the mathematical models for the PRP, and used the Cplex solver to perform extensive computational experiments on realistic test instances. Based on this work, Demir et al. (2012) presented an extended adaptive large-neighborhood search for the PRP. Their algorithm integrated the classical ALNS scheme with a specialized speed-optimization algorithm that computed optimal speeds on a given path to minimize fuel consumption, emissions and driver costs. Jabali et al. (2012) studied the trade-off between minimizing CO_2 emissions and minimizing the total travel times. As emissions were directly related to the vehicular speed, time-dependent travel times were included in their optimization models. Three different models were compared:

a model for minimizing the total travel time; a model for minimizing the total CO_2 emission (which in turn depended on travel times and speed); and a cost-based model that optimized the weighted average of travel time, emission and fuel costs. Assuming that carrier companies could limit the speed of their vehicles, the authors discussed the emission-optimal speed taking into account the impact exerted by traffic congestion on the actual travel time and emission. Later, Demir et al. (2014a) examined the bi-objective PRP, in which one of the objectives was related to carbon dioxide equivalent (CO_2e) emissions and the other to driving time. An adaptive large-neighborhood search algorithm combined with a speed-optimization procedure was presented to solve the bi-objective PRP. New sets of instances based on real geographic data were generated to evaluate the effectiveness of the algorithm. Oberscheider et al. (2013) discussed a multi-depot vehicle routing problem with pickup and delivery and time windows. The authors contrasted the results obtained by minimizing fuel consumption to those obtained by minimizing driving times. They found that a significant reduction of CO_2 emissions could be achieved by the former approach than the latter. Kramer et al. (2015) designed a matheuristic approach for the PRP, with the objective of minimizing the operational and environmental costs while respecting capacity constraints and service time windows. The costs were based on drivers' wages and fuel consumption, which depended on factors such as the travel distance and vehicular load. To solve this problem, the authors proposed a sophisticated hybrid algorithm, which combined a local search-based metaheuristic with an integer-programming approach over a set covering formulation and a speed-optimization algorithm. The algorithm was tested on both existing PRP benchmark instances and on the classical VRPTW benchmark. For a detailed review of such studies, reference may be made to Bektaş and Laporte (2011) and Demir et al. (2014b).

In summary, given its real-life applications, the load-dependent VRP has attracted much attention in the vehicle routing field. However, the time-window constraints have not been considered in the related problems. In this paper, we focus on the LDVRPTW which has a load-dependent objective and time windows, and propose a new heuristic for the problem.

4 An overview of the solution method

The VVTS is based on the tabu search framework proposed by Gendreau et al. (1994). We outline the general structure of VVTS in Algorithm 1. Firstly, the parameters and tabu list of the VVTS are initialized. Then, an initial solution is generated by an insert heuristic and set as the current solution s^c. The VVTS explores the inter-route neighborhood set $N(s^c)$ of s^c and selects the best non-tabu solution s^1 from $N(s^c)$ (lines 4–6). Then, a set of intra-route LS operators is applied to improve s^1 (line 7). Next, the tabu list and tabu tenure are updated (step 8) and solution s^1 is set as the new current solution. The subsequent iteration is repeated (steps 3–10) until VVTS reaches the stopping conditions.

Many algorithms relax the *vehicular load* and *time-window* constraints in the VRPTW to obtain a broader solution space. Such a notion is inspired by the

Lagrangian Relaxation method. In our algorithm, we also absorb the Lagrangian relaxation technology, i.e., three constraints can be violated by an intermediate solution of the algorithm (i.e. a solution obtained by each of the algorithmic iteration): (1) the vehicular capacity, (2) the time window of each client, and (3) the requirement that each client has to be visited by a vehicle. When the third one is violated, some clients are not served by any vehicle. In other words, in an intermediate solution, it is probable that only some of the customers are visited and the others have no vehicles assigned to them. In such a case, we suppose such un-visited customers are instead visited by a *virtual vehicle*. To simplify the notation, its route is called the *virtual route* or route $|K| + 1$.

For a given route $r=(r_0, r_1,\ldots, r_{n(r)}, r_{n(r)+1})$, characterized by load $q(r)=\sum_{i=1}^{n(r)} q_{r_i}$ and driving distance $d(r) = \sum_{i=0}^{n(r)} dr_i, r_{i+1}$, suppose $(t_0, t_1,\ldots, t_{n(r)}, t_{n(r)+1})$ be the visit times associated with each stop. Based on the above relaxation method, route r is feasible if $q(r) \le Q$ and $t_i \in [er_i, lr_i]$ for each $0 \le i \le n(r)+1$; and a feasible solution s is defined as a set of feasible routes such that each client is visited exactly once on a single route. In the VVTS, the generalized cost function of solution s is defined as follows,

$$\phi(s) = cost(s) + \alpha P_c(s) + \beta P_{tw}(s) + \gamma P_n(s)$$

where $cost(s)$ is the original objective value of solution s, $P_c(s)$ and $P_{tw}(s)$ represent the violations of vehicular capacity and of time-window constraints, $P_n(s)$ is the number of the clients not served by any vehicle in set K (i.e., the number of clients to whom the virtual vehicle is assigned), and α, β and γ are the penalty coefficient parameters. In terms of the virtual route, because it is introduced to absorb the unvisited clients and no service route is actually executed by this vehicle, we define its routing costs equal 0. We also define its violations of the vehicular capacity and of time-window constraints equal 0. Thus, the generalized cost function of this virtual route equals $\gamma P_n(s)$.

