Flex Serv Manuf J (2010) 22:83–108
DOI 10.1007/s10696-011-9077-4

A dynamic programming heuristic for vehicle routing with time-dependent travel times and required breaks

A. L. Kok · E. W. Hans · J. M. J. Schutten · W. H. M. Zijm

Received: 24 March 2010 / Accepted: 20 January 2011 / Published online: 18 February 2011

Abstract For the intensively studied vehicle routing problem (VRP), two real-life restrictions have received only minor attention in the VRP-literature: traffic congestion and driving hours regulations. Traffic congestion causes late arrivals at customers and long travel times resulting in large transport costs. To account for traffic congestion, time-dependent travel times should be considered when constructing vehicle routes. Next, driving hours regulations, which restrict the available driving and working times for truck drivers, must be respected. Since violations are severely fined, also driving hours regulations should be considered when constructing vehicle routes, even more in combination with congestion problems. The objective of this paper is to develop a solution method for the VRP with time windows (VRPTW), time-dependent travel times, and driving hours regulations. The major difficulty of this VRPTW extension is to optimize each vehicle's departure times to minimize the duty time of each driver. Having compact duty times leads to cost savings. However, obtaining compact duty times is much harder when time-dependent travel times and driving hours regulations are considered. We propose a restricted dynamic programming (DP) heuristic for constructing the vehicle routes, and an efficient heuristic for optimizing the vehicle's departure times for each (partial) vehicle route, such that the complete solution algorithm runs in

A. L. Kok (✉)
Algorithmic R&D, ORTEC, P.O. Box 490, 2800 AL Gouda, Netherlands
e-mail: leendert.kok@ortec.com

E. W. Hans · J. M. J. Schutten · W. H. M. Zijm
Operational Methods for Production and Logistics, University of Twente,
P.O. Box 217, 7500 AE Enschede, Netherlands
e-mail: e.w.hans@utwente.nl

J. M. J. Schutten
e-mail: m.schutten@utwente.nl

W. H. M. Zijm
e-mail: w.h.m.zijm@utwente.nl

polynomial time. Computational experiments demonstrate the trade-off between travel distance minimization and duty time minimization, and illustrate the cost savings of extending the depot opening hours such that traveling before the morning peak and after the evening peak becomes possible.

Keywords Restricted dynamic programming · Time-dependent travel times · Driving hours regulations · Vehicle routing problem with time windows · Duty time minimization

1 Introduction

For companies that practice vehicle routing, realizing compact driver duty times leads to substantial savings regarding, e.g., truck driver hiring costs, and the time vehicles are unavailable for other services. Compact duty times are in most countries even required by law: the European Community (EC) social legislation on driving and working hours (European Union 2006), for example, limits the daily driving and duty times of truck drivers.

To obtain compact driver duty times, the departure times within vehicle routes must be optimized within the applicable regulations. Two real-life restrictions make departure time optimization within vehicle routes particularly difficult: *time-dependent travel times* and *driving hours regulations* (Kok et al. 2010a). As traffic congestion typically occurs during peak hours, time-dependent travel times need to be accounted for to obtain robust vehicle routes. Driving hours regulations require the scheduling of mandatory breaks and rest periods after a certain amount of driving time. Therefore, solution approaches for vehicle routing problems and dedicated decision support systems should account for these real-life restrictions.

The difficulty of selecting feasible departure times under driving hours regulations is illustrated in the works of Xu et al. (2003), Archetti and Savelsbergh (2009), and Goel and Kok (2009a, b). Xu et al. (2003) conjecture that finding a feasible driver schedule for a given visit sequence under the US Federal Motor Carrier Safety Administration (2008) and multiple time windows is NP-hard. Archetti and Savelsbergh (2009) show that this problem under single time windows is polynomially solvable by presenting a cubic time algorithm for this problem. Goel and Kok (2009b) propose an improved algorithm for the problem considered in Archetti and Savelsbergh (2009) that runs in quadratic time. Goel and Kok (2009a) propose a quadratic time algorithm for a similar problem, but under the EC social legislation with team truck drivers.

The combination of duty time minimization within the construction of vehicle routes, accounting for time-dependent travel times, and obeying driving hours regulations is a highly complex problem, which has—to the best of our knowledge—not been addressed so far. The objective of this paper is to develop a solution method for the VRPTW with time-dependent travel times and the EC social legislation on driving and working hours (TDVRP-EC). Since the EC social legislation is more restrictive than the US Federal Motor Carrier Safety

Administration (2008), any solution method for the TDVRP-EC can also solve the TDVRP with the US Hours-Of-Service Regulations.

The VRP has been extensively studied in the literature (for an extensive overview, see Toth and Vigo 2002). The vehicle routing problem with time-dependent travel times (TDVRP, Malandraki and Daskin 1992) and the vehicle routing problem with the EC social legislation on driving and working hours (Goel 2009), however, have drawn only minor attention from scientists. To the best of our knowledge, this is the first paper that addresses these common timing restrictions together in one model.

Local search methods have proved to be successful in solving large vehicle routing and scheduling problems (Funke et al. 2005). However, it is hard to efficiently incorporate complex timing restrictions in local search methods, since customer insertions and removals have complex up- and downstream effects on the routes under consideration. This makes the evaluation of neighborhood solutions computationally expensive.

For the TDVRP, Ichoua et al. (2003) resolve this problem of computationally expensive checks by considering soft time windows and an estimation function for the neighborhood solutions. Only the most promising neighborhood solutions are evaluated explicitly. This procedure fails in case hard time windows are considered, since then the feasibility of a neighborhood solution must be evaluated exactly. Moreover, it is not possible to account for the EC social legislation with this procedure.

For the VRPTW under the EC social legislation, Goel (2009) proposes a large neighborhood search heuristic, which is based on successively applying customer removals and insertions to improve some initial solution. To account for the EC social legislation, Goel (2009) proposes a labeling algorithm that checks for each customer insertion and removal whether it is admissible. The solutions obtained by this method are substantially improved by the restricted DP heuristic of Kok et al. (2010b). This heuristic is an extension of the DP heuristic proposed by Gromicho et al. (2008), which is a construction heuristic that sequentially constructs vehicle routes by adding customers to the end of a partial vehicle route. The EC social legislation is accounted for by embedding a break scheduling algorithm within the DP heuristic. This break scheduling algorithm only schedules breaks locally, avoiding computationally expensive checks upstream in the partial vehicle routes, and runs in constant time. As a result, the running time complexity of the DP heuristic for the VRPTW with the EC social legislation is the same as the running time complexity of the DP heuristic for the traditional VRPTW. Following this promising result for the VRPTW with the EC social legislation, we propose a solution method for the TDVRP-EC based on the DP heuristic of Gromicho et al. (2008).

In the context of time-dependent vehicle routing, Hashimoto et al. (2008) also consider dynamic programming. However, they apply dynamic programming for determining the optimal start time of a given vehicle route, whereas we apply it for *constructing* the vehicle routes. Related works considering the TDVRP are of Fleischmann et al. (2004), Van Woensel et al. (2008), and Donati et al. (2008).

