IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 19, 2022, accepted May 1, 2022, date of publication May 10, 2022, date of current version May 13, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3174081

A Vehicle Routing Problem With Option for Outsourcing and Time-Dependent Travel Time

MARK POON[1], RUIXUE GU[2], AND YILIANG YUAN[2]
[1]Department of Logistics and Operations Management, HEC Montréal, Montréal, QC H3T 2A7, Canada
[2]Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore 119077

Corresponding author: Ruixue Gu (e0251049@u.nus.edu)

ABSTRACT This paper studies the time-dependent vehicle routing problem with private fleet and common carriers (TDVRPPC), which provides an option to outsource the customer requests and considers real-world time-dependent travel times. The problem is commonly seen in the transportation and logistics industries as it considers the impact of changing traffic conditions on travel times, maximum working hour regulations as well as vehicle capacity constraints. The time-dependent travel time is modeled as a piecewise linear function, based on which a mixed integer programming model is proposed for the TDVRPPC. To solve this NP-hard problem, we customize a hybrid algorithm to harness an adaptive large neighborhood search algorithm for exploration and a tabu search for the exploitation of the search. Through constraint relaxation, dynamic and coordinated adjustment of diversification and intensification strengths of the hybridized procedures, as well as an effective segment-based evaluation method, the proposed algorithm performs well on newly generated test instances for the TDVRPPC and on benchmark instances for the simplified vehicle routing problem with private fleet and common carriers (VRPPC).

INDEX TERMS Vehicle routing, outsourcing, time-dependent travel time, hybrid meta-heuristics, mathematical model.

I. INTRODUCTION

This paper investigates a goods distribution problem motivated by the challenges encountered by a supermarket operator in Singapore. The supermarket operator owns more than 160 outlets across the island-city and handles numerous delivery requests to the outlets each day. It adopts a mixed strategy to both dispatch self-owned trucks and outsource to third-party logistic (3PL) providers to meet the delivery requests of the outlets. The operator cannot neglect the impact of varied traffic conditions on the road throughout the day, which greatly undermines the decisions of outsourcing and route planning. This paper models the problem as the time-dependent vehicle routing problem with private fleet and common carriers (TDVRPPC), a generalization of the vehicle routing problem with private fleet and common carriers (VRPPC) problem [1]. In the VRPPC, a set of customers must be served by either third-party logistic (3PL) providers (common carriers) or the dispatchers' owned fleet (private fleet), subject to capacity constraints of the private fleet. The objective is to minimize the total cost, including both fixed and variable costs of the private fleet, and outsourcing cost charged by the 3PLs. Due to the high cost of owning and managing a private fleet and seasonable and/or unpredictable demands, outsourcing has become widely common in the transportation and logistics industries. The VRPPC and its variants have been investigated in the past decade [2]–[6]. However, existing research that considers the impact of real-world traffic conditions on the solution and algorithms for the problem appears to be minimal.

Typical vehicle routing problem (VRP) research implicitly assumes that travel times on the road are time-invariant [7], which is not close to realistic situations. Hence, the time-dependent VRP (TDVRP) has attracted much attentions in the academic research community in recent years. [8] proposed a mixed integer programming (MIP) model and several heuristics for the TDVRP for the first time with a stepwise travel time model, which could potentially produce solutions violating the first-in-first-out (FIFO) property. The FIFO property is enforced when [9] modelled the travel speed as a stepwise function with a piecewise linear time-dependent travel time function. Most subsequent research investigates TDVRP and

The associate editor coordinating the review of this manuscript and approving it for publication was Ugur Guvenc.

its variants with the proposed travel time model, such as under a congestion charge scheme and a speed-dependent fuel cost [10], determination of best paths between any pair of nodes [11], reduction of fuel consumption and carbon emission [12], [13], and deployment of electric vehicles [14]. Various heuristic algorithms are developed to solve related TDVRP, such as ant colony system [12], [15], [16], adaptive large neighborhood search [14], and hybrid algorithms [17], [18]. Reference [17] studied the multi-trip time-dependent vehicle routing problem with time windows (MT-TDVRPTW) and extended the efficient segment-based evaluation method in [19] with time-dependent travel times, which makes use of forward and backward propagation to generate the necessary breakpoints as speedup actions. Due to a lack of real-world time-dependent data in the research community, most of the literature relies on synthetic travel time functions where the computational campaign relies on time-dependent graphs [20]–[23]. Describing the time-dependent graphs considering less than ten speeds per time zone is generally used, because of the limitation of computational memory [24]–[26].

Though the VRPPC problem considers the capacity constraint of vehicles to minimize total cost, it ignores the working hours of the drivers. This is impractical when we consider real-world legal regulations on working hours and the varying travel times required on the road throughout the day. Specifically, [24] analyzed the working hours include driving time and service time under time-dependent travel times in the urban delivery model. The results show that working hours heavily affect road safety and routes efficiency. The TDVRPPC problem is hence an important research topic with great impact when deployed in the real world to minimize operational costs while adhering to both capacity and maximum route duration constraints.

As a generalization of both the TDVRP and the VRPPC, the TDVRPPC is NP-hard and requires planned problem modeling and specific algorithmic design. Various heuristic and meta-heuristic algorithms have been contemplated to study the VRPPC, the TDVRP and other VRP variants, for example tabu search (TS) [2], [27]–[32], variable neighborhood search (VNS) [3], [33], [34], ant colony optimization [16], and adaptive large neighbourhood search (ALNS) [17], [35]–[41]. Among them, the ALNS, which utilizes a set of destroying heuristics to remove part of the incumbent solution and recreates a new solution with another set of repair heuristics in an adaptive way based on the historical performances, has been shown to be effective in solving various problems. On the other hand, TS is a deterministic search algorithm that can flee from local optima efficiently by taking the best non-tabu move. Recently, [18] hybridized ALNS and TS (ALNS-TS) to solve the duration-minimizing time-dependent vehicle routing problem with time window (DM-TDVRPTW), in which the TS searches both feasible and infeasible solution spaces efficiently to identify local optima in a small solution region, while the ALNS performs intentional perturbation to the incumbent solution for adaptive exploration. We adopt the same algorithm framework in this study and tailor various algorithmic components based on the unique features of the TDVRPPC: (1) design removal operators and regret insert operation to handle outsourcing options; (2) relax both the maximum route duration constraint and the capacity constraints in the TS procedure to accept infeasible solutions with a penalty function; (3) propose neighbourhood structures with both private fleets and the common carrier; (4) extend the efficient segment-based evaluation method [17], [18] to the relaxed TDVRPPC (r-TDVRPPC) problem to speed up the feasibility checking. Specifically, capacity and maximum route duration constraints can be violated in the r-TDVRPPC and a forward and backward propagation algorithm is used to process the time-dependent ready time function and duration function.

Our main contributions are summarized as below: 1) we offer a formal description and a mixed integer programming (MIP) model for the TDVRPPC; 2) we tailor an effective hybrid algorithm that adjusts its behavior adaptively and dynamically to solve the problem; 3) we devise an efficient evaluation method for a vehicle route under the r-TDVRPPC, and 4) we derive benchmark test instances for the TDVRPPC and present computational studies that illustrate the performance of the hybrid algorithm.

The remainder of this paper is organized as follows. In Section II we introduce the problem with an arc-flow model. We then describe the ALNS procedure and the TS procedure in Section III and IV respectively. The Section V details the acceleration technique for route evaluation with the segment-based method for the r-TDVRPPC. Lastly, computational results and analysis are given in Section VI before the conclusion in Section VII. Readers may refer to Table 1 for the list of abbreviations used in this paper.

TABLE 1. Abbreviation table.

Abbreviation	Explanation
3PL	Third-party Logistic
ALNS	Adaptive Large Neighboorhood Search
BPS	Break Points Set
DM-TDVRPTW	Duration-Minimizing Time-Dependent Vehicle Routing Problem with Time Window
DUR	Duration Set
FIFO	First-In-First-Out
MD	Minimum Duration
MIP	Mixed Integer Programming
MT-TDVRPTW	Multi-Trip Time-Dependent Vehicle Routing Problem with Time Window
RI	Regret Insertion
RT_BPS	Ready Time Break Points Set
r-TDVRPPC	relaxed TDVRPPC
TDVRP	Time-dependent Vehicle Routing Problem
TDVRPPC	Time-dependent Vehicle Routing Problem with Private fleet and Commom carriers
TDVRPTW	Time-dependent Vehicle Routing Problem with Time Window
TS	Tabu Search
VNS	Variable Neighborhood Search
VRPPC	Vehicle Routing Problem with Private fleet and Commom carriers

II. PROBLEM DESCRIPTION

The TDVRPPC problem is defined over a directed graph $G = (V, A)$ with node set V and arc set A. $V = \{0\} \cup V_c$, where 0 represents the depot, and $V_c = \{1, 2, \ldots, n\}$ is the customer set. A is defined as $\{(i, j) : i, j \in V, i \neq j\}$, where each arc (i, j) is associated with a distance d_{ij}.

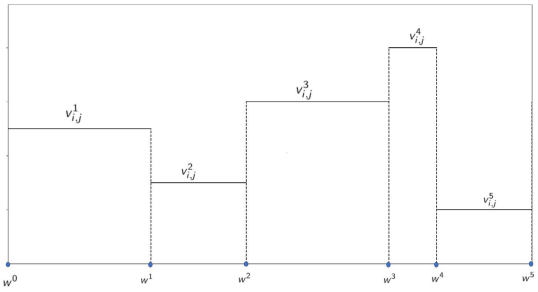

FIGURE 1. Example of a time-dependent travel speed function.

Each customer i has a demand q_i and must be served by exactly one vehicle from the private fleet or by the common carrier. The homogeneous private fleet is denoted by K, each of which has a maximum capacity Q. The working hour is from E to L and a private vehicle may be activated to start from the depot, serve some customers, and return to the depot within the working hour. Without loss of generality, E is set to 0 in this study. The route's duration, defined as the arrival time at the returning depot minus the departure time at the starting depot, should not exceed the driver's maximum working hour (T_{max}) as regulated by the government and the company's policy. This is also referred to as the maximum route duration constraint.

Each private vehicle has a fixed cost of f if it is activated. Distance-based travel cost d_{ij} will be incurred when a vehicle from the private fleet travels from node i to node j, while a vehicle from the common carrier incurs a charge of p_i if it serves node i. The TDVRPPC problem determines the set of customers to outsource and plans vehicle routes for the private fleet to serve the remaining customers. The problem then aims to minimize the sum of both the fixed and variable costs incurred by the private fleet as well as outsourced costs charged by the common carriers.

A. TIME DEPENDENT TRAVEL TIME FUNCTION

The TDVRPPC adopts the travel time model in [9], which defines the piecewise linear travel time function formally as below. A typical workday ($[E, L]$) is divided into non-overlapping time zones $T = \{1, 2, \ldots, |T|\}$, where the m-th time zone is $[w^{m-1}, w^m]$. The time values of $w^m, \forall m \in \{0\} \cup T$ are called break points. It is assumed that the travel speed is constant within a time zone and will change only at the end of each time zone. The arcs in A are assigned to a speed profile, which share the same travel speeds for each time zone. Let $v_{i,j}^m$ be the travel speed along (i, j) during the m-th time zone. The actual time-dependent travel time along (i, j) with departure time t_0 (denoted as $\bar{\tau}_{i,j}(t_0)$) depends on the speed profile of (i, j) and d_{ij}, which can be calculated using Algorithm 1. When given a departure time t_0, Algorithm 1 iteratively calculates the extra time required (line 2 and 6) to complete the remaining journey from i to j (line 4) under the travel speed of the current time zone. If the required travel time crosses into the next time zone (line 3), the calculation continues with the travel speed of the next time zone (line 7). Otherwise, the actual travel time required is determined (line 9).

