# Efficient neighborhood evaluations for the Vehicle Routing Problem with Multiple Time Windows

Maaike Hoogeboom, Wout Dullaert, David Lai[1] and Daniele Vigo[2]

[1]Department of Information, Logistics and Innovation, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands, m.hoogeboom@vu.nl

[2]Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", Alma Mater Università di Bologna, 40136 Bologna, Italy, daniele.vigo@unibo.it

April 27, 2018

---

**Abstract.** In the Vehicle Routing Problem with Multiple Time Windows (VRPMTW) a single time window must be selected for each customer from multiple time windows provided. As compared to classical vehicle routing problems with only a single time window per customer, multiple time windows increase the complexity of the routing problem. To minimize the duration of any given route, we present an exact polynomial time algorithm to determine efficiently the optimal start time for servicing each customer. The proposed algorithm has a lower worst-case and average complexity than existing exact algorithms. Furthermore, the proposed exact algorithm can be used to efficiently evaluate neighborhood operations during local search resulting in significant speed ups. To examine the benefits of exact neighborhood evaluations and to solve the VRPMTW, the proposed algorithm is embedded in a simple metaheuristic framework generating many new best-known solutions at competitive computation times.

**Keywords**: Vehicle Routing, Multiple Time Windows, Metaheuristics

---

The *Vehicle Routing Problem with Multiple Time Windows* (VRPMTW) arises naturally in delivery operations in which customers provide multiple time windows for service. Examples include: the distribution of industrial gases (Pesant et al. [1999]), long-haul transport (Rancourt et al. [2013], Goel and Kok [2012]), and city trips of tourists to various tourist attractions (Souffriau et al.

[2013]). The VRPMTW determines the minimum-cost vehicle routes that serve each customer in one of its specified time windows and satisfy vehicle capacity constraints.

The existing metaheuristics developed for the *Vehicle Routing Problem with Time Windows* (VRPTW) often involve the insertion and removal of customers from existing vehicle routes (see e.g., Bräysy and Gendreau [2005], Toth and Vigo [2014]). However, applying such local search operators to the scenario of multiple time windows becomes challenging. For a fixed vehicle route of $m$ customers where each customer has at most $t$ time windows, there are, in the worst-case, $t^m$ possible combinations of time windows to be considered for evaluation. Therefore, multiple time windows are portrayed as a difficult extension in the literature.

In various routing problems, multiple time windows are addressed, such as in the Traveling Salesman Problem, the Truck Driver Scheduling Problem, and the Team Orienteering Problem. Favaretto et al. [2007] were the first to tackle the VRPMTW. Their objective was to minimize the total duration by designing the routes and selecting the service time windows of customers. Belhaiza et al. [2014] were able to improve the results of Favaretto et al. [2007] by using an exact algorithm to determine the optimal departure time from the depot for a given route. Belhaiza et al. [2014] used the same route duration minimization algorithm for evaluating local search moves. This implied that although only a small part of the route is changed, the entire route is reevaluated. Tricoire et al. [2010] presented another exact route duration minimization algorithm for the related multi-period orienteering problem with multiple time windows. Their proposed algorithm finds the optimal departure time for every customer in a given route. Due to their computational complexity, the route duration minimization algorithm of Tricoire et al. [2010] is used only when the most promising move is performed. Promising moves are obtained by approximating the route duration when evaluating moves during the local search.

Research on the VRPTW has shown the benefits of exactly recalculating the route duration for local search moves. To recalculate the minimal route duration when neighborhood operators are applied, Savelsbergh [1992a] presented an efficient algorithm based on the forward and backward time slack of the departure time per customer. We call this algorithm efficient since the cost of a neighborhood operation can be calculated without reevaluating the entire route. However, as indicated by Tricoire et al. [2010] and Belhaiza et al. [2014], this approach to minimize the duration of a route cannot be easily extended to multiple time windows. To the best of our knowledge, there is no approach available to efficiently recalculate the minimal route duration when neighborhood operations are performed in multiple time window routing problems. Therefore, we have developed an exact polynomial time algorithm to both efficiently check the solution feasibility and determine the minimal route duration whenever a neighborhood operation is applied.

The proposed algorithm determines the optimal start time at each customer based on forward and backward start intervals and is inspired by the forward and backward algorithm of Savelsbergh [1992a]. These forward (resp. backward) intervals at any given customer represent the start times of servicing this customer such that all preceding (resp. succeeding) customers in the route are

served in an available time window. The complexity of the proposed algorithm is formally shown and its performance is examined by embedding the algorithm in a simple metaheuristic framework.

The contribution of this paper is threefold. First, we present a new exact polynomial time algorithm to calculate the minimal duration of a given route with a better worst-case complexity than the algorithms proposed in the literature. In addition, we show that the average computational time of our algorithm is at least twice as fast as existing algorithms when evaluating a given route. Second, we are the first to efficiently recalculate the minimum duration in local search operations (i.e., when customers are removed or inserted into a route). Computational experiments indicate an acceleration by a factor of four compared to existing exact route duration minimization algorithms. Therefore, our algorithm can provide computational benefits for many metaheuristic approaches. Last, we experimentally show that efficient exact neighborhood evaluations improve solution quality compared to an approximate evaluation method. By incorporating our exact algorithm in a simple metaheuristic, we were able to find 22 new best-known solutions to VRPMTW instances from the literature at competitive computation times.

The remainder of the paper is organized as follows. In Section 1, the literature on duration minimization in multiple time window routing problems is surveyed. In Section 2, the VRPMTW is defined and the subproblem of calculating the optimal departure time to minimize route duration is discussed. An exact polynomial algorithm to solve this subproblem is presented in Section 3. In Section 4, the metaheuristic is described, and in Section 5, the computational results are shown and discussed. The conclusions are presented in the last section.

# 1 Literature review

Compared to the VRPTW, the VRPMTW has received little attention in the literature. Problems with multiple time windows are often addressed within related problems, such as the Traveling Salesman Problem (TSP), the Truck Driver Scheduling Problem, and the Team Orienteering Problem.

Pesant et al. [1999] present a constraint programming formulation for the TSP with multiple time windows. They illustrate that multiple time windows result in discontinuities in the domain of the variable representing the start time of servicing customer $i$ and that these discontinuities can be used to sharpen the lower and upper bounds of this variable. As the goal is to minimize the total travel cost of the tour, waiting time is not taken into account; thus, the duration of a tour is not minimized. Recently Paulsen et al. [2015] presented a dynamic programming approach to minimize the tour duration for a TSP with multiple time windows. Their proposed dynamic programming algorithm solves the subproblem of finding a path of minimum duration from the depot to a final customer through a given subset of nodes. To solve the TSP, the algorithm uses labels of dominant arrival intervals with minimal duration till the final customer. The drawback of this method is that it is computationally intensive for large tours and the customers need to be

assigned to a vehicle before deploying the algorithm.

In the Truck Driver Scheduling Problem with multiple time windows, the goal is to find a schedule with minimal duration for a fixed route that satisfies regulations concerning hours of service (Goel and Kok [2012] and Goel [2012]). The main decision to be made is when to place the rest periods; therefore, the design of the algorithm is focused on rest periods. The algorithm can also be used for problems without rest periods, in which the algorithm would be similar to the approach of Belhaiza et al. [2014]. However, the dominance criteria used in Belhaiza et al. [2014] to eliminate dominated solutions are stronger than the dominance criteria in Goel and Kok [2012] and Goel [2012] when rest periods are not taken into account.

Souffriau et al. [2013] present a metaheuristic for the multi-constraint team orienteering problem with multiple time windows. To deal with multiple time windows, every vertex with more than one time window is replaced by a set of vertices with only one time window. An extra constraint is included to allow only one visit per vertex set. The goal is to maximize the score of the visited customers and no approach is given for minimizing the duration of a route. However, if there are duration limits then it is profitable to minimize route duration so that more customers can be visited in one trip. Therefore, Tricoire et al. [2010] present an algorithm to minimize the duration of a given route to address the multi-period orienteering problem with multiple time windows. They tighten the time windows for each customer by exploiting the fact that service at any given customer cannot start before the end of service at the previous customer. Tricoire's algorithm uses the concept of dominant solutions, which are solutions for which no other solution exists with the same finishing time and later departure time. The polynomial algorithm of Tricoire et al. [2010] enumerates all promising dominant solutions of an entire route in order of increasing finishing time and selects the one with the shortest duration. However, as the algorithm does not find the optimal solution for portions of a route, such as tails, it is not suited to quickly evaluate neighborhood operations in a local search.
In Hurkała [2015] the traveling salesman problem with multiple time windows and time-dependent travel times is presented. Their minimum route duration algorithm is based on, and therefore similar to the algorithm of Tricoire et al. [2010].

The first paper on the VRPTMW is published by Favaretto et al. [2007]. They consider the VRPMTW in a periodic setting in which some customers require service multiple times during the schedule horizon. Customers that are serviced multiple times must be visited in different time windows, so no time window can be used twice for the same customer. The objective is to minimize the total duration and the problem is solved using ant colony systems. The focus in Favaretto et al. [2007] is to divide time windows over multiple visit days; therefore, no algorithm is given to minimize the duration of a given route.

Belhaiza et al. [2014] minimize the total duration and a hybrid variable neighborhood tabu search heuristic is proposed to solve the VRPMTW. The duration of a route is minimized by

calculating the optimal departure time at the depot. Their proposed exact algorithm calculates the time delay interval of each time window at each customer when departing from the depot at time zero, i.e., the time that can be added to the arrival time to ensure the feasibility of each time window. Using these time delay intervals the algorithm enumerates all possible combinations of time windows and records the minimum waiting time and corresponding departure time from the depot. When waiting time occurs at a customer, the subsequent time windows of this customer are skipped. The enumeration is stopped when a solution with zero waiting time is found. Belhaiza et al. [2017] propose a new hybrid genetic variable neighborhood search heuristic for the VRPMTW that improves almost all results of Belhaiza et al. [2014].

The papers of Beheshti et al. [2015] and Belhaiza [2016] solve a multi objective VRPMTW. Beheshti et al. [2015] solve the vehicle routing problem with multiple prioritized time windows in which customers can prioritize the multiple available time windows and the customers are grouped based on their importance to the distributor. The multi-objective problem of minimizing the traveling cost and maximizing the customers' satisfaction is solved. In Belhaiza [2016] the multiple criteria of minimizing the total travel time and maximizing the utility of the customers and drivers are taken into account. Both approaches are searching for pareto optimal solutions and do not minimize the total duration.

Both exact algorithms of Tricoire et al. [2010] and Belhaiza et al. [2014] are developed to minimize the duration of a given route. In Tricoire et al. [2010] the exact algorithm is used only to calculate the minimal duration of a local optimum solution and an approximation of the route duration is used to evaluate the moves during the local search. In Belhaiza et al. [2014] the exact algorithm is also used to evaluate neighborhood operations during the local search. However, they use the same algorithm for a given route as for evaluating a neighborhood operation in which only a small part of a route changes. Therefore, we developed an approach which is efficient for evaluating neighborhood operations.

Savelsbergh [1992b] presented an efficient method to check if a move is feasible and profitable for the TSP with multiple time windows. He calls a move profitable if the completion time of the resulting route is earlier, while the departure time from the depot stays the same. Hence, the profitability of moves is calculated only for the current fixed departure time from the depot. Since the optimal departure time from the depot is not selected, the resulting route duration of a move is not necessarily minimized.

For the VRPTW, Savelsbergh [1992a] developed a route duration minimization algorithm, with variable departure times at the depot to recalculate the minimal waiting time when local search operations are applied. In this algorithm forward time slack is introduced, which indicates how far forward in time the departure time at any given customer can be shifted without causing the route to become infeasible. To measure the profitability of a move backward time slack is also calculated; this indicates how far the departure time at a customer can be shifted backward in

time without introducing waiting time. To the best of our knowledge, this approach has not been extended to a setting with multiple time windows. Therefore, we propose a new efficient algorithm based on forward and backward start intervals to recalculate the minimal waiting time of a route when customers are inserted or removed. Our algorithm can also be used to calculate the minimal duration of a given route similar to Tricoire et al. [2010] and Belhaiza et al. [2014], but at a lower computational complexity.

## 2 Problem description

The VRPMTW is defined on a complete directed graph $G = (V, A)$ with nodes $V = \{0, 1, \ldots, n\}$ and arcs $A = \{(i, j) \in V \times V : i \neq j\}$. Node 0 represents the depot and nodes $V' = \{1, \ldots, n\}$ correspond to the set of customers. Each arc $(i, j) \in A$ is associated with a non-negative travel time $\tau_{ij}$ of the shortest path from node $i$ to $j$. Each node $i \in V$ is associated with a demand $d_i$ and a service time $s_i$, where $d_i$ and $s_i$ are non-negative numbers. Let $\{[e_i^1, l_i^1], [e_i^2, l_i^2], \ldots, [e_i^{|T_i|}, l_i^{|T_i|}]\}$ be the time windows on node $i \in V$, with $T_i = \{1, \ldots, |T_i|\}$ the index set of the time windows. We assume that the time windows associated with node $i \in V$ are non-overlapping, with $0 \leq e_i^1 \leq l_i^1 < e_i^2 \leq l_i^2 < \ldots < l_i^{|T_i|}$. A preprocessing procedure is applied to remove or adapt the customer time windows that conflict with the time window of the depot, the details of this preprocessing can be found in Appendix A. If a vehicle arrives at a customer within one of the time windows, the service starts immediately upon arrival; otherwise, it waits until the opening time of the next time window and then starts the service. Servicing a customer after its last time window is not allowed. Furthermore, we set $s_0 = 0$, $d_0 = 0$ and define $[e_0, l_0]$ as the single time window of the depot.

