Efficient neighborhood evaluations for the vehicle routing problem with multiple time windows

Hoogeboom, Maaike; Dullaert, Wout; Lai, David; Vigoa, Daniele

published in
Transportation Science
2020

DOI (link to publisher)
10.1287/trsc.2019.0912

document version
Publisher's PDF, also known as Version of record

citation for published version (APA)
Hoogeboom, M., Dullaert, W., Lai, D., & Vigoa, D. (2020). Efficient neighborhood evaluations for the vehicle routing problem with multiple time windows. *Transportation Science*, *54*(2), 400-416. https://doi.org/10.1287/trsc.2019.0912

Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Efficient Neighborhood Evaluations for the Vehicle Routing Problem with Multiple Time Windows

Maaike Hoogeboom, Wout Dullaert, David Lai, Daniele Vigo

To cite this article:
Maaike Hoogeboom, Wout Dullaert, David Lai, Daniele Vigo (2020) Efficient Neighborhood Evaluations for the Vehicle Routing Problem with Multiple Time Windows. Transportation Science 54(2):400-416. https://doi.org/10.1287/trsc.2019.0912

http://pubsonline.informs.org/journal/trsc

TRANSPORTATION SCIENCE
Vol. 54, No. 2, March–April 2020, pp. 400–416
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

Efficient Neighborhood Evaluations for the Vehicle Routing Problem with Multiple Time Windows

Maaike Hoogeboom,[a] Wout Dullaert,[a] David Lai,[a] Daniele Vigo[a,b]

[a] Department of Supply Chain Analytics, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, Netherlands; [b] Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi," University of Bologna, 40136 Bologna, Italy

Contact: m.hoogeboom@vu.nl, https://orcid.org/0000-0001-8512-0762 (MH); w.e.h.dullaert@vu.nl, https://orcid.org/0000-0002-9416-211X (WD); david.lai@vu.nl (DL); daniele.vigo@unibo.it, https://orcid.org/0000-0002-1499-8452 (DV)

Received: March 1, 2018
Revised: October 5, 2018; March 8, 2019
Accepted: March 13, 2019
Published Online in Articles in Advance: January 6, 2020

https://doi.org/10.1287/trsc.2019.0912

Abstract. In the vehicle routing problem with multiple time windows (VRPMTW), a single time window must be selected for each customer from the multiple time windows provided. Compared with 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 efficiently determine the optimal start time for servicing each customer. The proposed algorithm has a reduced worst-case and average complexity than existing exact algorithms. Furthermore, the proposed exact algorithm can be used to efficiently evaluate neighborhood operations during a local search resulting in significant acceleration. To examine the benefits of exact neighborhood evaluations and to solve the VRPMTW, the proposed algorithm is embedded in a simple metaheuristic framework generating numerous new best known solutions at competitive computation times.

Funding: This work was partially supported by The Netherlands Organisation for Scientific Research (NWO) [Grant 407-13-050].
Supplemental Material: The online appendix is available at https://doi.org/10.1287/trsc.2019.0912.

Key words: vehicle routing • multiple time windows • metaheuristics

The *vehicle routing problem with multiple time windows* (*VRPMTW*) arises naturally in delivery operations where customers provide multiple time windows for service. Examples include the distribution of industrial gases (Pesant et al. 1999), long-haul transport (Goel and Kok 2012; Rancourt, Cordeau, and Laporte 2013), and city tourist trips to different tourist attractions (Souffriau et al. 2013). The VRPMTW determines the minimum-cost vehicle routes that serve each customer in one of their specified time windows and satisfy vehicle capacity constraints.

The existing metaheuristics developed for the *vehicle routing problem with time windows* (*VRPTW*) frequently 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 a maximum of 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.

Multiple time windows are addressed in different routing problems such as the traveling salesman problem (TSP), truck driver scheduling problem, and team orienteering problem. Favaretto, Moretti, and Pellegrini (2007) were the first to address the VRPMTW. Their objective was to minimize the total duration by designing the routes and selecting the service time windows of the customers. Belhaiza, Hansen, and Laporte (2014) (BHL14) improved the results of Favaretto, Moretti, and Pellegrini (2007) using an exact algorithm to determine the optimal departure time from the depot for a given route. Belhaiza, Hansen, and Laporte (2014) used the same route duration minimization algorithm to evaluate local search moves. This implied that although only a small part of the route was changed, the entire route was reevaluated. Tricoire et al. (2010) (TRDH10) presented another exact route duration minimization algorithm for the related multiperiod orienteering problem with multiple time windows. Their proposed algorithm determines the optimal departure time for every customer in a given route. Because of its 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 identified the benefits of exactly recalculating the route duration for local

search moves. To recalculate the minimal route duration when neighborhood operators are applied, Savelsbergh (1992b) presented an efficient algorithm based on the forward and backward time slack of the departure time per customer. We consider this algorithm efficient because the cost of a neighborhood operation can be calculated without reevaluating the entire route. However, as indicated by Tricoire et al. (2010) and Belhaiza, Hansen, and Laporte (2014), this approach to minimizing 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 (1992b). These forward (backward) intervals at any given customer represent the start times of servicing this customer such that all preceding (succeeding) customers in the route are served in an available time window. The complexity of the proposed algorithm is formally demonstrated, 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 an improved worst-case complexity compared with the algorithms proposed in the literature. Furthermore, we demonstrate that the average computational speed of the proposed algorithm is at least twice as fast as the existing algorithms. Second, we are the first to efficiently recalculate the minimum duration in local search operations (i.e., when customers are removed from or inserted into a route). Computational experiments indicate an acceleration by a factor of four compared with existing exact route duration minimization algorithms. Therefore, the proposed algorithm can provide computational benefits for numerous metaheuristic approaches. Last, we experimentally demonstrate that efficient exact neighborhood evaluations improve solution quality compared with an approximate evaluation method. By incorporating the proposed exact algorithm in a simple metaheuristic, we could identify 22 new best known solutions to VRPMTW instances from the literature at competitive computation times.

The remainder of this paper is organized as follows. In Section 1, the literature on duration minimization in multiple time window routing problems is reviewed. 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 presented and discussed. Conclusions are presented in the last section.

1. Literature Review

Compared with the VRPTW, the VRPMTW has received minimal attention in the literature. Problems with multiple time windows are frequently addressed within related problems such as the traveling salesman problem, truck driver scheduling problem, and team orienteering problem.

Pesant et al. (1999) presented a constraint programming formulation for the TSP with multiple time windows. They illustrated 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 considered; thus, the duration of a tour is not minimized. Recently, Paulsen, Diedrich, and Jansen (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 determining the 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 until the final customer. The drawback of this method is that it is computationally intensive for large tours, and the customers must be assigned to a vehicle before deploying the algorithm.

In the truck driver scheduling problem with multiple time windows, the goal is to determine a schedule with minimal duration for a fixed route that satisfies the regulations concerning hours of service (Goel 2012, Goel and Kok 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, where the algorithm becomes similar to the approach of Belhaiza, Hansen, and Laporte (2014). However, the dominance criterion used by Belhaiza, Hansen, and Laporte (2014) to eliminate dominated solutions is stronger than the dominance criteria used by Goel and Kok (2012) and Goel (2012) when rest periods are not considered.

