Multi-Trip Time-Dependent Vehicle Routing Problem with Time Windows

Binbin Pan[b], Zhenzhen Zhang[*,a], Andrew Lim[b]

[a] *School of Economics and Management, Tongji University, Shanghai 200092, China*
[b] *Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore 117576*

Abstract

In this study, we investigate a routing problem in urban transportation which considers time-dependent travel time, multiple trips per vehicle, and loading time at the depot simultaneously. Its objective is to minimize the total travel distance while satisfying the time windows, vehicle capacity, and maximum trip duration constraints. We model the problem as a multi-trip time-dependent vehicle routing problem with time windows (MT-TDVRPTW). We formulate the time-dependent ready time function and duration function for any segment of consecutive nodes as piecewise linear functions and develop an iterative algorithm to derive them efficiently. Then, these two functions are embedded in the segment-based evaluation scheme to accelerate the local search operators. Based on them, we design a hybrid meta-heuristic algorithm to solve the problem, leveraging the adaptive large neighborhood search (ALNS) for guided exploration and the variable neighborhood descend (VND) for intensive exploitation. Moreover, we propose problem-specific local search operators and removal operators to enhance the effectiveness of the algorithm. Extensive experiments are conducted to assess the performance of the algorithm on instances of varied sizes. The algorithm is shown to be robust and efficient under different speed profiles and maximum trip duration limits. Finally, we evaluate the performance of the algorithm on a special case: the multi-trip vehicle routing problem with time windows.

Key words: routing, time-dependent, multi-trip, mathematical model, meta-heuristic

[*]Corresponding author
Email addresses: panbinbin11@gmail.com (Binbin Pan), zhenzhenzhang222@gmail.com (Zhenzhen Zhang)

1. Introduction

This study is motivated by the routing design faced by a vending cafe company to replenish stocks for their geographically dispersed outlets around Singapore. Each outlet has a different capacity, demand, and time window for replenishment during the day. A fleet of small lorries is deployed to fulfill the delivery tasks each day. To ensure the welfare and safety of employees, the duration of each trip is limited, and drivers are entitled to a short break after the trip. Therefore, multiple trips are performed by each driver per day. Another challenging issue is the varying travel speeds throughout the day in Singapore. Indeed, drivers purposely avoid delivery to certain outlets during certain periods of the day due to the traffic situations. Moreover, the loading time at the depot needs to be considered explicitly in this context due to its impact on the departure time. This problem also resembles other real-world applications, such as the home delivery of perishable foods and the distribution of goods. We model this new variant as the multi-trip time-dependent vehicle routing problem with time windows (MT-TDVRPTW), which considers the following features together: time-dependent travel time, multiple trips per vehicle, required loading time at the depot, time windows, and trip duration limit.

Most existing studies on vehicle routing problem (VRP) and its variant with time windows (VRPTW) assumed implicitly that the travel time between two nodes is constant and independent of the departure time. However, this assumption is seriously challenged because travel speeds on the road vary substantially during peak and off-peak hours in the urban areas. Consequently, the routes generated by the traditional time-invariant VRP approach are sub-optimal or even infeasible under the time-dependent settings (Ichoua et al., 2003). As a result, the time-dependent vehicle routing problem (TDVRP) and its variant with time windows (TDVRPTW) have been developed to close the research gap (Malandraki & Daskin, 1992; Ichoua et al., 2003). However, they have received less attention compared to other VRP variants because of their complexity. Thus, further research should be conducted for the TDVRP and TDVRPTW (Gendreau et al., 2015).

It is common practice to allow vehicles to perform multiple trips in city logistics due to the proximity of customers and the depot and the possible restriction on maximum trip duration. The trip duration limit may be imposed by various managerial or legal constraints, such as the nature

of perishable goods, release dates of merchandise, the regulatory constraint on maximum driving hours, and regular disinfection requirement of ambulances in healthcare applications (Hernandez et al., 2014; Cattaruzza et al., 2016; Lim et al., 2017). Previous research has highlighted that allowing vehicles to perform multiple trips can significantly increase the utilization rate of vehicles and reduce the number of vehicles required, which further leads to lower fixed and overall costs (Cattaruzza et al., 2018). However, the consideration of time windows and maximum trip duration in the multi-trip vehicle routing problem with time windows (MT-VRPTW) dramatically increases the complexity of the problem (Azi et al., 2007; Zhang et al., 2015a; Lim et al., 2017), as it becomes important to determine the departure time of each trip precisely. Furthermore, it is desirable to consider the loading time of goods between trips under the multiple trip context.

The MT-TDVRPTW problem is NP-hard because it contains the TDVRPTW and the MT-VRPTW as its special cases. It is interesting to note that this problem models various practical features together, and this creates its unique characteristics and difficulties. First, the feasibility check of a trip depends on the actual departure time from the depot due to time window constraints, time-dependent travel time, and maximum trip duration constraint. It is possible that an infeasible trip that violates the maximum trip duration constraint can become feasible if the vehicle can start at an earlier or later time to avoid heavy traffic on the road. Consequently, the feasible start-time window of a trip may be divided into several disconnected intervals, which differs from the conclusion reported by Lim et al. (2017). Second, the feasibility check of a solution requires a thorough evaluation of the trip assignments to available vehicles and careful scheduling of the trips for each vehicle. These challenging characteristics necessitate a rigorous investigation of the problem to propose suitable models and to design tailored algorithms.

Our main contributions include a formal description with related properties, a mixed integer programming (MIP) model, an efficient evaluation scheme of a single trip, a hybrid meta-heuristic framework, and comprehensive computational experiments for the MT-TDVRPTW. First, we provide a formal description of the problem and define the time-dependent travel time function, the time-dependent ready time function, and the time-dependent duration function explicitly as piecewise linear functions. These functions can determine the feasible start-time windows and the corresponding durations of a given trip. Second, we formulate the problem as an MIP, which can

be solved directly by commercial solvers for small examples. Third, we extend the segment-based evaluation method, used in Vidal et al. (2013), to this problem. This is accomplished by developing an efficient iterative algorithm to concatenate two segments of nodes together and derive relevant information of the time-dependent functions for the resulted segment. Next, we develop a hybrid adaptive large neighborhood search (ALNS) algorithm (Ropke & Pisinger, 2006) that employs two variable neighborhood descend (VND) (Hansen et al., 2019) operators as local search operators. We then derive a new set of test instances and conduct extensive computational experiments for the problem. The algorithm can solve the base test instances from Dabia et al. (2013) efficiently and performs robustly on test instances with an increased number of time zones or with a shorter maximum trip duration. Last, the performance of the algorithm is compared with an existing algorithm on the special case MT-VRPTW.

The remainder of this paper is organized as follows. Section 2 presents a brief review on related works. Section 3 provides the formal description and an MIP model for the problem. Section 4 models the time-dependent ready time function and the time-dependent duration function before presenting the algorithm to concatenate two segments of consecutive nodes. The meta-heuristic algorithm is explained in Section 5, followed by extensive computational results in Section 6. Finally, the conclusion and possible direction of future studies are discussed in Section 7.

2. Literature review

Some existing researches have studied routing problems under time-dependent settings, such as time-dependent demands (Nguyen et al., 2013, 2017) or time-dependent cost (Liu et al., 2018), while the MT-TDVRPTW problem considers the time-dependent travel time as in Ichoua et al. (2003); Dabia et al. (2013); Sun et al. (2018b,a). To the best of our knowledge, only Sun et al. (2018c) has considered both time-dependent travel time and multiple trips together. However, Sun et al. (2018c) developed a piecewise linear travel speed model which leads to a quadratic travel time function, while the MT-TDVRPTW follows the widely used piecewise linear travel time function model (Ichoua et al., 2003). Besides, Sun et al. (2018c) considered the maximal working duration constraint per day for a vehicle, while the MT-TDVRPTW enforces the maximum trip duration constraint and loading time constraint as in Azi et al. (2010); Macedo et al. (2011); Azi

et al. (2014); Hernandez et al. (2014). Last, Sun et al. (2018c) extended the two-stage algorithm for MT-VRPTW (Lang et al., 2010), whereas we customize a hybrid meta-heuristic algorithm which employs an efficient and non-trivial segment-based evaluation scheme specially designed for the MT-TDVRPTW with a piecewise linear travel time function. The subsequent review will focus on TDVRPTW with a piecewise linear travel time model and MT-VRPTW for the sake of brevity.

2.1. Related works on TDVRPTW

The earliest work on TDVRP was reported in Malandraki & Daskin (1992) with an MIP model and several heuristics for the problem. However, their stepwise travel time model might allow solutions that violate the first-in first-out (FIFO) property. Later, Ichoua et al. (2003) resolved this issue by modelling the travel speed as a stepwise function, which leads to a piecewise linear time-dependent travel time function. Many studies have followed this model to investigate TDVRPTW variants with different heuristic algorithms, such as ant colony system (Donati et al., 2008; Balseiro et al., 2011; Liu et al., 2020), iterative route construction and improvement algorithm (Figliozzi, 2012), and adaptive large neighborhood search (Zhang et al., 2020). Besides, other interesting TDVRP variants were also proposed by considering different characteristics, such as a congestion charge scheme and a speed-dependent fuel cost (Wen & Eglese, 2015), multiple paths between any pair of nodes (Huang et al., 2017), fuel consumption and carbon emission (Liu et al., 2020; Xiao & Konak, 2016), and electric vehicles (Zhang et al., 2020). It is worthy to mention that Dabia et al. (2013) proposed a novel branch-and-price algorithm to solve the duration-minimizing TDVRPTW problem. The pricing problem is a time-dependent shortest path problem with resource constraints, which is solved with a tailored label setting algorithm. The interested reader is referred to Gendreau et al. (2015) for a comprehensive review of TDVRP.

Most research on the TDVRPTW aims to minimize the total travel time (Ichoua et al., 2003; Donati et al., 2008; Balseiro et al., 2011; Figliozzi, 2012; Dabia et al., 2013; Sun et al., 2018b,a), which differs from the objective of minimizing the total travel distance in the VRPTW. This is because minimizing the total travel time under the TDVRPTW requires contemplated determination of the actual departure times from the depot to minimize trip duration. Contrarily, so long as the trip is feasible, its departure time is insignificant insofar as minimizing the total travel distance is

concerned, as it is sufficient to set the departure time to the earliest possible time. Unfortunately, this trivial approach does not apply to the MT-TDVRPTW as the departure time of a trip must be carefully determined to satisfy the maximum trip duration. Moreover, the total travel distance is a good indicator of the actual operational costs concerning fuel consumption. Thus, the MT-TDVRPTW differs from previous research on TDVRPTW in two important aspects: 1) it aims to minimize the total travel distance, and 2) it allows multiple trips and enforces loading time constraint.