Consider a fleet of identical vehicles (denoted as set $K$), with capacity $Q$ and fixed vehicle cost $F$ when a vehicle is used. The objective value of a solution consist of the fixed vehicle cost and the total duration. The goal of the VRPMTW is to determine the vehicle routes with minimum objective value satisfying the following requirements.

1. Every customer is serviced exactly once by a single vehicle.
2. The service of every customer must start within one of their given time windows.
3. The total demand of a vehicle route cannot exceed the vehicle capacity, $Q$.

This paper focuses on determining the minimal route duration, hereafter referred to as the subproblem of the VRPMTW.

### 2.1 Subproblem: minimum route duration

A route satisfies time window constraints if the service of each customer in the route starts in one of its time windows. Different departure times from the depot can correspond to different time window selections and different durations. Hence, the subproblem is to select the optimal departure

time, such that service to all customers starts within one of its time window and route duration is minimized.

Let $\sigma$ be a given route of $m$ customers. For simplicity, suppose $\sigma = \{0, 1, \ldots, m, m+1\}$ with 0 and $m+1$ representing the depot and the customers are named $\sigma' = \{1, \ldots, m\}$, which can always be achieved by renumbering the customers. For all $i \in \sigma$ and $t \in T_i$, let $z_{it}$ be a binary decision variable, where $z_{it}$ equals 1 if time window $t$ of customer $i$ is selected; otherwise 0. Furthermore, let $a_i$ be the arrival time and $w_i$ be the waiting time at customer $i \in \sigma$, so the service time at customer $i$ starts at time $a_i + w_i$. The subproblem is formulated as the following mixed integer linear programming model.

$$\min \sum_{i=1}^{m} w_i, \tag{1}$$

$$\text{s.t. } a_i + w_i + s_i + \tau_{i,i+1} = a_{i+1}, \qquad \forall i \in \sigma, \tag{2}$$

$$a_i + w_i \leq \sum_{t \in T_i} l_i^t z_{it}, \qquad \forall i \in \sigma, \tag{3}$$

$$a_i + w_i \geq \sum_{t \in T_i} e_i^t z_{it}, \qquad \forall i \in \sigma, \tag{4}$$

$$\sum_{t \in T_i} z_{it} = 1, \qquad \forall i \in \sigma, \tag{5}$$

$$a_i, w_i \geq 0, \qquad \forall i \in \sigma, \tag{6}$$

$$z_{it} \in \{0, 1\}, \qquad \forall i \in \sigma,\ t \in T_i. \tag{7}$$

The objective function (1) is to minimize the total waiting time of the route, which corresponds to minimizing the duration since the travel time and service time are fixed. Constraints (2) prevent subtours and make the arrival times consistent. Constraints (3) - (5) ensure that the start time of servicing a customer lies within one of the given time window. In an optimal solution $w_i = 0$, so $a_0$ represents the optimal departure time.

Belhaiza et al. [2014] and Tricoire et al. [2010] both present an algorithm for this subproblem. In the worst-case, the approach of Belhaiza et al. [2014] has to check all possible combinations of time windows resulting in a complexity of $O(\prod_{i=1}^{m} |T_i|)$. The algorithm of Tricoire et al. [2010] has a better complexity of $O(m(1+\sum_{i=1}^{m}(|T_i|-1)))$. Our proposed exact polynomial algorithm, described in the following section, has the lowest complexity of $O(\sum_{i=1}^{m}(1+\sum_{j=1}^{i}(|T_j|-1)))$. Moreover, our algorithm can be used efficiently with local search operations in the metaheuristic search as will be shown in Section 3.4.

Note that the approach of Belhaiza et al. [2014] can handle overlapping time windows and Tricoire et al. [2010]'s and our algorithm can not. In practice there is no distinction between these cases, since overlapping time windows can easily be merged into non-overlapping time windows. If a decision about the start time at a customer has been made then the corresponding original time

window is communicated to the customer. For example, two overlapping time windows [3,5] and [4,6] at a customer are merged to the single time window [3,6]. Suppose that in the final solution the start time is 5, then the second time window is communicated to the customer.

# 3 Solution method subproblem

In this section, we introduce a novel approach to solve the subproblem defined in Section 2.1. First the proposed solution algorithm based on forward start intervals is described in Section 3.1. In Section 3.2, the complexity of the proposed algorithm is determined. Backward start intervals are introduced in Section 3.3; and we show how forward and backward start intervals can be used efficiently to evaluate local search operations in Section 3.4.

## 3.1 Route duration minimization algorithm: forward start intervals

In this section, we propose a new exact polynomial time algorithm based on forward start intervals to solve subproblem (1)-(7). The *forward start intervals* of customer $i \in \sigma'$ represent the start times of servicing customer $i$ such that the preceding customers can feasibly be serviced.

Let $[E_i^F(q), L_i^F(q)]$ be forward start interval $q$ of customer $i \in \sigma'$, where $E_i^F(q)$ and $L_i^F(q)$ are the end points of the interval. For all nodes $i \in \sigma'$, let $F_i$ be the index set of the forward start intervals associated with node $i$. The forward start intervals are sorted in increasing order, i.e., $E_i^F(1) \leq L_i^F(1) \leq E_i^F(2) \leq L_i^F(2) \leq \ldots \leq E_i^F(|F_i|) \leq L_i^F(|F_i|)$. The forward start intervals are recursively computed, from the first to the last customer in $\sigma'$. The forward start intervals of the first customer are equal to the time windows $T_1$ and the sets of forward start intervals of the other customers are initially empty. The forward start intervals of customer $i$ are recursively computed based on the forward start intervals of customer $i-1$ and the time windows of customer $i$. If a vehicle starts servicing customer $i-1$ during forward start interval $q$ and arrives at customer $i$ before or during time window $t \in T_i$, i.e., $E_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i} \leq l_i^t$, then a forward start interval within time window $t$ of node $i$ is created. This new forward start interval $p$ of customer $i$ can be determined by

$$\begin{aligned} E_i^F(p) &= \max\{E_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i}, e_i^t\}, \\ L_i^F(p) &= \min\{\max\{L_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i}, e_i^t\}, l_i^t\}, \end{aligned} \tag{8}$$

Waiting time will occur at customer $i$ if the customer is serviced in time window $t \in T_i$ while the vehicle arrives before this time window, i.e., if $L_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i} < e_i^t$. Hence, if the service of customer $i$ starts in $[E_i^F(p), L_i^F(p)]$, then the minimal total waiting time until this customer is given by:

$$w_i^F(p) = \begin{cases} w_{i-1}^F(q) + e_i^t - L_{i-1}^F(q) - s_{i-1} - \tau_{i-1,i} & \text{if } L_i^F(p) = e_i^t, \\ w_{i-1}^F(q) & \text{otherwise.} \end{cases} \tag{9}$$

Let $W_i^F = \{w_i^F(1), \ldots, w_i^F(|F_i|)\}$ be the set of minimal total waiting times until customer $i$ corresponding to the forward start intervals. As initialization, we set $w_1^F(p) = 0$ for all $p \in F_1 = T_1$. Note that if $w_i^F(p) > 0$ then $E_i^F(p) = L_i^F(p)$.

In Figure 1 the forward start intervals and corresponding waiting times of a small route of three customers are given. Note that every forward start interval corresponds to a different selection of time windows.

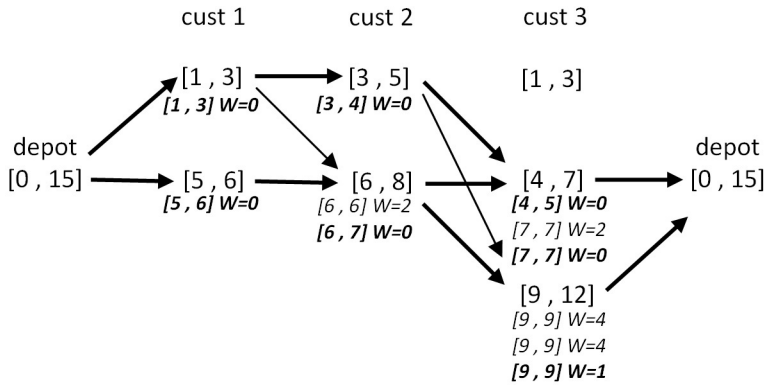


Figure 1: Time windows and forward start intervals, with corresponding waiting times (italics) of three customers in a route. Let all travel times be one time unit and all service times be zero. The arrows represent the feasible connections between time windows, where the bold arrows represent dominant combinations.

In Figure 1, customer 2 has forward start interval $[6, 6]$ with waiting time 2 and forward start interval $[6, 7]$ with zero waiting time. In this case, the first forward start interval will never be used, since it is dominated by the second forward start interval.

**Definition 1.** *A forward start interval $q \in F_i$ **dominates** forward start intervals $q' \in F_i$ at customer $i$ if all feasible arrival times $a_{i+1}$ at customer $i+1$ for $q'$ are also feasible for $q$ with lower or equal total waiting time until customer $i+1$. A forward start interval $q \in F_i$ is **dominant** if it is not dominated by another forward start interval.*

The dominance criteria used in our algorithm is given by the following proposition.

**Proposition 1.** *Forward start interval $q \in F_i$ **dominates** forward start intervals $q' \in F_i$ at customer $i$ if $E_i^F(q) \leq E_i^F(q')$ and $w_i^F(q') - w_i^F(q) \geq L_i^F(q') - L_i^F(q)$ hold.*

*Proof.* Suppose that we have two forward start intervals $q, q' \in F_i$ for which $E_i^F(q) \leq E_i^F(q')$ and $w_i^F(q') - w_i^F(q) \geq L_i^F(q') - L_i^F(q)$ hold. We have to prove that the condition from Definition 1 is satisfied. Let $a_{i+1}$ be a feasible arrival time at customer $i+1$ when departing from customer $i$ in forward start interval $q'$. Since $E_i^F(q) \leq E_i^F(q')$ arrival time $a_{i+1}$ is also feasible for forward start interval $q$. Let $w_{a_{i+1}}$ be the total waiting time at $a_{i+1}$ when departing from forward start interval

$q$ and $w'_{a_{i+1}}$ when departing from $q'$. We only have to show that $w'_{a_{i+1}} - w_{a_{i+1}} \geq 0$.

$$w'_{a_{i+1}} - w_{a_{i+1}} \tag{10}$$
$$= w_i^F(q') + \max\{0, a_{i+1} - L_i^F(q') - s_i - \tau_{i,i+1}\} - w_i^F(q) - \max\{0, a_{i+1} - L_i^F(q) - s_i - \tau_{i,i+1}\} \tag{11}$$
$$= w_i^F(q') - w_i^F(q) + \max\{0, a_{i+1} - L_i^F(q') - s_i - \tau_{i,i+1}\} - \max\{0, a_{i+1} - L_i^F(q) - s_i - \tau_{i,i+1}\} \tag{12}$$

There are two options to consider. First, if $L_i^F(q') < L_i^F(q)$, then $E_i^F(q) \leq E_i^F(q') \leq L_i^F(q') < L_i^F(q)$, so $E_i^F(q) < L_i^F(q)$ which implies that $w_i^F(q) = 0$. Since $w_i^F(q') \geq 0$ this implies that the equation in line (12) is greater or equal than zero. Hence, for the first option $w'_{a_{i+1}} - w_{a_{i+1}} \geq 0$ holds. Second, if $L_i^F(q') \geq L_i^F(q)$ then using the assumption $w_i^F(q') - w_i^F(q) \geq L_i^F(q') - L_i^F(q)$ we get

$$w'_{a_{i+1}} - w_{a_{i+1}}$$
$$\geq L_i^F(q') - L_i^F(q) + \max\{0, a_{i+1} - L_i^F(q') - s_i - \tau_{i,i+1}\} - \max\{0, a_{i+1} - L_i^F(q) - s_i - \tau_{i,i+1}\}. \tag{13}$$

Since $\max\{0, a_{i+1} - L_i^F(q') - s_i - \tau_{i,i+1}\} - \max\{0, a_{i+1} - L_i^F(q) - s_i - \tau_{i,i+1}\} \geq L_i^F(q) - L_i^F(q')$, equation (13) is larger or equal to zero, so also for the seconds case $w'_{a_{i+1}} - w_{a_{i+1}} \geq 0$ holds. □

In Figure 1 all dominant forward start intervals with corresponding waiting times are given in bold. In the remainder of the paper we take only the dominant forward start intervals into account. Let $\overline{F}_i$ be the index set of the dominant forward start intervals, which are also sorted in increasing order.