Souffriau et al. (2013) presented a metaheuristic for the multiconstraint team orienteering problem

with multiple time windows. To address 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 provided for minimizing the duration of a route. However, if there are duration limits, then it is profitable to minimize the route duration such that more customers can be visited in one trip. Therefore, Tricoire et al. (2010) presented an algorithm to minimize the duration of a given route to address the multiperiod orienteering problem with multiple time windows. They tightened the time windows for each customer by exploiting the fact that the service at any given customer cannot start before the end of the service at the previous customer. Their algorithm uses the concept of dominant solutions, which are solutions for which no other solution exists with the same finishing time and a later departure time. The polynomial algorithm of Tricoire et al. (2010) enumerates all the promising dominant solutions of an entire route in order of increasing finishing time and selects the solution with the shortest duration. However, as the algorithm does not identify the optimal solution for portions of a route, such as tails, it is not suited to quickly evaluating neighborhood operations in a local search.

Hurkała (2015) presented the TSP with multiple time windows and time-dependent travel times. Their minimum route duration algorithm is based on, and therefore similar to, the algorithm of Tricoire et al. (2010).

The first paper on the VRPMTW was published by Favaretto, Moretti, and Pellegrini (2007). They consider the VRPMTW in a periodic setting where customers can require service multiple times during the schedule horizon. Customers that are serviced multiple times must be visited in different time windows; hence, 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 the paper by Favaretto, Moretti, and Pellegrini (2007) is to divide the time windows over multiple visit days; therefore, no algorithm is offered to minimize the duration of a given route.

Belhaiza, Hansen, and Laporte (2014) minimized the total duration and proposed a hybrid variable neighborhood tabu search heuristic 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, that is, 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 terminated when a solution with zero waiting time is identified. Belhaiza, M'Hallah, and Brahim (2017) (BMB17) proposed a new hybrid genetic variable neighborhood search heuristic for the VRPMTW that improves virtually all the results of Belhaiza, Hansen, and Laporte (2014).

Beheshti, Hejazi, and Alinaghian (2015) and Belhaiza (2018) solve a multiobjective VRPMTW. Beheshti, Hejazi, and Alinaghian (2015) solve the vehicle routing problem with multiple prioritized time windows where customers can prioritize the multiple available time windows and the customers are grouped based on their importance to the distributor. The multiobjective problem of minimizing the traveling cost and maximizing the customer's satisfaction is solved. Belhaiza (2018) considers the multiple criteria of minimizing the total travel time and maximizing the utility of the customers and drivers. Both approaches search for Pareto optimal solutions and do not minimize the total duration.

The exact algorithms of Tricoire et al. (2010) and Belhaiza, Hansen, and Laporte (2014) were developed to minimize the duration of a given route. Tricoire et al. (2010) used the exact algorithm only to calculate the minimal duration of a local optimum solution, and used an approximation of the route duration to evaluate the moves during the local search. Belhaiza, Hansen, and Laporte (2014) also used the exact algorithm to evaluate neighborhood operations during the local search. However, they used the same algorithm for a given route as for evaluating a neighborhood operation where only a small part of a route changes. Therefore, we developed an approach that is efficient for evaluating neighborhood operations.

Savelsbergh (1992a) presented an efficient method to check whether a move is feasible and profitable for the TSP with multiple time windows. He considers a move profitable if the completion time of the resulting route is earlier, while the departure time from the depot remains the same. Hence, the profitability of moves is calculated only for the current fixed departure time from the depot. Because 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 (1992b) 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. The proposed algorithm can also be used to calculate the minimal duration of a given route, like those of Tricoire et al. (2010) and Belhaiza, Hansen, and Laporte (2014), but at a reduced 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 nonnegative travel time τ_{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 nonnegative 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 nonoverlapping, with $0 \leq e_i^1 \leq l_i^1 < e_i^2 \leq l_i^2 < \cdots < l_i^{|T_i|}$. 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 Online Appendix A. If a vehicle arrives at a customer within one of the time windows, the service starts immediately upon arrival; otherwise, the vehicle 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 permitted. Furthermore, we set $s_0 = 0$ and $d_0 = 0$, and define $[e_0, l_0]$ as the single time window of the depot.

Consider a fleet of identical vehicles (denoted by set K), with capacity Q and fixed vehicle cost F when a vehicle is used. The objective value of a solution consists 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 windows and the route duration is minimized.

Let σ 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 where 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 the time window t of customer i is selected and 0 otherwise. Furthermore, let a_i be the arrival time and w_i be the waiting time at customer $i \in \sigma$; hence, 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 because 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 is within one of the given time windows. In an optimal solution, $w_i = 0$; hence, a_0 represents the optimal departure time.

Belhaiza, Hansen, and Laporte (2014) and Tricoire et al. (2010) both presented an algorithm for this subproblem. In the worst case, the approach of Belhaiza, Hansen, and Laporte (2014) must check all possible combinations of the 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 $O(m(1 + \sum_{i=1}^{m}(|T_i| - 1)))$. The proposed exact polynomial algorithm, described in the following section, has the least complexity of $O(\sum_{i=1}^{m}(1 + \sum_{j=1}^{i}(|T_j| - 1)))$. Moreover, the proposed algorithm can be used efficiently with local search operations in the metaheuristic search, as will be demonstrated in Section 3.3.2.

Note that the approach of Belhaiza, Hansen, and Laporte (2014) can process overlapping time windows, whereas the approach of Tricoire et al. (2010) and the proposed algorithm cannot. In practice, there is no distinction between these cases, because overlapping time windows can easily be merged into nonoverlapping time windows. If a decision regarding 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. Efficient Methods for the Duration Minimization Subproblem

In this section, we introduce a novel approach to solve the route duration minimization subproblem defined in Section 2.1. In Section 3.1, the proposed solution algorithm based on forward start intervals is described. The complexity of this algorithm is determined in Section 3.2. The generation of the forward start intervals is sufficient to efficiently calculate the minimal route duration of a given route. In Section 3.3, we present a method to efficiently recalculate the minimal duration of a route when local search operations are performed.

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 the 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, that is, $E_i^F(1) \le L_i^F(1) \le E_i^F(2) \le L_i^F(2) \le \cdots \le E_i^F(|F_i|) \le L_i^F(|F_i|)$. The forward start intervals are recursively computed, from the first to the last customer in σ'. 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 $t \in T_i$, that is, $E_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i} \le l_i^t$, 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

$$E_i^F(p) = \max\left\{E_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i}, e_i^t\right\},$$
$$L_i^F(p) = \min\left\{\max\left\{L_{i-1}^F(q) + s_{i-1} + \tau_{i-1,i}, e_i^t\right\}, l_i^t\right\}. \quad (8)$$

Waiting time occurs at customer i if the customer is serviced in time window $t \in T_i$ and the vehicle arrives before this time window, that is, 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 serviced is given by

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

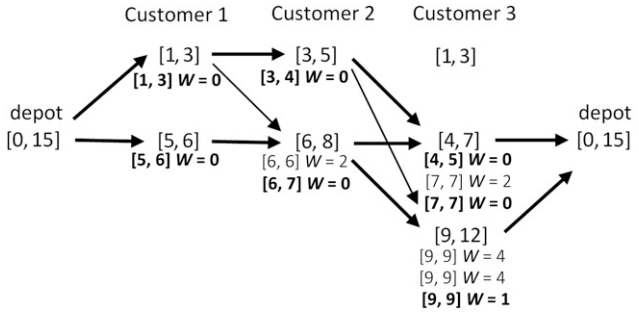
Let $W_i^F = \{w_i^F(1), \ldots, w_i^F(|F_i|)\}$ be the set of minimal total waiting times until customer i, corresponding to the forward start intervals. As initialization, we set $w_1^F(p) = 0$ for all $p \in F_1 = T_1$. Note that if $w_i^F(p) > 0$, then $E_i^F(p) = L_i^F(p)$.