Algorithm 1 Calculation of Actual Travel Time $\bar{\tau}_{i,j}(t_0)$

1: Determine the time zone of t_0 as M
2: $t \leftarrow t_0, d \leftarrow d_{ij}, t' \leftarrow t + (d/v_{ij}^M)$,
3: **while** $t' \geq w^M$ **do**
4: $\quad d \leftarrow d - v_{ij}^M \times (w^M - t)$
5: $\quad t \leftarrow w^M$
6: $\quad t' \leftarrow t + (d/v_{ij}^{M+1})$
7: $\quad M \leftarrow M + 1$
8: **end while**
9: **return** $t' - t_0$

We can now use the calculated actual travel times to construct the piecewise linear time-dependent travel time for an arc (i, j) with any departure time t as $\tau_{i,j}(t)$. As an example in Figure 2, this function can be fully determined using the break points ($w_{i,j}^0$ to $w_{i,j}^9$) of the ***arc time zones*** $T_{i,j}$ and their associated actual travel times (e.g, $\bar{\tau}_{i,j}(w_{i,j}^1)$, $\bar{\tau}_{i,j}(w_{i,j}^2)$, and etc, which are calculated with Algorithm 1). Note that the function contains two groups of breakpoints: the ***speed breakpoints*** $\{w_{i,j}^0, w_{i,j}^2, w_{i,j}^4, w_{i,j}^6, w_{i,j}^8, w_{i,j}^9\}$ corresponding to $\{w^0, w^1, w^2, w^3, w^4, w^5, \}$ in Figure 1 respectively, and the ***travel time breakpoints*** $\{w_{i,j}^1, w_{i,j}^3, w_{i,j}^5, w_{i,j}^7\}$ representing the time to depart from i so that the arrival times at j is exactly one of the speed breakpoints. The resulted time zones for (i, j) is called the arc time zones $T_{i,j}$, which is unique to each arc because of the arc's speed profile and d_{ij}. We denote the m-th arc time zone for (i, j) as $T_{i,j}^m$. For ease of calculation, the slope ($\theta_{i,j}^m$) and the intersection with y-axis ($\eta_{i,j}^m$) of the m-th line segment (representing the m-th arc time zone) can be pre-calculated and used to fully represent the piecewise linear travel time function $\tau_{i,j}(t)$ as below:

$$\tau_{i,j}(t) = \sum_{m \in T_{i,j}} \left(\theta_{i,j}^m t + \eta_{i,j}^m\right) \mathbf{1}_{T_{i,j}^m}(t), \ \forall t \in [E, L], \qquad (1)$$

where the indicator function $\mathbf{1}_{T_{i,j}^m}(t)$ indicates whether t belongs to $T_{i,j}^m$.

Similarly, the backward travel time function $\tau_{i,j}^{-1}(t)$, defined as the travel time required for the vehicle to definitely arrive at node j at time t along arc (i, j), is also a piecewise linear function and can be determined in a similar algorithm as Algorithm 1.

In the rest of this paper, we will use the travel time function $\tau_{i,j}(t)$, $\theta_{i,j}^m$ and $\eta_{i,j}^m$ directly for calculation of the travel time given (i, j) and t instead of invoking Algorithm 1.

B. MATHEMATICAL MODEL

This section presents an MIP model for the TDVRPPC, which incorporates $\theta_{i,j}^m$ and $\eta_{i,j}^m$ directly in the model instead of using $\tau_{i,j}(t)$. To construct the model, we first add node $n + 1$ to

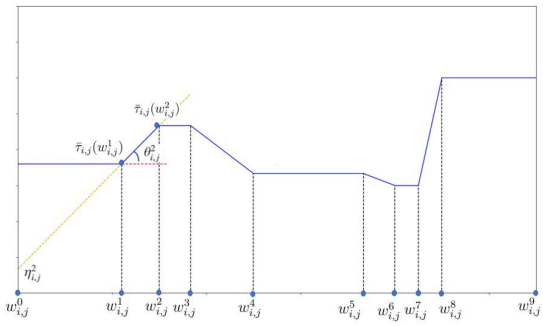

FIGURE 2. Example of a time-dependent travel time function.

graph G as the returning depot, and thereafter create a new graph $G' = (V', A')$, where $V' = V \cup \{n+1\}$ and $A' = A \cup \{(i, n+1) : i \in V_c\}$. The MIP is based on the decision variables as listed below:

- $x_{i,j}^k((i,j) \in A', k \in K)$: binary variable which is 1 if vehicle k travels from node i to node j, and 0 otherwise;
- $y_i^k(i \in V', k \in K)$: binary variable which is 1 if node i is served by the private vehicle k, and 0 otherwise;
- $s_i(i \in V_c)$: binary variable which is 1 if the demand of node i is served by the common carrier;
- $z_{i,j}^{m,k}((i,j) \in A', m \in T_{i,j}, k \in K)$: binary variable which is 1 if vehicle k travels from node i to node j during time zone $T_{i,j}^m$, and 0 otherwise;
- $t_{i,j}^{m,k}((i,j) \in A', m \in T_{i,j}, k \in K)$: the departure time if vehicle k travel from node i to node j during arc time zone $T_{i,j}^m$, and 0 otherwise;
- $a_i^k\ (i \in V', k \in K)$: arrival time of vehicle k at node i.

This TDVRPPC problem is formulated as below:

$$min\ z = \sum_{k \in K} f y_0^k + \sum_{(i,j) \in A'} \sum_{k \in K} d_{ij} x_{i,j}^k + \sum_{i \in V_c} p_i s_i \quad (2)$$

$$x_{i,j}^k = \sum_{m \in T_{i,j}} z_{i,j}^{m,k},\ \forall (i,j) \in A',\ k \in K \quad (3)$$

$$y_i^k = \sum_{(i,j) \in A'} x_{i,j}^k = \sum_{(j,i) \in A'} x_{j,i}^k,\ \forall i \in V_c,\ k \in K \quad (4)$$

$$y_0^k = \sum_{i \in V_c} x_{0,i}^k,\ \forall k \in K \quad (5)$$

$$y_{n+1}^k = \sum_{i \in V_c} x_{i,n+1}^k,\ \forall k \in K \quad (6)$$

$$w_{i,j}^m z_{i,j}^{m,k} \le t_{i,j}^{m,k} \le w_{i,j}^{m+1} z_{i,j}^{m,k},\ \times \forall (i,j) \in A', m \in T_{i,j},\ k \in K \quad (7)$$

$$a_i^k = \sum_{\forall (j,i) \in A'} \sum_{m \in T_{j,i}} \{(\theta_{j,i}^m + 1) t_{j,i}^{m,k} + \eta_{j,i}^m z_{j,i}^{m,k}\},\ \times \quad \forall i \in \{n+1\} \cup V_c,\ k \in K \quad (8)$$

$$a_i^k = \sum_{\forall (i,j) \in A'} \sum_{m \in T_{i,j}} t_{i,j}^{m,k}, \quad \forall i \in \{0\} \cup V_c,\ k \in K \quad (9)$$

$$a_{n+1}^k - a_0^k \le T_{max},\ \forall k \in K \quad (10)$$

$$\sum_{i \in V_c} q_i y_i^k \le Q,\ \forall k \in K \quad (11)$$

$$\sum_{k \in K} y_0^k = \sum_{k \in K} y_{n+1}^k \le |K| \quad (12)$$

$$\sum_{k \in K} y_i^k + s_i = 1,\ \forall i \in V_c \quad (13)$$

$$z_{i,j}^{m,k} \in \{0,1\},\ \forall (i,j) \in A',\ m \in T_{i,j},\ k \in K \quad (14)$$

$$x_{i,j}^k \in \{0,1\},\ \forall (i,j) \in A',\ k \in K \quad (15)$$

$$y_i^k \in \{0,1\},\ \forall i \in V',\ k \in K \quad (16)$$

$$s_i \in \{0,1\},\ \forall i \in V_c \quad (17)$$

$$E \le a_i^k \le L,\ \forall i \in V',\ k \in K \quad (18)$$

$$E \le t_{i,j}^{m,k} \le L,\ \forall (i,j) \in A', m \in T_{i,j},\ k \in K \quad (19)$$

Constraint (2) is the objective function to minimize the total cost, which includes fixed costs and routing costs of the private vehicles, and the costs of outsourcing. Constraints (3) define the relationship between $z_{i,j}^{m,k}$ and $x_{i,j}^k$ to ensure that $x_{i,j}^k$ is set correctly $\forall (i,j) \in A',\ k \in K$. Constraints (4) define y_i^k in terms of $x_{i,j}^k$ for the customers and is also the flow conservation constraints for the customer nodes. Furthermore, constraints (4) ensure that when a vehicle visits i, the customer must be served by the vehicle. Constraints (5) and (6) define y_i^k in terms of $x_{i,j}^k$ for the departure depot and the returning depot for all the private vehicles. Both will be set to 0 if the private vehicle is not used. Constraints (7) set the right range for $t_{i,j}^{m,k}$. It forces $t_{i,j}^{m,k}$ to be in the right arc time zone if the k-th vehicle travels from i to j during the m-th time zone; otherwise, $t_{i,j}^{m,k}$ is set to zero. Constraints (8) define the arrival time at a node i by consideration of all possible incoming arcs and the time-dependent travel time. The constraints use pre-computed parameters θ and η directly instead of the travel time function τ, which can be directly entered into any commercial solver, such as CPLEX Solver. Constraints (9) link arrival time at and the departure time from node i together to ensure that a vehicle will depart immediately from the node after arriving at and serving customer i. As no service time is used in the original VRPPC problem, the arrival time at and departure time from a visited customer are the same for the TDVRPPC. Constraints (9) and Constraints (8) together serve as subtour elimination constraints too. Constraints (10) calculate the length of the vehicle routes and ensure that the length does not exceed the maximum route duration allowed (T_{max}). Constraints (11) ensure capacity constraint for each vehicle used. Constraints (12) guarantee that at most $|K|$ private vehicles can be activated and all activated vehicles must start from and return to the depot. Constraints (13) ensure that each customer is served exactly once either by the private fleet or by the common carrier. Constraints (14), (15), (16), (17), (18) and (19) define the variables of the model.

III. ADAPTIVE EXPLORATION

The adaptive large neighbourhood search with embedded tabu search algorithm (ALNS-TS) is outlined in Algorithm 2

below. Note that the algorithm controls the strength of diversification and intensification adaptively and dynamically by setting the number of customer removals (parameter α) in the ALNS (Section III-A) and the maximum number of steps without improvement allowed (parameter ϵ_{TS}) in the TS (Section IV-B). When the ALNS-TS fails to achieve a better solution in an iteration, the values of α and ϵ_{TS} are increased gradually to increase the strength of ALNS and TS respectively for the subsequent iterations. A counter C_{cni}^{ALNS} is maintained for this purpose.