Algorithm 1: General structure of VVTS

```
Initialize parameters and tabu list
Generate an initial solution s^0, s^c = s^0
Repeat
    Apply cross-exchange neighborhood to s^c, to construct solution set N_1(s^c)
    Apply in-out heuristic L_cross·μ times to s^c, to construct solution set N_2(s^c)
    Select the best solution s^1 from N_1(s^c) ∪ N_2(s^c) that is not tabu or satisfies the
    aspiration criterion
    Apply intra-route LS procedures to improve s^1
    Update the tabu list and tabu tenure
    s^c = s^1
Until stop-criterion are satisfied
Output the best feasible solution s*
```

The change in $P_c(s)$ caused by a traditional LS move such as the inter-route swap can be easily computed in $O(1)$ time. As for the time penalty $P_{tw}(\sigma)$, different penalty settings are employed for the VRPTW. Two methods are frequently used. Firstly, the earliest departure time at depot t_0 for route r is e_0 whereas the earliest service time t_{r_i} for node r_i is $\max\left(t_{r_{i-1}} + \tau_{r_{i-1},r_i}, e_{r_i}\right)$, $0 < i \le n(r) + 1$. The associated penalty is defined as linear for tardiness but not for early activities. It costs $O(n)$ time to evaluate a classical LS move precisely. Secondly, the earliest departure time t_0 for route r is e_0 whereas the earliest service time for r_i is $t_{r_i} = \max\left(\min\left(t_{r_{i-1}} + \tau_{r_{i-1},r_i}, l_{r_i}\right), e_{r_i}\right), 0 < i \le n(r) + 1$. This associated penalty was proposed by Nagata et al. (2010) and Schneider et al. (2013). Classical LS operators, such as the 2-opt* and inter-route insert, are assessed in $O(1)$ time. Our VVTS adopts the second penalty function.

During algorithmic iterations, parameters α, β, and γ are self-adjusted to facilitate search-space exploration. They are initialized with α_0, β_0, and γ_0 and are limited within the intervals $(\alpha_{\min}, \alpha_{\max})$, $(\beta_{\min}, \beta_{\max})$ and $(\gamma_{\min}, \gamma_{\max})$, respectively. If the incumbent solution is feasible, then α is divided by factor $1+\varphi_1$; otherwise, it is multiplied by $1+\varphi_1$. If the incumbent solution is infeasible but the vehicular-capacity constraint is satisfied, then the parameter is divided by $1+\varphi_2$ $(0<\varphi_2<\varphi_1)$. Parameters β and γ are adjusted likewise.

5 Initial solution and inter-route neighborhoods

An insertion algorithm is used to construct the initial solution s^0. Firstly, we choose the client with the tightest time-window constraint and generate the first single-client route. If two or more clients have the same tightest time-window constraint, the client closest to the depot is chosen. Subsequently, we evaluate the increase in the generalized cost function when each un-inserted client is inserted into either a position on the existing routes, or an empty route if the solution has fewer than $|K|$ routes. At this step, the insertion must satisfy the vehicular-load constraints, but the process may violate the time-window constraints. We execute the cheapest insertion and repeat the procedure until all clients are contained in s^0. We cannot guarantee the feasibility of s^0 because the time-window constraints are not enforced in the insertion.

Starting from the first solution s^0, the VVTS chooses the best non-tabu solution s^1 from the inter-route neighborhood of s^0. The design of the neighborhood structures is crucial for the accuracy and speed of the VVTS. Two types of inter-route neighborhoods are adopted by the VVTS: *cross-exchange* and *in–out heuristic-based* neighborhoods.

5.1 Standard cross-exchange neighborhoods

The fundamental concept of the cross-exchange neighborhood involves removing two segments from two routes, of which the length (the number of clients on the

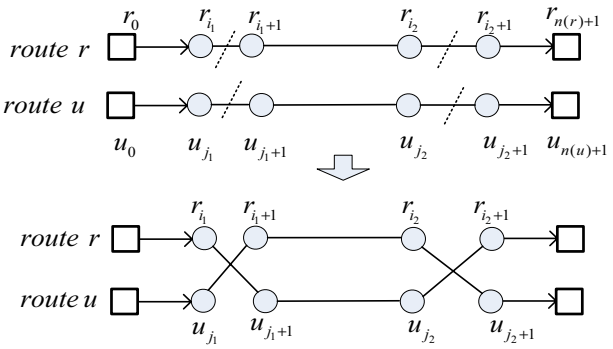

Fig. 1 An example of the standard cross-exchange operation

path) is at most L_{cross}, and exchanging them. The size of the cross-exchange neighborhood is $O(n^2(L_{\text{cross}})^2)$. As shown in Fig. 1, two edges (r_{i_1}, r_{i_1+1}) and (r_{i_2}, r_{i_2+1}) are removed from route r, and another two edges (u_{j_1}, u_{j_1+1}) and (u_{j_2}, u_{j_2+1}) are removed from route u. New edges (r_{i_1}, u_{j_1+1}), (r_{i_2}, u_{j_2+1}), (u_{j_1}, r_{i_1+1}) and (u_{j_2}, r_{i_2+1}) are then added to exchange the paths $(r_{i_1+1} \rightarrow r_{i_2})$ and $(u_{j_1+1} \rightarrow u_{j_2})$ between the routes. The generalized costs $\phi(r)$ and $\phi(u)$ are used to evaluate each cross-exchange move.

5.2 In–out cross-exchange neighborhood

Since a virtual vehicle is introduced to "serve" some clients, corresponding neighborhoods are designed to address this relaxation. Firstly, we apply the above cross-exchange neighborhood to a real route and the virtual vehicular route, i.e., we remove one segment from the real route and another from the virtual vehicular route, and then exchange them. We denote this neighborhood as the *in–out cross-exchange*. We use the generalized cost to evaluate each neighboring solution. Although it does not affect the generalized cost, the visit sequence of clients in the virtual route should be maintained in the in–out cross-exchange. This is because when a substring of clients is transferred from the virtual route to the real route, its visit sequence is important to the cost of the objective real route. Therefore, once a neighboring solution is identified, we regard the virtual route as the real route to adjust the position for insertion in the virtual route, i.e., use the sequence of clients on the virtual route to calculate the corresponding original objective value, capacity penalty and time-window penalty, and use the resultant sum of them to re-determine the best position for insertion in the virtual route.