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.

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 because 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 less or equal minimum total waiting time. A forward start interval $q \in F_i$ is *dominant* if it is not dominated by another forward start interval.

Figure 1. Time Windows and Forward Start Intervals, with Corresponding Waiting Times of Three Customers in a Route



Notes. Let all travel times be one time unit, and let all service times be zero. The arrows represent the feasible connections between time windows, where the bold arrows represent dominant combinations.

The dominance criteria used in the proposed algorithm are 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.*

In Online Appendix B, it is proved that these dominance criteria are sufficient to demonstrate that a forward start interval is dominated. In Figure 1, all dominant forward start intervals and corresponding waiting times are given in bold. In the remainder of this paper, we consider only the dominant forward start intervals. 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 σ, we present the dominant forward start interval (DFSI) algorithm. The pseudocode 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 nonoverlapping; hence, the extra waiting time obtained at customer i will 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; rather, we start with the last checked time window θ. The optimality of this approach is proved 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

Algorithm 1 (Dominant Forward Start Interval)

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 $W_i^F\ \forall i \in \{1, \ldots, m\}$

intervals. The details of the complexity are provided in Section 3.2.

The DFSI algorithm calculates all dominant forward start intervals of all customers $i \in \sigma'$. In Online Appendix C, 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 least total waiting time provides 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 determined.

The DFSI algorithm 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, for example, for the evaluation of local search operations as described in Section 3.3. If we wish to calculate only the minimal total duration of a given route, then the search should terminate as soon as a solution without waiting time is identified. In this case, a depth-first implementation as presented in Online Appendix D is more appropriate. Note, however, that both implementations have the same worst-case complexity.

3.2. Complexity Analysis

In this section, we formally demonstrate 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 require the following three lemmas.

The first lemma states that the dominant forward start intervals at every customer $i \in \sigma'$ are nonoverlapping and increasing in $q \in \overline{F}_i$. The proof can be found in Online Appendix E.

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 owing 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 $q' < q$* cannot *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 demonstrate that forward start interval p dominates p':

$$
\begin{aligned}
& w_j^F(p') - w_j^F(p) \\
&= w_{j-1}^F(q') + \max\left\{0, e_j^{t'} - L_{j-1}^F(q') - s_{j-1} - \tau_{j-1,j}\right\} \\
&\quad - \left(w_{j-1}^F(q) + \max\left\{0, e_j^{t} - L_{j-1}^F(q) - s_{j-1} - \tau_{j-1,j}\right\}\right) && (10) \\
&= w_{j-1}^F(q') - w_{j-1}^F(q) + e_j^{t'} - L_{j-1}^F(q') - s_{j-1} - \tau_{j-1,j} \\
&\quad - \max\left\{0, e_j^{t} - L_{j-1}^F(q) - s_{j-1} - \tau_{j-1,j}\right\} && (11) \\
&> L_{j-1}^F(q') - L_{j-1}^F(q) + e_j^{t'} - L_{j-1}^F(q') - s_{j-1} - \tau_{j-1,j} \\
&\quad - \max\left\{0, e_j^{t} - L_{j-1}^F(q) - s_{j-1} - \tau_{j-1,j}\right\}\right\} && (12) \\
&= e_j^{t'} - L_{j-1}^F(q) - s_{j-1} - \tau_{j-1,j} \\
&\quad - \max\left\{0, e_j^{t} - L_{j-1}^F(q) - s_{j-1} - \tau_{j-1,j}\right\}\right\} && (13) \\
&= e_j^{t'} - L_j^F(p) = L_j^F(p') - L_j^F(p). && (14)
\end{aligned}
$$

Line (10) follows from the definition given in (9). By Lemma 1, $L_{j-1}^F(q') + s_{j-1} + \tau_{j-1,j} < E_{j-1}^F(q) + s_{j-1} + \tau_{j-1,j} \leq l_j^t < e_j^{t'}$ holds, and by rearranging the terms, line (11) is obtained. Because forward start intervals q' and q are both dominant, we obtain $w_{j-1}^F(q') - w_{j-1}^F(q) > L_{j-1}^F(q') - L_{j-1}^F(q)$, which is used to deduce (12) from (11). Line (14) follows from the fact that $L_j^F(p) = L_{j-1}^F(q) + s_{j-1} + \tau_{j-1,j} + \max\{0, e_j^t - L_{j-1}^F(q) - s_{j-1} - \tau_{j-1,j}\}$ and $L_j^F(p') = e_j^{t'}$. We have demonstrated that because $E_j^F(p) < E_j^F(p')$ and $w_j^F(p') - w_j^F(p) > L_j^F(p') - L_j^F(p)$, $[E_j^F(p'), L_j^F(p')]$ is not a dominant forward start interval. □

Following Paulsen, Diedrich, and Jansen (2015), we formally demonstrate the upper bound on the number of dominant forward start intervals per customer in a route. Because every forward start interval corresponds to a different selection of time windows, it offers an upper bound to the maximum number of time window combinations that must 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 must be checked, that is, $|\overline{F}_m| \leq 1 + \sum_{j=1}^{m}(|T_j| - 1)$.*

Proof. By induction over the number of customers m, we prove that this lemma holds for all m. For $m = 1$, there is only one customer with $|T_1|$ time windows; hence, there are at most $|T_1|$ dominant forward start intervals. Suppose that the lemma is true for $m = i$. We prove 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 nonoverlapping and increasing. By setting both the index set of the dominant forward start intervals $\overline{F}_i$ and the time windows T_{i+1} in increasing order, the bipartite graph G can be drawn without crossings following Lemma 2. Because 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 obtain $|\overline{F}_{i+1}| = |\overline{F}_i| + |T_{i+1}| - 1 = 1 + \sum_{j=1}^{i+1}(|T_j| - 1)$. □

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 identifying 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. Extension to Local Search

In local search, neighborhood operations are used to insert and remove customers from a route σ. To efficiently check the time window feasibility, both forward and backward start time intervals are required. In Section 3.3.1, the backward start intervals are described, and Section 3.3.2 illustrates how the forward and backward start intervals are used to efficiently recalculate the minimum route duration when local search operations are applied.

3.3.1. Backward Start Intervals. 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. Similar to 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 σ', 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 $i+1$, that is, if $e_i^t + \tau_{i,i+1} + s_i \le L_{i+1}^B(q)$ ($\Longrightarrow L_{i+1}^B(q) - \tau_{i,i+1} - s_i \ge e_i^t$), then the new backward start interval p of customer i can be calculated by

$$E_i^B(p) = \max\left\{\min\left\{E_{i+1}^B(q) - \tau_{i,i+1} - s_i, l_i^t\right\}, e_i^t\right\},$$
$$L_i^B(p) = \min\left\{L_{i+1}^B(q) - \tau_{i,i+1} - s_i, l_i^t\right\}. \quad (15)$$

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.