Algorithm 2 The ALNS-TS Framework

```
Construct an initial solution S with Regret Insert
C_cni^ALNS ← 0, S_best ← S
while not exceeding time limit do
    α ← min{α_max, α_min + α_rate × C_cni^ALNS} × n
    Select removal operator to remove α customers from the S
    Select repair operator to repair S
    ε_TS ← min{C_cni^ALNS + φ_cni, 2φ_cni}
    S ← TS(S, ε_TS)
    if f(S) < f(S_best) then
        C_cni^ALNS ← 0, S_best ← S
    else
        C_cni^ALNS ← C_cni^ALNS + 1
    end if
    Update scores and probability for removal and repair operators
end while
return S_best
```

The ALNS adopts an adaptive selection of the removal and repair operators, whose main objective is to explore unexplored yet promising solution region through large neighborhood changes. Advanced removal methods based on the Shaw removal principle are designed to select customers either served by the private fleet or the common carrier for removal and repair (Section III-C).

A. ADAPTIVE EXPLORATION STRENGTH

The number of customers to be removed in an ALNS iteration (α) controls the diversification strength. Let the counter C_{cni}^{ALNS} denote the number of consecutive non-improving ALNS iterations. This counter is then used to update α adaptively as below:

$$\alpha = min\{\alpha_{max}, \alpha_{min} + \alpha_{rate} \times C_{cni}^{ALNS}\} \times n, \tag{20}$$

where α_{min} and α_{max} are the minimum and maximum perturbation rates, and α_{rate} is the unit increase rate. Initially, C_{cni}^{ALNS} is set to 0 and α is simply set to $\alpha_{min} \times n$.

B. ADAPTIVE OPERATOR SELECTION

The removal and repair operators are selected adaptively based on their historical performance with the roulette wheel selection principle [35]. Let O be the set of operators and w_o be the weight of an operator o. Then o is selected with probability $\frac{w_o}{\sum_{o' \in O} w_{o'}}$. We divide the ALNS iterations into phases of ϕ_{phase} consecutive iterations, within which the probability remains constant. w_o is re-calculated based on its scores ζ_o at the end of the phases as below:

$$w_o = \bar{w_o}(1-\gamma) + \gamma \frac{\zeta_o}{\theta_o}, \tag{21}$$

where $\bar{w_o}$ is the weight of o in the current phase, γ is the learning rate to control the speed of reaction, and θ_o is the number of times that o is selected during the current phase. ζ_o reflects the historical performance of the operator and is updated for each iteration when o is selected as below: 1) ζ_o is increased by λ_1 if an overall best solution is found; 2) otherwise, ζ_o is increased by λ_2 if the solution is better than the current solution. In the initial phase, all the operators are assigned the same weights and probabilities.

C. REMOVAL OPERATORS

One of the five removal operators is selected to remove α customers for each iteration. Note that at the end of each removal operator, the set of customers served by the common carriers will also be removed so that they will be considered for re-insertion in the repair stage.

1) WORST REMOVAL

A tailored worst removal aims to obtain better solutions by removing customers assigned wrongly in the vehicle routes. It calculates the insertion cost of a customer i as $\Delta C_i = f(S) - f(\bar{S}_i)$, where $\bar{S}_i$ is the partial solution generated when removing i from S. It then removes customer i with probability based on ΔC_i. Specifically, the probability of selecting a customer i at each round is equal to $\frac{\Delta C_i}{\sum_{j \in V(S)} \Delta C_j}$, where $V(S)$ represent the list of customers served by private fleet in the current solution S. The introduced randomness tends to remove customers with high insertion costs but avoids repeated selection of the same customers. Note that outsourced customers will not be removed by this operator.

2) ADVANCED SHAW REMOVAL

A special Shaw removal is designed to increase the possibilities of relocating customers to new positions in the solution by selecting customers based on the distance-based closeness measure and deleting neighboring customers served by the same vehicle route to create a bigger gap. The distance-based closeness measure is set to d_{ij} in this study. Algorithm 3 illustrates the pseudo-code of the operator. Initially, two empty lists are prepared for removed customers: L_1 stores customers selected based on the closeness measure, and L_2 stores customers based on the sequence of visits that are either before or after the removed customers in the vehicle route. The operator removes both L_1 and L_2 customers, but only uses L_1 customers to search for the next customer for removal based on the closeness measure. Note that no customer will be inserted into L_2 if the current customer being removed

is served by the common carrier. One initial customer is randomly selected and inserted into L_1 to initiate the removal process.

Algorithm 3 Advanced Shaw Removal

```
L1 ← {},L2 ← {}
while |L1| + |L2| < α do
    if L1 is empty or all L1 customers are processed then
        Select i randomly for L1.
        Randomly pick either the customer before or after i
        in the vehicle route for L2.
    end if
    Select an unprocessed customer i from L1
    Pick one or two close customers of i for L1
    Randomly pick either the customer before or after the
    selected customers in the vehicle route for L2
    Mark customer i as processed in L1
end while
Remove all customers from the common carriers
```

3) ROUTE-BASED SHAW REMOVAL

As opposed to the Advanced Shaw Removal operator, this operator differs in selecting initial customers removal as it considers the service efficiency of the vehicle route instead. Formally, we define route efficiency as $\phi(r) = f(r)/\sum_{i\in V(r)} q_i$ for a route r, where $f(r)$ is the total cost of the route r and $V(r)$ contains all the customers served in the route r. A route r is selected with probability $\phi(r)/\sum_{r'\in S}\phi(r')$ to initialize the Shaw removal process as in Algorithm 3.

4) MIXED SHAW REMOVAL

To further increase the diversity of the search procedure, we incorporate greater randomness by combining the methods for initial customer selection for the removal of the above two operators. Formally, a route is selected based on its effectiveness and another customer not served by the route is randomly selected for L_1. This operator then applies the Shaw removal procedure to select the rest of the customers.

5) RANDOM REMOVAL

This operator randomly selects α customers for removal from the current solution.

D. REGRET INSERTION AS REPAIR OPERATORS AND CONSTRUCTION

We treat the common carrier as a possible route for the insertion of customers to extend the Regret Insertion (RI) method for the TDVRPPC, which has been shown to be effective to repair partial solutions for various VRP problems [42], [43]. RI inserts unserved customers with the greatest regret value to the solution one by one. Let the minimum insertion cost of customer i to the r-th vehicle route be $\Delta C_{i,r}$, where $r \in K$ and $i \in V_c$. $\Delta C_{i,r}$ is set to $+\infty$ if i cannot be inserted into the route. Let $r_{i,k} \in K$ be a variable that indicates the route for which customer i has the k-th lowest insertion cost. RI inserts i with the highest $RV = \Delta C_{i,r_{i,2}} - \Delta C_{i,r_{i,1}}$ into the $r_{i,1}$-th route at the position with the lowest insertion cost. RI always maintains an empty route as a candidate route for the customers whenever possible.The advanced regret-k method adopts a look-ahead strategy and chooses i to maximize:

$$max_{i\in U}\sum_{j=1}^{k}(\Delta C_{i,r_{i,j}} - \Delta C_{i,r_{i,1}}), \tag{22}$$

where $U \subseteq V_c$ is the set of unserved customers. We create three repair operators by setting k to 1, 2 and 3 respectively. When $k = 1$, regret-k is equivalent to the greedy insertion method. We also employ the regret-k method to construct initial solutions in this paper.

IV. ADAPTIVE EXPLOITATION

While the ALNS steers the search to unexplored territory in the feasible solution space, the TS organizes an intensive and comprehensive local search to improve the solution generated by the ALNS. Hence, we permit the TS to explore infeasible solution spaces for r-TDVRPPC that may violate maximum duration and/or capacity constraints. Similar to diversification, the strength of intensification is dynamically determined based on the historical performance of the algorithm. The TS is restricted to return only feasible solutions to the ALNS loop since infeasible solutions are only used as a bridge between two otherwise unconnected feasible regions during local search and are not useful for large neighborhood changes in ALNS. The TS procedure for the r-TDVRPPC is shown in Algorithm 4.

Algorithm 4 TS for r-TDVRPPC

```
C_cni^TS ← 0, S_best ← S
while C_cni^TS < ε_TS do
    Select the best non-tabu move of S and obtain S'
    if S' is a feasible TDVRPPC solution then
        if f(S') < f(S_best) then
            S_best ← S', C_cni^TS ← 0
        else
            C_cni^TS ← C_cni^TS + 1
        end if
        β ← min{β_max, β/β_rate}
    else
        C_cni^TS ← C_cni^TS + 1, β ← max{β_min, β × β_rate}
    end if
    Update tabu list
    Reset β after every φ_p iterations
    S ← S'
end while
return S_best
```

A. ADAPTIVE PENALTY COST

A given solution S for the r-TDVRPPC is evaluated with the modified cost function below:

$$f'(S) = f(S) + \beta_{dur} P_{dur}(S) + \beta_{cap} P_{cap}(S) \tag{23}$$

where $f(S)$ is the original cost function of S for the TDVRPPC, $P_{dur}(S)$ and $P_{cap}(S)$ denote the amount of violation of the maximum route duration and the capacity constraints by S respectively, while β_{dur} and β_{cap} indicate the adaptive penalty factors. Both β_{dur} and β_{cap} are handled in a similar way and we omit the subscripts for clarity below. β is restricted to $[\beta_{min}, \beta_{max}]$ to avoid trapping the search procedure in the feasible or infeasible regions for too long. β is set to β_{min} initially and is adaptively updated based on the feasibility of the generated solution. If S is a feasible TDVRPPC solution, β is reduced to β/β_{rate} to encourage exploitation of the infeasible solution space. Otherwise, we increase β to $\beta \times \beta_{rate}$ to direct the search back to a feasible solution space.

B. ADAPTIVE EXPLOITATION STRENGTH

The strength of the TS depends on the termination criterion. Hence we confine the maximum number of consecutive non-improved steps allowed for the TS algorithm (ϵ_{TS}) as below:

$$\epsilon_{TS} = \min\{C_{cni}^{ALNS} + \phi_{cni}, 2\phi_{cni}\} \tag{24}$$

where ϕ_{cni} is a numerical parameter and C_{cni}^{ALNS} was defined in previous section. After each ALNS iteration, C_{cni}^{ALNS} is updated accordingly and the strength of the TS is determined dynamically based on the historical performance.

C. NEIGHBORHOOD STRUCTURE

Two groups of neighborhood structures are considered for the TS with the traditional relocation and swap operators. The first group is the intra-route operators, which relocate a customer or swap two customers within the same private vehicle. The second group is the inter-route operators, which relocate a customer or swap two customers between two privates vehicles, or between a private vehicle and the common carrier.

D. TABU DURATION AND ASPIRATION

A tabu duration of μ iterations is associated with each arc removed by the TS operators. The tabu status of an arc can be revoked if a new best feasible solution can be obtained by violating the tabu.