5.3 In–out heuristic neighborhoods

In addition to the in–out cross-exchange, we design several simple and effective methods to exchange clients between the virtual and real vehicles. Unlike the

cross-exchange that modifies two routes for a move, these methods handle several routes within a move. We use the *removal method* to select and remove clients from the real vehicular routes and then apply the *inserting method* to insert un-served clients into the real routes.

5.3.1 Shaw removal heuristic

The core idea of *Shaw removal* involves removing a given number of somewhat similar clients. For example, the clients with similar or conjoint service time windows are highly likely to be visited successively in a route. Removing such similar clients from separate routes and re-inserting them into the solution may generate a better result. The relatedness measure of two clients in this paper depends on the distance between them. The detailed steps of Shaw removal are shown in Algorithm 2. In Shaw removal, we first assess the number of clients in the virtual route, referred to as $n(|K|+1)$. If this number exceeds pn (where p is a parameter controlling the percentage of the clients in the virtual route), we deem that this virtual route has too many clients and that clients do not need to be removed from the real routes. Otherwise, we randomly choose a client from the real routes and the virtual route, depending on whether virtual route is empty. We then incorporate this client into a set N_T. Another client closest to the just-removed client is chosen and inserted into set N_T. The process is repeated until N_T contains $pn-n(|K|+1)$clients. During the selection and insertion of clients into N_T, a parameter q is introduced to randomize the client selection. Given that a maximum of pn clients can be removed from the real route, a maximum of pn major iterations are performed in Shaw removal. In each iteration, we can fill and sort set N_T' in $O(nlogn)$ time. Thus, the overall time complexity of the Shaw removal method is $O(pn^2logn)$.

Algorithm 2. **Shaw removal**

If the number of clients on the virtual vehicular route is less than *pn,* **then**
  If the virtual vehicle is empty, **then**
  Randomly select a client *i* from real vehicular routes, $N_T=\{i\}$
  Else
  Randomly select a client *i* from vehicle $|K|+1$, $N_T=\{i\}$
  End if
  Repeat
    Randomly select a client *i'* from N_T
    Set temporary client set N_T' as all clients on real vehicular routes and not in N_T
    Sort N_T' according to the distance to *i'*
    Choose a random number *y* in [0,1], and select a client $j= N_T' [y^q| N_T' |]$
    $N_T= N_T \cup \{j\}$
  Until $|N_T|= pn-n(|K|+1)$
  Remove the clients N_T from real vehicular routes
End if

5.3.2 Worst removal heuristic

The detailed steps of *worst removal* are shown in Algorithm 3. The heuristic first checks the initial condition. If this condition is satisfied, it computes the reduced cost resulting from the removal of each client from the real routes. It sorts clients in accordance with cost reduction and selects one client randomly. A parameter q controls the randomization, as in Shaw removal. This randomness prevents the repeated removal of the same clients when the worst removal is repeated many times. Finally, the selected client is deleted from the real routes and inserted into N_T. The process is repeated until $pn-n(|K|+1)$ clients are removed, i.e., $|N_T|=pn-n(|K|+1)$. The worst removal has the same $O(pn^2 logn)$ computation time as the Shaw removal heuristic.

Algorithm 3. **Worst removal**

```
If the number of clients on virtual vehicular routes is less than pn then
  N_T=∅
  Repeat
    Set N_T' contains all clients on real vehicular route, and not in N_T
    Sort N_T' according to the reducing cost that is obtained by removing a client from a
    real route
    Choose a random number y in [0,1], and select a client j= N_T' [y^g| N_T' |]
    N_T= N_T ∪{j}
  Until |N_T|=pn
  Remove the clients N_T from real vehicular routes
End if
```

5.3.3 Insertion heuristic

The insertion method comprises two phases. In the first phase, all clients of N_T are inserted into the virtual vehicular route. In the second phase, the client(s) from the virtual route are inserted into the real routes. The stepwise descriptions are shown in Algorithm 4.

Firstly, the method inserts all clients of set N_T into the virtual route. Because the visiting sequence of the clients on the virtual route affects the other inter-route in–out neighborhoods and the second phase of the insertion method, we use the corresponding generalized cost to evaluate each possible position for insertion. This process continues until N_T is empty. In the second phase, each client in the virtual route is selected, and we attempt to insert the client into an existing real route or an empty route so as to reduce the generalized value of this solution. In this phase, the cost of the real routes is computed based on the generalized cost whereas the cost of the virtual route is now proportional to the number of clients. Once a client cannot be re-located from the virtual route to any real route with a better generalized cost,

this client is reserved on the virtual route, and the insertion method proceeds to the next client.

In the first phase, given that both N_T and the virtual route have a maximum of pn clients, the complexity of this phase is $O(p^3n^3)$. Because the set value of parameter p is typically small, i.e., 10% in the VVTS, the speed of the insertion algorithm remains high. In the second phase, since $O(pn)$ clients are not served in the virtual route and $O(n)$ positions for insertion are possible for each selected client, the time complexity is $O(pn^2)$. Notice that another type of *deeper* insertion method is designed in the standard ALNS algorithm (Ropke and Pisinger 2006). This deeper method searches each client on the virtual route and in each possible position for insertion in the real route to determine the insertion with the minimum global cost until no more clients from the virtual route can be inserted into the real route. This deeper insertion prolongs the computation time to $O(p^2n^3)$. In the preliminary experiments, our insertion method does not differ from the deep insertion in terms of solution accuracy.