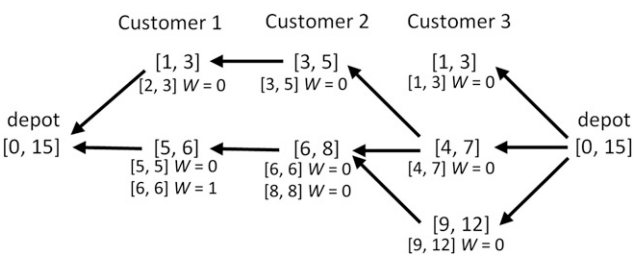
Similar to Definition 1, backward start interval q dominates backward start interval q' at customer i if all feasible start times at customer $i-1$ for q' are also feasible for q with less or equal waiting time. To check whether an interval is dominated, the following dominance criteria are used.

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

That these criteria imply dominance can be proved similarly to Proposition 1. Intuitively, the difference in waiting time of the two backward start intervals is greater than the time difference between the lower bounds of the intervals. Therefore, backward start interval q can identify the same arrival times at customer $i-1$, because $L_i^B(q) \ge L_i^B(q')$, with less 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 determine the solution with minimal duration, we must consider only the dominant backward start intervals. The dominant backward start intervals of the customers in the example route are given in Figure 2.

As with the DFSI algorithm, we can calculate all the dominant backward start intervals from the last to the first customer in a given route σ. Lemmas 1 and 3 also

Figure 2. Dominant Backward Start Intervals and Corresponding Waiting Times of Customers $\{1, 2, 3\}$



hold for the dominant backward start intervals; therefore, the calculation of all backward start intervals also has a polynomial time computational complexity of $O(\sum_{i=1}^{m}(1+\sum_{j=1}^{i}(|T_j|-1))$.

3.3.2. 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 are used to efficiently recompute the minimal total waiting time of a route when local search operators are applied. Because the majority of operators divide a route into two or more sections, the minimal route duration of a move connecting customers i and j is recalculated by comparing the forward and backward start intervals of customers i and j, respectively. We illustrate this for a deletion and insertion of a customer in a route because 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 must 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 accomplish 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 $E_{i-1}^F(f) + s_{i-1} + \tau_{i-1,i+1} \le L_{i+1}^B(b)$. The corresponding total waiting time for this feasible combination is given by $w^R(f,b) = w_{i-1}^F(f) + w_{i+1}^B(b) + \max\{0, E_{i+1}^B(b) - L_{i-1}^F(f) - s_{i-1} - \tau_{i-1,i+1}\}$, that is, the sum of the waiting time of sequence $\{1,\ldots,i-1\}$, waiting time of sequence $\{i+1,\ldots,m\}$, and waiting time between customers $i-1$ and $i+1$. The minimum waiting time of the new route $\tilde{\sigma}$ is given by $\min\{w^R(f,b) | f \in \overline{F}_{i-1}, b \in \overline{B}_{i+1} \text{ with } E_{i-1}^F(f) + s_{i-1} + \tau_{i-1,i+1} \le L_{i+1}^B(b)\}$.

If the new route $\tilde{\sigma}$ is accepted, then the new dominant start intervals must be calculated. Because a part of the route remains the same, only the dominant forward start intervals of customers $i+1,\ldots,m$ and dominant backward start intervals of customers $1,\ldots,i-1$ require recalculation.

The worst-case computational complexity of checking whether the 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 σ is feasible, then the customer deletion is always time window feasible if the triangle inequality holds.

Insertion. Suppose we have a route σ and we wish to insert customer α between i and $i+1$. To check whether the new route $\tilde{\sigma} = \{0,1,\ldots,i,\alpha,i+1,\ldots,m,m+1\}$ is feasible and to calculate the minimal total waiting time, we calculate the forward start intervals and corresponding waiting times at customer α 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 $E_\alpha^F(f) + s_\alpha + t_{\alpha,i+1} \le L_{i+1}^B(b)$. The corresponding total waiting time for this combination is $w_\alpha^F(f) + w_{i+1}^B(b) + \max\{0, E_{i+1}^B(b) - L_\alpha^F(f) - s_\alpha - \tau_{\alpha,i+1}\}$. The minimum waiting time of the new route $\tilde{\sigma}$ is given by $\min\{w^R(f,b) | f \in \overline{F}_\alpha, b \in \overline{B}_{i+1} \text{ with } E_\alpha^F(f) + s_\alpha + \tau_{\alpha,i+1} \le L_{i+1}^B(b)\}$.

The computational complexity to check whether 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 the removal and insertion of a sequence of customers and for more complex operators such as 2-opt* or cross exchanges.

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 numerous 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 different alternative components of a VNS that led to an overall adaptive VNS (AVNS) that outperformed the standard VNS both in terms of solution quality and convergence speed. Therefore, we propose an AVNS algorithm to solve the VRPMTW. In Section 5, the proposed AVNS algorithm is used to compare the average performance of the route duration minimization algorithms. This section presents the structure of the proposed metaheuristic framework (presented in Algorithm 2) and discusses its components.

The heuristic is initialized with a feasible solution determined by applying the route minimization (RM) heuristic of Nagata and Braysy (2009) that is described in Section 4.1. This solution is the start of the AVNS, where local search and shaking are iteratively called. A granular local search with multiple operators is used to identify 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 simulated annealing approach (Kirkpatrick, Gelatt, and Vecchi 1983), we attempt to diversify the search by accepting nonimproving solutions with probability $\exp(-(V(x'') - V(x'))/\theta)$, where $V(x)$ is the objective value defined in Section 4.2. The temperature θ is initially set to θ_0 and updated after every shake with factor θ^{update}. After I^θ nonimproving main AVNS iterations, θ is reset to θ_0. In the shaking phase, a set of k_{max} neighborhood structures is used. When a solution is accepted (Line 7),

Algorithm 2 (Adaptive Variable Neighborhood Search)

1. Initialization: Create an initial feasible solution x by the RM heuristic, $x^* = x'' = x$
2. Initialize kth 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

the search restarts at the first neighborhood $k = 1$, otherwise it continues with the next neighborhood $k = k + 1$. In the shaking phase (Line 13), a new solution x is generated from the kth neighborhood of the current local optimum x''; that is, $x \in \mathcal{N}_k(x'')$. 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 in S iterations, a large shaking is performed, and the local optimum x'' is reset to restart the search (see Lines 14–17 and Section 4.3).

4.1. Initial Solution

To minimize the number of vehicles, we use the RM heuristic of Nagata and Braysy (2009). This algorithm begins with an initial solution where 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 placed 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 attempt 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 reinserting 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 terminates 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 determine the local optimum solution. In the local search, seven different neighborhoods are explored in increasing order of complexity. If an improving solution is identified, the search restarts at the first neighborhood. First, intraroute neighborhood operations on a single route are applied. This includes the *relocate* neighborhood, where a customer is placed in a new best position in the same route; *exchange*, where two customers exchange positions; and the *relocate sequence* neighborhood, where a sequence of at least two and at most four customers is relocated to a new best position. Second, interroute neighborhood operations on two routes are applied. This includes the *relocate* neighborhood, which repositions a customer to the best position of another route; *exchange*, which exchanges the positions of two customers; 2-*opt**, which exchanges the tails of two routes; and *cross*, which exchanges two sequences of at most four customers of two routes.