V. SPEEDUP TECHNIQUE FOR ROUTE EVALUATION

The computation under time-dependent routing operations can be time-consuming since the TDVRPPC has to consider travel time between each customer node and in each discretized time zone (departure time) [25], [26]. Route evaluation is invoked regularly to assess the feasibility and cost of a route during local search and the repair stage of the ALNS. [19] developed an efficient segment based route evaluation method for traditional VRP problems, where the information about segments of node visits is pre-processed for faster concatenation operation and the segment of customer visits might violate certain constraints of the problem. Reference [18] extended the method in [19] with a penalty cost for violation of time windows for the time-dependent duration function, which is customized for the r-TDVRPPC with a modified cost function $f'(S)$ for the violation of both capacity constraints and maximum trip duration constraints below.

A. TIME DEPENDENT DURATION FUNCTION FOR R-TDVRPPC

Let $\sigma = (\sigma_1, \sigma_2, \ldots \sigma_L)$ be a segment of node visits for a vehicle from the private fleet with $\sigma_i \in V$, $\forall 1 \le i \le L$. The ready time function $\delta_{\sigma_i}^{\sigma}(t)$ is defined as the time when vehicle arrives at and is ready to depart from node σ_i if it starts at σ_1 at time t. Mathematically, it can be calculated recursively as

$$\delta_{\sigma_i}^{\sigma}(t) = \begin{cases} t, & \text{if } i = 1; \\ \delta_{\sigma_{i-1}}^{\sigma}(t) + \tau_{\sigma_{i-1},\sigma_i}\left(\delta_{\sigma_{i-1}}^{\sigma}(t)\right), & \text{otherwise.} \end{cases} \tag{25}$$

The duration function $\psi^{\sigma}(t)$ is defined as $\delta_{\sigma_L}^{\sigma}(t) - t$. The inverse function of the ready time function is denoted as $\pi_{\sigma_i}^{\sigma}(t)$, which is formally defined as the latest time when the vehicle should start to serve node σ_1 so that node σ_i will be ready by time t.

The following proposition is valid for the time-dependent ready time and duration functions under TDVRPPC.

Proposition 1: Under the piecewise linear travel time model for the TDVRPPC, 1) $\delta_{\sigma_i}^{\sigma}(t)$, $\pi_{\sigma_i}^{\sigma}(t)$ and $\psi^{\sigma}(t)$ are piecewise linear functions for any σ and $1 \le i \le L$. 2) $\delta_{\sigma_{L_1}}^{\sigma}(t)$, $\pi_{\sigma_1}^{\sigma}(t)$ and $\psi^{\sigma}(t)$ have the same set of break points; and 3) the minimum duration of σ must be associated with one of the break points for $\psi^{\sigma}(t)$.

Proof: The proof is based on the following: let $f(t)$ and $g(t)$ be two piecewise linear functions on t, then $f(t) + g(t)$, $f(g(t))$, and $f(t) + C$ are all piecewise linear functions, where C is a constant number. First, it is easy to see that $\delta_{\sigma_1}^{\sigma}(t) = t$ is a (piecewise) linear function. So $\tau_{\sigma_1,\sigma_2}(\delta_{\sigma_1}^{\sigma}(t))$ is a piecewise linear function on t as the travel time function is also a piecewise linear function. Therefore, $\delta_{\sigma_2}^{\sigma}(t)$ is a piecewise linear function as the sum of two piecewise linear functions. Similarly, it can be shown by induction that $\delta_{\sigma_i}^{\sigma}(t)$ is a piecewise linear function $\forall 1 \le i \le L$. Consequently, $\psi^{\sigma}(t)$ is also piecewise linear as the sum of $\delta_{\sigma_L}^{\sigma}(t)$ and $-t$. The proof for $\pi_{\sigma_i}^{\sigma}(t)$ is similar to the proof for $\delta_{\sigma_i}^{\sigma}(t)$ with the backward travel time function $\tau_{i,j}^{-1}(t)$.

(2) By definition of $\psi^{\sigma}(t)$, it is easy to see that $\psi^{\sigma}(t)$ and $\delta_{\sigma_{L_1}}^{\sigma}(t)$ share the same set of break points. On the other hand, the departure time break points from σ_1 and the arrival time break points at σ_L are the same for both $\delta_{\sigma_{L_1}}^{\sigma}(t)$ and $\pi_{\sigma_1}^{\sigma}(t)$ by definition of the $\pi_{\sigma_1}^{\sigma}(t)$.

(3) It can be shown by contradiction. Suppose the minimum duration of σ is associated with a departure time t, which is not one of the break points for $\psi^{\sigma}(t)$. Let t falls

under the line segment (the arc time zone) denoted by $[\underline{t}, \bar{t}]$. Then either $\psi^{\sigma}(\underline{t})$ or $\psi^{\sigma}(\bar{t})$ must have a lower duration than or both have the same duration as t. ■

Similar to the time-dependent travel time function, these functions can be fully represented by the values of their breakpoints, where the slopes and intersections of the line segments can be computed accordingly. Therefore, the method below will only focus on the breakpoints for storage and processing of $\delta^{\sigma}_{\sigma_{L_1}}(t)$, $\pi^{\sigma}_{\sigma_1}(t)$ and $\psi^{\sigma}(t)$ for any segment σ.

B. SEGMENT INITIALIZATION AND FEASIBILITY

A segment for the r-TDVRPPC stores its cost $C(\sigma)$, total load $Q(\sigma)$, the minimum duration $MD(\sigma)$, the break points set $BPS(\sigma)$ for the feasible start time window at σ_1, the associated ready time break points set $RT\_BPS(\sigma)$ and the associated duration set $DUR(\sigma)$. The segment for a single customer i is initialized with $C(\sigma) = 0, Q(\sigma) = q_i, MD(\sigma) = C(\sigma) = 0$, $BPS(\sigma)$ and $RT\_BPS(\sigma)$ contains all break points in T, and $DUR(\sigma)$ contains $|T|$ 0s. The segment for the departure depot is initialized in the same way except for $C(\sigma)$, which is set to the fixed cost of a vehicle f.

σ is feasible for the TDVRPPC if and only if $Q(\sigma) <= Q$ and $MD(\sigma) \leq T_{max}$. On the other hand, σ is feasible for the r-TDVRPPC as long as $MD(\sigma) \leq L - E$.

C. SEGMENT CONCATENATION

When two segments $\sigma^1 = (\sigma^1_1, \sigma^1_2, \ldots \sigma^1_{L_1})$ and $\sigma^2 = (\sigma^2_1, \sigma^2_2, \ldots \sigma^2_{L_2})$ are concatenated together to form $\sigma^1 \oplus \sigma^2$, $Q(\sigma^1 \oplus \sigma^2)$ is updated as $Q(\sigma^1) + Q(\sigma^2)$. Thereafter, we can use forward extension from $\sigma^1_{L_1}$ to σ^2_1 and backward extension to calculate the relevant values for $BPS(\sigma^1 \oplus \sigma^2)$, $RT\_BPS(\sigma^1 \oplus \sigma^2)$ and $DUR(\sigma^1 \oplus \sigma^2)$ as in Algorithm 5. Lastly, the cost is calculated as

$$\begin{aligned} C(\sigma^1 \oplus \sigma^2) = {} & C(\sigma^1) + C(\sigma^2) + d_{\sigma^1_{L_1}, \sigma^2_1} \\ & + \beta_{cap} \times \max\{0, Q(\sigma) - Q\} \\ & + \beta_{dur} \times \max\{0, MD(\sigma) - T_{max}\} \end{aligned} \quad (26)$$

Algorithm 5 Concatenation of σ^1 and σ^2 for r-TDVRPPC

for $t^1 \in RT\_BPS(\sigma^1)$ **do**
  Find the associated start time $\bar{t^1} \in BPS(\sigma^1)$ for t^1 at σ^1_1
  $t_{arr} \leftarrow t^1 + \tau_{\sigma^1_{L_1}, \sigma^2_1}(t^1)$
  $t_{ready} \leftarrow \delta^{\sigma^2}_{\sigma^2_{L_2}}(t_{arr})$
  $t_{dur} \leftarrow t_{ready} - \bar{t^1}$
  Insert $\bar{t^1}$, t_{ready} and t_{dur} into $BPS(\sigma^1 \oplus \sigma^2)$, $RT\_BPS(\sigma^1 \oplus \sigma^2)$, and $DUR(\sigma^1 \oplus \sigma^2)$ respectively
end for
for $t^2 \in RT\_BPS(\sigma^2)$ **do**
  Find the associated start time $\bar{t^2} \in BPS(\sigma^2)$ for t^2 at σ^2_1
  $t_{dep} \leftarrow \bar{t^2} - \tau^{-1}_{\sigma^1_{L_1}, \sigma^2_1}(\bar{t^2})$
  $t_{start} \leftarrow \pi^{\sigma^1}_{\sigma^1_{L_1}}(t_{dep})$
  $t_{dur} \leftarrow t^2 - t_{start}$
  Insert t_{start}, t^2 and t_{dur} into $BPS(\sigma^1 \oplus \sigma^2)$, $RT\_BPS(\sigma^1 \oplus \sigma^2)$, and $DUR(\sigma^1 \oplus \sigma^2)$ respectively
end for
$MD(\sigma^1 \oplus \sigma^2) = min_{d \in DUR(\sigma^1 \oplus \sigma^2)} d$
Determine the parameters of the time-dependent ready time functions for $\sigma^1 \oplus \sigma^2$

VI. COMPUTATIONAL EXPERIMENTS

In the remainder of this section, we first describe the test instances adopted from the existing VRPPC benchmark instances and the parameter tuning process. After that, we provide the computational results on the TDVRPPC test instances as well as the VRPPC test instances. All programs are coded in Java and run in a single-thread mode on a Ubuntu 18.04.3 LTS server with Intel(R) Xeon(R) Silver 4216 CPU at 2.10 GHz. The MILP model is solved with IBM ILOG CPLEX 12.8.0.

A. TEST INSTANCES

We adopt a total of 34 instances from [44] for the VRPPC, which are divided into two subsets: 14 instances originated from [45] (initialized with "C") and 20 instances proposed by [46] (initialized with "GE"). Note that our test instances contain the same information as the test instances in [44], including customer locations, customer demands, vehicle capacity, fixed cost and variable costs. The customer sizes of the instances may vary from 50 to 480 customers and up to 31 vehicles is allowed for some instances.

Although TDVRP and its variants have received increased attention from the scientific community in recent years, real-world time-dependent data is still rare. Only big IT players, such as Google, TomTom, AutoNavi(Gaode), etc., have the availability of high-quality historical time-dependent traffic data. As a result, there are no real travel time dataset freely available to the entire research community. To overcome this aspect, most of the literature on Time-Dependent Vehicle Routing problems relies on synthetic travel time functions. As far as synthetic time-dependent data is concerned, we observe that there are (highly-cited) contributions (see for example [9], [47]) on vehicle routing problems where the computational campaign relies on time-dependent graphs which satisfy the sufficient conditions partially introduced by [48] and further generalized by the path ranking invariance property defined in [22]. Therefore, we derive time-dependent travel time data for the adopted instances using the approach in [9]. First, We run a preliminary experiment to solve the VRPPC instances without time-dependent information to determine T_{max} and L for the instances. E is set to 0 without loss of generality in this study. Next, the workday $[0, L]$ is divided into five time zones with two peak hours to simulate the real world traffic conditions. The second and the fourth time zones represent the morning and afternoon

rush hours. The third time zone represents the off-peak hours during the day. The first and last time zones represent the beginning and end of the workday with less traffic on the road. We create three-speed profiles as in [9] to denote road segments with fast, medium, and slow traffic speeds. Each arc is then randomly assigned to one of the three speed profiles with different traffic speeds. The detailed speed profiles are shown in Table 2.