To calculate all dominant forward start intervals from the first to the last customer in a given route $\sigma$ we present the Dominant Forward Start Interval (DFSI) algorithm. The pseudo-code for the DFSI is given in Algorithm 1. In this algorithm the time windows of customer $i \in \{2, \ldots, m\}$ are compared with the dominant forward start intervals of the previous customer $i-1$, to construct the dominant forward start intervals of customer $i$.

If $E_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i} \leq l_i^t$, then customer $i-1$ is serviced at time $E_{i-1}^F(q)$ and the vehicle arrives before or during time window $t \in T_i$ at customer $i$. Hence, time window $t$ at customer $i$ is feasible and the corresponding forward start interval and total waiting time at customer $i$ are calculated.

If $L_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i} \leq l_i^t$, then forward start interval $q \in \overline{F}_{i-1}$ will not result in a dominant forward start interval within the next time windows $t' \in T_i$ with $t' > t$. This follows from the fact that the time windows are non-overlapping, so the extra waiting time obtained at customer $i$ would be $e_i^{t'} - L_{i-1}^F(q) - s_{i-1} - \tau_{i-1,i}$, which is equal to the difference in time with the previous forward start intervals. Hence, in this case we move to the next dominant forward start interval

of customer $i-1$. At this dominant forward start interval $q+1$ of customer $i-1$ we do not start comparing with the first time window of customer $i$, but rather with the last checked time window $\theta$. The optimality of this approach will be proven in Lemma 2 in the next subsection. Accordingly, not all combinations of dominant forward start intervals and time windows need to be checked. This results in an efficient generation of the dominant forward start intervals. The details on the complexity are given Section 3.2.

**Algorithm 1** Dominant Forward Start Interval (DFSI)

**Input:** $\overline{F}_1 = T_1$, $w_1^F(p) = 0\ \forall p \in \overline{F}_1$, $\overline{F}_i = W_i^F = \emptyset\ \forall i \in \{2,\ldots,m\}$ and $\theta = 1$

**for** $i \in \{2,\ldots,m\}$ **do**
  **for** $q \in \overline{F}_{i-1}$ **do**
    **for** $t \in \{\theta,\ldots,|T_i|\}$ **do**
      **if** $E_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i} \leq l_i^t$ **then**
        $p = |\overline{F}_i| + 1$
        **if** $L_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i} \geq e_i^t$ **then** ▷ Arrival in time window $t$ at customer $i$
          $E_i^F(p) = \max\{E_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i}, e_i^t\}$ ▷ Create a new forward start interval
          $L_i^F(p) = \min\{L_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i}, l_i^t\}$
          $w_i^F(p) = w_{i-1}^F(q)$
        **else** ▷ Arrival before time window $t$ at customer $i$
          $E_i^F(p) = L_i^F(p) = e_i^t$
          $w_i^F(p) = w_{i-1}^F(q) + e_i^t - L_{i-1}^F(q) - s_{i-1} - \tau_{i-1,i}$
        **end if**
        Check for dominance by comparing the last two start intervals $p$ and $p-1$, adjust $p$ if needed.
        **if** $L_{i-1}^F(q) + s_i + \tau_{i-1,i} \leq l_i^t$ **then**
          $\theta = t$ ▷ Set last visited time window
          break ▷ Go to the next forward start interval
        **end if**
      **end if**
    **end for** $t$
  **end for** $q$
**end for** $i$

**Output:** $\overline{F}_i$ and $W_i^F\ \forall i \in \{1,\ldots,m\}$

The DFSI algorithm calculates all dominant forward start intervals of all customers $i \in \sigma'$. In Appendix B it is proved that, for the optimal solution, the start time of servicing customer $i \in \sigma'$ is included in a dominant forward start interval. The dominant forward start interval at customer $i$ with the lowest total waiting time gives the minimal waiting time for subsequence $\{1,\ldots,i\}$. From this optimal forward start interval the optimal start times of servicing the other customers can be found.

The DFSI presented in this section is a breadth-first algorithm, which first calculates all forward

start intervals of customer 1, then of customer 2, and so on. A breadth-first algorithm is most suitable if all dominant forward start intervals are necessary, e.g., for the evaluation of local search operations as described in Section 3.4. If we want to calculate only the minimal total duration of a given route then the search should quit as soon as a solution without waiting time has been found. In that case, a depth-first implementation as presented in Appendix C is more appropriate. Note, however, that both implementations have the same worst-case complexity.

### 3.2 Complexity Analysis

In this section we will show that the DFSI algorithm has a polynomial time computational complexity of $O(\sum_{i=1}^{m}(1+\sum_{j=1}^{i}(|T_j|-1))$. To prove this complexity we need the following three lemmas.

The first lemma shows that the dominant forward start intervals at every customer $i \in \sigma'$ are non-overlapping and increasing in $q \in \overline{F}_i$. The proof can be found in Appendix D.

**Lemma 1.** *$E_i^F(q) \leq L_i^F(q) < E_i^F(q+1) \leq L_i^F(q+1)$ holds for all $i \in \sigma'$ and $q \in \overline{F}_i$.*

The next lemma states that, thanks to the dominance criteria, we are not required to check all combinations of time windows explicitly.

**Lemma 2.** *For all $j \in \sigma'$ and $q \in \overline{F}_{j-1}$, if dominant forward start interval $q$ of customer $j-1$ results in a dominant forward start interval within time window $t \in T_j$ at customer $j$, then dominant forward start interval $q' \in \overline{F}_{j-1}$ with $q' < q$ can **not** result in a dominant forward start interval within time window $t' \in T_j$ with $t' > t$ at customer $j$.*

*Proof.* Let $[E_j^F(p), L_j^F(p)]$ be a dominant forward start interval within time window $t \in T_j$ at customer $j$ originating from dominant forward start interval $q$ of customer $j-1$. Suppose that dominant forward start interval $q' \in \overline{F}_{j-1}$ with $q' < q$ results in forward start interval $[E_j^F(p'), L_j^F(p')]$ at customer $j$ within time window $t' \in T_j$ with $t' > t$. Using Proposition 1 we show that forward start interval $p$ dominates $p'$.

$$
\begin{aligned}
&w_j^F(p') - w_j^F(p) = \\
&= w_{j-1}^F(q') + \max\{0, e_j^{t'} - L_{j-1}^F(q') - s_{j-1} - \tau_{j-1,j}\} - (w_{j-1}^F(q) + \max\{0, e_j^t - L_{j-1}^F(q) - s_{j-1} - \tau_{j-1,j}\}) && (14)\\
&= w_{j-1}^F(q') - w_{j-1}^F(q) + e_j^{t'} - L_{j-1}^F(q') - s_{j-1} - \tau_{j-1,j} - \max\{0, e_j^t - L_{j-1}^F(q) - s_{j-1} - \tau_{j-1,j}\} && (15)\\
&> L_{j-1}^F(q') - L_{j-1}^F(q) + e_j^{t'} - L_{j-1}^F(q') - s_{j-1} - \tau_{j-1,j} - \max\{0, e_j^t - L_{j-1}^F(q) - s_{j-1} - \tau_{j-1,j})\} && (16)\\
&= e_j^{t'} - L_{j-1}^F(q) - s_{j-1} - \tau_{j-1,j} - \max\{0, e_j^t - L_{j-1}^F(q) - s_{j-1} - \tau_{j-1,j})\} && (17)\\
&= e_j^{t'} - L_j^F(p) = L_j^F(p') - L_j^F(p) && (18)
\end{aligned}
$$

Line (14) follows from the definition given in (9). By Lemma 1 $L^F_{j-1}(q') + s_{j-1} + \tau_{j-1,j} < E^F_{j-1}(q) + s_{j-1} + \tau_{j-1,j} \leq l^t_j < e^{t'}_j$ holds and by rearranging the terms line (15) is obtained. Because forward start intervals $q'$ and $q$ are both dominant by assumption, we get $w^F_{j-1}(q') - w^F_{j-1}(q) > L^F_{j-1}(q') - L^F_{j-1}(q)$, which is used to deduce (16) from (15). Line (18) follows from the facts that $L^F_j(p) = L^F_{j-1}(q) + s_{j-1} + \tau_{j-1,j} + \max\{0, e^t_j - L^F_{j-1}(q) - s_{j-1} - \tau_{j-1,j})\}$ and $L^F_j(p') = e^{t'}_j$. Because $E^F_j(p) < E^F_j(p')$ and $w^F_j(p') - w^F_j(p) > L^F_j(p') - L^F_j(p)$, we have shown that $[E^F_j(p'), L^F_j(p')]$ is not a dominant forward start interval. □

Following Paulsen et al. [2015], we will show the upper bound on the number of dominant forward start intervals. Since every forward start interval corresponds to a different selection of time windows, it gives an upper bound to the maximum number of time window combinations that need to be checked.

**Lemma 3.** *For a given route of $m$ customers, at most $1 + \sum_{j=1}^{m}(|T_j| - 1)$ selections of time windows need to be checked, i.e., $|\overline{F}_m| \leq 1 + \sum_{j=1}^{m}(|T_j| - 1)$.*

*Proof.* By induction over the number of customers $m$, we will show that this lemma holds for all $m$. For $m = 1$ there is only one customer with $|T_1|$ time windows so there are at most $|T_1|$ dominant forward start intervals. Suppose that the lemma is true for $m = i$, we will show that it is also true for $m = i + 1$. Define a bipartite graph $G = (\overline{F}_i, T_{i+1})$ with $\overline{F}_i$ the index set of dominant forward start intervals at customer $i$ and $T_{i+1}$ the index set of the time windows of customer $i + 1$. An edge $(q, t)$ exists if dominant forward start interval $q \in \overline{F}_i$ results in a dominant forward start interval within time window $t \in T_{i+1}$. Hence, the edge set of graph $G$ corresponds to the number of dominant forward start intervals at customer $i + 1$, $|\overline{F}_{i+1}|$. With Lemma 1 we know that the dominant forward start intervals are non-overlapping and increasing. By putting the index set of the dominant forward start intervals $\overline{F}_i$ and the time windows $T_{i+1}$ both in increasing order, the bipartite graph $G$ can be drawn without crossings following Lemma 2. Since $G$ is a bipartite graph without crossings, it does not contain cycles. This implies that $G$ is a forest, a union of disjoint trees, which has at most $|\overline{F}_i| + |T_{i+1}| - 1$ edges. By the induction hypothesis we get $|\overline{F}_{i+1}| = |\overline{F}_i| + |T_{i+1}| - 1 = 1 + \sum_{j=1}^{i+1}(|T_j| - 1)$. □

**Theorem 1.** *The computational complexity of the DFSI algorithm is $O(\sum_{i=1}^{m}(1 + \sum_{j=1}^{i}(|T_j| - 1))$*

*Proof.* With Lemma 3 we know that the maximum number of dominant forward start intervals at customer $i$ is bounded from above by $1 + \sum_{j=1}^{i}(|T_j| - 1)$. Hence, the complexity of finding all dominant forward start intervals of all $m$ customers in a route is $\sum_{i=1}^{m}(1 + \sum_{j=1}^{i}(|T_j| - 1))$. □

### 3.3 Backward start intervals

In local search, neighborhood operations are used to insert and remove customers from a route $\sigma$. Alongside forward start intervals, backward start intervals are needed to quickly check the time

window feasibility and to recalculate the minimal total waiting time of a route when neighborhood operations are applied. The *backward start intervals* of customer $i$ represent the start times of servicing customer $i$ such that all subsequent customers are served in one of their time windows. As for the forward start intervals, we associate a set of backward start intervals with each node $i \in \sigma'$. Let $B_i = \{1, \ldots, |B_i|\}$ be the index set of the backward start intervals of customer $i$ with $[E_i^B(q), L_i^B(q)]$ representing backward start interval $q \in B_i$. The backward start intervals are presented in increasing order and every backward start interval of node $i \in \sigma'$ corresponds to a different selection of time windows. The backward start intervals are computed in a backward recursion from the last customer to the first customer in $\sigma'$, with initialization $B_m = T_m$ and $w_m^B(p) = 0 \ \forall p \in B_m$. If a vehicle starts servicing customer $i$ in time window $t \in T_i$ and the vehicle arrives before or during the backward start interval $q$ at customer $i{+}1$, i.e., if $e_i^t + \tau_{i,i+1} + s_i \leq L_{i+1}^B(q)$ ( $\implies$ $L_{i+1}^B(q) - \tau_{i,i+1} - s_i \geq e_i^t$), then the new backward start interval $p$ of customer $i$ can be calculated by:

$$\begin{aligned} E_i^B(p) &= \max\{\min\{E_{i+1}^B(q) - \tau_{i,i+1} - s_i, l_i^t\}, e_i^t\} \\ L_i^B(p) &= \min\{L_{i+1}^B(q) - \tau_{i,i+1} - s_i, l_i^t\} \end{aligned} \tag{19}$$

The total waiting time of customer sequence $\{i, i+1, \ldots, m\}$ when service at customer $i$ starts within backward start interval $[E_i^B(p), L_i^B(p)]$ is given by:

$$w_i^B(p) = \begin{cases} w_{i+1}^B(q) + E_{i+1}^B(q) - \tau_{i,i+1} - s_i - l_i^t & \text{if } E_i^B(p) = l_i^t \\ w_{i+1}^B(q) & \text{otherwise} \end{cases}$$

Let $W_i^B = \{w_i^B(1), \ldots, w_i^B(|B_i|)\}$ be the set of minimal total waiting times corresponding to the backward start intervals of customer $i$.