2.2. Related works on MTVRPTW

The multi-trip vehicle routing problem (MTVRP) has been explored by various researchers (Taillard et al., 1996; Brandao & Mercer, 1997; Petch & Salhi, 2003; Olivera & Viera, 2007; Alonso et al., 2008; Cattaruzza et al., 2014; François et al., 2016). However, the complexity of the problem greatly increases with the introduction of time windows and maximum trip duration in the MT-VRPTW, and the body of literature is inadequate. Moreover, the MT-VRPTW is interesting and challenging as each trip is time-stamped and the scheduling of trips for a vehicle is not straightforward. Battarra et al. (2009) studied an MT-VRPTW variant where two commodities are incompatible to be transported together in the same vehicle. It applied a two-phase algorithm, which used a greedy procedure to construct MT-VRPTW solutions from the VRPTW trips. Cattaruzza et al. (2016) proposed a memetic algorithm to solve the MT-VRPTW with a release date which marks when the merchandise will be available for delivery from the depot. Lim et al. (2017) proposed a novel two-phase approach to solve a multi-trip pickup and delivery problem with manpower planning, which includes the MT-VRPTW as a special case. The problem originated from the public patient transportation service problem in Hong Kong. François et al. (2019) extended the two ALNS algorithms in François et al. (2016) with the extension of segment-based evaluation method in Vidal et al. (2013) for the MT-VRPTW. The computational experiments showed that the multi-trip operators are very efficient under the time window constraint. Exact approaches for MTVRP and MT-VRPTW are rare (Azi et al., 2007, 2010; Macedo et al., 2011; Hernandez et al., 2014; Paradiso et al., 2020) and discussion are omitted due to space constraint. Interested readers are referred to Cattaruzza et al. (2018).

The MT-TDVRPTW is significantly more challenging than the MT-VRPTW. First, it requires complicated calculations of the trip information under the time-dependent context. Second, the feasible start-time window of a trip may be disconnected. Third, the trip duration fluctuates with the departure time instead of a constant value. The aforementioned factors complicate the feasibility check of a single trip and the scheduling of multiple trips for a vehicle in MT-TDVRPTW.

3. Problem description and modelling

In this section, we describe the MT-TDVRPTW in detail before presenting an MIP model for the problem.

3.1. Basic notation

Let $V_c = \{1, 2, ..., n\}$ denote the set of customers and 0 and $n + 1$ denote the starting and the ending depots respectively. The MT-TDVRPTW is defined over a complete directed graph $G = (V, A)$, with node set $V = V_c \cup \{0, n + 1\}$ and arc set $A = \{(i, j) : i, j \in V, i \neq j\}$. Each node $i \in V$ is associated with a service time s_i, a demand q_i, and a time window $[e_i, l_i]$. Note that $q_0 = q_{n+1} = 0$, $e_{n+1} = e_0$, $l_{n+1} = l_0$, and $s_0 = s_{n+1} = 0$. The time span $[e_0, l_0]$ defines a typical workday. Additionally, each arc $(i, j) \in A$ is associated with a distance $d_{i,j}$.

Let K denote a fleet of homogeneous vehicles with capacity Q. A trip is defined as a sequence of node visits that starts from the depot, visits a sequence of customer nodes, and returns to the depot. For each trip, goods are loaded at the depot before departure, and the required loading time is proportional to the total service time incurred on the trip. The parameter of the proportion is set as $\lambda = 0.2$ in this paper. The duration of a trip is defined as the difference between the arrival time at the ending depot and the start time of loading at the starting depot, which must be smaller than the maximum trip duration limit T_{max}. As a result, a vehicle is allowed to perform several trips during the workday. Let R denote the set of possible trips for a vehicle, then the pair $(k, r), \forall k \in K, r \in R$ is used to represent the r-th trip performed by the k-th vehicle.

3.2. Time-dependent travel time

To capture the feature of time-dependent travel time, we follow the commonly used approach proposed by Ichoua et al. (2003). The workday $[e_0, l_0]$ is divided into non-overlapping time zones,

denoted as indexed set T. Although the travel speed during a time zone remains constant, it changes at the end of each time zone. The speed profile is modeled as a stepwise function (Figure 1) and is defined for each arc $(i, j) \in A$. The time-dependent travel time for each arc (i, j) is then derived based on its distance $d_{i,j}$ and its speed profile.

Proposition 1. *If the speed profile for an arc* (i, j) *is a stepwise function, then the derived time-dependent travel time function is a piecewise linear function as depicted in Figure 2.*

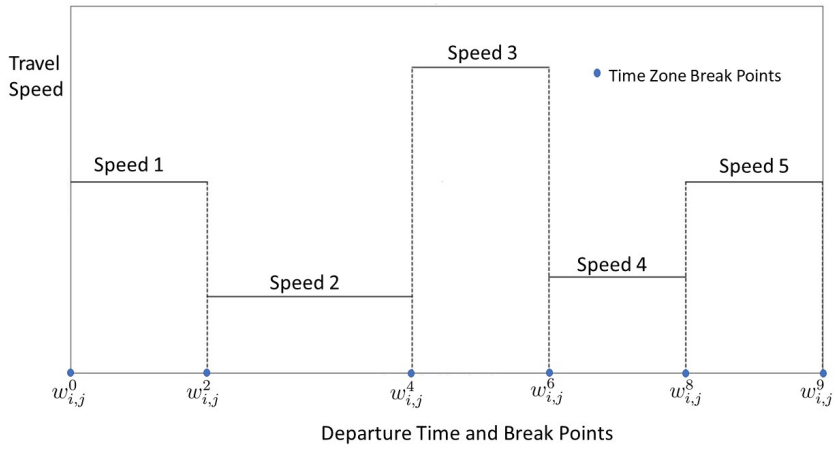

Figure 1: Travel Speed Function from i to j

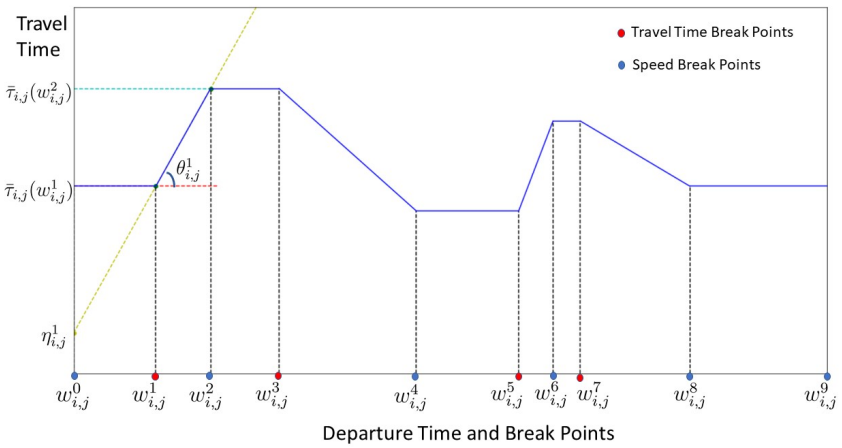

Figure 2: Travel Time Function from i to j (the solid blue lines). The extended yellow lines shows $\eta^1_{i,j}$.

Proposition 1 directly follows the claim reported by Ichoua et al. (2003) and proof is omit-

ted. There are two groups of break points in the travel time function: the ***speed break points*** $\{w_{i,j}^0, w_{i,j}^2, w_{i,j}^4, w_{i,j}^6, w_{i,j}^8, w_{i,j}^9\}$ which are the boundary points of time zones independent of arcs, and the ***travel time break points*** $\{w_{i,j}^1,\ w_{i,j}^3, w_{i,j}^5,\ w_{i,j}^7\}$ which are the departure times at node i such that the corresponding arrival time at node j is equal to one of the ***speed break points*** (Figure 2).

It should be noted that the ***travel time break points*** are usually unique for each arc (i, j) due to the distinctive distance $d_{i,j}$ and its speed profile. Therefore, the resulting time zones of the travel time function for an arc are also unique to the arc itself. We define the resulted ***arc time zones*** as an indexed set $T_{i,j} = \{0, 1, 2, ..., |T_{i,j}| - 1\}$ for each arc (i, j). We also define the b-th break point as $w_{i,j}^b$, $\forall b \in T_{i,j} \cup \{|T_{i,j}|\}$ and the m-th arc time zone as $T_{i,j}^m = [w_{i,j}^m, w_{i,j}^{m+1})$, $\forall m \in T_{i,j}$. The actual travel time $\bar{\tau}_{i,j}(w_{i,j}^b)$ at the b-th break point $w_{i,j}^b$ can be calculated recursively using the algorithm proposed by Ichoua et al. (2003).

The travel time function of an arc (i, j), denoted as $\tau_{i,j}(t)$, can now be uniquely determined by its break points and the associated time-dependent travel times (Dabia et al., 2013; Sun et al., 2018b,a). Without loss of generality, we let $w_{ij}^0 = e_0 = 0$. For any arc time zone $T_{i,j}^m,\ m \in T_{i,j}$, the slope $\theta_{i,j}^m$ of the travel time function is calculated as

$$\theta_{i,j}^m = \left(\bar{\tau}_{ij}(w_{ij}^{m+1}) - \bar{\tau}_{ij}(w_{ij}^m)\right) / \left(w_{ij}^{m+1} - w_{ij}^m\right); \tag{1}$$

and the intersection of the line segment with the y-axis $\eta_{i,j}^m$ is computed as

$$\eta_{ij}^m = \left(w_{ij}^{m+1}\bar{\tau}_{ij}(w_{ij}^m) - w_{ij}^m\bar{\tau}_{ij}(w_{ij}^{m+1})\right) / \left(w_{ij}^{m+1} - w_{ij}^m\right),\ \forall m \in T_{i,j}. \tag{2}$$

Subsequently, the travel time required to traverse the arc (i, j) for any departure time t can be directly computed as

$$\tau_{i,j}(t) = \sum_{m \in T_{i,j}} \left(\theta_{i,j}^m t + \eta_{i,j}^m\right) \mathbb{1}_{T_{i,j}^m}(t),\ \forall t \in [e_0, l_0], \tag{3}$$

where $\mathbb{1}_{T_{i,j}^m}(t)$ is the indicator function whether t is in $T_{i,j}^m$. Note that the actual travel time can be calculated in $O(log\ |T_{i,j}|)$ time with a binary search to find the correct time zone for a given

departure time. The backward travel time function, denoted as $\tau_{i,j}^{-1}(t)$, is defined as the travel time required if the vehicle must travel along arc (i, j) and arrive at node j at exactly time t. $\tau_{i,j}^{-1}(t)$ is also a piecewise linear function and can be determined in a similar way as $\tau_{i,j}(t)$. Both $\tau_{i,j}(t)$ and $\tau_{i,j}^{-1}(t)$ will be used in the segment concatenation algorithm (Algorithm 1).

The objective of the MT-TDVRPTW is to find the min-cost solution that satisfies the customers' time window constraints, maximum trip duration constraint, loading time constraint, and vehicle capacity constraint. Same as the classical VRPTW, the travel cost is defined as the total travel distance.