Algorithm 4. Insertion method

```
1  n(|K|+1) is the number of clients on the virtual route
2  Repeat   // phase 1
3    Δ*=+∞
4    for i∈N_T  do
5      for each position j on the virtual route do
6        calculate the generalized objective value change Δ when inserting i into position j
7        if Δ<Δ*  then i*=i; j*=j; Δ*=Δ  end if
8      end for
9    end for
10   remove i* from N_T and insert it into the virtual route at position j*
11 Until N_T=∅
12 for i=0  to  n(|K|+1)  do  // phase 2
13     Δ*=0
14     for each position for insertion j in the real routes do
15       calculate the generalized objective change Δ when inserting the ith client into the
         virtual route (denoted as (|K|+1)_i ) into position j
16       if  Δ<Δ* then j*=j; Δ=Δ*  end if
17     end for
18     if Δ*<0 then  (|K|+1)_i is removed from the virtual route, and is inserted into the real
       route at position j* end if
19 end for
```

The cross-exchange examines all possible neighborhoods and selects the best one, whereas the in–out heuristic simply constructs two neighborhood solutions from one run of the Shaw removal and one run of the worst removal. Thus, we execute the in–out heuristic $L_{cross}\mu$ times within each VVTS iteration

(step 4, Algorithm 1), where L_{cross} is the maximal length of segments in the cross-exchange.

6 Intra-route local search and execution scheme

After obtaining s^1 from the inter-route neighborhoods, we use the Or-opt intra-route local search (LS) to improve its route. The Or-opt LS removes a path with a maximum length of $L_{Or\text{-}opt}$ and inserts it into another position on the same route. The position for insertion is limited within a distance (the number of clients) of L_{dis} from the original position of the path. Figure 2 depicts an example of the Or-opt, in which path $(r_i \rightarrow r_{i+l_1-1})$, $1 \le l_1 \le L_{Or\text{-}opt}$, is removed from route r and inserted in between either two conjoint clients $r_{i+l_1-1+l_2}$ and $r_{i+l_1+l_2}(0 \le l_2 \le L_{dis}$, the middle part), or r_{i-l_2-1} and $r_{i-l_2}(0 \le l_2 \le L_{dis}$, the bottom part). Two cases are considered for each insertion, i.e., the case with the original order of path $(r_i \rightarrow r_{i+l_1-1})$ and the case with its reversed visiting order. The size of the intra-route Or-opt neighborhood is $O(L_{Or\text{-}opt}L_{dis}n)$.

Embedding above LS can intensify and improve the algorithm performances. Two LS execution schemes are typically employed. The first, known as the *feasible local search* (feasible-LS) because it starts from a feasible seed and enters the feasible solution space, applies the LS to a feasible solution within a feasible solution space. The second, known as *infeasible local search* (infeasible-LS), starts from an infeasible or feasible seed. During the infeasible-LS search, both feasible and infeasible neighbor solutions may be generated. The infeasible-LS has been integrated into the meta-heuristic that facilitates the exploration of infeasible solutions (Hemmelmayr et al. 2009; Stenger et al. 2013). Although both schemes have been widely adopted, nearly all relevant studies have used only one scheme. Liu et al. (2014) combined the feasible-LS and infeasible-LS in the solution-improvement phase of their heuristic algorithm. They noted that the sequential application of the infeasible-LS and feasible-LS to improve the incumbent solution was superior. In the VVTS, we consider the feasibility of the seed solution, and design a new hybrid for the feasible-LS and infeasible-LS. It is given in Algorithm 5. The VVTS first judges the feasibility of the input solutions

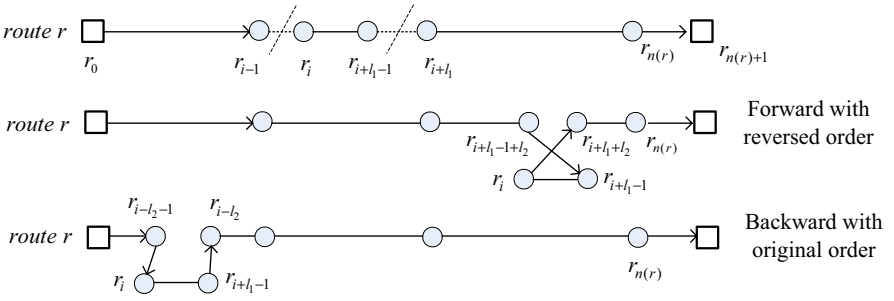

Fig. 2 An example of the intra-route Or-opt operation

s^c and s^1. If s^1 is feasible, then the VVTS applies the feasible-LS to extensively search its feasible neighborhood (steps 2–6). Otherwise, the VVTS applies the infeasible-LS to s^1 with a probability δ (steps 8–13).

Algorithm 5. Execution scheme of intra-route LS

1 **Input**: solution s^c, and s^1 obtained by inter-route neighborhood
2 **if** s^c and s^1 are feasible **then**
3 apply the intra-route feasible-LS to each modified route from s^c to s^1
4 **else if** s^c is infeasible and s^1 is feasible **then**
5 apply the intra-route feasible-LS to each route of s^1
6 **end if**
7 **if** s^1 is better than the incumbent best solution s^* **then** $s^* = s^1$ **end if**
8 **if** s^1 is infeasible **then**
9 apply the intra-route infeasible-LS to each route of s^1 with a probability δ
10 **if** s^1 is feasible **then**
11 Apply the intra-route feasible-LS to each route of s^1 and proceed to step 7
12 **end if**
13 **end if**
14 **output:** solution s^1

7 Tabu duration, aspiration and stopping criterion

The memory structure fundamentally guides the VVTS search process by preventing the search from being trapped into a local optimum. We record an attribute set $B(s) = \{(i, k) | i \in C, k \in K \cup \{|K| + 1\}$: client i is visited by vehicle $k\}$ for each solution during VVTS iterations. Thus, the transition from the current solution to a selected solution in the standard cross-exchange or in the in–out cross-exchange neighborhood can be regarded as the removal and addition of attributes in $B(s)$. The tabu duration is associated with each attribute. When one client i is removed from a route (including both the real vehicles and virtual vehicle), we assign a tabu status to attribute (i, k). The re-insertion of client i into route k is forbidden for the subsequent θ iterations of the VVTS. When we attempt to insert a set of clients into one or more routes simultaneously, this move is not tabu if the insertion of at least one client is not in its tabu status.