For the computational experiments, ALNS-TS runs 10 times with different random seeds per run for each test instance. We standardize the run time allocated for each run based on the customer size: 300 seconds per run for instances with less than or equal to 50 customers, 600 seconds for instances with 51 to 100 customers, 900 seconds for instances with l01 to 200 customers, and 1200 seconds for instances with 201 or more customers. On the other hand, the CPLEX Solver is given a maximum of 2 hours run time for small scale test instances with 15 and 25 customers.

TABLE 2. Travel speed profile and travel speed.

Time zone	$[0, 0.2L]$	$[0.2L, 0.3L]$	$[0.3L, 0.7L]$	$[0.7L, 0.8L]$	$[0.8L, L]$
Fast	1.5	1	1.67	1.17	1.33
medium	1.17	0.67	1.33	0.83	1
Slow	1	0.33	0.67	0.5	0.83

B. PARAMETER TUNING

We tune the parameters of the algorithm with the *IRACE* package [49], which adopts the iterated racing approach to find the best parameter settings of an optimizer automatically. IRACE defines a configuration as a set of values assigned to the algorithmic parameters. The inputs to the IRACE contains the optimizing algorithm itself, a set of algorithmic parameters, a set of training instances, and a training budget for the maximum number of algorithm runs. The iterated racing in IRACE consists of three iterative steps: (1) creating new configurations based on the current sampling distribution, (2) invoking the optimization algorithm with the sampled configurations and evaluating the configuration performance through racing, and (3) ranking and selecting configurations and updating the sampling distribution to bias towards the best configurations. At the end of the parameter tuning, IRACE returns a list of elite configurations based on performance and the best elite configuration returned is selected in this study.

A total of 10 test instances are randomly used as training instances. IRACE is given a budget of 2000 iterations and each iteration was given 300 seconds to execute the ALNS-TS algorithm. The list of the numerical parameters and their best configuration values returned by IRACE can be found in Table 3.

C. EVALUATION OVER SMALL SCALE INSTANCES

In the first experiment, we apply both the ALNS-TS and IBM ILOG CPLEX 12.8.0 to solve small-scale TDVRPPC instances to access the correctness of the algorithm in Table 4 and 5. Test instances used in Table 4 and 5 contain the first 15 customers and first 25 customers respectively from the corresponding TDVRPPC instances. In both tables, we present (1) the upper bound (UB, a.k.a the best feasible integer solution), the lower bound (LB), the gap between UB to LB (G_{opt}, calculated as $(UB - LB)/UB$), and the run time in seconds (T_{run}) for the CPLEX solver; and (2) the best solution cost found (C_{best}), the time in seconds to find the best solution (T_{best}), the gap of C_{best} to the UB (G_{UB}, calculated as $(C_{best} - UB)/UB$), the average solution cost over 10 runs (C_{avg}), as well as the gap of C_{avg} to the C_{best} (G_{avg}, calculated as $(C_{avg} - C_{best})/C_{best}$)) for the ALNS-TS algorithm. Note that the optimality gap threshold is set to $1e-4$ for CPLEX solver, which explains the inconsistencies over the UB and LB of some instances that are solved to optimality by CPLEX solver.

TABLE 3. Parameter tuning results for the ALNS-TS.

	Name	Type	Range	Final Value
TS	β_{min}	Real	[0.01,5]	2.13
	β_{max}	Integer	[500,30000]	7420
	β_{rate}	Real	[1.01,10]	3.04
	μ	Integer	[10,60]	21
	ϕ_p	Integer	[50,200]	100
	ϕ_{cni}	Integer	[40,300]	199
ALNS	α_{min}	Real	[0.1,0.49]	0.27
	α_{max}	Real	[0.5,0.8]	0.67
	α_{rate}	Real	[0.001,0.1]	0.01
	ϕ_{phase}	Integer	[50,150]	79
	γ	Real	[0.01,0.99]	0.27
	λ_1	Integer	[20,40]	35
	λ_2	Integer	[1,20]	6
	ω	Integer	[40,100]	62
TD	γ_{WT}	Real	[0.1,1]	0.66
	γ_{TW}	Real	[0.05,2]	0.11

For the instances with 15 customers, CPLEX cannot solve two instances to optimal values due to the complexity of the problem while ALNS-TS finds better feasible solutions for one of them (CE-13). Besides, ALNS-TS can obtain the same best solution for the rest of the 33 instances. The G_{avg} measures the consistencies of the algorithm over 10 runs for the same test instance. ALNS-TS performs consistently well as it can find the best solutions in each of the 10 runs for 32 out of the 34 test instances. For the other 2 instance, the G_{avg} is only 0.03% over 10 runs. Besides, ALNS-TS normally requires a much shorter run time to find the best solution compared to CPLEX solver.

The contrast of the performance for both solvers is even bigger for the instances with 25 customers and 3 vehicles. First, CPLEX solver can only solve 4 (G-13 to G-16) of the 34 instances to optimality within 2 hours. The G_{opt} can be as big as 133.2% for CE-13 while the average G_{opt} over the

34 instances is over 30%. These observations prove that the TDVRPPC instances are much more difficult to solve with the increase in customer size and CPLEX solver might not be a good tool for larger scale test instances. On the other hand, ALNS-TS is able to find the optimal solutions for the 4 test instances (G-13 to G-16) in shorter run time too and returns better solutions for all test instances that are not solved optimally by CPLEX solver. The G_{UB} can be very significant, for example −28.08% for CE-12. The average G_{UB} is −13%, namely, ALNS-TS finds better solutions whose solution cost is on average 13% lower than the UB provided by CPLEX solver. Lastly, ALNS-TS's performance is still consistent with the increase in complexity of the test instances. The average G_{avg} is only about 0.28% and ALNS-TS obtains the same best solution for 12 out of the 34 test instances for each of the 10 runs.

TABLE 4. ALNS-TS vs CPLEX (15 customers and 2 vehicles).

Inst	CPLEX					ALNS-TS				
	Opt?	UB	LB	G_{Opt}	T_{run}	C_{best}	T_{best}	G_{UB}	C_{avg}	G_{avg}
CE-01	Y	411.89	411.85	-	811.4	411.89	21.91	-	411.89	-
CE-02	Y	459.63	459.59	-	738.8	459.63	0.63	-	459.63	-
CE-03	Y	420.74	420.70	-	1721.5	420.74	2.64	-	420.74	-
CE-04	Y	425.08	425.04	-	895.8	425.08	1.00	-	425.08	-
CE-05	Y	444.42	444.38	-	516.6	444.42	0.12	-	444.42	-
CE-06	Y	419.11	419.11	-	813.4	419.11	0.23	-	419.11	-
CE-07	Y	476.33	476.28	-	656.0	476.33	0.18	-	476.33	-
CE-08	Y	426.82	426.77	-	1471.6	426.82	1.24	-	426.82	-
CE-09	Y	431.07	431.04	-	791.3	431.07	0.04	-	431.07	-
CE-10	Y	440.92	440.90	-	678.2	440.92	0.51	-	440.92	-
CE-11	**N**	**616.07**	**607.93**	**1.32%**	**7200.0**	**615.57**	**6.16**	**-0.08%**	**615.57**	**-**
CE-12	Y	328.40	328.36	-	724.1	328.40	0.33	-	328.40	-
CE-13	**N**	**627.57**	**616.66**	**1.74%**	**7200.0**	**627.57**	**10.31**	**-**	**627.57**	**-**
CE-14	Y	328.40	328.36	-	914.5	328.40	0.22	-	328.40	-
G-01	Y	599.08	599.02	-	1677.9	599.08	0.73	-	599.08	-
G-02	Y	613.08	613.02	-	990.5	613.08	2.50	-	613.08	-
G-03	Y	623.08	623.02	-	2715.9	623.08	0.55	-	623.08	-
G-04	Y	628.08	628.01	-	2324.8	628.08	0.74	-	628.08	-
G-05	Y	693.86	693.86	-	1225.6	693.86	2.18	-	693.86	-
G-06	Y	655.18	655.14	-	595.6	655.18	6.11	-	655.18	-
G-07	Y	628.29	628.27	-	515.1	628.29	0.59	-	628.29	-
G-08	Y	611.80	611.74	-	1085.2	611.80	1.80	-	611.80	-
G-09	Y	67.20	67.20	-	1104.6	67.20	65.67	-	67.22	0.03%
G-10	Y	67.04	67.04	-	570.6	67.04	0.69	-	67.04	-
G-11	Y	71.20	71.20	-	553.2	71.20	0.46	-	71.22	0.03%
G-12	Y	67.04	67.04	-	2082.6	67.04	1.70	-	67.04	-
G-13	Y	154.49	154.48	-	42.6	154.49	0.04	-	154.49	-
G-14	Y	154.39	154.37	-	74.3	154.39	0.04	-	154.39	-
G-15	Y	142.49	142.49	-	39.6	142.49	0.04	-	142.49	-
G-16	Y	142.39	142.38	-	44.5	142.39	0.05	-	142.39	-
G-17	Y	179.71	179.69	-	189.6	179.71	0.15	-	179.71	-
G-18	Y	186.71	186.69	-	53.1	186.71	0.37	-	186.71	-
G-19	Y	186.71	186.69	-	75.1	186.71	0.19	-	186.71	-
G-20	Y	186.71	186.69	-	145.7	186.71	1.32	-	186.71	-

D. EXPLORATION VS EXPLOITATION IN ALNS-TS

We design an experiment to illustrate the individual effects of exploration with ALNS and exploitation with TS and the necessity of combining both procedure to solve the TDVRPPC efficiently. First we tailor two variants of the ALNS-TS to solve the TDVRPPC instances, the first is the TS procedure in Section IV, the second is the ALNS procedure described in Section III. The detailed comparisons are shown in Table 6. For each of the three algorithms, the table shows the best solution cost found (C_{best}), the average best solution cost found over 10 runs (C_{avg}), and the gap between both (Gap_{avg}). For the ALNS and the TS procedure, we shows the gap between the C_{best} found by the ALNS-TS and the C_{best} found by the algorithmic variant too (Gap_{AT}).

A few observations can be made from the comparison. First, the performance of TS is close to the performance of

TABLE 5. ALNS-TS vs CPLEX (25 customers and 3 vehicles).