3.3. Mathematical formulation

The MIP model is based on the time-dependent travel time function with five indexes: the vehicle index, the trip index, the arc time zone index, and the node indexes. The decision variables are defined for all $i, j \in V, k \in K, r \in R$ and $m \in T_{i,j}$. The binary variable $x_{i,j}^{k,r,m}$ indicates whether vehicle trip (k, r) traverses arc (i, j) and departs from node i during arc time zone $T_{i,j}^{m}$. We use continuous variable $t_{i,j}^{k,r,m}$ to represent the departure time from node i if trip (k, r) departs from node i to node j during time zone $T_{i,j}^{m}$. It is set to 0 if trip (k, r) does not depart from node i to node j during time zone $T_{i,j}^{m}$. Similarly, binary variable $y_i^{k,r}$ represents whether trip (k, r) visits node i. We use continuous variable $tt_i^{k,r}$ to represent the departure time from node i if trip (k, r) visits node i, and it is set to 0 otherwise.

The MIP model is as follows:

$$\min \sum_{k\in K} \sum_{r\in R} \sum_{i\in\{0\}\cup V_c} \sum_{j\in\{n+1\}\cup V_c,\, i\neq j} \sum_{m\in T_{i,j}} d_{i,j} x_{i,j}^{k,r,m} \tag{4}$$

$$\text{s.t. } y_0^{k,r} = \sum_{j\in V_c\cup\{n+1\}} \sum_{m\in T_{0,j}} x_{0,j}^{k,r,m} = 1,\ \forall k \in K,\ r \in R, \tag{5}$$

$$y_{n+1}^{k,r} = \sum_{j\in\{0\}\cup V_c} \sum_{m\in T_{j,n+1}} x_{j,n+1}^{k,r,m} = 1,\ \forall k \in K,\ r \in R, \tag{6}$$

$$y_i^{k,r} = \sum_{j\in\{0\}\cup V_c\setminus\{i\}} \sum_{m\in T_{j,i}} x_{j,i}^{k,r,m} = \sum_{j\in V_c\cup\{n+1\}\setminus\{i\}} \sum_{m\in T_{i,j}} x_{i,j}^{k,r,m},\ \ \forall k \in K,\ r \in R,\ i \in V_c, \tag{7}$$

$$\sum_{k\in K} \sum_{r\in R} y_i^{k,r} = 1,\ \forall\ i \in V_c, \tag{8}$$

$$\sum_{i \in V_c} q_i y_i^{k,r} \leq Q, \ \forall k \in K, \ r \in R, \tag{9}$$

$$tt_i^{k,r} = \sum_{j \in V_c \cup \{n+1\} \setminus \{i\}} \sum_{m \in T_{i,j}} t_{i,j}^{k,r,m}, \ \ \forall k \in K, \ r \in R, \ i \in \{0\} \cup V_c, \tag{10}$$

$$tt_{n+1}^{k,r} = \sum_{i \in \{0\} \cup V_c} \sum_{m \in T_{i,n+1}} x_{i,n+1}^{k,r,m} \left\{ t_{i,n+1}^{k,r,m} + \tau_{i,n+1}(t_{i,n+1}^{k,r,m}) \right\}, \ \forall k \in K, \ r \in R, \tag{11}$$

$$\left(t_{i,j}^{k,r,m} + \tau_{i,j}(t_{i,j}^{k,r,m}) + s_j \right) x_{i,j}^{k,r,m} \leq tt_j^{k,r}, \ \forall k \in K, \ r \in R, \ i \in \{0\} \cup V_c, \ j \in \{n+1\} \cup V_c, \ i \neq j, \ m \in T_{i,j}, \tag{12}$$

$$(e_i + s_i)\, y_i^{k,r} \leq tt_i^{k,r} \leq (l_i + s_i)\, y_i^{k,r}, \ \forall k \in K, \ r \in R, \ i \in V, \tag{13}$$

$$w_{i,j}^m x_{i,j}^{k,r,m} \leq t_{i,j}^{k,r,m} \leq w_{i,j}^{m+1} x_{i,j}^{k,r,m}, \ \forall k \in K, \ r \in R, \ i \in \{0\} \cup V_c, \ j \in \{n+1\} \cup V_c, \ i \neq j, \ m \in T_{i,j}, \tag{14}$$

$$tt_0^{k,1} \geq \lambda \sum_{i \in V_c} s_i y_i^{k,1}, \ \forall k \in K, \tag{15}$$

$$tt_0^{k,r} - tt_{n+1}^{k,r-1} \geq \lambda \sum_{i \in V_c} s_i y_i^{k,r}, \ \forall k \in K, \ r \in \{2, 3, \ldots, |R|\}, \tag{16}$$

$$tt_{n+1}^{k,r} - tt_0^{k,r} + \lambda \sum_{i \in V_c} s_i y_i^{k,r} \leq T_{max}, \ \forall k \in K, \ r \in R, \tag{17}$$

$$x_{i,j}^{k,r,m} \in \{0, 1\}, \ \forall i, j \in V, \ k \in K, \ r \in R, \ m \in T_{i,j}^m, \tag{18}$$

$$y_i^{k,r} \in \{0, 1\}, \ \forall i \in V, \ k \in K, \ r \in R, \tag{19}$$

$$t_{i,j}^{k,r,m} \in [e_0, l_0], \ \forall i, j \in V, \ k \in K, \ r \in R, \ m \in T_{i,j}^m, \tag{20}$$

$$tt_i^{k,r} \in [e_0, l_0], \ \forall i \in V, \ k \in K, \ r \in R. \tag{21}$$

The objective (4) is to minimize the total travel distance of all trips. Constraints (5) and (6) ensure that all the trips start from and return to the depot. Constraints (7) represent the flow conservation constraints for all customer nodes for each trip. Constraints (8) guarantee that each customer is served in exactly one trip. Constraints (9) are the capacity constraints for the trips.

Constraints (10) define the departure time of each trip (k, r) from node i, namely $tt_i^{k,r}$, in terms of the decision variables $t_{i,j}^{k,r,m}$. Constraints (11) define the arrival time of each trip (k, r) at the ending depot based on the departure time from the last customer of the trip and the time-dependent travel time function. Constraints (12) are the time-consistent constraints. If arc (i, j) is traversed by trip

(k, r), the corresponding departure time at node j must be greater than or equal to the departure time at node i plus the time-dependent travel time from node i to node j and the service time of node j. Constraints (13) ensure the time window requirement at each node.

Constraints (14) require that if a trip (k, r) leaves from node i to node j within an arc time zone $T^m_{i,j}$, then the departure time $t^{k,r,m}_{i,j}$ must belong to the interval $[w^m_{i,j}, w^{m+1}_{i,j}]$; otherwise, $t^{k,r,m}_{i,j}$ should be set to 0. Constraints (15) and (16) guarantee sufficient loading time before departure for each trip. Constraints (17) enforce the maximum duration for all trips. Constraints (18-21) define the range of the decision variables.

With constraints (14) and the definition of function $\tau_{i,j}(t)$, we have $x^{k,r,m}_{i,n+1} t^{k,r,m}_{i,n+1} = t^{k,r,m}_{i,n+1}$ and $x^{k,r,m}_{i,n+1} \tau_{i,n+1}(t^{k,r,m}_{i,n+1}) = \theta^m_{i,n+1} t^{k,r,m}_{i,n+1} + \eta^m_{i,n+1} x^{k,r,m}_{i,n+1}$, $\forall k \in K, r \in R, i \in \{0\} \cup V_c$. Therefore, constraints (11) can be replaced by the following linear constraints:

$$tt^{k,r}_{n+1} = \sum_{i \in \{0\} \cup V_c} \sum_{m \in T_{i,n+1}} \left\{ \left(1 + \theta^m_{i,n+1}\right) t^{k,r,m}_{i,n+1} + \eta^m_{i,n+1} x^{k,r,m}_{i,n+1} \right\}, \ \forall k \in K, r \in R. \tag{22}$$

Similarly, constraints (12) are not linear but can be linearized with the big-M method in the following manner:

$$\begin{aligned} t^{k,r,m}_{i,j} \left(1 + \theta^m_{i,j}\right) + \left(\eta^m_{i,j} + M_{i,j}\right) x^{k,r,m}_{i,j} - tt^{k,r}_{j} \leq M_{i,j} - s_j, \ \forall k \in K, \\ r \in R, \ i \in \{0\} \cup V_c, \ j \in \{n+1\} \cup V_c, \ i \neq j, \ m \in T_{i,j}, \end{aligned} \tag{23}$$

where $M_{i,j} = l_i + s_i + \tau_{i,j}(l_i + s_i) + s_j$.

4. Segment-based trip evaluation

A key component of algorithms for routing problems is the feasibility evaluation of trips, which is frequently invoked during the search process but can be time-consuming under the time-dependent context. Vidal et al. (2013, 2014) developed a segment-based evaluation scheme for various VRP variants that reduces the complexity of trip evaluation to constant time independent of the length of the trip. Though Zhang et al. (2015a) and Lim et al. (2017) have extended the scheme to two variants of VRPTW with multiple trips, no trivial extension is available for the MT-TDVRPTW. Indeed, as highlighted by Vidal et al. (2014), this scheme is too complicated for the time-dependent

problems as the structure and updating of the segment are very complex. Therefore, we devote this section to discuss an efficient approach to the feasibility evaluation for the MT-TDVRPTW.

4.1. Ready time function and duration function

Dabia et al. (2013) introduced the ready time function and the duration function for a trip originating from the depot. In this section, we generalize the concepts to any segment of consecutive nodes, which is not required to originate from the depot.

Let $\sigma = (\sigma_1, \sigma_2, ...\sigma_L)$ denote a segment of nodes, where $\sigma_1 \in \{0\} \cup V_c$, $\sigma_L \in \{n+1\} \cup V_c$, and $\sigma_i \in V_c$, $\forall\ 1 < i < L$. We define the ready time function $\delta^{\sigma}_{\sigma_i}(t)$ for segment σ and node σ_i as the time when service at node σ_i is completed if the vehicle starts to serve node σ_1 at time t and visits segment σ in order. The required loading time of segment σ is represented as $loadtime(\sigma) = \lambda \sum_{\sigma_i \in \sigma} s_{\sigma_i}$. We enforce that $t \geq max\{e_0 + loadtime(\sigma), e_{\sigma_1}\}$, i.e., the service at the first node σ_1 of the segment can only be started after the loading time and the earliest specified time of node σ_1. Then, the ready time function can be defined recursively as

$$\delta^{\sigma}_{\sigma_i}(t) = \begin{cases} t + s_{\sigma_1}, & \text{if } i = 1; \\ \max\left\{e_{\sigma_i}, \delta^{\sigma}_{\sigma_{i-1}}(t) + \tau_{\sigma_{i-1},\sigma_i}\left(\delta^{\sigma}_{\sigma_{i-1}}(t)\right)\right\} + s_{\sigma_i}, & \text{otherwise.} \end{cases} \tag{24}$$

The duration function of the segment σ can be defined as $\phi_{\sigma}(t) = \delta^{\sigma}_{\sigma_L}(t) - t + loadtime(\sigma)$. A segment is only feasible for a start time t if its duration $\phi_{\sigma}(t)$ is not larger than T_{max}.