To accelerate the local search, a granular neighborhood is used where the arc set is restricted. Our restricted arc set consists of the arcs from each customer to its c% closest customers. All depot arcs are included because Schneider, Schwahn, and Vigo (2017) demonstrated that this improves the solution quality. As described by 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 nonmoved 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 be inserted, because 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 $t(x) = \sum_{i=1}^{n} \cdot \max\{a_i - l_i^{|T_i|}, 0\}$. 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 β_1 and β_2 are self-adjusting parameters that are updated every μ iterations. If, in the last μ iterations, the capacity constraint is violated in at least one route, then parameter β_1 is multiplied by $\delta > 1$. Otherwise, β_1 is divided by δ. The same update rule is applied to parameter β_2.

4.3. Shaking

An adaptive shaking approach based on Stenger et al. (2013) is used to determine a new starting point for the local search. In this shaking approach, one move of the kth neighborhood is performed. The $k_{max} = 18$ neighborhoods used in the shaking are all based on interroute 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 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 $L = 6$, 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. Then, the length of the sequences are fixed (if $L < 4$). Finally, the customer sequences involved in the move are selected.

The first route r_1 involved in the shaking operation is randomly selected. To avoid unpromising moves, the other routes in the shaking operation must be located close to route r_1. Therefore, the probability of selecting a route $r \neq r_1$ is determined by the number of arcs in the restricted arc set of the granular neighborhood that connect a customer in route r to a customer in route r_1.

The selection of the customer sequences is achieved 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. As in 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 determined 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 iterations, that is, 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 terminates after η restarts, if I iterations are performed, or if the time limit of T seconds is reached.

5. Computational Results

In this section, we compare the proposed exact algorithm with other route duration minimization algorithms presented in the literature. Furthermore, we incorporate the proposed algorithm into a new and effective AVNS algorithm, which is successfully used to solve the VRPMTW instances from the literature. The algorithms were implemented in C++ and executed on a single core of a Windows 10 computer with a 4.0 GHz Intel Core i7 processor and 24 GB of 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 remain fixed. For each parameter, three executions 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 begins at $c_0 = 10\%$ and increases by $c^{update} = 10\%$ after $I^c = 200$ nonimproving 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 $S = 5{,}000$ nonimproving iterations or if after $s = 2{,}000$ iterations, the current solution is not within $v = 10\%$ of the overall best solution. The search is terminated after $\eta = 10$ large shakes, after $I = 100{,}000$ iterations, or if the time limit of $T = 150$ seconds is reached. The RM heuristics executes for $T_{max} = 8$ seconds.

5.1. Data Set

Belhaiza, Hansen, and Laporte (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; that is, the vehicle costs F are 200, 700, 200, 1,000, 200, and 1,000 for instance sets CM1, CM2, RCM1, RCM2, RM1, and RM2, respectively. The instances have from 1 to 10 nonoverlapping time windows per customer, and the optimal solutions of the instances are unknown. The specific details of the instances are displayed in Table 1.

The published objective values of Belhaiza, Hansen, and Laporte (2014) include travel time, waiting time, service time, and vehicle cost. Because service time is a constant and can represent a large part of the objective value, we do not include it when reporting the duration and objective value to avoid distorting the calculation of savings. For example, for instance set CM, the fixed total service time is 9,000, whereas the total travel time plus waiting time is between 820 and 1,500.

As in Tricoire et al. (2010), the preprocessing steps described in Online 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 reduced worst-case complexity than the existing exact route duration minimization algorithms of Tricoire et al. (2010) and Belhaiza, Hansen, and Laporte (2014). In this section, we compare the average performance of the proposed route duration minimization algorithm to the existing algorithms. To do this, we implemented the exact algorithms described in Tricoire et al. (2010) and Belhaiza, Hansen, and Laporte (2014). In Belhaiza, Hansen, and Laporte (2014) and in the proposed 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 were required to make a small and straightforward

Table 1. Average Computational Times in Seconds of Different Route Duration Minimization Algorithms

Instance	m	nTW	Width TW	TRDH10	BHL14	DFSI	Instance	m	nTW	Width TW	TRDH10	BHL14	DFSI
RM101	10	5–9	10–30	14.0	21.1	9.4	RM201	2	5–8	50–100	25.0	515.4	8.9
RM102	9	5–7	10–30	12.9	20.9	9.4	RM202	2	3–5	50–100	19.8	755.2	4.8
RM103	9	4–7	10–30	14.3	23.5	9.5	RM203	2	2–5	50–100	15.5	306.1	3.7
RM104	9	3–6	10–30	13.6	17.5	9.2	RM204	2	2–4	50–100	21.6	186.5	4.3
RM105	9	2–6	10–30	13.7	16.6	9.2	RM205	2	1–4	50–100	17.6	87.7	3.7
RM106	9	2–3	30–50	12.4	14.9	9.0	RM206	2	1–3	100–200	19.9	45.3	3.8
RM107	9	1–3	30–50	12.0	13.1	8.9	RM207	2	1–3	100–200	19.4	33.5	3.0
RM108	9	1–2	50–100	13.4	13.9	9.2	RM208	2	1–5	100–200	19.7	32.6	3.2
Average	9.13			13.3	17.7	9.2	Average	2			19.8	245.3	4.4
CM101	10	5–10	50–100	6.7	14.8	2.2	CM201	5	5–10	50–100	14.9	48.3	10.1
CM102	11	5–7	50–100	14.7	23.7	9.9	CM202	6	5–7	50–100	13.3	37.2	9.6
CM103	11	3–7	50–100	13.9	17.1	9.5	CM203	5	3–7	50–100	15.6	24.9	9.5
CM104	13	3–5	50–100	14.1	16.9	9.4	CM204	5	3–5	50–100	13.8	25.5	9.5
CM105	10	2–5	50–100	4.5	7.2	1.2	CM205	4	2–5	100–200	21.2	25.0	9.8
CM106	10	2–4	100–200	4.0	5.5	1.0	CM206	4	2–4	100–200	14.4	17.2	9.3
CM107	10	1–3	100–200	4.0	4.6	0.9	CM207	4	1–3	100–300	13.7	16.0	9.1
CM108	10	1–3	100–500	3.6	4.0	0.8	CM208	4	1–3	100–500	13.7	15.1	9.1
Average	10.63			8.2	11.7	4.4	Average	4.63			15.1	26.2	9.5
RCM101	10	5–10	10–30	12.3	17.9	9.2	RCM201	2	5–10	50–100	26.1	1250.8	7.2
RCM102	10	5–7	10–30	12.3	15.7	9.1	RCM202	2	5–7	50–100	20.6	281.1	11.7
RCM103	10	3–7	10–50	12.0	15.8	9.0	RCM203	2	3–7	50–100	18.8	268.0	4.2
RCM104	10	3–5	10–50	12.1	14.4	9.0	RCM204	2	3–5	50–100	18.2	181.3	4.3
RCM105	10	2–5	10–70	12.7	14.9	9.1	RCM205	2	2–5	100–200	17.5	35.2	5.3
RCM106	10	2–4	30–70	11.7	13.3	8.8	RCM206	2	2–4	100–200	27.1	51.0	5.7
RCM107	11	1–3	30–70	11.0	11.3	8.7	RCM207	3	1–3	100–300	16.9	18.3	9.3
RCM108	11	1–3	30–100	12.5	13.7	9.0	RCM208	2	1–3	100–500	14.9	16.1	2.1
Average	10.25			12.1	14.6	9.0	Average	2.13			20.0	262.7	6.2

Note. nTW represents the number of time windows and Width TW the width of the time windows.

adjustment to the input for the algorithm of Tricoire et al. (2010). The upper bound of the time window of each customer was increased by its service time, allowing the time windows to overlap. However, this did not influence the Tricoire et al. (2010) algorithm as it continued to schedule the start of service within the original time windows.