A closely related topic to developing solution methods for the TDVRP is the topic of how to obtain the time-dependent travel times for real world vehicle routing problems (see, amongst others, Kolesar et al. 1975; see, amongst others, Ehmke et al. 2009, 2010). These works discuss data collection and conversion methods, such that it can be used as input for time-dependent vehicle routing problems.

In the VRPTW literature, heuristic solution methods generally use a lexicographic objective function in which the primary objective is to minimize the number of vehicles used and the secondary objective is to minimize the total distance traveled. However, within the VRPTW this secondary objective may lead to large waiting times, which are costly in practice. Moreover, traffic congestion makes the duration of travels (and thus also the costs of these travels) depend on the time of the day, while the distance remains the same. Therefore, a more relevant secondary objective is to minimize the total duty time (Savelsbergh 1992). We numerically analyze both travel distance and duty time as the secondary objective. Moreover, we quantify the impact of extending the depot opening hours, such that traveling before the morning peak and after the evening peak becomes possible.

This paper is organized as follows. Section 2 formally introduces the TDVRP-EC. Section 3 discusses some important assumptions considering waiting times at customers that have a strong impact on the complexity of the departure time optimization problem. Section 4 proposes a solution approach for the TDVRP-EC, based on the DP heuristic of Gromicho et al. (2008). In Sect. 5, we report on computational experiments to analyze the impact of different objective functions (minimize travel distance vs. minimize duty time) on the overall solution quality, and the impact of extending the depot opening hours. In Sect. 6, we summarize our main findings.

2 Problem description of the TDVRP-EC

We consider an extension of the classical VRPTW for which we first introduce some notation that we require throughout this paper. Within the VRPTW, we are given a set of vehicles $K = \{1, \ldots, m\}$ and a set of nodes $V = \{0, \ldots, n\}$ in which node 0 represents the depot. Nodes $i > 0$ represent customer requests with demands q_i and service time windows $[e_i, l_i]$. The problem is to find a set of routes, each starting and ending at the depot, such that the total demand along each route does not exceed the vehicle capacity Q, each service starts in the given time window, and some objective function is optimized.

We extend the VRPTW by considering time-dependent travel times and driving hours regulations. We assume that (aggregated) data is available for time-dependent travel speeds along customer-to-customer routes. In other words, we do not consider the underlying road network in which (time-dependent) shortest paths should be determined. The calculation of (time-dependent) shortest paths can be done in a pre-processing phase and from these paths the required aggregated travel data for customer-to-customer routes can be obtained, as demonstrated in Kok et al. (2009). To model the time-dependent travel times, we apply the time-dependent speed model of Ichoua et al. (2003), which satisfies the non-passing property (the

non-passing property states that overtaking is not possible). There are two main reasons for this approach: (1) the non-passing property is a realistic property (2) a more detailed travel time function (e.g., any differentiable travel time function) is not realistic to obtain from, e.g., historical travel time data.

In this paper, we consider Regulation (EC) 561/2006 on driving and working hours (European Union 2006), which is valid for all member countries of the European Union. Furthermore, we consider one-day planning in which all customer requests are known in advance and we assume that breaks and rests have to be scheduled at customer locations. The choice for one-day planning is motivated by practice, since duty time minimization is applied to one-day schedules because the costs applied for night rests on duty differ from those for working times. Considering one-day planning, Regulation (EC) 561/2006 poses the following requirements per driver:

1. A period between two breaks of at least 45 min is called a *driving period*. The accumulated driving time in a driving period may not exceed 4.5 h. The break that ends a driving period may be reduced to 30 min if an additional break of at least 15 min is taken anywhere during that driving period. The driving hours regulations do not allow service times at customers to be considered as break time.
2. The total accumulated driving time may not exceed 9 h.
3. The total accumulated duty time may not exceed 13 h.

The TDVRP-EC comprises three types of decisions: assigning customers to vehicles, sequencing customer visits for each vehicle, and selecting departure times for each vehicle. Departure times need not only be determined for the departure at the depot, but also at each customer to account for the driving hours regulations and the time windows. The opportunity to schedule waiting times at customers makes this departure time scheduling problem particularly difficult, as we shall illustrate in Sect. 3. Therefore, we discuss in Sect. 3 the scheduling of waiting times and our underlying assumptions in detail.

3 Waiting time assumptions

In order to construct feasible vehicle routes, we need a method that finds feasible departure times for these routes. Furthermore, the costs of such routes have to be determined in terms of duty times. Kok et al. (2010a) propose an ILP model to optimize vehicle departure times given the customer visit sequence of a vehicle route. We refer to this problem as the vehicle departure time optimization problem (VDO). When constructing vehicle routes in the DP heuristic (see Sect. 4), however, computation times to solve this ILP are too large to apply it for each (partial) vehicle route that is considered.

A complicating factor for the determination of the minimum duty time is the use of *unforced waiting time*. We define unforced waiting time as waiting time that is not forced by either time windows of customers or by driving hours regulations induced breaks. For example, if departing at time 0 from the depot leads to an arrival time of 2 at the first customer, but the earliest feasible time to start service at this customer is 5, then a waiting time of 3 is introduced. We call this *unforced*

waiting time, because it can be avoided by departing at time 3 from the depot (assuming time-independent travel times in this example). However, if departing from customer i at its latest feasible departure time (i.e., starting its service at its deadline l_i and departing directly after this service) still results in an early arrival at the next customer j, then we call this *forced* waiting time. As an illustration of how to profitably introduce unforced waiting time, suppose that direct continuation from a customer results in a total driving time of slightly more than 4.5 h, which requires an additional 45 min break before completing the vehicle route. However, if postponing the departure time by a small amount of time (unforced waiting) reduces the total driving time below 4.5 h (e.g., due to less traffic), then no additional break is required and we end up with an earlier completion.

The problem of exploiting unforced waiting time is that its profitability is difficult to measure, since it requires for each customer addition (or customer insertion, customer removal, etc.) a recheck at each visited customer for introducing unforced waiting time. To keep track of all possibly profitable unforced waiting times is thus computationally expensive. We consider the variant of the VDO in which introducing unforced waiting time is not considered. In addition, we choose to not schedule early breaks (which means that we also not split up breaks in a 15 min part and a 30 min part), even not when there is sufficient forced waiting time. In Sect. 5, we numerically analyze the effect of not considering unforced waiting time and early breaks by optimizing each vehicle route with the exact solution approach of Kok et al. (2010a) as a post-processing step. In the next section, we propose a solution method for the TDVRP-EC and for the VDO subproblem in which unforced waiting time and early breaks are not considered.

4 Solution approach

We solve the TDVRP-EC using the restricted dynamic programming framework of Gromicho et al. (2008). As illustrated by Kok et al. (2010b), this framework is suitable for incorporating complex timing restrictions such as driving hours regulations. The DP formulation constructs one tour and is applied to the VRP through the giant-tour representation (GTR) of vehicle routing solutions (Funke et al. 2005). The basic DP formulation for routing problems (without time-dependent travel times) is as follows.