For each attribute (i, k), its aspiration value is set to a large positive value. When an attribute (i, k) falls into the tabu status in a neighborhood move, the attribute can be revoked if a feasible solution s is identified by this move and s has a smaller cost than the best feasible solution determined with this attribute (i, k). When a set of clients is inserted into one or more routes simultaneously,

the aspiration criteria are satisfied if this move generates a feasible solution that can improve the aspiration values of all the corresponding attributes.

8 Numerical experiments

We conduct several sets of experiments to evaluate the performance of the VVTS. The VVTS is coded in C++. Firstly, we tune the VVTS parameter values. Next, we compare the VVTS results with those of existing state-of-the-art methods on the benchmarks of the LDVRPTW variants. All the experiments are run on an Intel E5-2670 clocked at 2.6 GHz and 4 GB memory.

8.1 Parameter-setting

We adopt the guidelines by Ropke and Pisinger (2006) to tune the parameters. Firstly, eighteen tuning instances are generated based on the well-known Solomon's VRPTW benchmark instances. An initial parameter-setting is estimated according to previous studies and a trial-and-error procedure during the development of the algorithm. This parameter-setting is improved when a single parameter is allowed to vary within an interval while the others are fixed. We apply the VVTS to the tuning instances 10 times to tune each parameter value. The appropriate value is selected based on solution quality and computation time. We then proceed to the next parameter until all parameters have been tuned. Following the first round of parameter-tuning, we apply this setting as a new start seed and repeat the procedure described above. This procedure stops after two rounds of tuning. The ultimate setting is summarized in Table 1. In terms of the stopping criterion of the VVTS, we use different settings either to suit the characteristics

Table 1 VVTS parameter-setting

Explanation	Values
Penalty relative parameters	
α_0, α_{min}, α_{max}	1, 0.001, 10,000
β_0, β_{min}, β_{max}	1, 0.001, 10,000
γ_0, γ_{min}, γ_{max}	$c(s^0)/n$, $0.001\times c(s^0)/n$, $1000\times c(s^0)/n$
φ_1, φ_2	1.05, 1.10
Tabu duration parameters: θ	$[5\log_{10}n]$
Diversification parameters: λ	0.045
Inter-route neighborhood parameters	
L_{cross}, p, q, μ	5, 10%,4, $[2\log n]$
Intra-route search parameters:	
$L_{Or\text{-}opt}$, L_{dis}, δ	6, 30, (30% and 100%)

of the test instances, or to enable fair comparisons with existing state-of-the-art algorithms, which are presented in the following subsections.

8.2 Results for benchmark problems

The VVTS is tested on the standard benchmarks of various types of problems. The results are compared with those of state-of-the-art methods. Unless otherwise specified, we apply the VVTS to each benchmark instance 10 times.

8.2.1 Results for the FCR-VRP benchmark instances

The VVTS is first tested on the FCR-VRP instances generated by Xiao et al. (2012). The original instances by Xiao et al. (2012) are grouped into two sets: the first set ***C-FCR*** contains seven test instances built on the small- and medium-scale VRP instances with 50–199 clients; the second set ***G-FCR*** contains 20 test instances generated from large-scale VRP instances with 200–483 clients. Zachariadis et al. (2015) used these test instances to perform their experiments and compare with the algorithm by Xiao et al. (2012). We note that Zachariadis et al. introduced an additional constraint in their problem and experiments: the total duration (length) of a route could not exceed an upper bound; in the VVTS, this constraint is not considered. Therefore, for a fair comparison, in our experiments we select test instances that have infinity duration, i.e., all 7 instances from the C-FCR set and the last 12 instances from the G-FCR set. For relatively small-scale C-FCR instances, the VVTS ceases after 3000 iterations, i.e., after 3000 algorithmic iterations, the VVTS yields the best feasible solution as the final output solution. For large-scale G-FCR instances, the VVTS ceases after 5000 iterations. Our VVTS is compared with two recent and powerful methodologies presented by Zachariadis et al. (2015) (ZTK) and by Kramer et al. (2015) (KSVC). For these 19 test instances, the duration constraint is relaxed (infinity duration) and the solutions of various approaches can be compared directly. The detailed results are reported in the online supplement; the results in boldface indicate the best known solutions (BKSs).

We find that the proposed VVTS is superior to existing state-of-the-art approaches in terms of solution quality. For the seven C-FCR test instances, the VVTS yields six existing BKSs and one new BKS (C-FCR-5). For the 12 G-FCR large-scale instances, three algorithms ZTK, KSVC and VVTS are able to find three, three and eight BKSs, respectively. Additionally, the average best solution costs found by the ZTK, KSVC and our VVTS are 1580.11, 1581.38 and 1579.56, respectively. Considering the computation time, the KSVC method appears to be the fastest among the three approaches, with average run times of 0.3 min for C-FCR instances and 1.7 min for G-FCR instances. The ZTK method increases run times to 2.3 and 19.1 min for two sets of instance, and run time of VVTS for C-FCR and G-FCR instances are 1.4 and 29.1 min, respectively.

8.2.2 Results for the realistic TBW-VRP instances

We test the VVTS on the TBW-VRP instances generated by Zhang et al. (2012). In these instances, the information is obtained from the actual expressway network of the Gansu and Jiangxi Provinces in China. Five subsets of instances are constructed based on the practical information of each province; each subset consists of five instances resulting in a total of 50 test instances. In all instances, the unladen weight of the vehicle is 5 tones, and a scheme involving a transportation toll of 0.08 Chinese Yuan (RMB) per kilometer and per ton is adopted. The objective of the TBW-VRP is to minimize the transportation costs under this toll scheme rather than those based on the total travel distance. Zhang et al. (2012) designed a branch-and-bound algorithm for solving such test instances. In this part of experiments, the VVTS ceases after 5000 iterations.