Proposition 2. *If the speed profile of the MT-TDVRPTW is a stepwise function, then the following claims hold for any segment σ and node σ_i: 1) the generated ready time function $\delta^{\sigma}_{\sigma_i}(t)$ is a non-decreasing piecewise linear function; 2) the generated duration function $\phi_{\sigma}(t)$ is also a piecewise linear function; and 3) the generated ready time function and duration function share the same set of break points.*

Proof. As the travel time satisfies the FIFO principle, the ready time function is non-decreasing. Next, the sum of two piecewise linear functions, the maximum of a constant value and a piecewise linear function, and the sum of a constant value and a piecewise linear function all result in another piecewise linear function. Therefore, both the ready time function and duration function are

piecewise linear functions. Last, adding a constant value to a piecewise linear function does not change its break points. □

We denote the inverse function of the ready time function as $\pi^{\sigma}_{\sigma_i}(t)$, 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 inverse of the ready time function has the same property as the ready time function and can be defined recursively too.

4.2. Time-dependent functions for concatenation

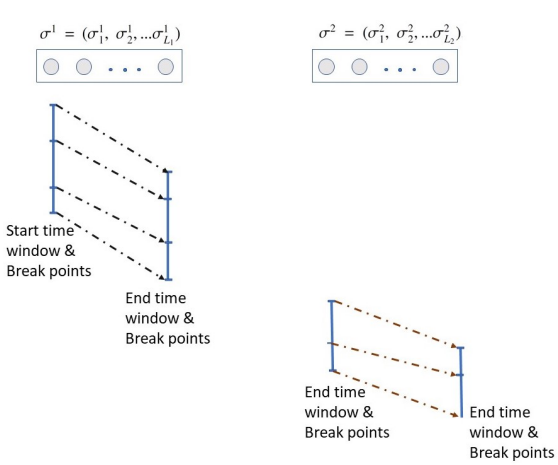

Figure 3: Concatenation 1. Two segments σ^1 and σ^2

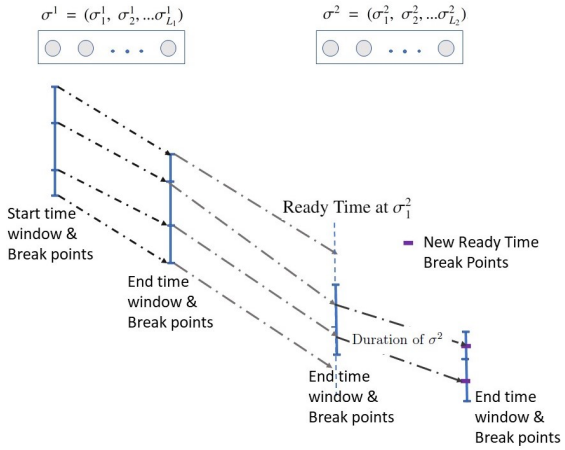

Figure 4: Concatenation 2. Forward calculation of Ready Time from $\sigma^1_{L_1}$ to σ^2_1

The concatenation process for the MT-TDVRPTW is illustrated in Figures 3-6. These four figures delineate the enumeration of all break points for the ready time function and the duration function, which result from the concatenation of two given segments $\sigma^1 = (\sigma^1_1, \sigma^1_2, ...\sigma^1_{L_1})$ and $\sigma^2 = (\sigma^2_1, \sigma^2_2, ...\sigma^2_{L_2})$. First, Figure 3 shows the original segments with their associated break points. Second, σ^1 is extended forward to σ^2 at each of its break points to find the associated ready time at the last node of σ^2 (Figure 4). Furthermore, a break point is dropped if it cannot lead to a feasible solution, i.e., violating the start-time window of σ^2. Similarly, σ^2 is extended backward to σ^1 at each of its break points to find the associated departure time at the first node of σ^1 (Figure 5). Finally, the resulting set of break points determine the piecewise linear ready time and duration functions for the concatenated segment denoted as $\sigma^1 \bigoplus \sigma^2$ (Figure 6).

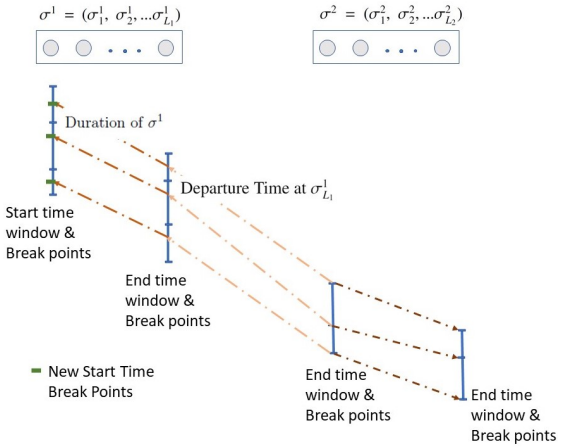

Figure 5: Concatenation 3. Backward calculation of Departure Time from σ^2_1 to $\sigma^1_{L_1}$

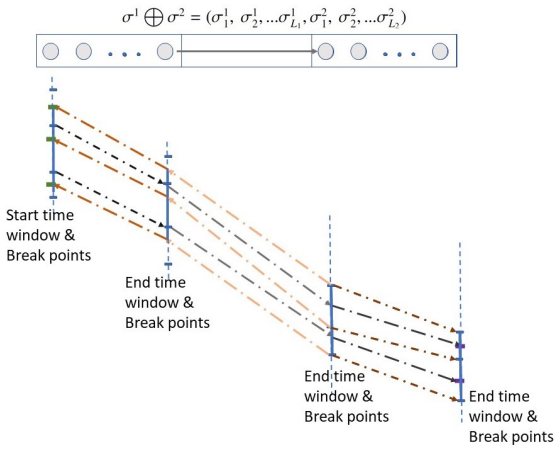

Figure 6: Concatenation 4. Concatenated segment with break points and duration

The feasibility of a given segment with respect to the duration limit can now be determined by inspecting the break points of the segment one by one. If any of the break points yields a duration less than or equal to T_{max}, the segment is feasible (Figure 7). However, the feasible start-time window for a segment may not be continuous because the segment must satisfy the maximum trip duration constraint (Figure 8). Therefore, it is necessary to clearly determine the boundary of the feasible start-time window by finding the time point in which the duration equals to T_{max} exactly. Note that in general, the inverse of the duration function is not well defined as there could be multiple start times with a same duration value. Yet this can be achieved by identifying all the line segments with one feasible and one infeasible end points, and then determining the new break points ($b1$, $b2$, $b3$, $b4$ in Figure 8) within the line segments. In the resulting duration function, the feasible time window is divided into three disconnected regions, i.e., $\{[b_0, b_1], [b_2, b_3], [b_4, b_5]\}$.

The procedure for the concatenation of two segments is summarized in Algorithm 1. The list of break points of the ready time function is defined as $BPS(\sigma)$ for a segment σ. For a segment with a single customer, the size of break points is upper bounded by $|T|$ and it is clear that for any segment σ, $|BPS(\sigma)| \leq |\sigma|T$. Indeed, if the time windows of customers are tight, the size of $BPS(\sigma)$ can be greatly reduced during the concatenation; thus, Algorithm 1 runs in polynomial time. If waiting for the concatenation is inevitable, the feasible time window for the concatenated segment will collapse into a single point in time. It is worthwhile noting that the time complexity

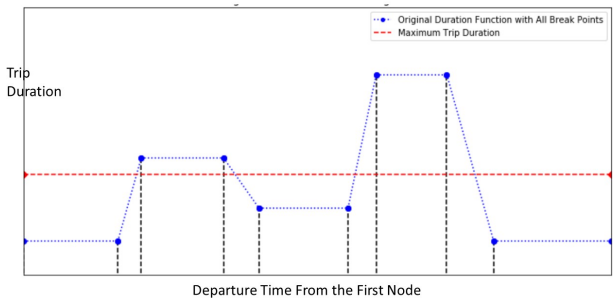

Figure 7: Duration Function

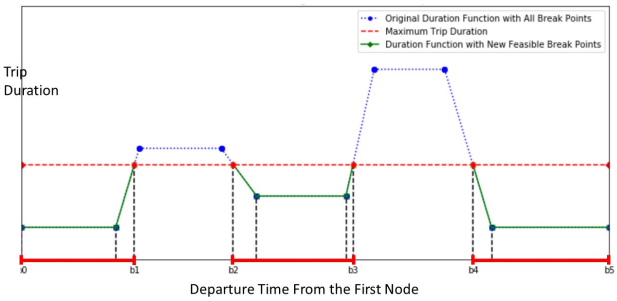

Figure 8: Reduced Duration Function

of concatenation with a segment consisting of only one single break point is constant. Moreover, if the earliest arrival time at σ_1^2 from $\sigma_{L_1}^1$ is greater than the latest feasible start time at σ_1^2 for the segment σ^2, the concatenation is infeasible.

4.3. Updating functions of concatenation

The original segment proposed by Vidal et al. (2013) contains single values for the segment's information, such as the total cost, the total loads, the earliest start time, and the latest start time, etc. For the MT-TDVRPTW, we also capture the cost $C(\sigma)$ and the loads $Q(\sigma)$ for each segment σ. The segment σ with a single customer i is initialized with $C(\sigma) = 0$ and $Q(\sigma) = q_i$. For the concatenated segment $\sigma^1 \bigoplus \sigma^2$, we trivially have $C(\sigma^1 \bigoplus \sigma^2) = C(\sigma^1) + C(\sigma^2) + c_{\sigma_{L_1}^1, \sigma_1^2}$ and $Q(\sigma^1 \bigoplus \sigma^2) = Q(\sigma^1) + Q(\sigma^2)$. The capacity constraint for a segment can then be checked to ascertain whether $Q(\sigma^1 \bigoplus \sigma^2) \leq Q$.

Next, we extend the segment to store the time-dependent ready time and duration functions for each segment, which includes the break points and their associated durations, the slopes and the intersections of the line segments, and the feasibility of the break points. This information is updated based on Algorithm 1.

All the information of the segments of an existing solution can be pre-computed and stored in the memory. As the basic swap and relocate operators involve the recombination of a small number of segments, they can be evaluated in $O(|\sigma||T|)$ time.