Each state (S, i) represents the minimum cost path of starting in node 0, visiting all customers in $S \subseteq V\backslash\{0\}$, and ending at customer $i \in S$. The costs of each state are represented by $C(S, i)$, and they are calculated by the following recurrence relation, in which c_{ij} represents the (time-independent) travel costs of traveling from node i to node j:

$$|S| = 1 : C(\{i\}, i) = c_{0i} \quad \forall i \in V\backslash 0.$$
$$|S| > 1 : C(S, i) = \min_{j \in S\backslash\{i\}} \{C(S\backslash\{i\}, j) + c_{ji}\}.$$

Finally, the costs of the optimal tour are calculated with:

$$\min_{j \in V \setminus \{0\}} \left\{ C(V \setminus \{0\}, j) + c_{j0} \right\}.$$

The giant-tour representation of vehicle routing solutions connects vehicle routes by ordering the vehicles and introducing start- and end-nodes for each vehicle route. Next, successive vehicles are connected by connecting the corresponding end- and start-node (i.e., we introduce precedence relations for the start- and end-nodes of the vehicles). Figure 1 presents an example of a VRP solution with three vehicles, two depots (A and B) and nine customers. Vehicle 1 starts at depot A and ends at depot B, vehicle 2 starts and ends at depot B, and vehicle 3 starts and ends at depot A. Figure 2 presents the same solution with its corresponding giant-tour representation.

To solve vehicle routing problems with the DP formulation, we apply it to the extended node set concerned with the giant-tour representation of vehicle routing solutions. When a state is expanded with a vehicle route-end node (e.g., node d_1 in Fig. 2), then the associated vehicle route is completed. In the next stage, we consider the route-start node of the successive vehicle (o_2 in this case) as the only feasible expansion, such that a new vehicle route is started. In order to obtain feasible vehicle routes, we add state dimensions that indicate, e.g., the remaining capacity of a vehicle, the current time (which is needed to determine the right travel times), the remaining travel time until a break must be scheduled. When we expand a state, we perform feasibility checks to ensure that vehicle capacities are not exceeded, time windows are not violated, etc. This implies that, for example, when a state is expanded by a vehicle end-node, then all state dimensions are set to the initial conditions of the next vehicle (remaining capacity is set to the vehicle's capacity, current time is set to 0, etc.).

All states with the same cardinality of S form a stage. The so-called stage width equals the total number of states in that stage. To obtain practical computation times, we bound the stage width with a value H, such that only the H lowest cost

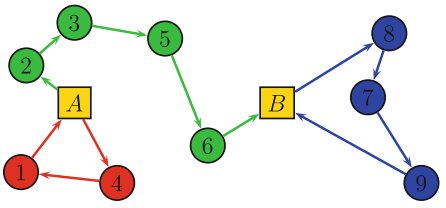

Fig. 1 Example of a solution to a VRP with three vehicles

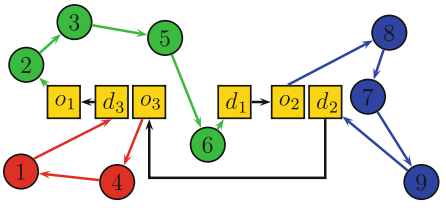

Fig. 2 The giant-tour representation of a solution to the VRP of Fig. 1

states in each stage are expanded. Since all states belonging to the same stage correspond to partial VRP solutions in which the same number of nodes are visited, low cost states are most likely to lead to good overall VRP solutions. The costs of each state are based on the partial VRP solution it represents.

To account for lexicographic objective functions, such as minimizing the number of vehicles used as the primary objective and minimizing the total duty time as the secondary objective, we define the cost of each state $C(S, i)$ as a tuple (number of vehicles used, total duty time). As a consequence, states are only compared with respect to their secondary criterion if they are equal with respect to their first criterion. Note that for each state the first criterion 'number of vehicles used' is only increased when a route-end node is added **and** the previously added node was **not** the route-start node (otherwise, we would count an empty route). Since this may imply that constructing empty routes is free with respect to the objective function (e.g., when the vehicle starts and returns at the same location), we forbid constructing empty routes when there are still customer nodes to be added.

In order to apply the DP heuristic to our problem, we need a method that checks for each state expansion whether there exists a feasible departure schedule for the corresponding partial vehicle route. Furthermore, the costs of such an expansion have to be determined in terms of duty times. In the remainder of this section, we propose a polynomial time algorithm for the VDO without unforced waiting time and early breaks. This VDO algorithm develops a time-dependent duty time function for the entire vehicle route under consideration. We describe how a duty time function based on time-dependent driving speeds can be represented in a duty time record with $O(p)$ elements, with p the maximum number of times the speed changes on a route. Section 4.1 describes how to update the duty time record each time a node is added to a partial vehicle route. We show that each such node addition introduces at most $O(p)$ new elements, resulting in $O(np)$ elements for the duty time record of the composite duty time function of an entire route. For simplicity, we first assume that service times are zero, no service time windows are given, and no driving hours regulations are present. Sections 4.2 and 4.3 then describe how service times and time windows can be incorporated, respectively, whilst maintaining the $O(np)$ running time complexity. Section 4.4 describes how breaks can be incorporated in order to respect the driving hours regulations. Section 5 derives the resulting running time complexity of the VDO algorithm.

4.1 Adding a node to a partial vehicle route

For simplicity reasons, we assume in this section that service times are zero, and time windows and driving hours regulations do not exist, which implies that driving times equal duty times. However, for reasons of generality, we set up an algorithm at the end of this section that remains valid when time windows are present.

Suppose that the number of speed changes on each route between two nodes is limited by p. These speed changes result in a piecewise linear duty time function. Figure 3a presents such a speed step function for a route $0 \rightarrow i$ with distance 2, and

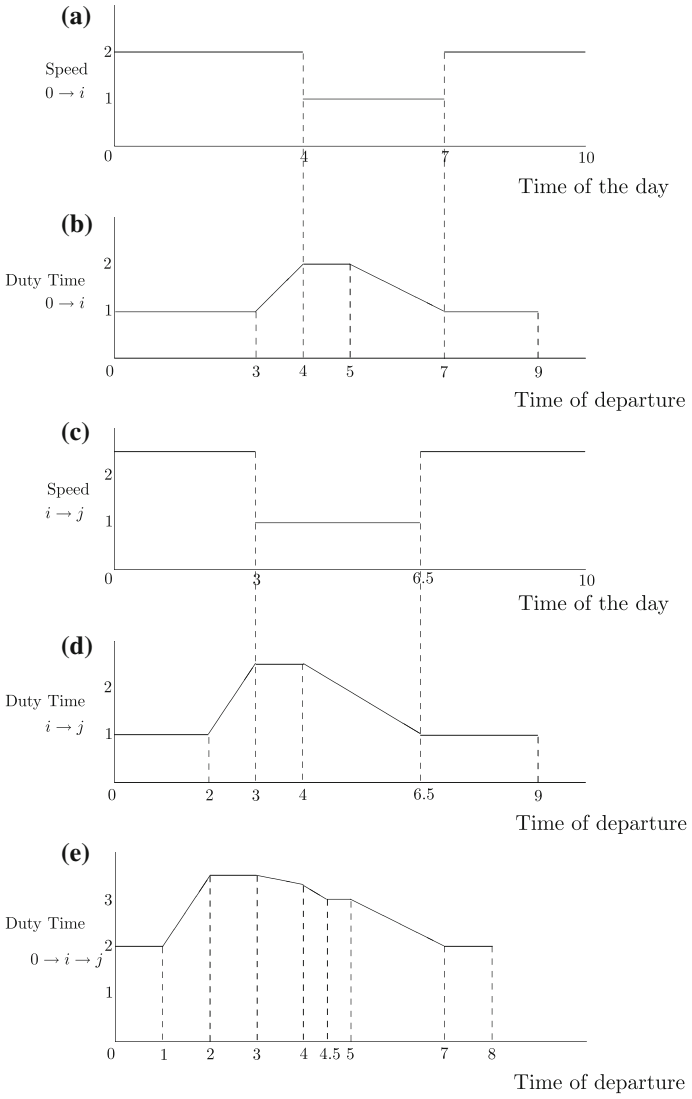

Fig. 3 **a** Speeds route $0 \to i$, **b** duty times route $0 \to i$, **c** speeds route $i \to j$, **d** duty times route $i \to j$, **e** duty times route $0 \to i \to j$