We present the detailed computational results for these instances in the Gansu and Jiangxi Provinces in the online supplement. For all test instances, the VVTS can determine the optimal solutions (compared with the known optimal solutions in Zhang et al. 2012). For the test instances from the Gansu Province, the optimal solution is obtained during each VVTS algorithmic run for each instance. For the test instances from the Jiangxi Province, considering the total of 250 runs (25 test instances, each of which to be solved 10 times), the maximum deviation between the VVTS costs and optimal costs is only 0.14%. These statistical findings illustrate the high quality of the VVTS solution. With respect to computation time, our VVTS (average running time of 0.09 min) is faster than the approach by Zhang et al. (average running time of 14.38 min).

We think it is also interesting to see the impact of load-dependency on the routing decisions on such realistic data. So, in the online supplement (Table EC.3 and EC.4), we provide the "VRP solution cost" on each instance, i.e., we solve each instance as classical VRP with the objective of minimizing the total traveling distances, and then we evaluate the VRP solution using transportation costs under toll scheme. In other words, we let the vehicle travels along the VRP-solution-routes and calculate the corresponding real transportation cost. We also present the percentage deviation between VRP solution cost and our solution cost, calculated by (VRP solution cost − our solution cost)/our solution cost × 100%.

Not surprising, we find that the costs of VRP solutions are much larger than that of our solutions. For example, for all instances from Gansu Province, the average percentage deviation between our solutions costs and VRP solution costs is up to 118.8%; in terms of all instances from Jiangxi Province, the average gap is 96.12%. Note that in such instances the time windows are not incorporated. These results indicate that for such realistic instances the VRP solution cost that only minimizes the total travel distances is about the double of our solution cost that considers the impact of load-dependency on the routing decisions. To more clearly illustrate the difference between two type of solutions, in Figs. 3 and 4, we show the routes of our solutions and VRP solutions to two instances (the first instance of Gansu Province with 25 customers, and the first one of Jiangxi Province with 30 customers).

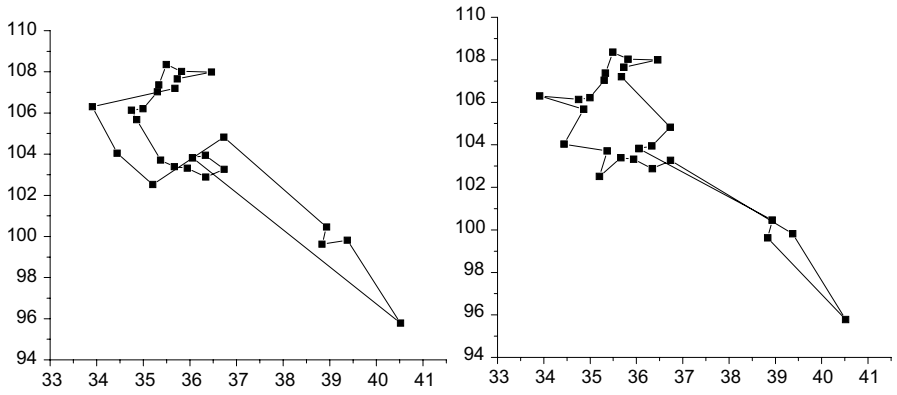

Fig. 3 Routes of our solution (left) and VRP solution (right) for a Gansu instance

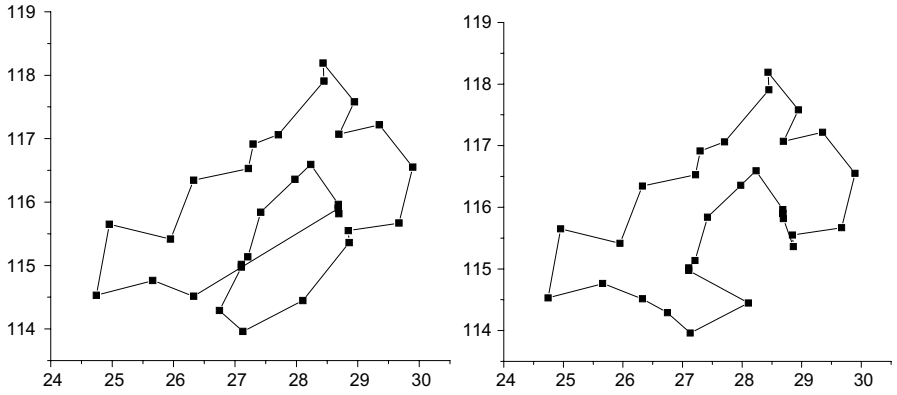

Fig. 4 Routes of our solution (left) and VRP solution (right) for a Jiangxi instance

8.2.3 Results for the VRPTW benchmark instances

In this subsection, we test the VVTS on the well-known VRPTW. The VVTS is then compared with existing state-of-the-art exact and heuristic methods. Specifically, the tests are performed on the Solomon benchmark (Solomon 1987) that includes 56 test instances. The Solomon instances consist of *C*, *R* and *RC* classes, which differ by the geographical distribution of the clients. Each class is divided into two series: 100-series instances with narrow time windows and 200-series instances with wide time windows. Different VRPTW objectives are considered in testing the Solomon instances: for example, minimizing the total traveling distance, the total duration, the number of vehicles, and any combinations of these factors (Braysy and Gendreau 2005a). We apply the total distance as our VRPTW objective in the experiment. Even with this simple objective, two arithmetic precisions were chosen by numerous studies to calculate the travel distance/time between each pair of clients: (1) the

travel time and distance are defined with the Euclidean distance, and (2) both are calculated with one decimal point and truncation. The former is generally used by heuristics whereas the latter is typically adopted by exact algorithms.