Algorithm 1 Calculate Break Points and Durations for Concatenation of two Segments

Break Point Set $B = \{\}$
for each ready time break point t^1 at $\sigma^1_{L_1}$ in set $BPS(\sigma^1)$ **do**
  Extend forward from $\sigma^1_{L_1}$ to σ^2_1 to determine the arrival time t_{arr} at σ^2_1 as $t^1 + \tau_{\sigma^1_{L_1},\sigma^2_1}(t^1)$
  Determine the associated ready time at $\sigma^2_{L_2}$ as $\delta^{\sigma^2}_{\sigma^2_{L_2}}(t_{arr})$
  Calculate the duration from σ^1_1 to $\sigma^2_{L_2}$
  Insert the break point into B
end for
for each ready time break point t^2 at $\sigma^2_{L_2}$ in set $BPS(\sigma^2)$ **do**
  Find the associated start time $\vec{t}^2$ at σ^2_1 for t^2
  Extend backward from σ^2_1 to $\sigma^1_{L_1}$ to obtain the departure time t_{dep} at $\sigma^1_{L_1}$ as $\vec{t}^2 - \tau^{-1}_{\sigma^1_{L_1},\sigma^2_1}(\vec{t}^2)$
  Determine the associated start time at σ^1_1 as $\pi^{\sigma^1}_{\sigma^1_1}(t_{dep})$
  Calculate duration from σ^1_1 to $\sigma^2_{L_2}$
  Insert the break point into B
end for
Remove any break points outside the time windows of σ^1 and σ^2
New Break Point Set $B' = \{\}$
for $i \in \{1, 2, .., |B| - 1\}$ **do**
  if exactly one of B_i and B_{i-1} is feasible **then**
    Find the boundary point $\bar{b}$ and insert into B'
  end if
end for
$B = B \cup B'$, and mark the feasibility of each break point with regard to maximum trip duration limit

5. ALNS-VND meta-heuristic

Hybrid meta-heuristic algorithms have been widely used in routing-related problems (Vidal et al., 2013; Akpinar, 2016; Zhang et al., 2015a; Lim et al., 2017). The variable neighborhood descend (VND) defines a list of neighbourhood structures and performs search in them in a cyclical and systematic way, which enables to escape the local minimum defined in a particular neighbourhood structure. With well-defined neighborhoods, VND is shown to be a good candidate for local search as it can intensify the search efficiently (Zhang et al., 2015b; Lim et al., 2017; Zeng et al., 2016). On the other hand, the adaptive large neighborhood search (ALNS) proposed by Ropke & Pisinger (2006) has gained popularity due to its efficiency and effectiveness in solving several VRPTW variants (Azi et al., 2014; Demir et al., 2012; Gschwind & Drexl, 2019; François et al., 2016,

2019). The ALNS adapts an "ruin-and-recreate" approach with a set of removal operators and repair operators. In each iteration, a pair of removal and repair operators are selected based on their historical performances, through which the ALNS adaptively destroys part of the incumbent solution and recreates a new solution. As the ALNS adaptively perturbs the incumbent solution with a large neighbourhood structure, it can diversify the search whenever the VND procedure is stuck in local optima. Hence, a hybrid ALNS-VND algorithm would be promising to produce high-quality solutions by leveraging the strengths of both. The segment-based evaluation in Section 4 is deployed to accelerate the feasibility checks in both components.

The pseudocode of the algorithm is shown in Algorithm 2. The solution representation is described in Section 5.1, and the construction algorithm is explained in Section 5.2. In the main ALNS loop, a new solution is first obtained by the removal and repair operators (Section 5.4), and then improved by two VND operators (Section 5.3). The new solution is accepted if its cost is better than the cost of the incumbent solution. The probability of removal and repair operators are updated based on the quality of the solutions obtained. This process is repeated until the stopping criteria are met.

Algorithm 2 ALNS-VND

```
Construct the initial solution S
S_best = S, NonImp = 0
while termination conditions are not met do
    Select removal & repair operators adaptively
    S' = RemovalAndRepair(S)
    S' = VND(S')
    if cost(S') < cost(S_best) then
        NonImp = 0, S_best = S'
    else
        NonImp = NonImp + 1
    end if
    if cost(S') < cost(S) then
        S = S'
    end if
    Update scores and probability for removal and repair operators
end while
return S_best
```

5.1. Solution representation and evaluation

A trip is the fundamental building block to the solution representation, which can be evaluated efficiently using the segment-based evaluation method (Section 4.2). Each vehicle performs a sequence of trips. The feasibility of a vehicle's assignment can be checked by scanning the trips in sequence to find a feasible schedule. This can be done in $O(|R|)$ time as the information has been pre-computed for all the trips. A solution is feasible if all trips are feasible and each vehicle has a feasible schedule for its assigned trips. An example of an MT-TDVRPTW solution is depicted in Figure 9.

V1	**[0, 5, 14, 15, 2, 16] , [0, 11, 7, 8, 6, 13, 4, 16]**
V2	**[0, 12, 9, 3, 10, 1, 16]**

Figure 9: Example of a multi-trip solution with 15 customers and 2 vehicles

5.2. Construction algorithm

The initial solution is constructed through a look-ahead approach named regret insert (RI), which has been proven to be very effective in problems with tight constraints (Ropke & Pisinger, 2006; Zhang et al., 2015a; Lim et al., 2017). Each vehicle is initialized with an empty trip, and then the customers are iteratively selected and inserted to the best place according to the following criterion. Let $\Delta C_{i,r}$ be the lowest additional cost to insert a customer i into a trip r, and $\Delta C_{i,r}$ is set to $+\infty$ if customer i cannot be inserted into the trip r. A greedy insertion algorithm always searches for the candidate customer i and trip r with the lowest insertion cost $\Delta C_{i,r}$. Conversely, the RI method chooses customer i with the largest *regret value* $RV_i = \sum_{j=1}^{k}(\Delta C_{i,r_j} - \Delta C_{i,r_1})$, where r_j is increasingly sorted by the additional cost of inserting customer i to route r_j. Note that RI always maintains an empty trip at the end of each vehicle. Besides, the parameter k in RV_i can take different values to increase the number of steps to look-ahead, thus the corresponding RI method is denoted as *reg-k*. In our implementation, k is randomly chosen as 1, 2, or 3 to increase the randomness of the initial solution. Note that the *reg-k* method becomes the greedy insertion when $k = 1$. Furthermore, the *reg-k* methods are also used in the VND operator (Section 5.3) and the repair stage of the ALNS algorithm (Section 5.4).

5.3. VND local search

We propose two types of moving operators for local search, which are used repeatedly one after another until no further improvement is obtained.

First, we extend the traditional relocation and swap of customer(s) to the multiple trip context. We adopt these two operators in three scenarios: intra-trip, inter-trip yet within the same vehicle, and inter-trip across two different vehicles. If the resulted trip is feasible, but there is no feasible schedule for the vehicle, the local search operator will relocate the trip to another position within the same vehicle if possible. If such attempt is unsuccessful, it will further attempt to shift the trip to another vehicle. The relocation of a trip will be made only if the resulted solution is feasible. Thus, the operators can explore a larger solution space.

However, traditional relocation and swap operators can still be trapped in sub-optimal solutions. We develop a second type of operator by enlarging the neighborhood through small scale removal and repair. A new solution is accepted only if it is better than the incumbent solution. The removal options include the removal of one trip from a vehicle, the removal of two consecutive trips from a vehicle, and removal of one trip from each of the two vehicles selected. The *reg-k* methods are randomly chosen to repair the solution by reinserting all removed customers one by one.

5.4. ALNS

We adopt the ALNS framework from Ropke & Pisinger (2006) with customization for the MT-TDVRPTW.

5.4.1. Removal and repair operators

The key idea of ALNS is to relocate misplaced customers to a better position in the solution to reduce the total travel cost (Ropke & Pisinger, 2006). Therefore, two factors considerably affect its effectiveness in a highly constrained problem like the MT-TDVRPTW: the closeness of the removed customers to each other and the gap created by the removal of customers within the trip. In this paper, we design a new *closeness measure* in the time-dependent context. For a pair of customers i and i', the *closeness measure* is defined as $CL(i,i') = \bar{t}_{i,i'} + \gamma_{wt} \max\{0, e_{i'} - l_i - s_i - \tau_{i,i'}(l_i + s_i)\} + \gamma_{tw} \max\{0, e_i + s_i + \tau_{i,i'}(e_i + s_i) - l_{i'}\}$, where $\bar{t}_{i,i'}$ is the average time needed to travel from customer

i to customer i' during the feasible time window; γ_{wt} is the penalty parameter for waiting time at customer i' when the vehicle leaves customer i at the latest possible time $l_i + s_i$; and γ_{tw} is the penalty parameter for violation of time window at customer i' even if the vehicle leaves from customer i at the earliest possible time $e_i + s_i$. If the path from customer i to customer i' is not feasible, $\bar{t}_{i,i'}$ and $CL(i,\ i')$ will be set as $+\infty$. Additionally, we remove one adjacent customer within the trip during the removal process to enlarge the gap and increase the chance of inserting new customers to fill the gap. We design two removal lists to differentiate between these two types of removal: 1) the first list contains customers selected based on the closeness measure, and 2) the second list stores the neighboring customers removed. The customers in both lists will be removed from the incumbent solution; however, only customers from the first list will be used to search for the next customer for removal.

We propose five removal operators: 1) random customer selection where customers are selected randomly; 2) related customer selection where two initial customers are randomly selected, and other customers are selected with probability based on the new closeness measure and the two removal lists are used; 3) trip effectiveness-based related customer selection which is similar to the second operator except for the way that initial customers are selected. A trip is chosen and removed with probability based on its effectiveness in meeting the demands of the customers over its incurred cost; 4) mixed related customer selection which uses both the second and third operators to select the initial customers; 5) worst customer selection which adopts a similar idea introduced by Ropke & Pisinger (2006) to select customers with probability based on the insertion cost of the customer.

The *reg-k* methods are used to reinsert the removed customers back to the solution where k is selected from $\{1, 2, 3\}$.

5.4.2. Roulette wheel selection

The removal and repair operators are selected using the roulette wheel selection approach. Each operator j in the operator set O is given a dynamic weight of w_j based on its historical performance. The probability of selecting operator j is $w_j / \sum_{i \in O} w_i$. In the initial stage, all operators have equal weights and equal probabilities of selection. For each ALNS iteration, an integer score of ρ_1 or ρ_2 is awarded to the selected operator if it yields a new global best solution or a better solution than

the incumbent solution respectively, and no score would be awarded otherwise. The entire search is then broken down into phases of 100 iterations, during which the weights of operators are not changed. Let $w_{j,p}$ be the weight of operator j for phase p. Then at the end of the p-th phase, we update the weight of operator j for phase $p+1$ as follows:

$$w_{j,p+1} = (1-r)w_{j,p} + r\frac{\pi_{j,p}}{max\{1,\ \theta_{j,p}\}}, \tag{25}$$

where $\pi_{j,p}$ is the sum of score for operator j in the phase p, $\theta_{j,p}$ is the number of times operator j is invoked, and $r \in (0, 1)$ is the reaction rate to control the speed of weight adjustment. The larger the value of r is, the faster the weights are changed based on the latest score.