Fig. 3b presents the resulting duty time function for that route as a function of the departure time. Each speed change causes the slope of the duty time function to change at most two times: (1) when the arrival time at node i equals the moment that the speed changes (2) when the departure time from node 0 equals the moment that the speed changes. For example, the speed change at time 4 causes the slope of the duty time function to change at departure times 3 and 4. Therefore, the number of linear pieces of the duty time function is $O(p)$.

Each duty time function of a route z can be represented by a duty time record $r^z = \left(r_1^z, \ldots, r_{U^z}^z\right)$ of $O(p)$ elements. Each record entry $r_u^z = (d_u^z, t_u^z)$ contains two elements: the start time d_u^z of the u-th linear piece of the duty time function and initial height of this piece (i.e., the duty time t_u^z required to (completely) travel route z when departing at time d_u^z from the first node in route z). We assume that for each route $i \rightarrow j$, the travel speeds are given for the entire planning horizon, i.e., for the depot opening hours $[e_0, l_0]$. Therefore, for each route $i \rightarrow j$ we have $d_0^{i \rightarrow j} = e_0$ and $d_{U^{i \rightarrow j}}^{i \rightarrow j} + t_{U^{i \rightarrow j}}^{i \rightarrow j} = l_0$. This allows us to construct the duty time records for each route $i \rightarrow j$ in a pre-processing step without knowing the actual nodes that will be visited before arriving at node i in a solution. Note that when time windows are present, departing at e_0 from node i may not make sense, even when we do not consider the nodes that may be visited before node i in a solution. Section 4.3 describes how to include time windows at customers in the construction of the duty time records during the pre-processing step, which may then result in $d_0^{i \rightarrow j} > e_0$ and $d_{U^{i \rightarrow j}}^{i \rightarrow j} + t_{U^{i \rightarrow j}}^{i \rightarrow j} < l_0$ for certain routes.

The duty time record for the duty time function in Fig. 3b is:

$$r^{0 \rightarrow i} = ((0,1),(3,1),(4,2),(5,2),(7,1),(9,1)).$$

The minimum duty time equals the minimum of all duty time entries. The duty time for a given departure time can be calculated by interpolation. We define the function $T^z(d)$ as the function that gives the duty time needed to travel route z for a given departure time d from the first node in route z.

The duty time for a given arrival time a at the last node in route z can also be calculated using the duty time record r^z. Each departure time $d^z{}_u$ from the first node in route z results in an arrival time of $a_u^z = d_u^z + t_u^z$ at the last node in route z. This arrival time a_u^z corresponds to a duty time of t_u^z. We can determine the duty time for a given arrival time a at the last visited node in route z by interpolation. We define the function $F^z(a)$ as the function that gives the departure time d from the first node in route z that exactly results in an arrival time of a at the last node in route z (i.e., the difference between arrival time a at the last node in route z and the corresponding duty time). A call to this function requires a run through the duty time record. However, the calls we make in Algorithm 1 are with non-decreasing arrival times a. Therefore, we only require one run through the duty time record for all calls to $F^z(a)$ in Algorithm 1. We can do this by storing for each call the required positions in the duty time record to calculate $F^z(a)$, and to continue the search from these positions for the successive call.

We now describe how to derive a new duty time record when a node is added to the end of a partial vehicle route. Suppose that we add a node j to the end of a partial vehicle route corresponding to a state (S, i), i.e., route $i \rightarrow j$ is added to the partial vehicle route. Then, we need to determine the duty time record r^{new} of the new partial vehicle route, which is the composite record of the duty time record r^{old} of the old partial vehicle route from node 0 to node i and the duty time record r^{add} of route $i \rightarrow j$. The duty time function of the new route is the composite function of two piecewise linear functions, which in our case is again a piecewise linear function.

Algorithm 1 VDO algorithm

// Initialization
if $d_0^{old} + t_0^{old} > d_{U^{add}}^{add}$ **then**
  STOP
end if
if $d_{U^{old}}^{old} + t_{U^{old}}^{old} < d_0^{add}$ **then**
  $d_0^{new} \Leftarrow d_{U^{old}}^{old}$
  $t_0^{new} \Leftarrow d_0^{add} + t_0^{add} - d_{U^{old}}^{old}$
  STOP
end if
if $d_0^{old} + t_0^{old} \geq d_0^{add}$ **then**
  $d_0^{new} \Leftarrow d_0^{old}$
else
  $d_0^{new} \Leftarrow F^{old}\left(d_0^{add}\right)$
end if
$t_0^{new} \Leftarrow T^{old}\left(d_0^{new}\right) + T^{add}\left(d_0^{new} + T^{old}\left(d_0^{new}\right)\right)$
if $d_{U^{old}}^{old} + t_{U^{old}}^{old} \leq d_{U^{add}}^{add}$ **then**
  $d_{max}^{new} \Leftarrow d_{U^{old}}^{old}$
else
  $d_{max}^{new} \Leftarrow F^{old}\left(d_{U^{add}}^{add}\right)$
end if
$v \Leftarrow 0$
$u^{add} \Leftarrow 0$
// Main procedure
while $d_v^{new} < d_{max}^{new}$ **do**
  $u^{old} \Leftarrow \arg\min_u \left\{d_u^{old} | d_u^{old} > d_v^{new}\right\}$
  while $d_{u^{add}}^{add} \leq d_v^{new} + T^{old}\left(d_v^{new}\right)$ **do**
    $u^{add} \Leftarrow u^{add} + 1$
  end while
  $v \Leftarrow v + 1$
  if $d_{max}^{new} + T^{old}\left(d_{max}^{new}\right) \geq d_{u^{add}}^{add}$ **then**
    $d_v^{new} \Leftarrow \min\left\{d_{u^{old}}^{old}, F^{old}\left(d_{u^{add}}^{add}\right)\right\}$
  else
    $d_v^{new} \Leftarrow d_{u^{old}}^{old}$
  end if
  $t_v^{new} \Leftarrow T^{old}\left(d_v^{new}\right) + T^{add}\left(d_v^{new} + T^{old}\left(d_v^{new}\right)\right)$
end while

Suppose that r^{old} is the duty time record of the duty time function in Fig. 3b (i.e., the old partial vehicle route is route $0 \rightarrow i$). Furthermore, suppose that the distance of route $i \rightarrow j$ is 2.5 with a speed step function as in Fig. 3c, and resulting duty times as in Fig. 3d. Then we get:

$$r^{old} = ((0,1),(3,1),(4,2),(5,2),(7,1),(9,1)).$$
$$r^{add} = ((0,1),(2,1),(3,2.5),(4,2.5),(6.5,1),(9,1)).$$