The proposed start interval algorithm was implemented in two different manners. First, it used a depth-first implementation that terminated when a solution with zero waiting time was identified (see Online Appendix D). This implementation used only the forward start intervals and was useful for comparing the proposed method with existing methods that calculate the minimum duration of a given route. Second, it applied an embedded start interval (ESI) algorithm that used both the backward and forward start intervals, as described in Section 3.3.2.

In the first experiment, the running times of the exact route duration minimization algorithms are compared. In the second experiment, we demonstrate the benefits of using the embedded start interval algorithm where the local search moves are efficiently evaluated compared with using a standard route duration minimization algorithm. Furthermore, the influence 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, Hansen, and Laporte (2014) (BHL14). To perform this comparison, we executed the AVNS algorithm described in Section 4 on all instances for $I = 1{,}000$ iterations without imposing a time limit. Every time the route duration required minimization, each of the three algorithms was used and its computational time was reported. Note that because all algorithms are exact, the final solution was the same for all algorithms. The minimum total duration was, therefore, not reported in Table 1. Furthermore, the RM algorithm used to create an initial solution was the same for all algorithms. Its computational time varied across problem instances; it required a maximum of eight seconds.

The first column of Table 1 reports the instances, and the second column indicates the number of routes m in the final solution. The number of time windows and the width of the time windows are indicated in the third and fourth columns, respectively.

For each instance set, the number of time windows per customer decreased per instance. The decrease in computational time from the first to the last instance in a set was, on average, 20%, 65%, and 36% for the TRDH10, BHL14, and DFSI algorithms, respectively. Hence, Table 1 indicates that the computational times of all algorithms increased if more time windows were considered. The computational times of both the TRDH10 and BHL14 algorithms clearly increased when route length increased, especially for the BHL14 algorithm, this increase in computational time was significant. This can be seen when comparing Set 1 instances with Set 2 instances. The average computational times of the TRDH10 and BHL14 algorithms were, respectively, 11.2 and 14.7 seconds for the Set 1 instances, and 18.3 and 178.1 for the Set 2 instances. The computational times of the DFSI algorithm were between 0.8 and 11.7 seconds for all instances, which indicates that the DFSI algorithm is less influenced by problem specific characteristics.

The proposed DFSI algorithm outperformed the other algorithms in terms of average computational time. The DFSI algorithm had the lowest computational time on all instances. On average, it was 2.1 and 13.5 times faster than the approaches of TRDH10 and BHL14, respectively. The TRDH10 method outperformed the BHL14 method on all instances. These results are in line with the worst-case computational complexity analysis of the algorithms in Section 2.1.

5.2.2. Influence of the Embedded Forward Start Interval. In the second test, the performance of different methods to evaluate neighborhood solutions in a local search was tested. Because the previous test indicated that the proposed DFSI algorithm outperformed the existing route duration minimization algorithms, we compared only the average computational times of the DFSI algorithm and the ESI algorithm. These are both exact methods to calculate the minimal duration of a route. We also implemented an approximate evaluation method where the waiting time and time window infeasibility of a route were approximated by servicing the customers as early as possible and accumulating the waiting times and delays at every customer in the route. The approximate method was used when evaluating a move, and the exact DFSI algorithm was used only when a move was performed. The AVNS algorithm was 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 is 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 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"

indicate the average computational times of the corresponding exact evaluation methods. Note that the two algorithms follow exactly the same optimization path and thus yield the same final solution.

The results in Table 2 indicate 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 with the approximate evaluation approach. However, the approximate method is, on average, 2.7 times faster than the exact DFSI algorithm, yet only 1.2 times faster than the exact ESI algorithm. The ESI is, on average, 2.25 times faster than the DFSI method. Because it was demonstrated in the previous test 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 savings of a factor of four compared with using the existing exact algorithms for local search evaluations. This indicates that an efficient evaluation method using the forward and backward start intervals significantly accelerates the evaluation of local search moves.

5.3. Comparison with State-of-the-Art Results

In the final experiment, we compared the performance of the proposed AVNS with state-of-the-art results published by Belhaiza, Hansen, and Laporte (2014) and those reported in the recent conference paper by Belhaiza, M'Hallah, and Brahim (2017). The 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 influence of exact local search evaluations.

Belhaiza, Hansen, and Laporte (2014) used a computer with a 3.3 GHz Intel Core i5 vPro processor and 3.2 GB of RAM, and performed 10 runs of at most 25,000 iterations per instance. Belhaiza, M'Hallah, and Brahim (2017) performed a single run of 100,000 iterations on a computer with a 3.3 GHz Intel Core i5 vPro processor and 4 GB of RAM. The proposed AVNS was executed one time for 100,000 iterations. The computational results are presented per instance in Table 3. For each solution method, the 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." For Belhaiza, Hansen, and Laporte (2014), the best results over the 10 runs are reported.

The best known solution for every instance is indicated in bold. Of the 48 instances, the E-AVNS improved 39 instances of the best published results of Belhaiza, Hansen, and Laporte (2014) and 22 instances compared with Belhaiza, M'Hallah, and Brahim (2017). The RM heuristic used in the AVNS performed extremely well because the AVNS always identifies a solution with the minimal number of vehicles. The number of vehicles used in the AVNS was on average 1.3% and 1.9% less than those of BMB17 and BHL14, respectively. The E-AVNS had, on average, the lowest objective value, and the BMB17 algorithm had, on average, the lowest duration.

The two-sided sign test indicates that neither the E-AVNS nor BMB17 algorithm dominates the other. With a p-value of 0.77, the null hypothesis that both algorithms are equally likely to produce better results compared with the other is not rejected. Conversely, the significantly improved results of the E-AVNS over the BHL14 method are confirmed by the two-sided sign test with a significance level of 1.5×10^{-5}. The Wilcoxon two-sided rank test confirms these results, with p-values of 0.3 and 1.4×10^{-5}.

In Table 4, the average running time and average execution time in seconds to obtain the best result are given in rows "Total time" and "Time best," respectively. The BHL14 algorithm had an average computational time of 70.2 seconds per run (they executed their algorithm 10 times). BMB17 reported an average execution time of 81.2 seconds to obtain the best solution per instance. The average computational time to obtain the best solution of the E-AVNS was less; however, we were executing on a marginally faster computer. Therefore, we consider the computational times of the BMB17 and E-AVNS algorithms to be comparable.