Similar as Definition 1, backward start interval $q$ dominates backward start interval $q'$ at customer $i$ if all feasible start times at customer $i-1$ for $q'$ are also feasible for $q$ with lower or equal waiting time. To check if an interval is dominated the following dominance criteria is used:

**Proposition 2.** *Backward start interval $q \in B_i$ dominates backward start interval $q' \in B_i$ if $L_i^B(q) \geq L_i^B(q')$ and $w_i^B(q') - w_i^B(q) \geq E_i^B(q) - E_i^B(q')$.*

That this criteria implies dominance can be proven similar as for Proposition 1. Intuitively, the difference in waiting time of the two backward start intervals is higher than the time difference between the lower bounds of the intervals. Therefore, backward start interval $q$ can find the same arrival times at customer $i-1$, because $L_i^B(q) \geq L_i^B(q')$, with lower or equal total waiting time. Let $\overline{B}_i$ be the index set of the dominant backward start intervals of customer $i$, which are presented in increasing order. To find the solution with minimal duration we have to take only the dominant backward start intervals into account.

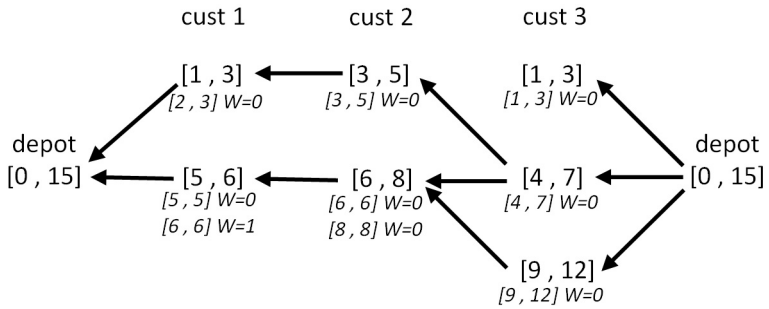


Figure 2: The dominant backward start intervals and corresponding waiting times of the customers $\{1, 2, 3\}$ are presented.

Similar to the DFSI algorithm we can calculate all dominant backward start intervals from the last to the first customer in a given route $\sigma$. Lemma 1 and Lemma 3 also hold for the dominant backward start intervals, therefore, the calculation of all backward start intervals also has a polynomial time computational complexity of $O(\sum_{i=1}^{m}(1 + \sum_{j=1}^{i}(|T_j| - 1))$.

### 3.4 Efficient evaluation of local search moves

For all customers in a route $i \in \sigma'$, the dominant forward start intervals $f \in \overline{F}_i$ and the dominant backward start intervals $b \in \overline{B}_i$, with corresponding waiting times $W_i^F$ and $W_i^B$, can be calculated in polynomial time. These sets will be used to recompute efficiently the minimal waiting time of a route when local search operators are applied. We will illustrate this for a deletion and insertion of a customer in a route, since these approaches can be easily extended to more complex operations.

*Deletion*
Suppose we have a route $\sigma = \{0, 1, \ldots, m, m+1\}$ and we want to remove customer $i$, then we need to check whether the new route $\tilde{\sigma} = \{0, 1, \ldots, i-1, i+1, \ldots, m+1\}$ is feasible and, if so, compute the minimal total waiting time. To do this, we compare the dominant forward start intervals of customer $i-1$ with the dominant backward start intervals of customer $i+1$. The route is feasible if there exist $f \in \overline{F}_{i-1}$ and $b \in \overline{B}_{i+1}$ such that $E_{i-1}^F(f) + s_{i-1} + \tau_{i-1,i+1} \leq L_{i+1}^B(b)$. The corresponding total waiting time to this feasible combination is given by $w^R(f, b) = w_{i-1}^F(f) + w_{i+1}^B(b) + \max\{0, E_{i+1}^B(b) - L_{i-1}^F(f) - s_{i-1} - \tau_{i-1,i+1}\}$, i.e., the sum of the waiting time of sequence $\{1, \ldots, i-1\}$, the waiting time of sequence $\{i+1, \ldots, m\}$, and the waiting time between customer $i-1$ and $i+1$. The minimum waiting time of the new route $\tilde{\sigma}$ is given by $\min\{w^R(f, b) | f \in \overline{F}_{i-1}, b \in \overline{B}_{i+1}$ with $E_{i-1}^F(f) + s_{i-1} + \tau_{i-1,i+1} \leq L_{i+1}^B(b)\}$.

If the new route $\tilde{\sigma}$ is accepted, then the new dominant start intervals need to be calculated. Since a part of the route stays the same, only the dominant forward start intervals of customers $i+1, \ldots, m$ and the dominant backward start intervals of customers $1, \ldots, i-1$ need to be recalculated.

The worst-case computational complexity, to check if a removal of customer $i$ results in a

feasible solution and to calculate the new minimal total waiting time, is $|\overline{F}_{i-1}| + |\overline{B}_{i+1}| - 1 = 2 - |T_i| + \sum_{j=1}^{m} |T_j| - 1$. Note that if route $\sigma$ is feasible then customer deletion is always time window feasible if the triangle inequality holds.

*Insertion*
Suppose we have a route $\sigma$ and we want to insert customer $\alpha$ between $i$ and $i+1$. To check if the new route $\tilde{\sigma} = \{0, 1, \ldots, i, \alpha, i+1, \ldots, m, m+1\}$ is feasible and to calculate the minimal waiting time, we will calculate the forward start intervals and corresponding waiting times at customer $\alpha$ by the forward recursion given in (8) and (9). These forwards start intervals $\overline{F}_\alpha$ are compared with the dominant backward start intervals of customer $i+1$ and route $\tilde{\sigma}$ is feasible if there exists $f \in \overline{F}_\alpha$ and $b \in \overline{B}_{i+1}$ such that $E^F_\alpha(f) + s_\alpha + t_{\alpha,i+1} \leq L^B_{i+1}(b)$. The corresponding total waiting time to this combination is $w^F_\alpha(f) + w^B_{i+1}(b) + \max\{0, E^B_{i+1}(b) - L^F_\alpha(f) - s_\alpha - \tau_{\alpha,i+1}\}$. The minimum waiting time of the new route $\tilde{\sigma}$ is given by $\min\{w^R(f,b) | f \in \overline{F}_\alpha, b \in \overline{B}_{i+1}$ with $E^F_\alpha(f) + s_\alpha + \tau_{\alpha,i+1} \leq L^B_{i+1}(b)\}$.

The computational complexity to check if an insertion is feasible and to calculate the new minimal total waiting time is $O(|T_\alpha| + \sum_{j=1}^{m} |T_j - 1|)$.

The same approach can be used for removal and insertion of a sequence of customers, but also for more complex operators such as 2-Opt* or Cross exchanges. Since most operators split a route into two or more sections, we can efficiently check whether a move that connects customers $i$ and $j$ is feasible by comparing the forward and backward start intervals of customer $i$ and $j$, respectively.

# 4 Metaheuristic

Metaheuristics based on (large) variable neighborhood search (VNS, see Mladenović and Hansen [1997]) are simple and effective algorithms that have been successful in solving many variants of the vehicle routing problem. Stenger et al. [2013] proposed an adaptive mechanism that was inspired by the work of Ropke and Pisinger [2006] on large neighborhood search. It combined various alternative components of a VNS that led to an overall Adaptive VNS that outperformed the standard VNS both in terms of solution quality and convergence speed. Therefore, we propose an Adaptive Variable Neighborhood Search (AVNS) to solve the VRPMTW. In Section 5, the proposed AVNS will be used to compare the average performance of the route duration minimization algorithms. This section presents the structure of the proposed metaheuristic framework (shown in Algorithm 2) and discusses its components.

The heuristic is initialized with a feasible solution found by applying the route minimization heuristic of Nagata and Braysy [2009] that is described in Section 4.1. This solution is the start of the AVNS, in which local search and shaking are iteratively called. A granular local search with multiple operators is used to find a new local optimum $x'$ (line 4) as described in Section 4.2. The new local optimum $x'$ is always accepted if it improves the incumbent $x''$. Inspired by the

**Algorithm 2** Adaptive variable neighborhood search

1: Initialization: Create an initial feasible solution $x$ by the RM heuristic, $x^* = x'' = x$
2: Initialize $k$th neighborhood $k = 1$
3: **while** number of iterations $\leq I$ AND elapsed time$\leq T$ **do**
4: *local search:* perform local search on $x$ to determine local optimum $x'$
5: *acceptance criteria:*
6: **if** $x'$ improves $x''$ or $x'$ is accepted **then**
7: $x'' = x'$ and $k = 1$
8: **if** current best solution $x''$ improves best solution $x^*$ **then**
9: $x^* = x''$
10: **end if**
11: **else**$k = k + 1$
12: **end if**
13: *adaptive shaking:* generate solution $x \in \mathcal{N}_k(x'')$ using selected shaking method
14: **if** large shaking condition holds **then**
15: generate new solution $x$ by removing and reinserting $r\%$ of the customers
16: $x'' = x$
17: **end if**
18: **end while**

simulated annealing approach (Kirkpatrick et al. [1983]), we try to diversify the search by accepting non improving solutions with probability $\exp(-(V(x'') - V(x'))/\theta)$, where $V(x)$ is the objective value defined in Section 4.2. As usual, the temperature $\theta$ is initially set to $\theta_0$ and updated after every shake with factor $\theta^{update}$. After $I^\theta$ non-improving main AVNS iterations, $\theta$ is reset to $\theta_0$. In the shaking phase, a set of $k_{max}$ neighborhood structures is used. When a solution is accepted (line 7), the search restarts at the first neighborhood $k = 1$, or otherwise it continues with the next neighborhood $k = k+1$. In the shaking phase (line 13), a new solution $x$ is generated from the $k$th neighborhood of the current local optimum $x''$, i.e., $x \in \mathcal{N}_k(x'')$. Instead of using a random shaking, three shaking methods are proposed and the weights of these methods are adaptively adjusted over time. The details of this shaking procedure are described in Section 4.3. When the incumbent is not improved for $S$ iterations, a large shaking is performed and the local optimum $x''$ is reset to restart the search (see line 14-17 and Section 4.3).

## 4.1 Initial solution

To minimize the number of vehicles, we use the route minimization (RM) heuristic of Nagata and Braysy [2009]. The algorithm starts with an initial solution in which each customer is serviced by a separate vehicle. The RM heuristic attempts to reduce the number of routes by removing one route from the solution. The unserviced customers are put in the ejection pool. At each iteration, a customer from the ejection pool is inserted in the position that minimizes the penalized objective function $V(x)$ defined in the next section. If the new solution is infeasible, we try to restore the

solution by local search moves that minimize the capacity and time window violations. If this fails, up to three customers are removed from the existing routes and are placed in the ejection pool. The ejected customers are selected using the concept of guided local search (Voudouris and Tsang [2003]), which estimates the difficulty of re-inserting the ejected customers. After customers are ejected from a route, the search is diversified by 20 local search moves (described in the next section). This process is repeated until the ejection pool is empty. Then, the next route is selected for removal. The RM heuristic stops if the number of vehicles is equal to the lower bound $\lceil \sum_{i=1}^{N} \frac{q_i}{Q} \rceil$ or if the maximum execution time $T^{max}$ is reached.

### 4.2 Local search

A local search is performed to find the local optimum solution. In the local search, seven different neighborhoods are explored in increasing order of complexity. If an improving solution is found, the search restarts at the first neighborhood. First, intra-route neighborhood operations on a single route are applied. This includes the *relocate* neighborhood in which a customer is replaced to a new best position in the same route, *exchange* in which two customers exchange positions, and the *relocate sequence* neighborhood in which a sequence of at least two and at most four customers is relocated to a new best position. Second, inter-route neighborhood operations on two routes are applied. This includes the *relocate* neighborhood that repositions a customer to the best position of another route, *exchange* that swaps the positions of two customers, *2-Opt** that exchanges the tails of two routes, and *cross* that exchanges two sequences of at most four customers of two routes.

To speed up the local search, a granular neighborhood is used in which the arc set is restricted. Our restricted arc set consists of the arcs from each customer to its $c\%$ closest customers. Furthermore, all depot arcs are included, since Schneider et al. [2017] have shown that this improves solution quality. As described in Toth and Vigo [2003], in the local search, only the moves with a generator arc in the restricted arc set are performed. The generator arc is the newly inserted arc connecting a non-moved customer with a moved customer. After choosing the generator arc $(i, j)$, the other arcs involved in a neighborhood move are uniquely specified. Arcs that do not belong to the restricted arc set can still be inserted, since only the generator arc is checked. The search starts with $c = c_0\%$, and if the local optimum does not improve for $I^c$ iterations, the granular neighborhood increases by $c^{update}\%$ up to a maximum of $c_{max}\%$. If the local optimum solution is improved, $c$ is decreased to $c = c_0$.