The earliest feasible departure time from the first node in the new route $0 \rightarrow i \rightarrow j$ equals d_0^{old} (an earlier departure is not possible and departing at this time does not lead to any waiting time at node i). Therefore, $d_0^{new} := d_0^{old} = 0$. This departure time from node 0 results in an arrival time of 1 at node i. Then, departing at node i at time 1 results in an additional duty time of 1 for traveling from node i to node j (since $d_0^{add} = 0, d_1^{add} = 2$, and $t_0^{add} = t_1^{add} = 1$), which results in a total duty time for route $0 \rightarrow i \rightarrow j$ of $t_0^{new} := 2$. Next, we need to determine the first departure time from node 0 after d_0^{new} at which the slope of the duty time function of the new route changes. This happens at $min\{d_1^{old}, F^{old}(d_1^{add})\}$. We have $d_1^{old} = 3$ and $F^{old}(d_1^{add}) = F^{old}(2) = 1$. Therefore, $d_1^{new} := 1$ with corresponding duty time $t_1^{new} := 2$.

We continue this process, each time determining which departure time is the first to change the slope of the duty time function and calculating the corresponding duty time. This process continues until either $d_{U^{old}}^{old}$ or $F^{old}(d_{U^{add}}^{add})$ has been added. This leads to:

$$r^{new} = ((0,2),(1,2),(2,3.5),(3,3.5),(4,3.3),(4.5,3),(5,3),(7,2),(8,2)).$$

Figure 3e presents the duty time function of the new route.

Algorithm 1 describes a general procedure for determining the composite duty time record r^{new} of the duty time records of the old route r^{old} and the route to be added r^{add}. Recall that when time windows are present, d_0^{add} does not need to be equal to 0. We already account for such cases in Algorithm 1. Note that $F^{old}(a)$ is only defined for the interval $[d_0^{old} + t_0^{old}, d_{U^{old}}^{old} + t_{U^{old}}^{old}]$. We now describe the steps of the algorithm.

In the initialization, we abort if no feasible departure time from the first node in the new route exists (Line 1–3). Next, we check whether departing at the latest feasible departure time from the first node in the old route, i.e. $d_{U^{old}}^{old}$, still results in an early arrival at the first node of the route to be added (Line 4). If this is the case, then the only feasible departure time from the first node in the new route without *unforced* waiting time is $d_{U^{old}}^{old}$. The duty time is then the difference between the earliest completion time at the last node in the new route (which equals $d_0^{add} + t_0^{add}$) and the latest feasible departure time from the first node in the new route (Line 5 and 6). For the remainder, we know that there are multiple feasible departure times without unforced waiting time from the first node in the new route. The earliest of such departure times is either d_0^{old} or $F^{old}(d_0^{add})$ (Line 9–13). Note that we cannot use $F^{old}(d_0^{add})$ in the check in Line 9, since it is not defined when $d_0^{old} + t_0^{old} > d_0^{add}$. The duty time t_0^{new} is equal to the sum of the duty time needed for visiting the nodes in the old route and the duty time needed for visiting the nodes in the route to be added (Line 14). The next step is to determine the latest feasible departure time from the first node in the new route (Line 15–19). This departure time equals either $d_{U^{old}}^{old}$ or $F^{old}(d_{U^{add}}^{add})$. The final step in the initialization is to initialize v and u^{add} (Line

21 and 22). Index v represents the index of the current entry in r^{new}. Index u^{add} is the index of the entry in r^{add} that contains the earliest departure time from the first node in the route to be added that requires a new record entry for r^{new} (i.e., when departing later than d_v^{new} from the first node in the new route, arrival time $d_{u^{add}}^{add}$ at the first node in the route to be added is the earliest arrival time at this node that changes the slope of the duty time function of the new route).

The main procedure adds record entries to r^{new} for each change in the slope of the duty time function of the new route until an entry with departure time d_v^{new} is added. A later departure time than d_v^{new} may cause a change in the slope of the duty time function of the new route both because of a change in the slope of the duty time function of the 'old' part of the new route and because of a change in the slope of the duty time function of the 'added' part of the new route. Therefore, we determine the earliest departure time from the first node in the old route later than d_v^{new} that changes the slope of the duty time function of the old route (Line 23) and we determine u^{add} (Line 24–26). Next, we increase index v (Line 27), and we determine d_v^{new} (Line 28–32). Note that we have to be careful again with the usage of $F^{old}(a)$. If $d_{max}^{new} + T^{old}\left(d_{max}^{new}\right) < d_{u^{add}}^{add}$, then $F^{old}\left(d_{u^{add}}^{add}\right)$ is not defined. When this situation appears, only departure times corresponding to $d_{u^{old}}^{old}$ will be added until $u^{old} = U^{old}$.

4.2 Incorporating service times

Service times can be incorporated by adding them to the driving times. Since service times are constant, they do not affect any of the calculations described before. What typically happens is that the duty time function for a route $i \rightarrow j$ is shifted up and to the left by the service time at node i. By doing this, the duty times include both driving times and service times.

4.3 Incorporating time windows

Suppose we have a route $i \rightarrow j$ with corresponding duty time function (e.g., as in Fig. 3d), and given time windows $[e_i, l_i]$ and $[e_j, l_j]$ for starting service at node i and node j, respectively. For ease of explanation, we again assume that service times are zero. Then, three cases may appear.

Case 1 is when $e_i + T^{i \rightarrow j}(e_i) > l_j$. In that case, the route $i \rightarrow j$ is infeasible, since the earliest feasible time to start service at node i is already too late to arrive ultimately at l_j at node j.

Case 2 is when $l_i + T^{i \rightarrow j}(l_i) < e_j$. This means that, even if we start service at node i as late as possible, we arrive before the earliest feasible time to start service at node j. In this case, the only way to avoid introducing *unforced* waiting time is to start serving node i as late as possible, implying one feasible departure time from node i: l_i. The corresponding duty time is equal to the travel time plus the forced waiting time: $T^{i \rightarrow j}(l_i) + \left(e_j - (l_i + T^{i \rightarrow j}(l_i))\right) = e_j - l_i$.

Case 3 is the remaining case, i.e., the interval of possible arrival times at node j intersects with $[e_j, l_j]$. We then restrict the feasible departure times from node i to the interval in which we arrive in time at node j (i.e., before or at l_j) and we do not

introduce unforced waiting time (i.e., we do not arrive before e_j). This implies that for the earliest feasible departure time from node i without unforced waiting time at node j we get $d_0^{i\to j} := e_i$ if $e_i + T^{i\to j}(e_i) \geq e_j$, and $d_0^{i\to j} := F^{i\to j}(e_j)$ otherwise. Furthermore, we get $d_{U^{i\to j}}^{i\to j} := l_i$ if $l^i + T^{i\to j}(l_i) \leq l^j$, and $d_{U^{i\to j}}^{i\to j} := F^{i\to j}(l_j)$ otherwise.