5.4.3. Dynamic perturbation strength

The strength of perturbation in ALNS is defined as the number of customers removed from the incumbent solution. When the VND local search and ALNS perturbation fail to find a better solution, it is desirable to increase the strength of the perturbation gradually. Therefore, we maintain a counter $NonImp$, which is increased by 1 if the perturbation fails to find a better global solution and is reset to 0 otherwise. At the beginning of the algorithm, we set the number of requests to be removed as $\eta_{min} = \alpha n$, where α is the factor for the minimum number of customers to be removed. Subsequently, the number of requests $\eta_{strenth}$ is updated dynamically as $\min\{0.5,\ \alpha + 0.005NonImp\} \times n$.

6. Computational experiments

This section presents the test instances, configuration of the algorithm, and computational results on MT-TDVRPTW and MT-VRPTW. Our algorithm is coded in Java and the MIP model is solved with IBM ILOG CPLEX 12.8.0 (IBM CPLEX, 2017). All experiments are run in an Ubuntu 18.04.3 LTS server with Intel(R) Xeon(R) Silver 4216 CPU of 2.10 GHz. All test instances and the detailed routing plans are available online at `http://www.computational-logistics.org/orlib/MTTDVRPTW`.

6.1. Test instances for MT-TDVRPTW

The test instances are derived based on the TDVRPTW instances proposed by Dabia et al. (2013), who extended Solomon's benchmark data with time-dependent travel time information in a similar way of Ichoua et al. (2003). The TDVRPTW instances randomly assign each arc with one of three speed profiles, representing heavy, medium, and light traffic conditions respectively. The workday $[e_0, l_0]$ is divided into five time zones of varied width. For the MT-TDVRPTW, we only use the C2, R2, and RC2 instances as in the MT-VRPTW (Azi et al., 2010; Macedo et al., 2011; Hernandez et al., 2014; Lim et al., 2017), since type-1 instances limit the number of possible trips performed by a vehicle due to their narrow time windows.

The newly generated instances are grouped with the naming convention of "T$n - |T| - |R|$", where n is the number of customers, $|T|$ denotes the number of time zones, and $|R|$ indicates the number of allowed trips per vehicle. As shown in Table 1, instances with 15, 50, and 100 customers and 2, 3, and 5 vehicles are generated respectively. Two values are given for T_{max} because C2 instances have a longer service time and require a higher T_{max} than R2 and RC2.

Table 1: MT-TDVRPTW test instance groups

Group	**n**	**|T|**	**|R|**	T_{max}	**|K|**	MAX_{RT}
T15-5-2	15	5	2	550/1500	2	600
T15-11-2	15	11	2	550/1500	2	600
T50-5-5	50	5	5	300/750	3	1800
T50-7-5	50	**7**	5	300/750	3	1800
T50-11-5	50	**11**	5	300/750	3	1800
T50-5-10	50	5	**10**	**250/550**	3	1800
T100-5-5	100	5	5	300/750	5	3600

The last column (MAX_{RT}) is the maximum runtime allowed for the ALNS-VND algorithm. It may terminate earlier if no improving solutions are obtained for consecutive ωn iterations, where the parameter ω is determined during the parameter tuning stage. The ALNS-VND is run 10 times for each instance with different random seeds. Additionally, the exact solver is given a maximum runtime of 7200 seconds for groups T15-5-2 and T15-11-2.

Table 2: Numerical parameter list

Name	Type	Range	Description
γ_{wt}	Real	[0.1, 1]	Penalty parameter for waiting time at next customer i' when the vehicle leaves previous customer i at the latest possible time $l_i + s_i$
γ_{tw}	Real	[0.1, 2]	Penalty parameter for violation of time window at next customer i' even if the vehicle leaves from previous customer i at the earliest possible time $e_i + s_i$
α	Real	[0.2, 0.35]	Factor for minimum number of customers to be removed in ALNS
ω	Integer	[40, 100]	Factor for maximum number of non-improving iterations allowed
ρ_1	Integer	[25, 40]	Score of finding a better global solution in an ALNS iteration
ρ_2	Integer	[5, 20]	Score of finding a better solution than the incumbent solution in an ALNS iteration
r	Real	[0.85, 0.99]	The reaction rate to control the speed of weight adjustment in wheel roulette

6.2. Automatic configuration

The numerical parameters of the ALNS-VND algorithm listed in Table 2 are tuned with an automatic algorithm configuration tool, the IRACE package (López-Ibáñez et al., 2016). A configuration in IRACE refers to a set of values assigned to the algorithmic parameters. IRACE is an implementation of the iterated racing, which consists of three iterative steps: (1) sampling new configurations based on the current sampling distribution, (2) evaluating and selecting the best configurations through racing, and (3) updating the sampling distribution to bias towards the best configurations. The IRACE package takes in the algorithm itself, a set of parameters to be tuned,and a set of training instances as input. A training budget is set to control the maximum number of configuration runs allowed, which is set to 10,000 in this experiment. For each configuration run, the algorithm is applied to one of the given training instances with a maximum run time of 600 seconds and returns the best solution cost found. IRACE is executed in parallel mode with a maximum of 100 concurrent runs. Upon termination of the configuration runs, IRACE returns a set of elite configurations ordered from best to worst.

As in François et al. (2016, 2019), IRACE is used to select algorithmic design options, for example, which removal and repair operators are effective and should be included in the ALNS-VND algorithm. New boolean parameters are introduced to turn the operators on or off, which is then tuned by IRACE together with the numerical parameters.

Table 3 shows the values of all the algorithmic parameters of the 6 elite configurations returned by the IRACE package. For the design options, "1" indicates that IRACE chooses to turn on the design option, and "0" indicates otherwise. It is interesting to note that the related customer removal

Table 3: Elite configurations by rank of performance

	Configuration #	#1	#2	#3	#4	#5	#6
Numerical							
	γ_{wt}	0.26	0.62	0.32	0.35	0.19	0.25
	γ_{tw}	1.02	0.41	1.08	0.97	1.17	1
	α	0.32	0.24	0.33	0.32	0.33	0.35
	ω	79	61	89	99	99	86
	ρ_1	30	36	31	33	33	30
	ρ_2	9	14	6	7	6	7
	r	0.92	0.91	0.9	0.9	0.91	0.92
Removal							
	Random customer	1	1	1	1	1	1
	Related customer	0	1	0	0	0	0
	Trip effectiveness based	1	1	1	1	1	1
	Mixed trip and customer	1	0	1	1	1	1
	Worst customer	1	0	1	1	1	1
Repair							
	reg-1 (Greedy)	0	1	0	0	0	0
	reg-2	1	1	1	1	1	1
	reg-3	1	1	1	1	1	1

operator and the greedy *reg*-1 repair operator are only turned on by IRACE once, while the other operators are turned on in almost all the elite configurations. The possible explanation is that the related customer removal operator is very similar to the mixed trip and customer removal operator, except for how the initial customers are selected. Hence, its contributions in practice could be overshadowed. On the other hand, the greedy *reg*-1 repair operator is ineffective compared to the *reg*-2 and *reg*-3 operators for MT-TDVRPTW with tight constraints.

In the subsequent experiments, the ALNS-VND adopts the best configuration (#1) with the highest score among the elite configurations as the actual parameter values.

6.3. Results and analysis on the MT-TDVRPTW instances

The main computational results are covered in this section. We first evaluate the correctness of the proposed algorithm in 6.3.1 by comparing results obtained by CPLEX on small instances and then provide benchmark results for the medium and large scale instances in 6.3.2. We then study the performances of the algorithm under a larger $|T|$ in 6.3.3 and evaluate the impact of a smaller T_{max} on solutions in 6.3.4.

6.3.1. Results on the small instances

The small instances of 15 customers are solved with both the ALNS-VND and the CPLEX optimizer to evaluate the correctness of the ALNS-VND algorithm. The detailed results are reported in Table 4. For the CPLEX optimizer, we report whether the optimal value is found, the best solution cost found (C_{CPLEX}), and the run time required (T_{CPLEX}). For the ALNS-VND, the table shows the best solution cost (C_{bst}), the average best solution cost (C_{avg}) among 10 runs, and the fastest time to find the best solution (T_{bst}). According to the results, only 12 instances in group T15-11-2 are solved to optimality within 7200 seconds, compared to 23 instances in T15-5-2. This demonstrates the growing complexity of the problem with the increased number of time zones. However, the ALNS-VND can obtain the optimal or best solutions for all instances in short run times, and even better solutions for three instances in group T15-11-2.

Table 4: IBM CPLEX Optimizer vs ALNS-VND with 15 customers

	T15-5-2						**T15-11-2**					
	IBM CPLEX			**ALNS-VND**			**IBM CPLEX**			**ALNS-VND**		
Inst.	Optimal?	C_{CPLEX}	T_{CPLEX}	C_{bst}	C_{avg}	T_{bst}	Optimal?	C_{CPLEX}	T_{CPLEX}	C_{bst}	C_{avg}	T_{bst}
C201	Y	211.75	0.7	211.75	211.75	0.1	Y	211.75	0.9	211.75	211.75	0.2
C202	Y	203.07	409.9	203.07	203.07	1.3	**N**	203.07	7206.5	203.07	203.07	2.4
C203	Y	203.07	2773.7	203.07	203.07	2.1	**N**	203.07	7214.8	203.07	203.07	4.7
C204	Y	188.68	4127.8	188.68	188.68	5.2	**N**	188.68	7218.9	188.68	188.68	12.5
C205	Y	208.79	8.3	208.79	208.79	0.2	Y	208.79	19.6	208.79	208.79	0.3
C206	Y	208.79	14.7	208.79	208.79	0.3	Y	208.79	239.2	208.79	208.79	0.5
C207	Y	204.85	161.6	204.85	204.85	0.6	Y	204.85	1050.9	204.85	204.85	1.0
C208	Y	203.07	86.3	203.07	203.07	0.6	Y	203.07	1094.7	203.07	203.07	1.1
R201	Y	343.68	5.9	343.68	343.68	1.0	Y	341.03	15.0	341.03	341.03	2.7
R202	Y	297.80	1133.7	297.80	297.80	3.6	**N**	297.52	7208.0	297.52	297.52	5.2
R203	Y	297.52	2030.4	297.52	297.52	2.0	**N**	297.52	7213.0	297.52	297.55	4.1
R204	Y	266.59	1785.8	266.59	266.59	21.7	**N**	266.59	7212.4	266.59	266.59	59.6
R205	Y	278.08	14.9	278.08	278.08	0.9	Y	278.08	31.0	278.08	278.08	1.3
R206	Y	248.91	70.6	248.91	248.91	3.0	Y	248.91	5465.3	248.91	248.91	7.4
R207	Y	248.91	196.8	248.91	248.91	3.6	**N**	248.91	7209.3	248.91	248.91	7.9
R208	Y	238.23	1834.0	238.23	238.23	4.8	**N**	238.23	7214.8	238.23	238.23	14.2
R209	Y	263.48	36.4	263.48	263.48	12.5	Y	263.48	127.4	263.48	263.48	9.9
R210	Y	292.15	273.9	292.15	292.15	22.1	**N**	292.15	7214.8	292.15	293.23	45.1
R211	Y	247.23	96.0	247.23	247.23	1.5	**N**	**261.16**	**7215.5**	**256.66**	**257.37**	**28.8**
RC201	Y	313.08	18.7	313.08	313.08	0.3	Y	288.84	22.0	288.84	288.84	0.5
RC202	**N**	235.37	7210.9	235.37	235.37	1.4	**N**	235.37	7211.0	235.37	235.37	4.5
RC203	**N**	198.64	7210.3	198.64	198.64	1.5	**N**	198.64	7216.5	198.64	198.64	5.2
RC204	**N**	192.18	7212.0	192.18	192.18	27.6	**N**	-	**7200.0**	**192.18**	**192.38**	**86.8**
RC205	Y	236.69	140.6	236.69	236.69	0.6	Y	200.76	44.8	200.76	200.76	1.8
RC206	Y	204.22	80.7	204.22	204.22	0.8	Y	204.22	580.3	204.22	204.22	1.1
RC207	Y	189.06	278.5	189.06	189.06	2.6	**N**	189.06	7210.6	189.06	189.06	9.1
RC208	**N**	157.01	7200.0	157.01	157.01	2.4	**N**	**161.51**	**7200.0**	**158.53**	**159.59**	**44.4**