During the search, we allow for the violation of the capacity and time window constraints at the expense of a penalty to reach different areas of the solution space. Let $x$ be a solution. Then, the overload $q(x)$ is calculated by $q(x) = \sum_{k=1}^{|K|} \max\{\sum_{(i,j)\in A} q_i x_{ij}^k - Q, 0\}$, and the time window violation is given by $t(x) = \sum_{i=1}^{n} \max\{a_i - l_i^{|T_i|}, 0\}$. Let $F \times m$ be the total vehicle cost, with $F$ the cost per used vehicle and $m$ the number of vehicles used in solution $x$. The total duration of all $m$ routes is denoted by $d(x)$. Therefore, the penalized objective function of solution $x$ is given

by $V(x) = F \times m + d(x) + \beta_1 q(x) + \beta_2 t(x)$, where $\beta_1$ and $\beta_2$ are self-adjusting parameters that are updated every $\mu$ iterations. If in the last $\mu$ iterations the capacity constraint is violated in at least one route, then parameter $\beta_1$ is multiplied by $\delta > 1$. Otherwise, $\beta_1$ is divided by $\delta$. The same update rule is applied to parameter $\beta_2$.

### 4.3 Shaking

An adaptive shaking approach based on Stenger et al. [2013] is used to find a new starting point for the local search. In this shaking approach, one move of the $k$th neighborhood is performed. The $k_{max} = 18$ neighborhoods used in the shaking are all based on inter-route cross exchanges that exchange sequences of at most $L$ customers. For neighborhoods with length $L < 4$, the sequence length is a random value between one and the maximum length $L$. For the neighborhoods with $L \geq 4$, the sequence lengths are fixed to $L$. For neighborhoods $k = 1$ until $k = 6$, one sequence has length zero and the other has length $L = 1$ until $L = 6$, respectively. Therefore, these neighborhoods' operations are relocations of a sequence with $L$ customers. For the next neighborhoods $k = 7$ until $k = 12$, exchange two sequences of length $L = 1$ until $k = 6$, respectively. In the last neighborhoods ($k = 13$ until $k = 18$), three routes are involved in exchanging sequences of length $L = 1$ until $k = 6$.
When the shaking approach of neighborhood $k$ is called, first, the routes involved are selected. Second, the length of the sequences are fixed (if $L < 4$). Last, the customer sequences involved in the move are selected.

The first route involved in the shaking operation is randomly selected. To avoid unpromising moves, the other routes involved in the operation have to be located close to the first selected route. The probability of selecting a route $r$ corresponds to the number of arcs in the network that connect the customers in route $r$ to the customers in the first selected route. Since a granular neighborhood is used, this probability measures the closeness of the routes.

For a neighborhood with $L < 4$, the length of the sequences to be exchanged is determined by randomly selecting a value between 1 and $L$.

The selection of the customer sequences is done by one of these three methods:
1. All sequences involved are randomly selected.
2. The sequence of the first route is randomly selected and the sequences of the other routes are selected such that the objective of the resulting solution is minimized.
3. All sequences are selected such that the objective of the resulting solution is minimized.
Similar to Stenger et al. [2013], the weight of selecting one of these methods is updated by an adaptive mechanism. During the search, the number of times a shaking method $i$ is selected (denoted by $c_i$) is counted. When shaking method $i$ leads to an improvement of the local best solution ($x''$), a value of two is added to the performance score $s_i$. If the overall best solution ($x^*$) is improved, six additional points are added to $s_i$. After $z$ shaking rounds, the weights of the shaking types

are updated based on the newly found scores: $w_i = \rho w_i^n + (1-\rho)w_i$, with $w_i^n = \lceil \frac{s_i}{c_i} \rceil / \sum_i \lceil \frac{s_i}{c_i} \rceil$ and $1 \leq i \leq 3$ and $0 \leq \rho \leq 1$.

A large shaking is called if the local optimum $x''$ does not improve for $S$ iteration, i.e., if the adaptive shaking proves to be insufficient to escape from a local optimum. In that case, $r\%$ of the customers are deleted from the routes and reinserted in random order into the best position. The search is restarted from this solution by resetting the local optimum $x''$ as described in line 16 of Algorithm 2. This larger shaking procedure is also called $s$ iterations after the last restart if the current solution is not within $v\%$ of the overall best solution $x^*$. The search stops after $\eta$ restarts, if a total of $I$ iterations are performed or if the time limit of $T$ seconds is reached.

# 5 Computational Results

In this section, we evaluate the impact of the exact start interval algorithm we proposed towards the solution of the VRPMTW. To this end, we compare our exact algorithm with other route duration minimization algorithms presented in the literature. Furthermore, we incorporate our algorithm into a new and effective AVNS, which is successfully used to solve the VRPMTW instances from the literature. The algorithms are implemented in C++ and executed on a single core of a Windows 10 computer with a 4.0 GHz Intel core i7 processor and 24GB memory.

To tune the parameters of the AVNS, the strategy described by Ropke and Pisinger [2006] is used. During the tuning, one parameter value is changed while the other parameters stay fixed. For each parameter, three runs on a randomly selected subset of the instance set are conducted (with two instances out of each set C1, C2, R1, R2, RC1 and RC2). The parameter value with the best average results is selected and this process is repeated for all parameters, resulting in the following parameter setting. The granular neighborhood starts at $c_0 = 10\%$ and increases by $c^{update} = 10\%$ after $I^c = 200$ non-improving iterations until a maximum of $c_{max} = 40\%$. The simulated annealing components are set to $\theta_0 = 20$, $\theta^{update} = 0.85$ and $I^\theta = 20$. The penalty parameters are adjusted every $\mu = 20$ iterations with factor $\delta = 1.2$ and the parameters are initialized at $\beta_1 = \beta_2 = \beta_3 = 100$. After $z = 20$ main AVNS iterations, the weights of the different shaking methods are updated with $\rho = 0.3$. In the large shaking, $r = 30\%$ of the customers are relocated. The large shaking phase is called after $S = 5,000$ non-improving iterations or if after $s = 2,000$ iterations the current solution is not within $v = 10\%$ of the overall best solution. The search is stopped after $\eta = 10$ large shakes, after $I = 100,000$ iterations or if the time limit of $T = 150$ seconds is reached. The RM heuristics runs for $T_{max} = 8$ seconds.

## 5.1 Dataset

Belhaiza et al. [2014] generated VRPMTW instances from the Solomon [1987] VRPTW instances, all of which contain 100 customers. Only the first eight instances of every Solomon [1987] set are

used. The vehicle cost is set equal to the vehicle capacity, i.e., the vehicle costs $F$ are 200, 700, 200, 1000, 200 and 1000 for instance sets CM1, CM2, RCM1, RCM2, RM1 and RM2, respectively. The instances have from one to ten non-overlapping time windows per customer and the optimal solutions of the instances are unknown. The specific details of the instances can be found in Table 1.

The published objective values of Belhaiza et al. [2014] include travel time, waiting time, service time, and vehicle cost. Since service time is a constant and can represent a large part of the objective value, we do not include it in our objective value in order to avoid distorting the calculation of savings. For example, for instance set CM, the fixed total service time is 9000, while the total travel time plus waiting time is between 820 and 1500.

Similar to Tricoire et al. [2010], the preprocessing steps described in Appendix A are performed before running the benchmark heuristic. In these steps, the time windows that conflict with the time window of the depot are adjusted or eliminated.

### 5.2 Comparison of the different exact algorithms

Section 3.2 demonstrates that our start interval algorithm has a lower worst-case complexity than the existing exact route duration minimization algorithms of Tricoire et al. [2010] and Belhaiza et al. [2014]. In this section, we compare the average performance of our route duration minimization algorithm to the existing algorithms. To do so, we implement the exact algorithms described in Tricoire et al. [2010] and Belhaiza et al. [2014]. In Belhaiza et al. [2014] and in our algorithm, the service must start within a time window. However, in Tricoire et al. [2010], the service of a customer must both start and end in a single selected time window. To align the settings of the algorithms, we had to make a small and straightforward adjustment to the input for Tricoire's algorithm. The upper bound of the time window of each customer is increased by its service time. As a result, the time windows could be overlapping. However, this does not impact the algorithm of Tricoire et al. [2010], as it still schedules the start of service within the original time windows.

Our start interval algorithm is implemented in two different ways. First, it uses a depth-first implementation that stops as soon as a solution with zero waiting time has been found (see Appendix C). This implementation uses only the forward start intervals and is useful for comparing our method with existing methods that calculate the minimum duration of a given route. Second, it applies of an embedded start interval algorithm that uses both the backward and forward start intervals, as described in Section 3.4.

In the first experiment, the running times of the exact route duration minimization algorithms are compared. In the second experiment, we show the benefits of using the embedded start interval algorithm in which the local search moves are efficiently evaluated compared to using a standard route duration minimization algorithm. Furthermore, the impact of using exact and approximate evaluations for local search moves is tested.

### 5.2.1 Average case analysis of exact evaluations

In this section, we compare the average running times of the newly proposed depth-first start interval algorithm (DFSI) to the existing route duration minimization algorithms of Tricoire et al. [2010] (TRDH10), and Belhaiza et al. [2014] (BHL14). To perform this comparison, we first use the AVNS described in Section 4 to generate a solution for each problem instance. For every route in these solutions, the minimal duration is calculated by the three algorithms. Because the computational times of the algorithms are small, the minimum duration of each route is calculated ten times to better highlight the differences. Per algorithm, the summed computational times of the ten runs are reported in seconds in Table 1. Note that because the minimum total duration of every algorithm is the same, it is not reported in Table 1.

| instance | $m$ | nTW | width TW | TRDH10 | BHL14 | DFSI | instance | m | nTW | width TW | TRDH10 | BHL14 | DFSI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| cm101 | 10 | 5-10 | 50-100 | 0.099 | 0.091 | 0.034 | cm201 | 5 | 5-10 | 50-100 | 0.100 | 0.058 | 0.034 |
| cm102 | 11 | 5-7 | 50-100 | 0.089 | 0.065 | 0.031 | cm202 | 6 | 5-7 | 50-100 | 0.091 | 0.089 | 0.032 |
| cm103 | 11 | 3-7 | 50-100 | 0.075 | 0.055 | 0.030 | cm203 | 5 | 3-7 | 50-100 | 0.078 | 0.058 | 0.030 |
| cm104 | 13 | 3-5 | 50-100 | 0.073 | 0.053 | 0.029 | cm204 | 5 | 3-5 | 50-100 | 0.071 | 0.050 | 0.028 |
| cm105 | 10 | 2-5 | 50-100 | 0.064 | 0.047 | 0.028 | cm205 | 4 | 2-5 | 100-200 | 0.065 | 0.050 | 0.028 |
| cm106 | 10 | 2-4 | 100-200 | 0.061 | 0.047 | 0.025 | cm206 | 4 | 2-4 | 100-200 | 0.063 | 0.046 | 0.027 |
| cm107 | 10 | 1-3 | 100-200 | 0.054 | 0.040 | 0.025 | cm207 | 4 | 1-3 | 100-300 | 0.051 | 0.040 | 0.027 |
| cm108 | 10 | 1-3 | 100-500 | 0.052 | 0.040 | 0.026 | cm208 | 4 | 1-3 | 100-500 | 0.054 | 0.040 | 0.025 |
| average | 10.6 | | | 0.071 | 0.055 | 0.029 | average | 4.6 | | | 0.072 | 0.054 | 0.029 |
| | | | | | | | | | | | | | |
| rcm101 | 10 | 5-10 | 10-30 | 0.080 | 0.103 | 0.030 | rcm201 | 2 | 5-10 | 50-100 | 0.081 | 0.060 | 0.029 |
| rcm102 | 10 | 5-7 | 10-30 | 0.071 | 0.064 | 0.029 | rcm202 | 2 | 5-7 | 50-100 | 0.075 | 1.030 | 0.030 |
| rcm103 | 10 | 3-7 | 10-50 | 0.064 | 0.050 | 0.027 | rcm203 | 2 | 3-7 | 50-100 | 0.070 | 0.051 | 0.026 |
| rcm104 | 10 | 3-5 | 10-50 | 0.063 | 0.052 | 0.030 | rcm204 | 2 | 3-5 | 50-100 | 0.062 | 0.050 | 0.028 |
| rcm105 | 10 | 2-5 | 10-70 | 0.060 | 0.045 | 0.024 | rcm205 | 2 | 2-5 | 100-200 | 0.055 | 0.063 | 0.027 |
| rcm106 | 10 | 2-4 | 30-70 | 0.058 | 0.042 | 0.027 | rcm206 | 2 | 2-4 | 100-200 | 0.055 | 0.052 | 0.023 |
| rcm107 | 11 | 1-3 | 30-70 | 0.051 | 0.040 | 0.025 | rcm207 | 3 | 1-3 | 100-300 | 0.047 | 0.041 | 0.024 |
| rcm108 | 11 | 1-3 | 30-100 | 0.050 | 0.039 | 0.024 | rcm208 | 2 | 1-3 | 100-500 | 0.046 | 0.032 | 0.023 |
| average | 10.3 | | | 0.062 | 0.054 | 0.027 | average | 2.1 | | | 0.061 | 0.172 | 0.026 |
| | | | | | | | | | | | | | |
| rm101 | 10 | 5-9 | 10-30 | 0.086 | 0.096 | 0.030 | rm201 | 2 | 5-8 | 50-100 | 0.091 | 0.060 | 0.029 |
| rm102 | 9 | 5-7 | 10-30 | 0.080 | 0.057 | 0.027 | rm202 | 2 | 3-5 | 50-100 | 0.073 | 0.055 | 0.029 |
| rm103 | 9 | 4-7 | 10-30 | 0.073 | 0.050 | 0.026 | rm203 | 2 | 2-5 | 50-100 | 0.065 | 0.052 | 0.027 |
| rm104 | 9 | 3-6 | 10-30 | 0.070 | 0.059 | 0.029 | rm204 | 2 | 2-4 | 50-100 | 0.061 | 0.049 | 0.027 |
| rm105 | 9 | 2-6 | 10-30 | 0.065 | 0.049 | 0.027 | rm205 | 2 | 1-4 | 50-100 | 0.056 | 0.045 | 0.026 |
| rm106 | 9 | 2-3 | 30-50 | 0.053 | 0.043 | 0.027 | rm206 | 2 | 1-3 | 100-200 | 0.053 | 0.058 | 0.021 |
| rm107 | 9 | 1-3 | 30-50 | 0.053 | 0.041 | 0.028 | rm207 | 2 | 1-3 | 100-200 | 0.051 | 0.044 | 0.025 |
| rm108 | 9 | 1-2 | 50-100 | 0.047 | 0.037 | 0.026 | rm208 | 2 | 1-5 | 100-200 | 0.053 | 0.048 | 0.027 |
| average | 9.125 | | | 0.066 | 0.054 | 0.028 | average | 2 | | | 0.063 | 0.051 | 0.026 |