Suppose in our example node i has a time window [2, 9] and node j has a time window [6, 10]. Furthermore, Fig. 3d presents the duty time record without time windows:

$$r^{i\to j} = ((0,1),(2,1),(3,2.5),(4,2.5),(6.5,1),(9,1)).$$

The time window at node i causes the feasible departure time interval to be restricted to [2, 9], such that:

$$r^{i\to j} := ((2,1),(3,2.5),(4,2.5),(6.5,1),(9,1)).$$

Next, the time window at node j causes that departing from node i earlier than time 3.5 will result in unforced waiting time at node j, resulting in:

$$r^{i\to j} := ((3.5,2.5),(4,2.5),(6.5,1),(9,1)).$$

Figure 4a presents the resulting duty time function.

We construct the duty time records for each route between two nodes in this way during the pre-processing step. Then, we apply Algorithm 1 again to obtain the duty time records for the (partial) vehicle routes. Note that the time windows may substantially reduce the number of record entries. In the extreme case, only one feasible departure time remains, which implies that there is forced waiting time on the route and continuing ASAP is the best we can do in the remainder. Figure 4b

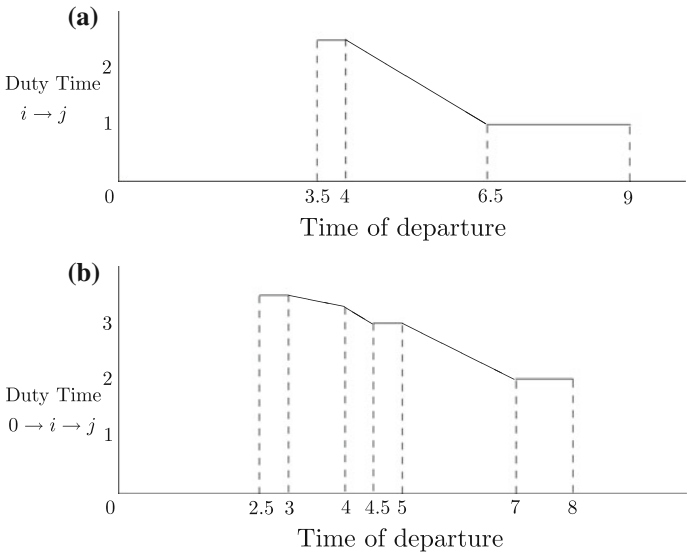

Fig. 4 Duty time records with time windows: **a** duty times route $i \to j$, **b** duty times route $0 \to i \to j$

presents the duty time function of the new route in our example. The number of record entries reduces from 9 to 7.

4.4 Scheduling breaks

To comply with the EC social legislation, we schedule a 45 min break whenever the accumulated driving time of a partial vehicle route is about to exceed 4.5 h. To account for the accumulated driving time, we add an element ta_u^z to each duty time record entry r_u^z, indicating the total accumulated driving time in route z since the last break taken at a customer. Note that the accumulated driving time depends on the chosen departure time d_u^z from the first node in route z. Therefore, we have to account for the accumulated driving time ta_u^z for each departure time d_u^z from the first node in route z. Since we only schedule breaks at customer sites, the values of ta_u^z for the duty time records of each route $i \rightarrow j$, which are constructed in the pre-processing step, equal the driving time from node i to node j for departure time $d_u^{i \rightarrow j}$ from node i. For simplicity reasons, we again assume all service times to be zero.

We assume that driving times between node pairs do not exceed 4.5 h. In case a route $i \rightarrow j$ has a departure time that results in more than 4.5 h of driving time, we assume this route is infeasible. Note that such a route is very unlikely to be selected in a good VRP solution, since the shortest vehicle route in such a solution would be the tour $depot \rightarrow i \rightarrow j \rightarrow depot$ and the total driving time in this tour is likely to exceed its maximum of 9 h. Within the problem instances used for the computational experiments in Sect. 5, the driving time between each pair of nodes and for each departure time does not exceed 4.5 h. If VRPs with a long time horizon are considered, or VRPs with only few customers per vehicle, then it might become necessary to include also routes between two nodes exceeding 4.5 h of driving time. This can be done by, e.g., modeling parking lots along such routes, or by assuming that breaks can be taken anywhere along the routes. These model assumptions do not affect the algorithmic framework, they only affect the calculation of the duty time records.

Now, suppose we add a node j to a partial vehicle route represented by a state (S, i), again with duty time records r^{old}, r^{add}, and r^{new} for the duty time functions of the old route, the route to be added, and the new route, respectively. We define $\tilde{r}^{new}$ to be the duty time record of the new route in which we ignore that a break may have to be scheduled at node i. We use $\tilde{r}^{new}$ to derive for which departure times we do have to schedule a break at node i. Each record entry $\tilde{r}_u^{new}$ contains a departure time $\tilde{d}_u^{new}$, a corresponding duty time $\tilde{t}_u^{new}$, and a corresponding accumulated driving time $\tilde{ta}_u^{new}$ since the last break without a possibly needed break at node i. We can derive $\tilde{r}^{new}$ by applying Algorithm 1 in which we can calculate each $\tilde{ta}_u^{new}$ in a similar way as how we calculate each $\tilde{t}_u^{new}$. Then, three cases may appear:

1. After adding route $i \rightarrow j$, $\tilde{ta}_u^{new} \leq 4.5$ for all $u = 0, \ldots, U^{new}$.
2. After adding route $i \rightarrow j$, $\tilde{ta}_u^{new} > 4.5$ for all $u = 0, \ldots, U^{new}$.
3. After adding route $i \rightarrow j$, $\tilde{ta}_u^{new} > 4.5$ for some, but not all $u = 0, \ldots, U^{new}$.

In Case 1, we do not need to schedule a break for any feasible departure time and we get $r^{new} = \tilde{r}^{new}$. We describe the other two cases in detail.

In Case 2, a break is required at node i regardless of the departure time from the first node in the old route, since we assume that breaks are only taken at customers. With this break, the departure time from node i is delayed by 45 min. The same procedure as in Algorithm 1 can be applied to determine the duty times of the new route, but with 45 min added to all duty times in r^{old}. Since a break is taken at node i, such that the accumulated driving time is reset to 0 when departing from node i, all ta_u^{new} are set to $t_u^{i \to j}$.

In Case 3, we have to split the new duty time record, such that for each partial duty time record either a break is scheduled at node i for each departure time, or no break is scheduled for any departure time. Therefore, we first determine the series of departure times δ_w at which the new duty time record should be split. This is the case if departure time δ_w results in exactly 4.5 h of accumulated driving time (when no break is scheduled at node i), while departing directly before or directly after δ_w results in more than 4.5 h of accumulated driving time (both is also possible). Suppose that u_w is such that $\tilde{d}_{u_w}^{new}$ is the earliest departure time in duty time record $\tilde{r}^{new}$ larger than δ_w (if $\delta_w = \tilde{d}_{U^{new}}^{new}$, we set $\tilde{d}_{u_w}^{new} := \delta_w$). Then, each departure time δ_w results in exactly 4.5 h of accumulated driving time, while $\tilde{ta}_{u_w-1}^{new} > 4.5$ or $\tilde{ta}_{u_w}^{new} > 4.5$. This leads to a series of strictly increasing departure times $\{\delta_1, \ldots, \delta_{W^{new}}\}$ at which the new duty time record should be split. Let's set $\delta_0 := \tilde{d}_0^{new}$ and $\delta_{W^{new}+1} := \tilde{d}_{U^{new}}^{new}$. Then, we split the duty time record of the new route in duty time records $r^{new_w}, w = 0, \ldots, W^{new}$ with earliest and latest departure times δ_w and δ_{w+1}, respectively. Now, for each duty time record r^{new_w} either Case 1 applies, such that we follow the procedure described in Case 1 for this duty time record, or we follow the procedure described in Case 2. There is one exception: when $\tilde{ta}_{u_w-1}^{new} > 4.5$ *and* $\tilde{ta}_{u_w}^{new} > 4.5$. In that situation, we apply the procedure described in Case 2 to the departure intervals $[\delta_{w-1}, \delta_w]$ and $[\delta_w, \delta_{w+1}]$. However, we also have to consider departing exactly at δ_w without scheduling a break at node i. We resolve this by creating an additional duty time record with only one feasible departure time (δ_w) for which Case 1 applies.