6.3.2. Results on the medium and large scale instances

The time zones of instances in groups T50-5-5 and T100-5-5 are the same as TDVRPTW instances Dabia et al. (2013). They are solved by the ALNS-VND algorithm and the results are presented in Table 5. With the doubling of the customer size, the T100-5-5 instances become much more difficult to solve and require on average 2.7 times T_{bst} as T50-5-5 does. Additionally, the solutions are more diverse as the average standard deviation (S.D) of solution costs increases significantly from 11.8 to 36.8. As there are no existing benchmark results on the MT-TDVRPPTW, these results can serve as benchmark for future studies.

Table 5: ALNS-VND on T50-5-5 and T100-5-5

	T50-5-5				**T100-5-5**			
Inst.	C_{bst}	C_{avg}	S.D	T_{bst}	C_{bst}	C_{avg}	S.D	T_{bst}
C201	779.72	780.67	1.89	90.4	1562.79	1581.08	21.76	3553.8
C202	716.39	720.78	5.16	327.9	1541.90	1563.86	17.21	2927.3
C203	697.39	709.55	6.56	694.5	1502.68	1555.95	29.61	3129.8
C204	653.84	661.46	5.81	1712.6	1488.92	1532.59	32.39	2424.7
C205	748.32	754.72	3.12	676.4	1515.94	1578.57	39.83	2768.2
C206	731.44	738.72	5.42	1461.9	1510.00	1545.03	20.06	3180.6
C207	709.39	718.89	7.10	1005.9	1504.16	1522.65	17.55	2575.0
C208	670.07	675.87	5.49	247.9	1481.65	1507.64	22.11	2474.1
R201	998.81	1009.50	8.42	852.5	1478.71	1506.99	27.85	3478.4
R202	861.81	875.56	10.99	677.5	1296.00	1325.08	21.11	3093.5
R203	748.29	764.32	8.88	1384.5	1119.34	1144.87	18.19	2829.1
R204	580.01	592.31	8.08	1692.7	956.21	1002.44	54.92	2395.4
R205	783.46	798.14	9.02	597.7	1194.20	1240.74	40.12	3165.5
R206	713.36	727.55	9.09	1396.4	1115.28	1162.80	36.85	2842.9
R207	644.97	677.68	21.55	1678.0	1052.67	1109.96	27.96	1944.8
R208	550.88	580.10	20.56	1144.5	956.71	1006.53	31.37	2855.7
R209	688.03	710.21	16.07	1699.5	1072.51	1114.54	42.90	3469.9
R210	742.39	766.37	12.92	1124.4	1134.07	1158.22	23.15	1886.7
R211	613.05	635.86	21.22	1407.9	979.90	1070.49	64.64	2810.7
RC201	1060.56	1076.83	15.92	565.5	1713.99	1776.96	49.23	2847.9
RC202	982.64	991.03	10.13	568.2	1500.71	1534.27	36.58	2987.1
RC203	819.38	832.79	17.22	882.3	1330.17	1375.18	40.94	1935.5
RC204	604.62	621.38	28.51	452.2	1048.58	1141.94	76.50	939.6
RC205	1049.40	1082.68	18.98	868.9	1595.45	1655.52	44.96	2161.4
RC206	817.59	818.23	0.70	1096.6	1378.03	1435.29	34.40	2212.5
RC207	701.15	737.43	16.41	1617.8	1241.56	1304.69	38.91	3368.9
RC208	566.15	617.93	27.95	1155.9	1079.17	1206.22	81.18	1652.6

6.3.3. Analysis on instances with more time zones

In real life applications, the travel speeds change more frequently and rapidly throughout the day, which might make the instances more difficult to solve. It is therefore interesting to evaluate the performance of the proposed algorithm on instances with larger value of $|T|$. The three groups (T15-11-2, T50-7-5, and T50-11-5) are served for this purpose where all the time zones are set to have equal widths.

The results of group T15-11-2 in Table 4 already demonstrate the significant growing of complexity for commercial solvers, while medium instances in groups T50-7-5 and T50-11-5 are used to investigate the impact on the proposed ALNS-VND algorithm. The comparison results with base group T50-5-5 are summarized in Table 6. Generally speaking, the instances with more time zones lead to longer travel distance and computational times, and larger standard deviation among solutions. However, the trend is not stable. For example, the S.D of RC2 in T50-7-5 is the smallest among the three instance groups, and the average C_{bst} for T50-7-5 is the highest among the three instance groups across C2, R2, and RC2. Those suggest that the solutions found for T50-7-5 may deviate far away from the solutions found for T50-5-5 and T50-11-5.

Table 6: Impact of increased number of time zones on test instances with 50 customers

Group	Set	C_{bst}	S.D	T_{bst}
T50-5-5	C2	713.32	5.07	777.2
	R2	720.46	13.35	1241.4
	RC2	825.19	16.98	900.9
T50-7-5	C2	723.87	5.20	883.4
	R2	739.43	15.00	1071.0
	RC2	868.95	13.85	1299.0
T50-11-5	C2	710.36	8.33	1119.0
	R2	734.08	14.16	1148.1
	RC2	851.13	26.80	1275.7

6.3.4. Impact of a smaller T_{max}

The value of T_{max} places an important restriction on the feasibility of a single trip. A smaller T_{max} may increase the number of trips performed by a vehicle and then the complexity of multi-trip

scheduling, which makes the problem even tougher to solve. In this section, we analyze the impact of a smaller T_{max} through the group T50-5-10.

Table 7 presents the comparison between the results of T50-5-5 and T50-5-10. First, it shows that the smaller T_{max} affects C2 more significantly than R2 and RC2 in terms of the average cost C_{bst} and the average number of trips per vehicle. The average cost C_{bst} for C2 increases by more than 25% to 896.31 and the average number of trips per vehicle increases by 38%, while the increases for the same measures are less significant for R2 and RC2. The observed increases could be attributed to the fact that customers are located in clusters in C2. When T_{max} is larger as in T50-5-5, a vehicle could potentially serve more clustered customers in a single trip, and hence reduce the total travel distance, and the number of trips required to serve all customers. Second, the average T_{bst} for R2 is reduced, while it has increased for C2 and RC2. It shows that the ALNS-VND does not necessarily require longer run times with a smaller T_{max}. Third, the average S.Ds of the best solution costs for C2 and R2 increase slightly, while the value for RC2 is reduced. Lastly, we compare the ratio of the average trip duration against the maximum trip duration, which is reported in the column "Util(%)". The results show that the utilization rates have indeed increased for T50-5-10.

Table 7: Impact of reduced maximum trip duration on test instances with 50 customers

Group	Set	C_{bst}	S.D	T_{bst}	N_{trips}	Util(%)
T50-5-5	C2	713.32	5.07	777.2	3.17	87.58
	R2	720.46	13.35	1241.4	2.01	85.91
	RC2	825.19	16.98	900.9	2.04	86.56
T50-5-10	C2	896.31	5.64	891.1	4.38	88.59
	R2	758.34	13.39	1173.6	2.28	87.51
	RC2	895.99	12.50	1306.7	2.43	87.35

6.4. Comparison on MT-VRPTW instances

To further evaluate the performance of the ALNS-VND algorithm, we modify it to solve the MT-VRPTW, which can be viewed as a special case of the MT-TDVRPTW problem without the time-dependent information. Its performance is then compared with the combined iterated local search and VND (ILS-VND) algorithm in Lim et al. (2017). To maintain consistency with the notation used for the MT-VRPTW, we change the maximum trip duration limit to the goods travel

duration (T_{GT}) for MT-VRPTW, which is defined as the arrival time at the last customer in the route minus the departure time from the depot. We solve four groups of MT-VRPTW instances, namely, M1 and M2 with 25 customers and M3 and M4 with 40 customers (Table 8). M2 and M4 differ from M1 and M3 with a larger T_{GT}, which enlarges the solution spaces.

Table 8: MT-VRPTW test instance groups

Group	**n**	**|K|**	**|R|**	T_{GT}	MAX_{RT}
M1	25	2	10	75/220	1800
M2	25	2	10	100/250	1800
M3	40	2	10	75/220	1800
M4	40	2	10	100/250	1800

Table 9 summarizes the number of the Same or Better Solutions (SBS) found with both algorithms, the average of the best solution costs (Avg C_{bst}), the average of the solution costs found over 10 runs (Avg $Cost$), and the average of the best run times to find the best solutions (Avg T_{bst}). The details of each instance are reported in the supplemental material. For M1 and M2 with 25 customers, the ALNS-VND performs very well as it finds the same C_{bst} for all the 27 test instances with a reasonable computational time compared to those required by (Lim et al., 2017). The ALNS-VND finds the same C_{bst} for each of the 10 runs for 24 of the test instances in M1 and 23 of the test instances in M2. For the other 7 instances, the ALNS-VND finds sub-optimal solutions with slightly higher costs in some of the 10 runs (See Supplemental Material). Due to the increased number of customers in M3 and M4, it is impossible to serve all customers in some instances. Therefore, these test instances are excluded from the comparisons. Overall, we use 15 instances in M3 and 20 instances in M4 for evaluation. Generally, the performances of the ALNS-VND on M3 and M4 instances deviate slightly from the previous algorithm. The ALNS-VND performs better for M4 than for M3 when compared to the results reported by Lim et al. (2017): 1) it finds better solutions for 4 test instances in M4, and 2) it yields better average C_{bst} than Lim et al. (2017) for C2 and RC2 in M4. One possible reason is that the ILS-VND algorithm in Lim et al. (2017) is designed specifically to search both feasible and infeasible solution spaces and normally performs better for problems with very tight constraints, such as M3.