Table 1: Average computational times in seconds of the different route duration minimization algorithms.

The first column of Table 1 reports the instances, and the second column indicates the number

of routes $m$ of the solution. The number of time windows and the width of the time windows are indicated in the third and fourth columns, respectively. Note that all instances contain 100 customers.

The results show that the computational times of all algorithms increase if more time windows are taken into account. The BHL14 algorithm is the most sensitive to the number of time windows. For example, for instances RCM20, the average computational times are between 0.03 and 1.03. These results are in line with the computational complexity analysis of the BHL14 algorithm. In the worst-case scenario, the computational time of the BHL14 algorithm can indeed be high. For example, in instance rcm202, the computational time of BHL14 is 34 times higher than of the DFSI algorithm. In this instance, the waiting times of the routes are not equal to zero, and therefore many combinations of time windows have to be checked. In instances rcm201, rcm203 and rcm204, the waiting times of the routes are zero, thus resulting in a lower computational time for BHL14. The computational times of the TRDH10 algorithm are slightly lower for short routes and for instances with tighter time windows. In almost all instances, the BHL14 method outperforms the TRDH10, but on some long routes with many customers, the TRDH10 has a lower computational time.

The proposed DFSI outperforms the other algorithms in average computational time. The DFSI has the lowest computational time on all instances. On average, it is 2.7 and 2.4 times faster than the approaches of BHL14 and TRDH10, respectively. The computational times of the DFSI algorithm are between 0.021 and 0.034 seconds for all instances, thus indicating that it is not much influenced by specific problem characteristics.

### 5.2.2 Impact of embedded forward start interval

In the second test, the performance of different methods to evaluate neighborhood solutions in a local search is tested. Since the previous test showed that the proposed DFSI algorithm outperforms the existing route duration minimization algorithms, we only compare the average computational times of the depth-first start interval (DFSI) algorithm and the embedded start interval algorithm (ESI). These are both exact methods to calculate the minimal duration of a route. We also implemented an approximate evaluation method in which the waiting time and time window infeasibility of a route are approximated by servicing the customers as early as possible and adding up the waiting times and delays at every customer in the route. The approximate method is used when evaluating a move and the exact DFSI algorithm is used only when a move is performed. The AVNS is used to test the three different evaluation methods with a maximum number of $I = 50,000$ iterations and no time limit. The obtained average results on the different instance sets are summarized in Table 2. The first column gives the name of the instance set. The next three columns represent the results for the approximate evaluation method. The average number of vehicles used, the average duration, and the average computational time in seconds are reported

| | Approximate | | | Exact | | | |
|---|---|---|---|---|---|---|---|
| Instance | nVeh | Duration | time | nVeh | Duration | DFSI | ESI |
| CM1 | 10.6 | 1277.5 | 34.9 | 10.6 | 1198.7 | 91.7 | 44.1 |
| CM2 | 4.6 | 1038.1 | 57.0 | 4.6 | 982.4 | 128.6 | 62.5 |
| RCM1 | 10.3 | 1234.8 | 33.6 | 10.3 | 1214.7 | 78.9 | 47.7 |
| RCM2 | 2.1 | 809.2 | 99.8 | 2.1 | 769.1 | 269.6 | 100.7 |
| RM1 | 9.1 | 959.7 | 40.5 | 9.1 | 936.9 | 101.4 | 51.6 |
| RM2 | 2.0 | 725.5 | 109.2 | 2.0 | 705.1 | 346.8 | 145.3 |
| Average | 6.5 | 1007.5 | 62.5 | 6.5 | 967.8 | 169.5 | 75.3 |

Table 2: The average number of vehicles, average total duration, and average running time in seconds, using different evaluation algorithms for different instance sets, with $I = 50,000$ and no time limit.

in columns 'nVeh', 'Duration' and 'time', respectively. The results for the exact evaluations are reported in the last four columns, where columns 'DFSI' and 'ESI' show the average computational times of the corresponding exact evaluation method. Note that the two algorithms follow exactly the same optimization path and thus yield the same final solution.

The results in Table 2 show a trade-off between solution quality and computational time when comparing the approximate evaluation approach with the exact evaluation approach. Using exact evaluations during the local search reduces the average duration by 4% compared to the approximate evaluation approach. However, the approximate method is on average 2.7 times faster than the exact DFSI algorithm, but only 1.2 times faster than the exact ESI algorithm. The ESI is on average 2.25 times faster than the DFSI method. Since in the previous test is shown that the DFSI algorithm is on average twice as fast as the existing route duration minimization algorithms, we can conclude that the ESI results in an average computational saving of a factor of 4 compared to using the existing exact algorithms for local search evaluations. This shows that the efficient evaluation method using the forward and backward start intervals significantly accelerates the evaluation of local search moves.

### 5.3 Comparison to state-of-the-art results

In the final experiment, we compare the performance of our proposed AVNS with state-of-the-art results published in Belhaiza et al. [2014] (BHL14) and those reported in the recent conference paper of Belhaiza et al. [2017] (BMB17). Our proposed AVNS uses either the embedded start interval algorithm (denoted by E-AVNS) or the approximate evaluation method (denoted by A-AVNS) described in the previous section to test the impact of exact local search evaluations.

Belhaiza et al. [2014] used a computer with a 3.3 GHz Intel core i5 vPro processor and 3.2GB RAM and performed ten runs of at most 25,000 iterations per instance. Belhaiza et al. [2017] performed a single run of 100,000 iterations on a computer with a 3.3 GHz Intel Core i5 vPro

processor and 4GB RAM. Our proposed AVNS runs one time for 100,000 iterations. The computational results are presented per instance in Table 3. For each solution method, the average number of vehicles used and the total duration (i.e., the total travel time plus the total waiting time) are reported in columns 'nVeh' and 'Dur', respectively. The objective value consisting of the duration and vehicle costs is presented in column 'Obj'.

The best-known solution for every instance is given in bold. Out of the 48 instances, the E-AVNS improved 39 instances of the best published results of Belhaiza et al. [2014] and 22 instances compared to Belhaiza et al. [2017]. The RM heuristic used in the AVNS performs very well, since the AVNS always finds the solution with the minimal number of vehicles. The number of vehicles used in the AVNS is on average 1.3% and 1.9% lower than those of BMB17 and BHL14, respectively. The E-AVNS has on average the lowest objective value and the BMB17 has on average the lowest duration.

The two-sided sign test shows that the algorithms E-AVNS and BMB17 do not dominate each other. With a $p$-value of 0.77, the null hypothesis that there is no significant difference between the solution quality of the E-AVNS and BMB17 is not rejected. On the other hand, the significantly better results of the E-AVNS over the method of BHL14 are confirmed by the two-sided sign test with a significance level of $1.5 \times 10^{-5}$. The Wilcoxon two-sided rank test confirms these results.

In Table 4, the average running time and the average execution time to reach the best result in seconds are given in 'Total time' and 'Time best', respectively. BHL14 has an average computational time of 70.2 seconds per run, but they run their algorithm ten times. BMB17 reports an average execution time of 81.2 seconds to reach the best solution per instance. The average computational time to reach the best solution of the E-AVNS is lower, but we are using a slightly faster computer. Therefore, we consider the computational times of BMB17 and E-AVNS to be comparable.

Lastly, the impact of the exact evaluations in the local search can be found by comparing the E-AVNS with the A-AVNS. The average total computational time of the E-AVNS is 14.4% higher than the A-AVNS, but for both methods the calculation times remain very short. The duration of the E-AVNS is lower in 46 of the 48 instances with an average improvement of 3.5%. The A-AVNS is not able to find any best-known solutions of the benchmark instances. Therefore, we conclude that including the proposed efficient exact route duration minimization algorithm enables finding state-of-the-art results in a simple metaheuristic framework.

# 6 Conclusions

In the Vehicle Routing Problem with Multiple Time Windows, the duration of a route can be minimized by determining the optimal selection of time windows. Routing problems with multiple time windows occur frequently in practice but receive relatively little attention in the literature. Tricoire et al. [2010] and Belhaiza et al. [2014] developed exact algorithms to minimize the duration of a given route. Belhaiza et al. [2014] use the same exact approach for evaluating a move in the