For example, suppose a node k is added to the route $0 \to i \to j$ presented in Fig. 4b. Furthermore, suppose that all service times are 0 such that the duty times in Fig. 4b equal the accumulated driving times. Finally, suppose that the travel time from node j to node k is 1.5 h, independent of the time of departure. Then, for departure times 2.5 until 4.5 from node 0, the accumulated driving times exceed 4.5 h. This results in 2 duty time records with departure intervals [2.5, 4.5] and [4.5, 8], respectively. For the first interval we have to apply the procedure described in Case 2, for the second interval we have to apply the procedure described in Case 1. Figure 5a and b present the resulting duty times and accumulated driving times, respectively.

Note that, for example, departing at time 4 from node 0 leads to a later arrival time at node k than departing at time 4.5. Time windows might allow departure at

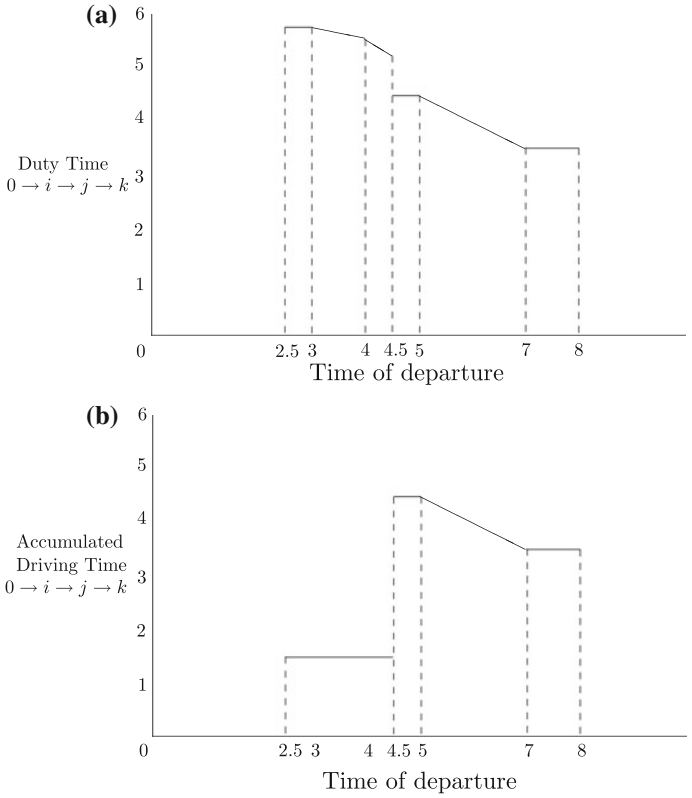

Fig. 5 Duty time records with time windows and breaks: **a** duty times route $0 \rightarrow i \rightarrow j \rightarrow k$, **b** accumulated driving times route $0 \rightarrow i \rightarrow j \rightarrow k$

time 4.5, but not at time 4. Therefore, there might be gaps between succeeding feasible departure intervals.

To account for the total driving time available for each day, we add an element to each duty time record entry accounting for the total accumulated driving time over the entire route. If this element exceeds the total available driving time of 9 h for a certain departure time, then we determine a similar series of departure times as described in Case 3 above. However, the intervals corresponding to total accumulated driving times exceeding 9 h are left out of consideration, thereby possibly introducing gaps between departure intervals. We follow a similar strategy for the total duty times, such that non-feasible departure times are left out of consideration.

4.5 Running time complexity

The procedure for adding the breaks increases the running time complexity of the VDO algorithm. To derive this complexity, it is crucial to know how many breaks could maximally be scheduled in a route for a certain departure time from the first

node in that route. In Appendix A, we derive that this number equals 4, given the total daily driving time of 9 h. We now derive how many additional duty time record entries each break might introduce.

Suppose that after adding a route $i \rightarrow j$ to a partial solution we would have $\tilde{ta}_u^{new} < 4.5$ for some entry $\tilde{r}_u^{new}$ and $\tilde{ta}_{u+1}^{new} > 4.5$ for the next entry $\tilde{r}_{u+1}^{new}$. Then, the break requirement introduces two duty time record entries $(r_{U^{new_w}}^{new_w}, r_0^{new_{w+1}})$ for two successive duty time records r^{new_w} and $r^{new_{w+1}}$, both with the same departure time, but with different duty times and accumulated driving times. The first entry $r_{U^{new_w}}^{new_w}$ represents the case where no break is scheduled at node i, while the second entry $r_0^{new_{w+1}}$ represents the case where a break is scheduled at node i. Suppose next that $\tilde{ta}_{u+2}^{new} < 4.5$. Then, again the break requirement introduces two duty time record entries: $r_{U^{new_{w+1}}}^{new_{w+1}}$ and $r_0^{new_{w+2}}$. When another node is added to the route, a similar procedure may apply to the successive record entries $\left(r_{U^{new_w}-1}^{new_w}, r_{U^{new_w}}^{new_w}\right)$ and the successive record entries $\left(r_0^{new_{w+2}}, r_1^{new_{w+2}}\right)$. In the worst case, each node addition results in four new duty time record entries caused by the break requirement for the original duty time record entries $\tilde{r}_u^{new}$ and $\tilde{r}_{u+1}^{new}$, because of ascending (descending) $\tilde{ta}_u^{new}$ that cross the 4.5 h driving limit. Since there are at most $n + 1$ node additions per vehicle route, this leads to at most $2(n + 1)$ additional entries for the original entry $\tilde{r}_u^{new}$ (and $2(n + 1)$ additional entries for the original entry $\tilde{r}_{u+1}^{new}$).

Since the number of existing entries without considering breaks is $O(np)$, the total number of entries with at most one break scheduled is $O(n^2p)$. The same procedure applies for each additional break, i.e., introducing at most $2n$ entries for each existing entry. Therefore, given that at most 4 breaks will be scheduled for each departure time, the running time complexity of the algorithm with scheduling breaks is $O(n^5p)$.

5 Computational experiments

In this section, we test the solution approach described in Sect. 4. We ran our experiments on a PC with a Core 2 Quad, 2.83 GHz CPU and 4 GB of RAM. Section 5.1 describes our test instances, Sect. 5.2 describes our test approach, and Sect. 5.3 presents the results.