In summary, although the ALNS-VND algorithm is not specifically designed for the MT-

VRPTW, the experiment confirms its correctness and competitiveness in solving the MT-VRPTW instances.

Table 9: ALNS-VND (AV) vs Lim et al. (2017)(Lim) on MT-VRPTW

			SBS		**Avg C_{bst}**			**Avg $Cost$**			**Avg T_{bst}**	
Group	Set	#Inst.	AV	Lim	AV	Lim	Gap	AV	Lim	Gap	AV	Lim
M1	C2	8	8	8	631.43	631.43	-	631.59	631.43	0.03%	25.0	42.6
	R2	11	11	11	620.13	620.13	-	620.13	620.13	-	6.1	13.2
	RC2	8	8	8	804.07	804.07	-	804.15	804.07	0.01%	87.9	51.2
M2	C2	8	8	8	530.27	530.27	-	530.27	530.27	-	1.5	1.6
	R2	11	11	11	544.99	544.99	-	545.27	544.99	0.05%	10.0	28.0
	RC2	8	8	8	629.29	629.29	-	629.52	629.29	0.04%	5.8	7.6
M3	C2	7	6	7	1076.74	1076.14	0.06%	1082.48	1077.58	0.45%	617.4	677.8
	R2	8	6	8	912.23	910.69	0.17%	919.10	915.14	0.43%	722.2	820.6
	RC2	-	-	-	-	-	-	-	-	-	-	-
M4	**C2**	**8**	**8**	**6**	**922.18**	**922.43**	**-0.03%**	924.54	922.94	0.17%	414.7	680.1
	R2	10	6	9	798.09	797.41	0.09%	801.76	798.88	0.36%	611.1	424.6
	RC2	**2**	**2**	**1**	**958.89**	**960.43**	**-0.16%**	996.10	975.86	2.07%	798.5	753.7

7. Conclusion

This paper has investigated a new and important VRP variant in city logistics that simultaneously considers the time-dependent travel time, multiple trips, time windows, and maximum trip duration constraint together. This problem bears significant and practical value due to its realistic modelling of city logistics. We first presented an MIP model and modelled the time-dependent travel time, ready time, and duration functions explicitly as piecewise linear functions. More importantly, we developed the segment-based evaluation scheme to the time-dependent setting, which can dramatically accelerate the meta-heuristic algorithm. Subsequently, we proposed the ALNS-VND to solve the problem and the related MT-VRPTW problem. Extensive experiments were conducted to demonstrate the performance of the proposed algorithm.

The characteristics of multiple trips and time-dependent travel time usually cannot be ignored in the practical applications of city logistics. The existing algorithms for many kinds of VRP variants cannot be trivially extended to deal with these considerations. Thus, future research efforts should aim to design specially tailored algorithms. We believe that our work will serve to raise awareness and attract attention to this area. Moreover, further research should be undertaken to

develop improved meta-heuristic algorithm, potential decomposition methods and branch-and-price algorithm for the MT-TDVRPTW.

Acknowledgements

This research was partially supported by NRF Singapore [Grant NRFRSS2016-004], and MOE -AcRF-Tier 1 [Grants R-266-000-096-133 , R-266-000-096-731 , and R-266-000-100-646] and Tier 2 [Grant MOE2017-T2-2-153].

References

Akpinar, S. (2016). Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem. *Expert Systems with Applications*, *61*, 28–38.

Alonso, F., Alvarez, M. J., & Beasley, J. E. (2008). A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. *Journal of the Operational Research Society*, *59*, 963–976.

Azi, N., Gendreau, M., & Potvin, J.-Y. (2007). An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. *European Journal of Operational Research*, *178*, 755–766.

Azi, N., Gendreau, M., & Potvin, J.-Y. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, *202*, 756–763.

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

Balseiro, S., Loiseau, I., & Ramonet, J. (2011). An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows. *Computers & Operations Research*, *38*, 954–966.

Battarra, M., Monaci, M., & Vigo, D. (2009). An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers & Operations Research*, *36*, 3041–3050.

Brandao, J., & Mercer, A. (1997). A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European Journal of Operational Research*, *100*, 180–191.

Cattaruzza, D., Absi, N., & Feillet, D. (2016). The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science*, *50*, 676–693.

Cattaruzza, D., Absi, N., & Feillet, D. (2018). Vehicle routing problems with multiple trips. *Annals of Operations Research*, *271*, 127–159.

Cattaruzza, D., Absi, N., Feillet, D., & Vidal, T. (2014). A memetic algorithm for the multi trip vehicle routing problem. *European Journal of Operational Research*, *236*, 833–848.

Dabia, S., Ropke, S., Van Woensel, T., & De Kok, T. (2013). Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science*, *47*, 380–396.

Demir, E., Bektaş, T., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, *223*, 346–359.

Donati, A. V., Montemanni, R., Casagrande, N., Rizzoli, A. E., & Gambardella, L. M. (2008). Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research*, *185*, 1174–1191.

Figliozzi, M. A. (2012). The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E: Logistics and Transportation Review*, *48*, 616–636.

François, V., Arda, Y., & Crama, Y. (2019). Adaptive large neighborhood search for multitrip vehicle routing with time windows. *Transportation Science*, *53*, 1706–1730.

François, V., Arda, Y., Crama, Y., & Laporte, G. (2016). Large neighborhood search for multi-trip vehicle routing. *European Journal of Operational Research*, *255*, 422–441.

Gendreau, M., Ghiani, G., & Guerriero, E. (2015). Time-dependent routing problems: A review. *Computers & Operations Research*, *64*, 189–197.

Gschwind, T., & Drexl, M. (2019). Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Transportation Science*, *53*, 480–491.

Hansen, P., Mladenović, N., Brimberg, J., & Pérez, J. A. M. (2019). Variable neighborhood search. In *Handbook of metaheuristics* (pp. 57–97). Springer.

Hernandez, F., Feillet, D., Giroudeau, R., & Naud, O. (2014). A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration. *4OR*, *12*, 235–259.

Huang, Y., Zhao, L., Van Woensel, T., & Gross, J.-P. (2017). Time-dependent vehicle routing problem with path flexibility. *Transportation Research Part B: Methodological*, *95*, 169–195.

IBM CPLEX (2017). *IBM ILOG CPLEX 12.8.0 callable library*.

Ichoua, S., Gendreau, M., & Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, *144*, 379–396.

Lang, M., Wang, Y., & Zhou, X. (2010). A two-stage algorithm for a dynamic multi-trip vehicle scheduling problem. In *2010 WASE International Conference on Information Engineering* (pp. 188–191). IEEE volume 2.

Lim, A., Zhang, Z., & Qin, H. (2017). Pickup and delivery service with manpower planning in Hong Kong public hospitals. *Transportation Science*, *51*, 688–705.

Liu, C., Kou, G., Zhou, X., Peng, Y., Sheng, H., & Alsaadi, F. E. (2020). Time-dependent vehicle routing problem with time windows of city logistics with a congestion avoidance approach. *Knowledge-Based Systems*, *188*, 104813.

Liu, S., Qin, S., & Zhang, R. (2018). A branch-and-price algorithm for the multi-trip multi-repairman problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, *116*, 25–41.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, *3*, 43–58.

Macedo, R., Alves, C., de Carvalho, J. V., Clautiaux, F., & Hanafi, S. (2011). Solving the vehicle routing problem with time windows and multiple routes exactly using a pseudo-polynomial model. *European Journal of Operational Research*, *214*, 536–545.

Malandraki, C., & Daskin, M. S. (1992). Time dependent vehicle routing problems: formulations, properties and heuristic algorithms. *Transportation Science*, *26*, 185–200.

Nguyen, P. K., Crainic, T. G., & Toulouse, M. (2013). A tabu search for time-dependent multi-zone multi-trip vehicle routing problem with time windows. *European Journal of Operational Research*, *231*, 43–56.

Nguyen, P. K., Crainic, T. G., & Toulouse, M. (2017). Multi-trip pickup and delivery problem with time windows and synchronization. *Annals of Operations Research*, *253*, 899–934.

Olivera, A., & Viera, O. (2007). Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research*, *34*, 28–47.

Paradiso, R., Roberti, R., Laganá, D., & Dullaert, W. (2020). An exact solution framework for multitrip vehicle-routing problems with time windows. *Operations Research*, .

Petch, R. J., & Salhi, S. (2003). A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics*, *133*, 69–92.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*, 455–472.

Sun, P., Veelenturf, L. P., Dabia, S., & Van Woensel, T. (2018a). The time-dependent capacitated profitable tour problem with time windows and precedence constraints. *European Journal of Operational Research*, *264*, 1058–1073.

Sun, P., Veelenturf, L. P., Hewitt, M., & Van Woensel, T. (2018b). The time-dependent pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, *116*, 1–24.

Sun, Y., Wang, D., Lang, M., & Zhou, X. (2018c). Solving the time-dependent multi-trip vehicle routing problem with time windows and an improved travel speed model by a hybrid solution algorithm. *Cluster Computing*, (pp. 1–12).

Taillard, É. D., Laporte, G., & Gendreau, M. (1996). Vehicle routeing with multiple use of vehicles. *Journal of the Operational Research Society*, *47*, 1065–1070.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, *40*, 475–489.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, *234*, 658–673.

Wen, L., & Eglese, R. (2015). Minimum cost vrp with time-dependent speed data and congestion charge. *Computers & Operations Research*, *56*, 41–50.

Xiao, Y., & Konak, A. (2016). The heterogeneous green vehicle routing and scheduling problem with time-varying traffic congestion. *Transportation Research Part E: Logistics and Transportation Review*, *88*, 146–166.

Zeng, Z., Yu, X., He, K., Huang, W., & Fu, Z. (2016). Iterated tabu search and variable neighborhood descent for packing unequal circles into a circular container. *European Journal of Operational Research*, *250*, 615–627.

Zhang, R., Guo, J., & Wang, J. (2020). A time-dependent electric vehicle routing problem with congestion tolls. *IEEE Transactions on Engineering Management*, .

Zhang, Z., Liu, M., & Lim, A. (2015a). A memetic algorithm for the patient transportation problem. *Omega*, *54*, 60–71.

Zhang, Z., Wei, L., & Lim, A. (2015b). An evolutionary local search for the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints. *Transportation Research Part B: Methodological*, *82*, 20–35.