| | BHL14 | | | BMB17 | | | A-AVNS | | | E-AVNS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | nVeh | Dur | Obj | nVeh | Dur | Obj | nVeh | Dur | Obj | nVeh | Dur | Obj |
| cm101 | 10 | 1320 | 3320 | 10 | 1319.1 | **3319.1** | 10 | 1496.47 | 3496.47 | 10 | 1345.4 | 3345.4 |
| cm102 | 12 | 1092.1 | 3492.1 | 11 | 1210.7 | **3410.7** | 11 | 1258.67 | 3458.67 | 11 | 1282.3 | 3482.3 |
| cm103 | 12 | 1241.1 | 3641.1 | 12 | 1232.4 | 3632.4 | 11 | 1466.72 | 3666.72 | **11** | 1392.2 | **3592.2** |
| cm104 | 14 | 1287.8 | 4087.8 | 14 | 1298 | 4098 | 13 | 1378.44 | 3978.44 | **13** | 1327.8 | **3927.8** |
| cm105 | 11 | 883.4 | 3083.4 | 10 | 1027 | **3027** | 10 | 1097.5 | 3097.5 | 10 | 1066.3 | 3066.3 |
| cm106 | 10 | 1073.9 | 3073.9 | 10 | 1059 | **3059** | 10 | 1191.3 | 3191.3 | 10 | 1066.4 | 3066.4 |
| cm107 | 11 | 1124.2 | 3324.2 | 11 | 1118 | 3318 | 10 | 1138.62 | 3138.62 | **10** | 1108.4 | **3108.4** |
| cm108 | 10 | 990.4 | 2990.4 | 10 | 986 | 2986 | 10 | 1010.29 | 3010.29 | 10 | 985.9 | **2985.9** |
| cm201 | 5 | 1020.1 | 4520.1 | 5 | 998.8 | 4498.8 | 5 | 1076.64 | 4576.64 | 5 | 968.4 | **4468.4** |
| cm202 | 6 | 827.3 | 5027.3 | 6 | 825.1 | 5025.1 | 6 | 879.55 | 5079.55 | 6 | 820.2 | **5020.2** |
| cm203 | 5 | 997.2 | 4497.2 | 5 | 965.8 | **4465.8** | 5 | 1062.59 | 4562.59 | 5 | 986.5 | 4486.5 |
| cm204 | 5 | 859.8 | 4359.8 | 5 | 844 | **4344** | 5 | 899.405 | 4399.405 | 5 | 856.9 | 4356.9 |
| cm205 | 4 | 1084.1 | 3884.1 | 4 | 1027.8 | **3827.8** | 4 | 1107.74 | 3907.74 | 4 | 1096.8 | 3896.8 |
| cm206 | 4 | 967.7 | 3767.7 | 4 | 913.2 | **3713.2** | 4 | 991.122 | 3791.122 | 4 | 933.4 | 3733.4 |
| cm207 | 4 | 1209.7 | 4009.7 | 4 | 1163.7 | **3963.7** | 4 | 1207.54 | 4007.54 | 4 | 1163.7 | **3963.7** |
| cm208 | 4 | 988.1 | 3788.1 | 4 | 949.7 | **3749.7** | 4 | 983.701 | 3783.701 | 4 | 956.8 | 3756.8 |
| rcm101 | 10 | 1098.9 | 3098.9 | 10 | 1081.2 | 3081.2 | 10 | 1093.25 | 3093.25 | 10 | 1080.6 | **3080.6** |
| rcm102 | 10 | 1222.6 | 3222.6 | 10 | 1188.3 | 3188.3 | 10 | 1194.78 | 3194.78 | 10 | 1184.3 | **3184.3** |
| rcm103 | 10 | 1174.3 | 3174.3 | 10 | 1150.4 | 3150.4 | 10 | 1171.04 | 3171.04 | 10 | 1148.3 | **3148.3** |
| rcm104 | 10 | 1156.3 | 3156.3 | 10 | 1144 | 3144 | 10 | 1157.84 | 3157.84 | 10 | 1141.2 | **3141.2** |
| rcm105 | 10 | 1216.7 | 3216.7 | 10 | 1207 | **3207** | 10 | 1233.32 | 3233.32 | 10 | 1208.2 | 3208.2 |
| rcm106 | 10 | 1219.9 | 3219.9 | 10 | 1181.7 | **3181.7** | 10 | 1211.12 | 3211.12 | 10 | 1191.8 | 3191.8 |
| rcm107 | 11 | 1342.4 | 3542.4 | 11 | 1321.5 | 3521.5 | 11 | 1325.46 | 3525.46 | 11 | 1316.5 | **3516.5** |
| rcm108 | 11 | 1414.5 | 3614.5 | 11 | 1365.2 | **3565.2** | 11 | 1389.46 | 3589.46 | 11 | 1366.2 | 3566.2 |
| rcm201 | 2 | 783.6 | 2783.6 | 2 | 783.2 | **2783.2** | 2 | 890.075 | 2890.075 | 2 | 800.1 | 2800.1 |
| rcm202 | 2 | 847.1 | 2847.1 | 2 | 779.4 | **2779.4** | 2 | 838.434 | 2838.434 | 2 | 822.9 | 2822.9 |
| rcm203 | 2 | 721.9 | **2721.9** | 2 | 722 | 2722 | 2 | 814.213 | 2814.213 | 2 | 771.7 | 2771.7 |
| rcm204 | 2 | 726.5 | 2726.5 | 2 | 708.5 | **2708.5** | 2 | 758.866 | 2758.866 | 2 | 716.0 | 2716.0 |
| rcm205 | 2 | 754.5 | 2754.5 | 2 | 754.5 | **2754.5** | 2 | 796.543 | 2796.543 | 2 | 756.0 | 2756.0 |
| rcm206 | 2 | 812.7 | 2812.7 | 2 | 803.3 | 2803.3 | 2 | 822.724 | 2822.724 | 2 | 725.0 | **2725.0** |
| rcm207 | 3 | 764.2 | 3764.2 | 3 | 761.5 | 3761.5 | 3 | 789.891 | 3789.891 | 3 | 757.1 | **3757.1** |
| rcm208 | 2 | 791.4 | 2791.4 | 2 | 742.7 | 2742.7 | 2 | 759.418 | 2759.418 | 2 | 735.1 | **2735.1** |
| rm101 | 10 | 1041.9 | 3041.9 | 10 | 1027.1 | 3027.1 | 10 | 1064.54 | 3064.54 | 10 | 1026.1 | **3026.1** |
| rm102 | 9 | 965.1 | 2765.1 | 9 | 951.2 | **2751.2** | 9 | 988.636 | 2788.636 | 9 | 974.8 | 2774.8 |
| rm103 | 9 | 908.5 | 2708.5 | 9 | 903 | 2703 | 9 | 927.014 | 2727.014 | 9 | 900.6 | **2700.6** |
| rm104 | 9 | 918 | 2718 | 9 | 901.2 | **2701.2** | 9 | 933.155 | 2733.155 | 9 | 907.1 | 2707.1 |
| rm105 | 9 | 888.8 | 2688.8 | 9 | 887.2 | **2687.2** | 9 | 913.245 | 2713.245 | 9 | 890.5 | 2690.5 |
| rm106 | 9 | 892.9 | **2692.9** | 9 | 908.4 | 2708.4 | 9 | 927.626 | 2727.626 | 9 | 914.8 | 2714.8 |
| rm107 | 9 | 901.4 | 2701.4 | 9 | 892.8 | **2692.8** | 9 | 916.079 | 2716.079 | 9 | 900.4 | 2700.4 |
| rm108 | 9 | 929.1 | 2729.1 | 9 | 922.6 | **2722.6** | 9 | 935.899 | 2735.899 | 9 | 938.1 | 2738.1 |
| rm201 | 3 | 808.2 | 3808.2 | 3 | 805.4 | 3805.4 | 2 | 925.931 | 2925.931 | **2** | 888.9 | **2888.9** |
| rm202 | 2 | 739 | 2739 | 2 | 706.8 | **2706.8** | 2 | 753.957 | 2753.957 | 2 | 721.9 | 2721.9 |
| rm203 | 2 | 710.3 | 2710.3 | 2 | 696.9 | 2696.9 | 2 | 700.178 | 2700.178 | 2 | 693.2 | **2693.2** |
| rm204 | 2 | 691.9 | 2691.9 | 2 | 674.5 | 2674.5 | 2 | 692.692 | 2692.692 | 2 | 671.7 | **2671.7** |
| rm205 | 2 | 689.9 | 2689.9 | 2 | 668.1 | **2668.1** | 2 | 681.688 | 2681.688 | 2 | 668.4 | 2668.4 |
| rm206 | 2 | 703.4 | 2703.4 | 2 | 684.9 | 2684.9 | 2 | 689.184 | 2689.184 | 2 | 672.6 | **2672.6** |
| rm207 | 2 | 701.7 | 2701.7 | 2 | 664.3 | 2664.3 | 2 | 667.886 | 2667.886 | 2 | 662.4 | **2662.4** |
| rm208 | 2 | 682.8 | 2682.8 | 2 | 664.3 | 2664.3 | 2 | 678.367 | 2678.367 | 2 | 663.6 | **2663.6** |
| | 6.58 | 962.2 | 3231.0 | 6.54 | **949.8** | 3210.2 | 6.46 | 997.7 | 3224.8 | **6.46** | 961.9 | **3189.0** |

Table 3: The number of vehicles, total duration, and objective value of Belhaiza et al. [2014], Belhaiza et al. [2017], the A-AVNS and the E-AVNS.

| | BHL14 | BMB17 | A-AVNS | E-AVNS |
|---|---|---|---|---|
| Total time | $70.2 \times 10$ | - | 92.1 | 105.4 |
| Time best | - | 81.2 | 51.7 | 67.6 |

Table 4: The average total running time and time to reach the best solution in seconds for the different heuristics.

local search. In the metaheuristic of Tricoire et al. [2010], the exact algorithm is used only to calculate the minimum duration of a local optimum. An approximation of the route duration is used when evaluating moves in the local search.

In this paper, we present an efficient approach to recalculate the exact route duration when neighborhood operations are evaluated during the local search based on forward and backward start intervals. These forward (backward) start intervals represent the start times of servicing a customer such that the preceding (succeeding) customers in a route are serviced in one of their time windows. Furthermore, forward start intervals can be used to efficiently calculate the minimal duration of a given route (i.e., to determine the optimal selection of time windows).

First, we have formally shown that the forward start interval algorithm has a lower worst-case complexity than the existing exact route duration minimization algorithms of Tricoire et al. [2010] and Belhaiza et al. [2014]. Second, the experimental tests on given routes show that the forward start interval algorithm is at least twice as fast as existing algorithms in terms of average computational effort. Third, the start interval approach is embedded in an adaptive variable neighborhood search in order to efficiently recalculate the exact route duration when neighborhood operations are evaluated in the local search. Experimental tests show that these efficient exact local search evaluations lead to an acceleration by a factor of 4 compared to using the existing exact algorithms in the local search. Furthermore, we show that the exact evaluations significantly improve solution quality compared to an approximate evaluation approach, while the increase in the computational time is relatively small. Finally, our proposed algorithm found new best-known solutions for 22 of the 48 benchmark instances. Therefore, we believe that our proposed algorithm will be useful for solving other routing and scheduling problems involving multiple time windows.

acknowledgment: This work was partially supported by The Dutch Science Foundation under Grant 407-13-050. This support is gratefully acknowledged.

## Appendix A Preprocessing time windows

A preprocessing procedure is applied to remove or adapt the customer time windows that conflict with the time window of the depot $[e_0, l_0]$. For all $i \in V'$ and $t \in T_i$, we set $e_i^t = \max\{e_i^t, e_0 + \tau_{0i}\}$ and if $l_i^t < \tau_{0i}$, then time window $t$ is removed from $T_i$. Similarly, for all $i \in V'$ and $t \in T_i$, we set $l_i^t = \min\{l_i^t, l_0 - \tau_{0i} - s_i\}$ and if $e_i^t > l_0 - \tau_{0i} - s_i$, then time window $t$ is removed from $T_i$. Hence, after the preprocessing step the time window bounds satisfy $\tau_{0i} \leq e_i^t \leq l_i^t \leq l_0 - \tau_{i0} - s_i$ for all

$i \in V'$, and $t \in T_i$.

An additional preprocessing step can be executed to check if the first and last time window of a customer are "useful". If $e_i^2 + s_i + \tau_{ij} \leq e_j^1$ for all $(i,j) \in A$, then the first time window of customer $i$ is useless, since it is possible to use the second time window of customer $i$ and arrive before the first time window at all other customer. Similarly, the latest time window of customer $i$ is useless if $l_i^{|T_i|-1} - \tau_{ji} - s_j \geq l_j^{|T_j|}$ for all $(j,i) \in A$.

# Appendix B Optimality proof of DFSI algorithm

Let $\{\zeta_1^*, \ldots, \zeta_m^*\}$ be the optimal start times of customers sequence $\sigma' = \{1, \ldots, m\}$ resulting in a minimal route duration. In this appendix we show that $\zeta_i^*$ lies in a dominant forward start interval $\forall i \in \sigma'$.

**Theorem 2.** *The optimal start time of servicing customer $i \in \sigma'$ lies in a dominant forward start interval, i.e., $\zeta_i^* \in \overline{F}_i \ \forall i \in \sigma'$.*

*Proof.* We will prove this by induction. Let $\zeta^* = \{\zeta_1^*, \ldots, \zeta_m^*\}$ be the start times of servicing the customers in $\sigma'$ in the optimal solution. By definition $\zeta_1^*$ lies in a dominant forward start interval, which are equal to the time windows $T_1$. Suppose that the theorem holds for $\zeta_1^*, \ldots, \zeta_{i-1}^*$, then we will show that it also holds for customer $i$.

By the induction hypothesis we know that $\zeta_{i-1}^*$ lies in dominant forward start interval $q \in \overline{F}_{i-1}$, i.e., $E_{i-1}^F(q) \leq \zeta_{i-1}^* \leq L_{i-1}^F(q)$. We will distinguish two cases: $\zeta_{i-1}^* + s_{i-1} + \tau_{i-1,i}$ lies in a time window at customer $i$ or not. First, suppose that $\exists t \in T_i$ such that $e_i^t \leq \zeta_{i-1}^* + s_{i-1} + \tau_{i-1,i} \leq l_i^t$. Then $\zeta_i^* = \zeta_{i-1}^* + s_{i-1} + \tau_{i-1,i}$ is optimal since it lies in a time window, so no waiting time occurs at customer $i$, therefore start times $\zeta_i > \zeta_i^*$ will never results in a better solution. Since $E_{i-1}^F(q) \leq \zeta_{i-1}^* \leq L_{i-1}^F(q)$ the dominant forward start interval $q$ at customer $i-1$ and time window $t$ will generate a new dominant forward start interval $p$ at customer $i$, $e_i^t \leq E_i^F(p) \leq \zeta_i^* \leq L_i^F(p) \leq l_i^t$ with waiting time $w_i^F(p) = w_i^F(q)$. Hence, $\zeta_i^*$ lies in a dominant forward start interval

Second, suppose that $\exists t \in T_i$ such that $l_i^{t-1} \leq \zeta_{i-1}^* + s_{i-1} + \tau_{i-1,i} \leq e_i^t$, where the first inequality can be ignored for $t = 1$. To get the solution with minimal waiting time we set $\zeta_i^* = e_i^t$. Since $E_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i} \leq l_i^t$ forward start interval $q$ and time window $t$ generate forward start interval $p$ at customer $i$ with $E_i^F(p) = e_i^t$. Now we only have to show that forward start interval $p$ is dominant.