Firstly, we consider the Euclidean distance with one precision as the travel distance and time. In this case, the VRPTW has been studied extensively using various effective algorithms, such as the branch-and-price (Danna and Le Pape 2005; Desaulniers et al. 2008) and branch-and-cut (Bard et al. 2002) algorithms. In terms of the exact approaches, the best ones have been designed by Desaulniers et al. (2008), Baldacci et al. (2011) and Røpke (2012). The optimal solutions of such single-decimal instances have been obtained from such sophisticated approaches. In terms of the heuristic algorithms, the most recent and highly competitive ones have been presented by Muter et al. (2010), Demir et al. (2012) and Kramer et al. (2015). Muter et al. (2010) proposed a MetaOpt approach that integrated a column generation-based exact algorithm into a metaheuristic algorithm. This MetaOpt approach was applied to the VRPTW twice through double- and one-decimal truncated travel distances. The approach improved some best solutions identified prior to 2010. The ALNS in Demir et al. (2012) and the KSVC method in Kramer et al. (2015) were designed for the PRP and have been successfully applied to these VRPTW benchmarks with single decimals. Our VVTS also addresses such Solomon VRPTW problems through one-precision truncation, and the results are compared with previous state-of-the-art results. In this test, to fairly compete with these sophisticated metaheuristics, we solve each instance 15 times, with a maximum running time of 180 s as the stopping criterion of VVTS.

The detailed results are reported in the online supplement, where the BKS is in boldface. Despite the competition from significant approaches, the performance of the VVTS is Satisfied. The quality of the VVTS solutions exceeds that of the MetaOpt and ALNS by Demir et al. (2012). For example, out of the 56 instances with known optimal solutions, the MetaOpt and ALNS yield optimal solutions for 15 instances and 36 instances, respectively, whereas VVTS can solve all 56 instances optimally. It is noteworthy that KSVC also obtains outstanding results, yielding all optimal solutions except for one instance, RC106. In terms of the computation time, Muter et al. applied their approach three times for each test instance and set a time limit of 20 min for each execution. For almost all of these instances, the average computation time of the VVTS was 3 min, which was considerably shorter than that of the MetaOpt. However, our computation time is longer than that of the ALNS and KSVC.

Next, we do not truncate the travel times and distances among locations. We apply the VVTS to such VRPTW instances, and obtain the solution costs in treble precision. For such a case, to the best of our knowledge, the most recent studies are those by Muter et al. (2010), Vidal et al. (2013) and Kramer et al. (2015). The MetaOpt method by Muter et al. (2010) has been found to improve 17 BKSs. Then, Vidal et al. (2013) adopted a significant metaheuristic, namely, the hybrid genetic algorithm with advanced population diversity management (Vidal et al. 2012) (denoted as HGA), to solve the VRPTW and obtained positive results. Vidal et al. (2013) thereafter updated all the previous BKSs in the literature, yielding 27 new best solutions. Kramer et al. (2015) reported their results obtained with

double-precision distances. We compare the VVTS solutions with those obtained by Muter et al. (2010), Vidal et al. (2013) and Kramer et al. (2015). Again, we limit the computation time of the VVTS to 180 s. The detailed results are presented in the online supplement.

We find that the HGA by Vidal et al., the KSVC by Kramer et al., and our VVTS can obtain significantly positive results for such test instances. Compared with the MetaOpt method, the approaches of HGA, KSVC and VVTS improve the mean value of the best solution costs over all test instances at a gap of 0.64%. For 51 out of the 56 test instances, both the HGA and VVTS determine the same best solutions. For four instances (R102, RC108, R205, and RC202), the VVTS obtains new BKSs whereas, for one instance, the HGA obtains a better solution than VVTS. Despite the difference in the acquisition of results by VVTS and KSVC (the VVTS uses full-precision distances whereas the KSVC uses double-precision distances), the superiority of the VVTS is apparent: for three test instances (R107, RC106 and R211), the VVTS yields better solutions whereas, for other test instances, the two approaches obtain the same best solutions. In summary, we can conclude that the results of the HGA, KSVC and VVTS are comparable for the Solomon VRPTW benchmark and better than the existing state-of-the-art approaches. Furthermore, the VVTS in this paper can yield solutions of higher quality than the HGA and KSVC.

9 Conclusions

We study the LDVRPTW, a generalization of the VRPTW in which the traveling costs are computed based on not only the travel distance but also the vehicular load on travel arcs. Such increasingly-common cost structures are important for modern-day transportation problems. We propose a new constraint relaxation-based heuristic to efficiently address the LDVRPTW. The new relaxation scheme allows some clients to be served by a virtual vehicle, resulting in new structural differences to the algorithm compared with the existing relaxations. Based on this new relaxation scheme, we design corresponding neighborhood structures to broaden the access to the solution space. We report the computational results for test instances derived from the literature on the LDVRPTW-variant problems. The results of our experiments and comparisons with existing state-of-the-art algorithms demonstrate that the new heuristic is powerful and exhibits great potential for a wide range of complex practical problems.

Acknowledgements This work is supported by the National Natural Science Foundation of China (71672112), Shanghai Science and Technology Committee Major Program (17DZ1101202).