Inst	CPLEX					ALNS-TS				
	Opt?	UB	LB	G_{Opt}	T_{run}	C_{best}	T_{best}	G_{UB}	C_{avg}	G_{avg}
CE-01	N	695.92	561.63	19.30%	7200.0	613.75	189.07	**-11.81%**	619.37	0.92%
CE-02	N	829.82	693.89	16.38%	7200.0	789.91	122.09	**-4.81%**	789.91	-
CE-03	N	855.72	606.04	29.18%	7200.0	656.87	87.93	**-23.24%**	656.94	0.01%
CE-04	N	735.15	601.58	18.17%	7200.0	666.97	0.18	**-9.27%**	666.97	-
CE-05	N	843.44	607.31	28.00%	7200.0	731.49	26.07	**-13.27%**	738.80	1.00%
CE-06	N	798.09	574.66	28.00%	7200.0	630.82	102.86	**-20.96%**	633.40	0.41%
CE-07	N	993.62	716.88	27.85%	7200.0	851.91	0.42	**-14.26%**	852.13	0.03%
CE-08	N	890.61	615.80	30.86%	7200.0	658.68	7.73	**-26.04%**	660.77	0.32%
CE-09	N	710.18	633.82	10.75%	7200.0	682.84	0.14	**-3.85%**	682.84	-
CE-10	N	791.28	632.84	20.02%	7200.0	734.65	30.05	**-7.16%**	734.65	-
CE-11	N	1281.78	948.78	25.98%	7200.0	1109.88	3.23	**-13.41%**	1110.48	0.05%
CE-12	N	718.52	388.16	45.98%	7200.0	516.75	106.68	**-28.08%**	517.24	0.09%
CE-13	N	1383.84	593.39	57.12%	7200.0	1110.96	240.28	**-19.72%**	1112.88	0.17%
CE-14	N	675.56	432.35	36.00%	7200.0	517.59	234.50	**-23.38%**	518.22	0.12%
G-01	N	958.33	889.87	7.14%	7200.0	912.74	4.44	**-4.76%**	912.74	-
G-02	N	1181.69	909.21	23.06%	7200.0	933.74	1.08	**-20.98%**	933.74	-
G-03	N	1120.35	921.21	17.77%	7200.0	946.74	99.66	**-15.50%**	946.74	-
G-04	N	1085.50	584.01	46.20%	7200.0	956.74	300.01	**-11.86%**	956.74	-
G-05	N	1383.16	1150.65	16.81%	7200.0	1191.53	61.21	**-13.85%**	1193.48	0.16%
G-06	N	1355.05	977.40	27.87%	7200.0	1002.92	44.55	**-25.99%**	1005.52	0.26%
G-07	N	1143.78	930.95	18.61%	7200.0	956.13	4.56	**-16.41%**	956.13	-
G-08	N	1164.28	906.90	22.11%	7200.0	932.05	14.09	**-19.95%**	932.05	-
G-09	N	127.58	103.58	18.81%	7200.0	109.25	1.82	**-14.37%**	110.02	0.70%
G-10	N	110.42	97.62	11.59%	7200.0	105.25	53.63	**-4.68%**	105.98	0.69%
G-11	N	119.71	103.64	13.42%	7200.0	111.25	3.27	**-7.07%**	111.80	0.49%
G-12	N	115.15	99.94	13.21%	7200.0	108.25	8.79	**-5.99%**	108.72	0.43%
G-13	Y	242.57	242.56	-	2412.1	242.57	0.85	-	243.65	0.45%
G-14	Y	239.46	239.43	-	7187.1	239.46	0.92	-	240.00	0.23%
G-15	Y	225.57	225.55	-	1819.8	225.57	62.36	-	225.57	-
G-16	Y	225.46	225.44	-	3705.5	225.46	101.95	-	225.46	-
G-17	N	276.80	216.96	21.62%	7200.0	221.14	56.05	**-20.11%**	223.19	0.93%
G-18	N	288.11	226.92	21.24%	7200.0	232.14	4.59	**-19.43%**	232.29	0.06%
G-19	N	260.48	226.57	13.02%	7200.0	232.14	164.29	**-10.88%**	234.67	1.09%
G-20	N	256.06	226.52	11.54%	7200.0	232.14	10.11	**-9.34%**	234.38	0.96%

the ALNS-TS, with an average GAP_{AT} of 1.9%, which shows that the TS performs relatively well with its comprehensive exploitation. Second, ALNS has the largest Gap_{avg} and a larger Gap_{AT}. ALNS is designed to jump to unexplored regions in the search process, which explains the large value for Gap_{avg} for solution diversity. Furthermore, each iteration in ALNS destroys the current solution through a large neighbourhood change controlled by parameters α_{min}, α_{max}, and α_{rate}. No guarantee of solution improvement is enforced in the design of ALNS. This explains the larger Gap_{AT}. Lastly, ALNS-TS performs best among the three algorithms as benefits from the exploitation strength of TS with an additional perturbation through ALNS iterations.

Furthermore, even though the Gap_{avg} are in general larger than those reported in Table 4 and 5 for the small scale instances, most of the Gap_{avg} are in the range of 1% to 3% with a median value of 2.0%.

E. CONSISTENCY OF SOLUTION COSTS FOUND

We design a box plot to illustrate the consistency of solution costs found by the ALNS-TS for the TDVRPPC test instances. We denote the test instances in Table 4 with 15 customers as "$C-15$", the test instances in Table 5 with 25 customers as "$C-25$", and the test instances in Table 6 as "*TDVRPPC*", and compare the results for the three groups. We first determine the standard score (z score) of the solution cost for each of the 10 solutions found by the ALNS-TS algorithm for a test instance. The standard score is formally calculated as $(cost-\mu)/\sigma$, where *cost* is the actual solution cost found in the run, μ represents the mean of the costs, and σ represents the standard deviation of the costs for the test instance. Lastly, we group the standard score of a total of $10 \times 34 = 340$ runs per group together and plot number by instance group in Figure 3 below.

TABLE 6. ALNS-TS vs ALNS vs TS (TDVRPPC).

			ALNS-TS			ALNS				TS			
Inst	**n**	**|K|**	C_{best}	C_{avg}	Gap_{avg}	C_{best}	Gap_{AT}	C_{avg}	Gap_{avg}	C_{best}	Gap_{AT}	C_{avg}	Gap_{avg}
CE-01	50	4	1125.85	1138.90	1.2%	1615.16	74.7%	1990.01	23.2%	1128.80	1.6%	1157.45	2.5%
CE-02	75	9	1881.83	1899.15	0.9%	3140.56	99.6%	3789.97	20.7%	1883.13	0.7%	1912.09	1.5%
CE-03	100	6	1984.46	2008.86	1.2%	3055.52	83.4%	3683.42	20.5%	1994.19	1.4%	2036.23	2.1%
CE-04	150	9	2653.97	2699.18	1.7%	4868.81	116.4%	5841.46	20.0%	2648.61	2.5%	2767.08	4.5%
CE-05	199	13	3305.85	3371.55	2.0%	5936.85	120.1%	7421.64	25.0%	3278.28	1.3%	3415.88	4.2%
CE-06	50	4	1223.80	1230.29	0.5%	1676.03	76.4%	2170.45	29.5%	1226.35	2.5%	1261.37	2.9%
CE-07	75	9	2084.24	2115.56	1.5%	3293.45	95.7%	4140.40	25.7%	2089.83	0.9%	2134.12	2.1%
CE-08	100	6	2149.56	2179.52	1.4%	3256.15	73.8%	3788.73	16.4%	2141.57	0.8%	2197.63	2.6%
CE-09	150	10	2649.34	2675.28	1.0%	5659.03	140.1%	6424.68	13.5%	2632.56	1.6%	2718.70	3.3%
CE-10	199	13	3581.27	3688.21	3.0%	6803.39	112.0%	7819.78	14.9%	3568.64	0.7%	3713.42	4.1%
CE-11	120	6	2381.03	2440.94	2.5%	7575.61	253.0%	8615.75	13.7%	2470.25	6.5%	2599.72	5.2%
CE-12	100	8	2016.99	2057.77	2.0%	3599.80	119.6%	4518.73	25.5%	2051.74	3.3%	2126.65	3.7%
CE-13	120	6	2902.54	2978.78	2.6%	7034.06	199.1%	8910.04	26.7%	2950.07	2.8%	3063.23	3.8%
CE-14	100	7	2266.47	2308.86	1.9%	4009.49	112.5%	4907.18	22.4%	2241.98	1.2%	2336.55	4.2%
G-01	240	7	14568.86	14725.38	1.1%	31330.98	130.8%	33981.19	8.5%	14533.91	0.2%	14752.41	1.5%
G-02	320	8	19783.22	20243.96	2.3%	40814.44	131.6%	46884.07	14.9%	19782.95	0.2%	20290.57	2.6%
G-03	400	8	26504.81	28480.74	7.5%	56159.00	119.1%	62406.62	11.1%	26653.15	3.3%	29410.68	10.3%
G-04	480	8	38201.97	45101.07	18.1%	72323.93	85.2%	83544.27	15.5%	38116.64	1.9%	45972.20	20.6%
G-05	200	4	14852.89	15323.96	3.2%	32548.15	135.3%	36052.08	10.8%	14849.20	0.6%	15422.51	3.9%
G-06	280	5	22478.92	23189.73	3.2%	44245.52	115.3%	49936.07	12.9%	22443.55	0.6%	23337.01	4.0%
G-07	360	7	24965.84	28206.24	13.0%	52057.28	110.2%	59293.00	13.9%	25328.53	-0.7%	28000.69	10.5%
G-08	440	8	32960.50	37238.53	13.0%	64591.85	91.4%	71291.25	10.4%	33093.99	0.7%	37489.02	13.3%
G-09	255	11	1405.36	1428.33	1.6%	4727.55	258.6%	5122.56	8.4%	1399.96	-1.1%	1412.35	0.9%
G-10	323	13	1699.83	1737.41	2.2%	6446.27	306.1%	7055.42	9.4%	1682.02	2.2%	1776.16	5.6%
G-11	399	14	2294.82	2444.16	6.5%	9088.38	298.9%	9748.96	7.3%	2276.01	10.6%	2704.46	18.8%
G-12	483	15	2718.88	3346.86	23.1%	11528.53	260.0%	12050.31	4.5%	2700.33	12.8%	3774.39	39.8%
G-13	252	21	2355.94	2394.21	1.6%	4801.99	117.2%	5199.22	8.3%	2355.80	1.8%	2436.52	3.4%
G-14	320	23	3179.85	3263.77	2.6%	6418.27	117.8%	7107.47	10.7%	3062.21	-1.0%	3229.99	5.5%
G-15	396	26	5795.59	5881.75	1.5%	9068.37	57.4%	9260.63	2.1%	5599.16	-2.1%	5758.48	2.8%
G-16	480	29	4706.17	5268.55	11.9%	10657.08	113.9%	11268.10	5.7%	4613.15	3.1%	5432.42	17.8%
G-17	240	18	1784.06	1827.32	2.4%	3134.45	116.7%	3960.55	26.4%	1803.78	0.6%	1839.15	2.0%
G-18	300	22	2915.62	2971.39	1.9%	4987.35	115.7%	6410.25	28.5%	2882.73	-0.3%	2963.73	2.8%
G-19	360	26	3926.64	4004.70	2.0%	6966.26	106.7%	8278.27	18.8%	3796.62	-1.6%	3941.30	3.8%
G-20	420	31	5145.59	5252.29	2.1%	8758.57	103.4%	10682.56	22.0%	4937.83	4.2%	5471.19	10.8%
Avg					4.2%		134.4%		16.1%		1.9%		6.7%

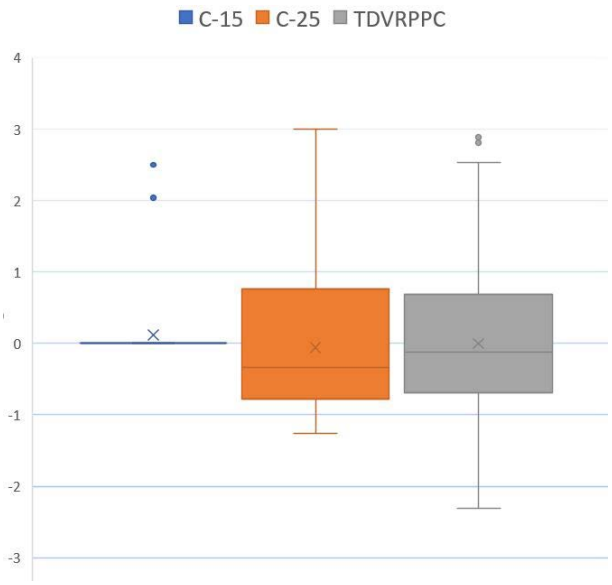

FIGURE 3. Standard score of solution cost per run by instance group.