If $L_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i} \geq e_i^t$, then no waiting time occurs at customer $i$, i.e., $w_i^F(p) = w_i^F(q)$, so forward start interval $p$ is dominant. If $l_i^{t-1} < L_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i} < e_i^t$, then $E_i^F(p) = L_i^F(p) = e_i^t$ and since $q$ is dominant we know that the forward start intervals at customer $i$ originating from $q' \leq q$ can not lead to a better solution. Therefore, to proof that $p$ is dominant we only have to check forward start intervals $p'$ originating from $q' > q$. Suppose there exists a dominant forward start interval $q' > q$ that generates a new forward start interval $p'$ in time window $T$ with

$E_i^F(p') = E_i^F(p) = e_i^t$, then we have

$$w_i^F(p) - w_i^F(p') \geq w_{i-1}^F(q) + e_i^t - L_{i-1}^F(q) - s_{i-1} - \tau_{i-1,i} - w_{i-1}^F(q') - \max\{e_i^t - L_{i-1}^F(q') - s_{i-1} - \tau_{i-1,i}, 0\} \tag{20}$$

$$= w_{i-1}^F(q) - w_{i-1}^F(q') + e_i^t - L_{i-1}^F(q) - s_{i-1} - \tau_{i-1,i} - \max\{e_i^t - L_{i-1}^F(q') - s_{i-1} - \tau_{i-1,i}, 0\} \tag{21}$$

$$> L_{i-1}^F(q) - L_{i-1}^F(q') + e_i^t - L_{i-1}^F(q) - s_{i-1} - \tau_{i-1,i} - \max\{e_i^t - L_{i-1}^F(q') - s_{i-1} - \tau_{i-1,i}, 0\} \tag{22}$$

$$= e_i^t - L_{i-1}^F(q') - s_{i-1} - \tau_{i-1,i} - \max\{e_i^t - L_{i-1}^F(q') - s_{i-1} - \tau_{i-1,i}, 0\} \tag{23}$$

$$= L_i^F(p) - L_i^F(p') \tag{24}$$

The inequality follows from the assumption that $q'$ and $q$ are both dominant, so with Proposition 1 $w_{i-1}^F(q') - w_{i-1}^F(q) < L_{i-1}^F(q') - L_{i-1}^F(q)$ holds. In the last equation the definition of $L_i^F$ in (8) is used. Hence, by Proposition 1 $p$ is dominated by $q'$. To be optimal $\zeta_{i-1}^*$ has to lie in interval $q'$, this is in contradiction with the induction hypothesis, since dominant forward start interval $q'$ and $q$ are non-overlapping as proven in Lemma 1. Therefore, we have shown that for all cases $p$ has to be dominant. Hence, $\zeta_i^*$ lies in a dominant forward start interval of customer $i$. □

# Appendix C Pseudo-code depth-first start interval algorithm

In this appendix the pseudo-code of the depth-first variant of the start interval algorithm is given. Let $\theta_i$ be the last checked time window of customer $i$, with $\zeta = \{\zeta_1, \ldots, \zeta_m\}$. The main advantage is that the Depth-First Algorithm stops if a solution with zero waiting time has been found or when a delay occurs, i.e., if $\theta_i = |T_i| + 1$ then the vehicle arrives after the last time window at customer $i$. The disadvantage is that the forward start intervals of some non-dominant selection of time windows will be calculated.

**Algorithm 3** Depth-First Algorithm

```
θ = {θ_1, ..., θ_m} = {1, ..., 1}, W_min = 10^10 and Stop=0
for t = 1 : |T_1| do
    FSI(2, θ_2, e_1^t, l_1^t, 0, θ, W_min, Stop)
    if W_min = 0 OR Stop = 1 then
        return W_min
    end if
end for
```

**Algorithm 4** $FSI(i, t, E^F_{i-1}, L^F_{i-1}, w^F_{i-1}, \theta, W_{min}, \text{Stop})$

**if** $W_{min} = 0$ OR Stop $= 1$ **then**
  return $W_{min}$
**else if** $i \leq m$ AND $t \leq |T_i|$ **then**
  **if** $E^F_{i-1} + s_{i-1} + \tau_{i-1,i} \leq l^t_i$ **then**
    $\theta_i = t$ ▷ Adjust last used time window
    **if** $L^F_{i-1} + s_{i-1} + \tau_{i-1,i} \geq e^t_i$ **then** ▷ Arrival in time window $t$ at customer $i$
      $E^F_i = \max\{E^F_{i-1} + s_{i-1} + \tau_{i-1,i}, e^t_i\}$
      $L^F_i = \min\{L^F_{i-1} + s_{i-1} + \tau_{i-1,i}, l^t_i\}$
      $w^F_i = w^F_{i-1}$
      $FSI(i+1, \theta_{i+1}, E^F_i, L^F_i, w^F_i, \theta, W_{min}, Stop)$
      **if** $L^F_{i-1} + s_i + \tau_{i-1,i} > l^t_i$ **then**
        $FSI(i, t+1, E^F_{i-1}, L^F_{i-1}, w^F_{i-1}, \theta, W_{min}, Stop)$
      **end if**
    **else** ▷ Arrival before time window $t$ at customer $i$
      $E^F_i = L^F_i = e^t_i$
      $w^F_i = w^F_{i-1} + e^t_i - L^F_{i-1} - s_{i-1} - \tau_{i-1,i}$
      $FSI(i+1, \theta_{i+1}, E^F_i, L^F_i, w^F_i, \theta, W_{min}, Stop)$
    **end if**
  **else** ▷ Arrival after time window $t$
    $FSI(i, t+1, E^F_{i-1}, L^F_{i-1}, w_{i-1}, \theta, W_{min}, Stop)$
  **end if**
**else if** $i \leq m$ AND $\theta_i = |T_i| + 1$ **then** ▷ Arrival after last time window
  Stop $= 1$ ▷ The next paths arrive later at this customer so are infeasible
**else if** $i > m$ **then**
  **if** $w^F_i < W_{min}$ **then** ▷ Check if minimum waiting time has to be updated
    $W_{min} = w^F_i$
  **end if**
**end if**

# Appendix D Proof Lemma 1

**Lemma 1.** *$E^F_i(q) \leq L^F_i(q) < E^F_i(q+1) \leq L^F_i(q+1)$ holds for all $i \in \sigma'$ and $q \in \overline{F}_i$*

*Proof.* We will prove this by induction on $i$. By definition, the time windows for every customer $i \in V'$ are non-overlapping and increasing, i.e., $e^1_i \leq l^1_i < e^2_i \leq l^2_i < \ldots \leq l^{|T_i|}_i$, therefore the lemma holds for $i = 1$. Assuming that the lemma is true for $i = j - 1$, we will show that the lemma also holds for $i = j$.

Order the dominant forward start intervals at customer $j$ with $E^F_j(1) \leq E^F_j(2) \leq \ldots \leq E^F_j(|\overline{F}_j|)$ and let $[E^F_j(p), L^F_j(p)]$ be a dominant forward start interval of customer $j$. By definition of the forward start intervals we know that $E^F_j(p) \leq L^F_j(p)$ for all $p \in F_{j+1}$, so we only have to show that $L^F_j(p) < E^F_j(p+1)$. We will prove this with contradiction.

Suppose that $E_j^F(p) \leq E_j^F(p+1) \leq L_j^F(p)$, then forward start interval $p$ and $p+1$ lie in the same time window $t \in T_j$. Let $p$ be based on dominant forward start interval $q \in \overline{F}_{j-1}$ and $p+1$ on $q' \in \overline{F}_{j-1}$ with $q \neq q'$.
Suppose that $q < q'$ then $E_j^F(p+1) = \max\{E_{j-1}^F(q') + s_{j-1} + \tau_{j-1,j}, e_j^t\} \geq \max\{L_{j-1}^F(q) + s_{j-1} + \tau_{j-1,j}, e_j^t\} \geq \min\{\max\{L_{j-1}^F(q) + s_{j-1} + \tau_{j-1,j}, e_j^t\}, l_j^t\} = L_j^F(p)$. In the first inequality the induction hypothesis is used. Note that $E_j^F(p+1) = L_j^F(p)$ only if $E_j^F(p) = L_j^F(p) = E_j^F(p+1) = e_j^t$, this will lead to a contradiction with the assumption that $p$ and $p+1$ are both dominant. We will prove that forward start interval $p+1$ dominates forward start interval $p$. We distinguish two cases: First, if $E_j^F(p+1) = e_j^t < L_j^F(p+1)$, then $w_j^F(p+1) = 0$ which implies that $w_j^F(p) - w_j^F(p+1) = w_j^F(p) \geq 0 \geq L_j(p)^F - L_j^F(p+1)$. Using Proposition 1 we have proven shown that forward start interval $p+1$ dominates forward start interval $p$.

Second, if $L_j^F(p+1) = e_j^t$ then we get

$$w_j(p) - w_j(p+1) \tag{25}$$
$$= w_{j-1}^F(q) + e_j^t - L_{j-1}^F(q) - s_{j-1} - \tau_{j-1,j} - (w_{j-1}^F(q') + e_j^t - L_{j-1}^F(q') - s_{j-1} - \tau_{j-1,j}) \tag{26}$$
$$= w_{j-1}^F(q) - w_{j-1}^F(q') + L_{j-1}^F(q') - L_{j-1}^F(q) \tag{27}$$
$$> 0 = L_j^F(p) - L_j^F(p+1) \tag{28}$$

The inequality in line (28), follows from the assumption that forward start interval $q$ and $q'$ are both dominant, so by Proposition 1 we have $w_{j-1}^F(q) - w_{j-1}^F(q') > L_{j-1}^F(q) - L_{j-1}^F(q')$. Hence, for both cases we have shown that $w_j(p) - w_j(p+1) > L_j^F(p) - L_j^F(p+1)$, so with Proposition 1 forward start interval $p+1$ dominates $p$. Hence, we have proven that if $q < q'$ then $L_j^f(p) < E_j^F(p+1)$.
Suppose that $q > q'$ then $E_j^F(p) \leq E_j^F(p+1)$. In this case $L_j^F(p+1) = E_j^F(p)$ is only possible if $e_j^t = E_j^F(p+1) = L_j^F(p+1) = E_j^F(p) \leq L_j(p)$. We can prove in a manner similar to the proof above that in this case forward start interval $p$ dominates forward start interval $p+1$. □

## References


A. K. Beheshti, S. R. Hejazi, and M. Alinaghian. The vehicle routing problem with multiple prioritized time windows: A case study. *Computers & Industrial Engineering*, 90:402–413, 2015.

S. Belhaiza. A game theoretic approach for the real-life multiple-criterion vehicle routing problem with multiple time windows. *IEEE Systems Journal*, 2016.

S. Belhaiza, P. Hansen, and G. Laporte. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. *Comput Oper Res*, 52:269–281, 2014.

S. Belhaiza, R. M'Hallah, and G. B. Brahim. A new hybrid genetic variable neighborhood search heuristic for the vehicle routing problem with multiple time windows. In *IEEE Congress on Evolutionary Computation (CEC), 2017*, pages 1319–1326, 2017.

O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation science*, 39(1):119–139, 2005.

D. Favaretto, E. Moretti, and P. Pellegrini. Ant colony system for a vrp with multiple time windows and multiple visits. *Journal of Interdisciplinary Mathematics*, 10(2):263–284, 2007.

A. Goel. The minimum duration truck driver scheduling problem. *EURO Journal on Transportation and Logistics*, 1(4):285–306, 2012.

A. Goel and L. Kok. Truck driver scheduling in the united states. *Transportation science*, 46(3): 317–326, 2012.

J. Hurkała. Time-dependent traveling salesman problem with multiple time windows. *Annals of Computer Science and Information Systems*, 6:71–78, 2015.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220 (4598):671–680, 1983.

N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

Y. Nagata and O. Braysy. A powerful route minimization heuristic for the vehicle routing problem with time windows. *Operations Research Letters*, 37(5):333–338, 2009.

N. Paulsen, F. Diedrich, and K. Jansen. Heuristic approaches to minimize tour duration for the tsp with multiple time windows. In *OASIcs-OpenAccess Series in Informatics*, volume 48. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015. ISBN 393989799X.

G. Pesant, M. Gendreau, J. Y. Potvin, and J. M. Rousseau. On the flexibility of constraint programming models: From single to multiple time windows for the traveling salesman problem. *European Journal of Operational Research*, 117(2):253–263, 1999.

M.-E. Rancourt, J.-F. Cordeau, and G. Laporte. Long-haul vehicle routing and scheduling with working hour rules. *Transportation Science*, 47(1):81–107, 2013.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006.

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

M. W. Savelsbergh. *Computer aided routing*. PhD thesis, CWI, 1992b. page 42-48.

M. Schneider, F. Schwahn, and D. Vigo. Designing granular solution methods for routing problems with time windows. *European Journal of Operational Research*, 2017.

M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, and D. Van Oudheusden. The multiconstraint team orienteering problem with multiple time windows. *Transportation Science*, 47(1):53–63, 2013.

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. *Transportation Science*, 47(1): 64–80, 2013.

P. Toth and D. Vigo. The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing*, 15(4):333–346, 2003.

P. Toth and D. Vigo. *Vehicle routing: problems, methods, and applications*, volume 18. SIAM, Philadelphia, 2014.

F. Tricoire, M. Romauch, K. F. Doerner, and R. F. Hartl. Heuristics for the multi-period orienteering problem with multiple time windows. *Comput Oper Res*, 37(2):351–367, 2010.

C. Voudouris and E. P. Tsang. *Chapter 7 Guided local search.* Handbook of metaheuristics. Springer, 2003. ISBN 1402072635.