In the above computational results, the objective consisted of minimizing the total duration and vehicle cost. Belhaiza, Hansen, and Laporte (2014) and Belhaiza, M'Hallah, and Brahim (2017) developed their algorithms to also minimize a second objective of minimizing the total travel time (without waiting

Table 2. Results from Different Evaluation Methods in the AVNS Algorithm

	Approximate			Exact			
Instance	nVeh	Duration	Time	nVeh	Duration	DFSI	ESI
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
CM1	10.6	1,277.5	34.9	10.6	1,198.7	91.7	44.1
CM2	4.6	1,038.1	57.0	4.6	982.4	128.6	62.5
RCM1	10.3	1,234.8	33.6	10.3	1,214.7	78.9	47.7
RCM2	2.1	809.2	99.8	2.1	769.1	269.6	100.7
Average	6.5	1,007.5	62.5	6.5	967.8	169.5	75.3

Note. The table shows the average number of vehicles (nVeh), 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.

Table 3. Number of Vehicles (nVeh), Total Duration (Dur), and Objective Value (Obj) of Belhaiza, Hansen, and Laporte (2014), Belhaiza, M'Hallah, and Brahim (2017), A-AVNS, and E-AVNS

	BHL14			BMB17			A-AVNS			E-AVNS		
Instance	nVeh	Dur	Obj	nVeh	Dur	Obj	nVeh	Dur	Obj	nVeh	Dur	Obj
RM101	10	1,041.9	3,041.9	10	1,027.1	3,027.1	10	1,064.5	3,064.5	10	1,026.1	**3,026.1**
RM102	9	965.1	2,765.1	9	951.2	**2,751.2**	9	988.6	2,788.6	9	974.8	2,774.8
RM103	9	908.5	2,708.5	9	903.0	2,703.0	9	927.0	2,727.0	9	900.6	**2,700.6**
RM104	9	918.0	2,718.0	9	901.2	**2,701.2**	9	933.2	2,733.2	9	907.1	2,707.1
RM105	9	888.8	2,688.8	9	887.2	**2,687.2**	9	913.2	2,713.2	9	890.5	2,690.5
RM106	9	892.9	**2,692.9**	9	908.4	2,708.4	9	927.6	2,727.6	9	914.8	2,714.8
RM107	9	901.4	2,701.4	9	892.8	**2,692.8**	9	916.1	2,716.1	9	900.4	2,700.4
RM108	9	929.1	2,729.1	9	922.6	**2,722.6**	9	935.9	2,735.9	9	938.1	2,738.1
RM201	3	808.2	3,808.2	3	805.4	3,805.4	2	925.9	2,925.9	**2**	888.9	**2,888.9**
RM202	2	739.0	2,739.0	2	706.8	**2,706.8**	2	754.0	2,754.0	2	721.9	2,721.9
RM203	2	710.3	2,710.3	2	696.9	2,696.9	2	700.2	2,700.2	2	693.2	**2,693.2**
RM204	2	691.9	2,691.9	2	674.5	2,674.5	2	692.7	2,692.7	2	671.7	**2,671.7**
RM205	2	689.9	2,689.9	2	668.1	**2,668.1**	2	681.7	2,681.7	2	668.4	2,668.4
RM206	2	703.4	2,703.4	2	684.9	2,684.9	2	689.2	2,689.2	2	672.6	**2,672.6**
RM207	2	701.7	2,701.7	2	664.3	2,664.3	2	667.9	2,667.9	2	662.4	**2,662.4**
RM208	2	682.8	2,682.8	2	664.3	2,664.3	2	678.4	2,678.4	2	663.6	**2,663.6**
cm101	10	1,320.0	3,320.0	10	1,319.1	**3,319.1**	10	1,496.5	3,496.5	10	1,345.4	3,345.4
cm102	12	1,092.1	3,492.1	11	1,210.7	**3,410.7**	11	1,258.7	3,458.7	11	1,282.3	3,482.3
cm103	12	1,241.1	3,641.1	12	1,232.4	3,632.4	11	1,466.7	3,666.7	**11**	1,392.2	**3,592.2**
cm104	14	1,287.8	4,087.8	14	1,298.0	4,098.0	13	1,378.4	3,978.4	**13**	1,327.8	**3,927.8**
cm105	11	883.4	3,083.4	10	1,027.0	**3,027.0**	10	1,097.5	3,097.5	10	1,066.3	3,066.3
cm106	10	1,073.9	3,073.9	10	1,059.0	**3,059.0**	10	1,191.3	3,191.3	10	1,066.4	3,066.4
cm107	11	1,124.2	3,324.2	11	1,118.0	3,318.0	10	1,138.6	3,138.6	**10**	1,108.4	**3,108.4**
cm108	10	990.4	2,990.4	10	986.0	2,986.0	10	1,010.3	3,010.3	10	985.9	**2,985.9**
cm201	5	1,020.1	4,520.1	5	998.8	4,498.8	5	1,076.6	4,576.6	5	968.4	**4,468.4**
cm202	6	827.3	5,027.3	6	825.1	5,025.1	6	879.6	5,079.6	6	820.2	**5,020.2**
cm203	5	997.2	4,497.2	5	965.8	**4,465.8**	5	1,062.6	4,562.6	5	986.5	4,486.5
cm204	5	859.8	4,359.8	5	844.0	**4,344.0**	5	899.4	4,399.4	5	856.9	4,356.9
cm205	4	1,084.1	3,884.1	4	1,027.8	**3,827.8**	4	1,107.7	3,907.7	4	1096.8	3,896.8
cm206	4	967.7	3,767.7	4	913.2	**3,713.2**	4	991.1	3,791.1	4	933.4	3,733.4
cm207	4	1,209.7	4,009.7	4	1,163.7	**3,963.7**	4	1,207.5	4,007.5	4	1,163.7	**3,963.7**
cm208	4	988.1	3,788.1	4	949.7	**3,749.7**	4	983.7	3,783.7	4	956.8	3,756.8
RCM101	10	1,098.9	3,098.9	10	1,081.2	3,081.2	10	1,093.3	3,093.3	10	1,080.6	**3,080.6**
RCM102	10	1,222.6	3,222.6	10	1,188.3	3,188.3	10	1,194.8	3,194.8	10	1,184.3	**3,184.3**
RCM103	10	1,174.3	3,174.3	10	1,150.4	3,150.4	10	1,171.0	3,171.0	10	1,148.3	**3,148.3**
RCM104	10	1,156.3	3,156.3	10	1,144.0	3,144.0	10	1,157.8	3,157.8	10	1,141.2	**3,141.2**
RCM105	10	1,216.7	3,216.7	10	1,207.0	**3,207.0**	10	1,233.3	3,233.3	10	1,208.2	3,208.2
RCM106	10	1,219.9	3,219.9	10	1,181.7	**3,181.7**	10	1,211.1	3,211.1	10	1,191.8	3,191.8
RCM107	11	1,342.4	3,542.4	11	1,321.5	3,521.5	11	1,325.5	3,525.5	11	1,316.5	**3,516.5**
RCM108	11	1,414.5	3,614.5	11	1,365.2	**3,565.2**	11	1,389.5	3,589.5	11	1,366.2	3,566.2
RCM201	2	783.6	2,783.6	2	783.2	**2,783.2**	2	890.1	2,890.1	2	800.1	2,800.1
RCM202	2	847.1	2,847.1	2	779.4	**2,779.4**	2	838.4	2,838.4	2	822.9	2,822.9
RCM203	2	721.9	**2,721.9**	2	722.0	2,722.0	2	814.2	2,814.2	2	771.7	2,771.7
RCM204	2	726.5	2,726.5	2	708.5	**2,708.5**	2	758.9	2,758.9	2	716.0	2,716.0
RCM205	2	754.5	2,754.5	2	754.5	**2,754.5**	2	796.5	2,796.5	2	756.0	2,756.0
RCM206	2	812.7	2,812.7	2	803.3	2,803.3	2	822.7	2,822.7	2	725.0	**2,725.0**
RCM207	3	764.2	3,764.2	3	761.5	3,761.5	3	789.9	3,789.9	3	757.1	**3,757.1**
RCM208	2	791.4	2,791.4	2	742.7	2,742.7	2	759.4	2,759.4	2	735.1	**2,735.1**
Average	6.58	962.2	3,231.0	6.54	**949.8**	3,210.2	6.46	997.7	3,224.8	**6.46**	1,033.7	**3,189.0**