References

Baldacci R, Mingozzi A, Roberti R (2011) New Route relaxation and pricing strategies for the vehicle routing problem. Oper Res 59:1269–1283

Bard JF, Kontoravdis G, Yu G (2002) A branch-and-cut procedure for the vehicle routing problem with time windows. Transp Sci 36:250–269

Bektaş T, Laporte G (2011) The pollution-routing problem. Transp Res Part B: Methodol 45:1232–1250
Braysy O, Gendreau M (2005a) Vehicle routing problem with time windows, Part I: route construction and local search algorithms. Transp Sci 39:104–118
Braysy O, Gendreau M (2005b) Vehicle routing problem with time windows, Part II: metaheuristics. Transp Sci 39:119–139
Danna E, Le Pape C (2005) Branch-and-price heuristics: a case study on the vehicle routing problem with time windows. In: Desaulniers G, Desrosiers J, Solomon M (eds) Column generation. Springer, Berlin, pp 99–129
Demir E, Bektaş T, Laporte G (2012) An adaptive large neighborhood search heuristic for the pollution-routing problem. Eur J Oper Res 223:346–359
Demir E, Bektaş T, Laporte G (2014a) The bi-objective pollution-routing problem. Eur J Oper Res 232:464–478
Demir E, Bektaş T, Laporte G (2014b) A review of recent research on green road freight transportation. Eur J Oper Res 237:775–793
Desaulniers G, Lessard F, Hadjar A (2008) Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transp Sci 42:387–404
Desaulniers G, Madsen OBG, Ropke S (2014) Chapter 5: the vehicle routing problem with time windows. In: Vehicle routing: problems, methods, and applications, pp 119–159
Figliozzi M (2010) Vehicle routing problem for emissions minimization. In: Transportation research record: journal of the transportation research board, pp 1–7
Gaur DR, Mudgal A, Singh RR (2013) Routing vehicles to minimize fuel consumption. Oper Res Lett 41:576–580
Gendreau M, Hertz A, Laporte G (1994) A tabu search heuristic for the vehicle routing problem. Manage Sci 40:1276–1290
Grabenschweiger J, Tricoire F, Doerner KF (2017) Finding the trade-off between emissions and disturbance in an urban context. Flex Serv Manuf J. https://doi.org/10.1007/s10696-017-9297-3
Hemmelmayr VC, Doerner KF, Hartl RF (2009) A variable neighborhood search heuristic for periodic routing problems. Eur J Oper Res 195:791–802
Jabali O, Woensel T, De Kok A (2012) Analysis of travel times and CO2 emissions in time-dependent vehicle routing. Prod Oper Manag 21:1060–1074
Kara İ, Kara B, Yetis MK (2007) Energy minimizing vehicle routing problem. In: Dress A, Xu Y, Zhu B (eds) Combinatorial optimization and applications. Springer, Berlin, pp 62–71
Kopfer HW, Schönberger J, Kopfer H (2014) Reducing greenhouse gas emissions of a heterogeneous vehicle fleet. Flex Serv Manuf J 26:221–248
Kramer R, Subramanian A, Vidal T, Cabral LDAF (2015) A matheuristic approach for the pollution-routing problem. Eur J Oper Res 243:523–539
Kuo Y (2010) Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. Comput Ind Eng 59:157–165
Kuo Y, Wang C-C (2011) Optimizing the VRP by minimizing fuel consumption. Manag Environ Qual: Int J 22:440–450
Laporte G (2009) Fifty years of vehicle routing. Transp Sci 43:408–416
Liu R, Xie X, Garaix T (2014) Hybridization of tabu search with feasible and infeasible local searches for periodic home health care logistics. Omega 47:17–32
Muter İ, Birbil Şİ, Şahin G (2010) Combination of metaheuristic and exact algorithms for solving set covering-type optimization problems. Inform J Comput 22:603–619
Nagata Y, Bräysy O, Dullaert W (2010) A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Comput Oper Res 37:724–737
Oberscheider M, Zazgornik J, Henriksen CB, Gronalt M, Hirsch P (2013) Minimizing driving times and greenhouse gas emissions in timber transport with a near-exact solution approach. Scand J For Res 28:493–506
Røpke S (2012) Branching decisions in branch-and-cut-and-price algorithms for vehicle routing problems. In: Presentation in international workshop on column generation. http://www.gerad.ca/colloques/ColumnGeneration2012/presentations/session7/Ropke.pdf
Ropke S, Pisinger D (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp Sci 40:455–472
Schneider M, Sand B, Stenger A (2013) A note on the time travel approach for handling time windows in vehicle routing problems. Comput Oper Res 40:2564–2568

Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res 35:254–265

Stenger A, Vigo D, Enz S, Schwind M (2013) An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. Transp Sci 47:64–80

Vidal T, Crainic TG, Gendreau M, Lahrichi N, Rei W (2012) A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Oper Res 60:611–624

Vidal T, Crainic TG, Gendreau M, Prins C (2013) Time-window relaxations in vehicle routing heuristics. Tech. rep., CIRRELT, Montréal

Xiao Y, Zhao Q, Kaku I, Xu Y (2012) Development of a fuel consumption optimization model for the capacitated vehicle routing problem. Comput Oper Res 39:1419–1431

Zachariadis EE, Tarantilis CD, Kiranoudis CT (2015) The load-dependent vehicle routing problem and its pick-up and delivery extension. Transp Res Part B: Methodol 71:158–181

Zhang Z, Qin H, Lim A, Guo S (2010) Branch and bound algorithm for a single vehicle routing problem with toll-by-weight scheme. In: García-Pedrajas N, Herrera F, Fyfe C, Benítez J, Ali M (eds) Trends in applied intelligent systems. Springer, Berlin, pp 179–188

Zhang Z, Qin H, Zhu W, Lim A (2012) The single vehicle routing problem with toll-by-weight scheme: a branch-and-bound approach. Eur J Oper Res 220:295–304

Ran Liu is an Associate Professor at Department of Industrial Engineering & Management, in Shanghai Jiao Tong University, Shanghai, China. He received his Ph.D. from Shanghai Jiao Tong University, Shanghai, China, in 2012. His research interests include combinational optimization, algorithm design and analysis.

Zhibin Jiang received his Ph.D. degree in manufacturing engineering and engineering management from City University of Hong Kong, Hong Kong, China, in 1999. He is Changjiang Scholar Chair Professor of MOE, China, and Distinguished Professor of Shanghai Jiao Tong University, Shanghai, China. His research interests include discrete-event modeling and simulation, operation management in manufacturing industries and health-care systems.