The box plot shows that the solution costs are very consistent as most of the values are near to the reference point, especially for C-15 and C-25. Indeed, the standard scores for 322 of the 340 runs for the C-15, 199 runs for the C-50 test instances, and 237 runs for the TDVRPPC test instances falls within 1σ away from the mean value.

F. RESULTS ON TDVRPPC AND VRPPC

The introduction of the time-dependent travel time and the maximum trip duration constraint in the TDVRPPC places extra constraints to the VRPPC solutions. Hence, we apply the ALNS-TS algorithm to solve both the VRPPC and the TDVRPPC instances to evaluate the impact and tightness of the capacity constraint and the maximum trip duration constraint. Besides, it is necessary to evaluate the changes in the cost allocation between internal cost and outsourced cost. Table 7 compares the best solution cost found (C_{best}), the average percentage of outsourcing cost over total solution cost (R_{cs}), the average loads of the vehicle routes over Q in terms of percentage (R_{cap}), and the average route duration of the vehicle routes over T_{max} in terms of percentage (R_{dur}). Based on the results, the new constraints have forced more customers to be outsourced and in general increased both C_{best} by 1.9% and R_{cs} by 3.1%. Therefore, the TDVRPPC constraints have positive impact on the solution cost. Though R_{cap} is reduced by about 1.1%, R_{dur} is about 5.6% smaller than R_{cap} for the TDVRPPC. This may suggest that either capacity constraints are tighter than maximum route duration constraints, or that the ALNS-TS prioritizes to maximize the vehicle capacity to reduce outsourcing cost and total cost. The next experiment is hence designed to evaluate a scenario when the maximum route duration is tightened.

TABLE 7. VRPPC vs TDVRPPC.

Inst.	VRPPC			TDVRPPC			
	C_{best}	R_{cs}	R_{cap}	C_{best}	R_{cs}	R_{cap}	R_{dur}
CE-01	1119.47	14.4	98.7	1125.85	15.3	98.8	95.1
CE-02	1828.94	5.9	99.1	1881.83	7.9	98.2	91.6
CE-03	1942.52	17.3	99.7	1984.46	18.8	99.5	94.4
CE-04	2549.56	20.2	99.8	2653.97	23.7	99.3	94.5
CE-05	3191.44	19.7	99.8	3305.85	23.2	99.5	93.4
CE-06	1207.47	13.8	98.7	1223.8	15.6	98.8	94.3
CE-07	2023.89	5.9	99.2	2084.24	8.8	98.1	93.3
CE-08	2075.59	16.9	99.7	2149.56	19.2	99.5	94.8
CE-09	2499.56	10.2	99.7	2649.34	14.9	99	95.3
CE-10	3471.61	19.4	99.8	3581.27	22.1	99.4	93.6
CE-11	2435.05	10.1	100	2381.03	11.9	99.8	86.4
CE-12	1990.76	11.3	100	2016.99	14.4	99.3	91.2
CE-13	2979.6	10.1	100	2902.54	10.1	99.9	86
CE-14	2225.49	24.2	100	2266.47	25.3	99.9	91.9
G-01	14398.8	23.5	100	14568.9	23.6	100	96.8
G-02	19617.9	13.9	100	19783.2	14.8	100	96.3
G-03	25308.8	10.9	100	26504.8	21.4	98.7	94.4
G-04	35711.3	20.6	100	38202	36.6	97.5	90.3
G-05	14531.1	9.6	100	14852.9	9.8	100	98
G-06	22324.6	22.5	100	22478.9	23.7	100	95.6
G-07	24361.4	13.8	100	24965.8	23.1	99.8	93.5
G-08	30331.7	22.4	100	32960.5	35.6	99.1	93.3
G-09	1345.47	8.4	99.2	1405.36	8.6	98.7	91.5
G-10	1631.07	6.2	99.1	1699.83	9.2	98.2	90.5
G-11	2239.1	7.7	99.3	2294.82	16.2	91.7	90.8
G-12	2587.93	9.2	99.3	2718.88	29.4	91.3	87.5
G-13	2319.7	10.1	98.4	2355.94	12.4	97.6	89.7
G-14	2785.08	14.1	98.2	3179.85	23.2	95.8	94
G-15	3279.17	13.2	98.3	5795.59	52.7	83.7	97.9
G-16	3762.81	12	98.4	4706.17	35.8	85.1	91.9
G-17	1776.86	22.2	100	1784.06	21.8	100	92.5
G-18	2834.32	19.6	100	2915.62	20.9	100	91.5
G-19	3617.54	21.1	100	3926.64	24.1	100	93.3
G-20	4466.92	18.6	100	5145.59	28.6	97	91.5
Average	7375.66	14.68	99.54	7778.02	20.67	97.74	92.84

G. IMPACT OF CAPACITY AND MAXIMUM ROUTE DURATION CONSTRAINTS

We reduce T_{max} by 10% to generate another instance group (TDVRPPC2) for further evaluation of the impact of both constraints. As shown in Table 8, we group the instances by either type CE or G and present the average of R_{cap}, R_{dur} and R_{cs} in percentage values. As shown in the results, the average R_{dur} increased more significantly than the reduction in the average R_{cap} for both groups in TDVRPPC2 instances when T_{max} is reduced by 10%. The results also show that introducing T_{max} and reducing T_{max} increases the average R_{cs}, while means the company is forced to outsource more customers using 3PL services due to tighter constraints on vehicle route duration. This ultimately resulted in a higher average solution cost ($Cost$) for both groups. However, the average R_{cap} is still higher than the average R_{dur} for both groups, which again suggests that the ALNS-TS may prefer to maximize vehicle capacity utilization than to maximize the route duration to reduce outsourcing costs and total costs.

H. VRPPC BENCHMARK RESULTS

Lastly, we present a comparison on the VRPPC benchmark data with two state-of-the-art algorithms, namely the tabu search algorithm in [2] (TS+) and the AVNS in [3], in Table 9.

TABLE 8. Impact of constraints on the solution.

	VRPPC		TDVRPPC		TDVRPPC2	
	CE	G	CE	G	CE	G
R_{cap}	99.6	99.5	99.2	96.7	98.9	97.3
R_{dur}	-	-	92.6	93	95.2	97
R_{CS}	14.2	15	16.5	23.6	17.8	20.6
$Cost$	2252.9	10961.6	2300.5	11612.3	2312.7	11635.6

TABLE 9. VRPPC benchmark comparisons.

Inst.	TS+	AVNS	ALNS-TS	Inst.	TS+	AVNS	ALNS-TS
CE-01	**1119.47**	1123.95	**1119.47**	G-04	34802.08	34415.82	35711.26
CE-02	1814.52	1814.52	1828.94	G-05	14261.31	14272.32	14531.12
CE-03	1930.66	1920.86	1942.52	G-06	21498.03	21440.79	22324.59
CE-04	2525.17	2512.05	2549.56	G-07	23513.06	23375.60	24361.35
CE-05	3117.10	3099.77	3191.44	G-08	30073.56	29797.62	30331.67
CE-06	**1207.47**	1207.81	**1207.47**	G-09	1325.62	1335.45	1345.47
CE-07	2006.52	2013.93	2023.89	G-10	1590.82	1604.50	1631.07
CE-08	2056.59	2052.05	2075.59	G-11	2173.80	2189.02	2239.10
CE-09	2435.97	2432.51	2499.56	G-12	2495.02	2520.29	2587.93
CE-10	3401.83	3391.35	3471.61	G-13	2274.12	2291.83	2319.70
CE-11	2332.36	2332.21	2435.05	G-14	2703.31	2708.22	2785.08
CE-12	1952.86	1953.55	1990.76	G-15	3161.26	3194.82	3279.17
CE-13	2860.89	2858.94	2979.60	G-16	3638.39	3671.34	3762.81
CE-14	2216.97	2215.38	2225.49	G-17	1633.35	1682.49	1776.86
G-01	14190.01	14157.08	14398.80	G-18	2710.21	2741.80	2834.32
G-02	19208.52	19204.36	19617.86	G-19	3497.72	3507.94	3617.54
G-03	24592.18	24602.61	25308.81	G-20	4306.89	4332.44	4466.92

The TS+ algorithm [2] extends the TS with a neighbourhood structure based on ejection chains. The AVNS [3] develops a group of 51 cyclic-exchange neighborhoods and incorporates an adaptive mechanism to bias the random shaking step. As shown in the table, the ALNS-TS algorithm obtains best-known solutions for two instances and the average deviation of the best found solution cost from the best solutions is about 2.6%.We conclude that the performances of the ALNS-TS on the VPRPV test instance is acceptable since it is not designed specially to solve the VRPPC problem.

VII. CONCLUSION

In this work, we examine the practical time-dependent travel times on the road and the maximum route duration constraints for the drivers for the TDVRPPC problem. Two popular heuristic algorithms, namely the ALNS and the TS, are hybridized together to minimize the total cost consisting of fixed cost, variable cost, and outsourced cost. The ALNS broadens the exploration through adaptively controlling the neighborhood size and TS dynamically deepens the exploitation while relaxing duration and/or capacity-related constraints. Computational experiments show that our algorithm performs well on the TDVRPPC benchmark data. For further research, we are interested to extend the current TDVRPPC model with richer real-world considerations, such as adding multi-trip per vehicle, customer time window limitation, as well as multi-depot for vehicles. At the same time, new efficient algorithms can be proposed and evaluated on the benchmark data.

REFERENCES

[1] C.-W. Chu, "A heuristic algorithm for the truckload and less-than-truckload problem," *Eur. J. Oper. Res.*, vol. 165, no. 3, pp. 657–667, 2005.

[2] J.-Y. Potvin and M.-A. Naud, "Tabu search with ejection chains for the vehicle routing problem with private fleet and common carrier," *J. Oper. Res. Soc.*, vol. 62, no. 2, pp. 326–336, Feb. 2011.

[3] A. Stenger, D. Vigo, S. Enz, and M. Schwind, "An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping," *Transp. Sci.*, vol. 47, no. 1, pp. 64–80, Feb. 2013.

[4] Z. Zhang, Z. Luo, H. Qin, and A. Lim, "Exact algorithms for the vehicle routing problem with time windows and combinatorial auction," *Transp. Sci.*, vol. 53, no. 2, pp. 427–441, Mar. 2019.