Notes. The service time is not included in the duration or objective. The results including service time as in Belhaiza, Hansen, and Laporte (2014) and Belhaiza, M'Hallah, and Brahim (2017) are presented in Online Appendix F. The best known solutions are given in bold.

time) and vehicle cost. Because the proposed start interval algorithm was developed to minimize the waiting time of a route, it is less useful for minimizing travel time excluding waiting time. However, the proposed algorithm without modification could improve 17 of the 48 best known results of Belhaiza, Hansen, and Laporte (2014) and Belhaiza, M'Hallah, and Brahim (2017), and has an average saving of 0.5% per instance. The results per instance can be found in Online Appendix G.

Table 4. Average Total Running Time and Time to Obtain the Best Solution in Seconds for Different Heuristics

	BHL14	BMB17	A-AVNS	E-AVNS
Total time	70.2×10		92.1	105.4
Time best		81.2	51.7	67.6

Last, the influence of the exact evaluations in the local search can be determined by comparing the E-AVNS with the A-AVNS. The average total computational time of the E-AVNS was 14.4% greater than that of the A-AVNS, yet for both methods the calculation, time remains short. The duration of the E-AVNS was less in 46 of the 48 instances, with an average improvement of 3.5%. The A-AVNS could not identify any best known solutions of the benchmark instances. Therefore, we conclude that including the proposed efficient exact route duration minimization algorithm enables determining state-of-the-art results in a simple metaheuristic framework.

6. Conclusions

In the VRPMTW, the duration of a route can be minimized by determining the optimal selection of the time windows. Routing problems with multiple time windows occur frequently in practice, yet receive relatively minimal attention in the literature. Tricoire et al. (2010) and Belhaiza, Hansen, and Laporte (2014) developed exact algorithms to minimize the duration of a given route. Belhaiza, Hansen, and Laporte (2014) used the same exact approach to evaluate a move in the 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 presented 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 formally demonstrated that the forward start interval algorithm has a reduced worst-case complexity than the existing exact route duration minimization algorithms of Tricoire et al. (2010) and Belhaiza, Hansen, and Laporte (2014). Second, the experimental tests indicate that the forward start interval algorithm is at least twice as fast as the existing algorithms in terms of average computational effort. Third, the start interval approach was embedded in an AVNS to efficiently recalculate the exact route duration when neighborhood operations are evaluated in the local search. Experimental tests indicated that these efficient exact local search evaluations resulted in an acceleration by a factor of four compared with using the existing exact algorithms in the local search. Furthermore, we demonstrated that the exact evaluations significantly improved the solution quality compared with an approximate evaluation approach, where the increase in the computational time was relatively small. Finally, the proposed algorithm identified new best known solutions for 22 of the 48 benchmark instances. Therefore, we believe that the proposed algorithm can be useful for solving other routing and scheduling problems involving multiple time windows, such as routing problems with arrival time diversification (Hoogeboom and Dullaert 2019) and long-haul transport (Goel and Kok 2012).

Acknowledgments

The authors thank the referees for valuable comments.

References

Beheshti AK, Hejazi SR, Alinaghian M (2015) The vehicle routing problem with multiple prioritized time windows: A case study. *Comput. Indust. Engrg.* 90(December):402–413.

Belhaiza S (2018) A game theoretic approach for the real-life multiple-criterion vehicle routing problem with multiple time windows. *IEEE Systems J.* 12(2):1251–1262.

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

Belhaiza S, M'Hallah R, Brahim GB (2017) A new hybrid genetic variable neighborhood search heuristic for the vehicle routing problem with multiple time windows. Lozano JA, ed. *IEEE Congress Evolutionary Comput.* (Institute of Electrical and Electronics Engineers, Piscataway, NJ), 1319–1326.

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

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

Goel A (2012) The minimum duration truck driver scheduling problem. *EURO J. Transportation Logist.* 1(4):285–306.

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

Hoogeboom M, Dullaert W (2019) Vehicle routing with arrival time diversification. *Eur. J. Oper. Res* 275(1):93–107.

Hurkała J (2015) Time-dependent traveling salesman problem with multiple time windows. *Ann. Comput. Sci. Inform. Systems* 6(September): 71–78.

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. *Science* 220(4598):671–680.

Mladenović N, Hansen P (1997) Variable neighborhood search. *Comput. Oper. Res.* 24(11):1097–1100.

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

Paulsen N, Diedrich F, Jansen K (2015) Heuristic approaches to minimize tour duration for the TSP with multiple time windows. Italiano GF, Schmidt M, eds. *15th Workshop Algorithmic*

Approaches Transportation Model. Optim. Systems, OpenAccess Series in Informatics, vol. 48 (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany), 42–55.

Pesant G, Gendreau M, Potvin JY, Rousseau JM (1999) On the flexibility of constraint programming models: From single to multiple time windows for the traveling salesman problem. *Eur. J. Oper. Res.* 117(2):253–263.

Rancourt ME, Cordeau JF, Laporte G (2013) Long-haul vehicle routing and scheduling with working hour rules. *Transportation Sci.* 47(1):81–107.

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

Savelsbergh MW (1992a) Computer aided routing. Unpublished doctoral thesis, Centrum voor Wiskunde en Informatica, Amsterdam.

Savelsbergh MW (1992b) The vehicle routing problem with time windows: Minimizing route duration. *ORSA J. Comput.* 4(2):146–154.

Schneider M, Schwahn F, Vigo D (2017) Designing granular solution methods for routing problems with time windows. *Eur. J. Oper. Res.* 263(2):493–509.

Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* 35(2): 254–265.

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

Stenger A, Vigo D, Enz S, Schwind M (2013) An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. *Transportation Sci.* 47(1):64–80.

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

Toth P, Vigo D (2014) *Vehicle Routing: Problems, Methods, and Applications*, vol. 18 (SIAM, Philadelphia).

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

Voudouris C, Tsang EP (2003) Guided local search. Glover F, Kochenberger GA, eds. *Handbook of Metaheuristics* (Springer, Berlin), 185–218.