[5] C. Gahm, C. Brabänder, and A. Tuma, "Vehicle routing with private fleet, multiple common carriers offering volume discounts, and rental options," *Transp. Res. E, Logistics Transp. Rev.*, vol. 97, pp. 192–216, Jan. 2017.

[6] S. Dabia, D. Lai, and D. Vigo, "An exact algorithm for a rich vehicle routing problem with private fleet and common carrier," *Transp. Sci.*, vol. 53, no. 4, pp. 986–1000, Jul. 2019.

[7] P. Toth and D. Vigo, *Vehicle Routing: Problems, Methods, and Applications*. Philadelphia, PA, USA: SIAM, 2014.

[8] C. Malandraki and M. S. Daskin, "Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms," *Transp. Sci.*, vol. 26, no. 3, pp. 185–200, 1992.

[9] S. Ichoua, M. Gendreau, and J.-Y. Potvin, "Vehicle dispatching with time-dependent travel times," *Eur. J. Oper. Res.*, vol. 144, no. 2, pp. 379–396, Jan. 2003.

[10] L. Wen and R. Eglese, "Minimum cost VRP with time-dependent speed data and congestion charge," *Comput. Oper. Res.*, vol. 56, pp. 41–50, Apr. 2015.

[11] Y. Huang, L. Zhao, T. Van Woensel, and J.-P. Gross, "Time-dependent vehicle routing problem with path flexibility," *Transp. Res. B, Methodol.*, vol. 95, pp. 169–195, Jan. 2017.

[12] C. Liu, G. Kou, X. Zhou, Y. Peng, H. Sheng, and F. E. Alsaadi, "Time-dependent vehicle routing problem with time windows of city logistics with a congestion avoidance approach," *Knowl.-Based Syst.*, vol. 188, Jan. 2020, Art. no. 104813.

[13] Y. Xiao and A. Konak, "The heterogeneous green vehicle routing and scheduling problem with time-varying traffic congestion," *Transp. Res. E, Logistics Transp. Rev.*, vol. 88, pp. 146–166, Apr. 2016.

[14] R. Zhang, J. Guo, and J. Wang, "A time-dependent electric vehicle routing problem with congestion tolls," *IEEE Trans. Eng. Manag.*, early access, Jan. 1, 2020, doi: 10.1109/TEM.2019.2959701.

[15] A. V. Donati, R. Montemanni, N. Casagrande, A. E. Rizzoli, and L. M. Gambardella, "Time dependent vehicle routing problem with a multi ant colony system," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1174–1191, 2008.

[16] S. R. Balseiro, I. Loiseau, and J. Ramonet, "An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows," *Comput. Oper. Res.*, vol. 38, no. 6, pp. 954–966, 2011.

[17] B. Pan, Z. Zhang, and A. Lim, "Multi-trip time-dependent vehicle routing problem with time windows," *Eur. J. Oper. Res.*, vol. 291, no. 1, pp. 218–231, May 2021.

[18] B. Pan, Z. Zhang, and A. Lim, "A hybrid algorithm for time-dependent vehicle routing problem with time windows," *Comput. Oper. Res.*, vol. 128, Apr. 2021, Art. no. 105193.

[19] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "Time-window relaxations in vehicle routing heuristics," *J. Heuristics*, vol. 21, no. 3, pp. 329–358, Jun. 2015.

[20] M. Gendreau, G. Ghiani, and E. Guerriero, "Time-dependent routing problems: A review," *Comput. Oper. Res.*, vol. 64, pp. 189–197, Dec. 2015.

[21] L. Foschini, J. Hershberger, and S. Suri, "On the complexity of time-dependent shortest paths," in *Proc. 22nd Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2011, pp. 327–341.

[22] T. Adamo, G. Ghiani, and E. Guerriero, "On path ranking in time-dependent graphs," *Comput. Oper. Res.*, vol. 135, Nov. 2021, Art. no. 105446.

[23] S. Allahyari, S. Yaghoubi, and T. Van Woensel, "The secure time-dependent vehicle routing problem with uncertain demands," *Comput. Oper. Res.*, vol. 131, Jul. 2021, Art. no. 105253.

[24] N. Rincon-Garcia, B. Waterson, T. J. Cherrett, and F. Salazar-Arrieta, "A metaheuristic for the time-dependent vehicle routing problem considering driving hours regulations—An application in city logistics," *Transp. Res. A, Policy Pract.*, vol. 137, pp. 429–446, Jul. 2020.

[25] A. L. Kok, E. W. Hans, and J. M. J. Schutten, "Vehicle routing under time-dependent travel times: The impact of congestion avoidance," *Comput. Oper. Res.*, vol. 39, no. 5, pp. 910–918, May 2012.
[26] T. Erdelić, T. Carić, M. Erdelić, L. Tišljarić, A. Turković, and N. Jelušić, "Estimating congestion zones and travel time indexes based on the floating car data," *Comput., Environ. Urban Syst.*, vol. 87, May 2021, Art. no. 101604.
[27] J.-F. Cordeau, M. Gendreau, and G. Laporte, "A Tabu search heuristic for periodic and multi-depot vehicle routing problems," *Netw., Int. J.*, vol. 30, no. 2, pp. 105–119, 1997.
[28] J.-F. Cordeau, G. Laporte, and A. Mercier, "A unified Tabu search heuristic for vehicle routing problems with time windows," *J. Oper. Res. Soc.*, vol. 52, no. 8, pp. 928–936, Aug. 2001.
[29] H. Qin, W. Ming, Z. Zhang, Y. Xie, and A. Lim, "A Tabu search algorithm for the multi-period inspector scheduling problem," *Comput. Oper. Res.*, vol. 59, pp. 78–93, Jul. 2015.
[30] M. Gmira, M. Gendreau, A. Lodi, and J.-Y. Potvin, "Tabu search for the time-dependent vehicle routing problem with time windows on a road network," *Eur. J. Oper. Res.*, vol. 288, no. 1, pp. 129–140, Jan. 2021.
[31] Z. Luo, M. Poon, Z. Zhang, Z. Liu, and A. Lim, "The multi-visit traveling salesman problem with multi-drones," *Transp. Res. C, Emerg. Technol.*, vol. 128, Jul. 2021, Art. no. 103172.
[32] Y. Wu, B. Zheng, and X. Zhou, "A disruption recovery model for time-dependent vehicle routing problem with time windows in delivering perishable goods," *IEEE Access*, vol. 8, pp. 189614–189631, 2020.
[33] L. Wei, Z. Zhang, and A. Lim, "An adaptive variable neighborhood search for a heterogeneous fleet vehicle routing problem with three-dimensional loading constraints," *IEEE Comput. Intell. Mag.*, vol. 9, no. 4, pp. 18–30, Oct. 2014.
[34] L. Wei, Z. Zhang, D. Zhang, and A. Lim, "A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints," *Eur. J. Oper. Res.*, vol. 243, no. 3, pp. 798–814, Jun. 2015.
[35] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transp. Sci.*, vol. 40, no. 4, pp. 455–472, 2006.
[36] V. C. Hemmelmayr, J.-F. Cordeau, and T. G. Crainic, "An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics," *Comput. Oper. Res.*, vol. 39, no. 12, pp. 3215–3228, 2012.
[37] E. Demir, T. Bektaş, and G. Laporte, "An adaptive large neighborhood search heuristic for the pollution-routing problem," *Eur. J. Oper. Res.*, vol. 223, no. 2, pp. 346–359, Dec. 2012.
[38] Y. Adulyasak, J.-F. Cordeau, and R. Jans, "Optimization-based adaptive large neighborhood search for the production routing problem," *Transp. Sci.*, vol. 48, no. 1, pp. 20–45, 2012.
[39] Z. Luo, H. Qin, D. Zhang, and A. Lim, "Adaptive large neighborhood search heuristics for the vehicle routing problem with stochastic demands and weight-related cost," *Transp. Res. E, Logistics Transp. Rev.*, vol. 85, pp. 69–89, Jan. 2016.
[40] V. François, Y. Arda, and Y. Crama, "Adaptive large neighborhood search for multitrip vehicle routing with time windows," *Transp. Sci.*, vol. 53, no. 6, pp. 1706–1730, Nov. 2019.
[41] P. Sun, L. P. Veelenturf, M. Hewitt, and T. Van Woensel, "Adaptive large neighborhood search for the time-dependent profitable pickup and delivery problem with time windows," *Transp. Res. E, Logistics Transp. Rev.*, vol. 138, Jun. 2020, Art. no. 101942.
[42] Z. Zhang, M. Liu, and A. Lim, "A memetic algorithm for the patient transportation problem," *Omega*, vol. 54, pp. 60–71, Jul. 2015.
[43] A. Lim, Z. Zhang, and H. Qin, "Pickup and delivery service with manpower planning in Hong Kong public hospitals," *Transp. Sci.*, vol. 51, no. 2, pp. 688–705, May 2017.
[44] M.-C. Bolduc, J. Renaud, F. Boctor, and G. Laporte, "A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers," *J. Oper. Res. Soc.*, vol. 59, no. 6, pp. 776–787, Jun. 2008.
[45] N. Christofides and S. Eilon, "An algorithm for the vehicle-dispatching problem," *J. Oper. Res. Soc.*, vol. 20, no. 3, pp. 309–318, 1969.
[46] B. L. Golden, E. A. Wasil, J. P. Kelly, and I.-M. Chao, "The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results," in *Fleet Management and Logistics*. Boston, MA, USA: Springer, 1998, pp. 33–56.
[47] H. Hashimoto, M. Yagiura, and T. Ibaraki, "An iterated local search algorithm for the time-dependent vehicle routing problem with time windows," *Discrete Optim.*, vol. 5, no. 2, pp. 434–456, May 2008.
[48] J.-F. Cordeau, G. Ghiani, and E. Guerriero, "Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem," *Transp. Sci.*, vol. 48, no. 1, pp. 46–58, Feb. 2014.
[49] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, L. P. Cáceres, and M. Birattari, "The irace package: Iterated racing for automatic algorithm configuration," *Oper. Res. Perspect.*, vol. 3, pp. 43–58, Sep. 2016.

MARK POON received the B.S. and M.S. degrees in computer science and the Ph.D. degree in industrial system engineering from the National University of Singapore (NUS), in 2006, 2009, and 2021, respectively.

He has been a Postdoctoral Research Fellow at HEC Montréal, Canada, since 2021. His research interest includes applied operations research and he has published papers in various operations research and transportation research journals, such as *Transportation Research Part C: Emerging Technologies*, *European Journal of Operational Research*, and *Computers and Operations Research*.

RUIXUE GU received the B.S. degree in electronic engineering from the University of Electronic Science and Technology of China, in 2017, and the M.S. degree in electrical and computer engineering from the National University of Singapore, in 2018, where she is currently pursuing the Ph.D. degree in industrial systems engineering and management. Her research interests include operation research in vehicle routing problems and heuristic algorithms.

YILIANG YUAN received the B.S. degree in computer science from the University of Science and Technology of China, in 2018. He is currently pursuing the Ph.D. degree in industrial systems engineering and management with the National University of Singapore. His research interests include operation research in vehicle routing problem and heuristic algorithms.