5.1 Test instances

To test our heuristic, we use a modification of the set of benchmark instances for the VRPTW with time-dependent travel times proposed by Figliozzi (2009). These Figliozzi benchmark instances are themselves modifications of the well-known Solomon (1987) benchmark instances for the VRPTW. We selected these instances, because the Solomon benchmarks are standard reference in the VRP literature and they represent an extensive set of VRPTW instances with various characteristics. Moreover, Figliozzi's modification of the Solomon instances for the VRPTW with time-dependent travel times is—to the best of our knowledge—the only set of benchmark instances available in the literature for this type of problem. Below we

explain both (Figliozzi's and our) modifications with respect to the Solomon instances.

Figliozzi proposed the following modification of the Solomon instances to make them applicable to the VRPTW with time-dependent travel times. The opening hours of the depot ($[e_0, l_0]$) are divided in 5 equally spread time intervals. The first and the last time interval correspond to the morning and evening peak with a reference speed of 1.00. In the remaining intervals, the speeds are higher. Figliozzi proposed the following three speed patterns, representing traffic congestion during the peak hours to an increasing extent:

$$TD1 = [1.00, 1.60, 1.05, 1.60, 1.00]$$
$$TD2 = [1.00, 2.00, 1.50, 2.00, 1.00]$$
$$TD3 = [1.00, 2.50, 1.75, 2.50, 1.00]$$

We add one speed pattern ($TD0$) in which speeds are constant (1.00) over the day.

Since these benchmarks do not include driving hours regulations, we modify them for the TDVRP-EC as follows. We assume that the opening hours of the depot correspond to a working day of 12 h: from 7 AM until 7 PM. With Figliozzi's speed patterns, this implies that the morning and evening peak last from 7 AM until 9:24 AM and from 4:36 PM until 7 PM, respectively, which is similar to the observations of the Dutch Motorists' Organization ANWB of the traffic peak periods in the Netherlands (ANWB Reisinformatie 2010). To obtain these depot opening hours, we scale the time windows and travel distances in each problem instance. In summary, the resulting problem instances for the TDVRP-EC consist of the scaled modified Solomon instances with the speed patterns proposed by Figliozzi, and the EC social legislation on driving and working hours. We refer to this test set as Set 1.

The speed patterns in Set 1 do not allow driving before the morning peak or after the evening peak. Moreover, since the depot is open for 12 h, the EC regulation on daily duty times—which restricts daily duty times to 13 h—is always satisfied. In order to quantify the benefits of allowing travels before the morning peak and after the evening peak, we propose a second test set in which driving before and after the morning peak is possible, and for which the EC regulation on daily duty times can be restrictive. For this purpose, we introduce Set 2 in which we extend the depot opening hours to 16 by advancing the opening time by 2 h and by postponing the closing time by 2 h. The speeds during these new periods represent free-flow speeds before the morning peak and after the evening peak, respectively. Therefore, we set the speed during these periods to the maximum speed for each speed pattern, i.e., we get the following speed patterns:

$$TD0' = [1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00]$$
$$TD1' = [1.60, 1.00, 1.60, 1.05, 1.60, 1.00, 1.60]$$
$$TD2' = [2.00, 1.00, 2.00, 1.50, 2.00, 1.00, 2.00]$$
$$TD3' = [2.50, 1.00, 2.50, 1.75, 2.50, 1.00, 2.50]$$

Note that the first and the last speed last for 2 h, while the other speeds last for 2.4 h.

In addition to these extra depot opening hours, we adjust a selection of the customer service time windows in Set 2. If the opening (closing) time of a time window is non-restrictive in the original Solomon instance, then we make it also non-restrictive in the new problem instance. This implies that if the opening time in the original Solomon instance equals the opening time of the depot, then we set this opening time accordingly in Set 2. The closing times in the original Solomon instances are integer and they are constructed such that they always allow a direct return to the depot after starting service at this closing time. Therefore, we consider closing times non-restrictive if starting service at this closing time and directly returning to the depot results in an arrival time (after rounding up) equal to the closing time of the depot. In our new test set, we set such closing times equal to the closing time of the depot. We refer to this test set as Set 2. Note that Set 2 is less restrictive than Set 1, since some time windows are increased and the average travel speed is increased (every feasible solution in Set 1 is also a feasible solution in Set 2). However, the EC regulation on daily duty times can be restrictive in Set 2 as opposed to Set 1.

5.2 Test approach

Our test approach is as follows. We solve all problem instances twice. Both times, we use a lexicographic objective function in which we set the primary objective to minimize the number of vehicles used. The first time, we set the secondary objective to minimize the total travel distance, the second time to minimize the total duty time. In the remainder, we refer to the DP heuristic with minimizing travel distance as secondary objective as DP^{dist}, and we refer to the DP heuristic with minimizing duty time as secondary objective as DP^{duty}. We compare the results of these two heuristics in terms of all relevant cost factors (number of vehicles, travel distance, duty time).

For both DP heuristics we set $H = 10{,}000$, which means that in each stage in the DP heuristic only the 10,000 best states are selected to be expanded in the next stage. For this selection procedure, we use the following hierarchical criteria: (1) number of vehicles used (2) earliest completion time of vehicle route being constructed (3) secondary objective. We added the secondary cost criterion 'earliest completion time', because preliminary tests showed that this criterion has a positive impact on minimizing the number of vehicles used. Within the DP heuristic, the primary criterion 'number of vehicles used' starts to play a role when a node representing the depot is about to be added to a state. However, when a customer with a late window opening time is selected, then there is little room for adding customers to the end of this partial vehicle route, such that extra vehicles are needed in the complete solution. Setting the secondary selection criterion to 'earliest completion time of the partial vehicle route being constructed' increases the room for adding customers such that less vehicles are needed in the complete solution.

5.3 Test results

Table 1 presents the results for the two heuristics on Set 1 in terms of number of vehicles used, total travel distance, total duty time, and the required cpu time (in

Table 1 Results set 1

Speed pattern	DP^{dist}				DP^{duty}			
	# Veh	Dist	Duty	Cpu(s)	# Veh	Dist	Duty	Cpu(s)
TD0	9.18	1,294	4,992	148	9.34	1,314	4,860	148
TD1	8.23	1,261	4,730	397	8.82	1,318	4,540	397
TD2	7.75	1,265	4,501	407	8.18	1,326	4,352	408
TD3	7.48	1,258	4,413	415	8.18	1,330	4,228	415
Average	8.16	1,269	4,659	342	8.63	1,322	4,495	342

seconds). DP^{dist} leads to better results than DP^{duty} in terms of travel distance (-4.1%, on average), and in terms of number of vehicles used (-5.7%, on average). The latter result can be explained as follows. If the secondary objective is set to minimize the total duty time, then routes that start late and complete early are preferable. Therefore, customers with either an early or late time window are not preferable with this objective. The first two criteria (number of vehicles used and earliest completion time of the route being constructed) for selecting the H best states in each stage try to avoid missing such customers, but only for the route that is being constructed. These criteria do not have any effect on the routes that have already been completed in the partial solution. Therefore, for those completed routes only the tertiary criterion plays a role. Since for DP^{duty} this criterion is 'total duty time', it is likely that only a few customers with either an early or a late time window are in the completed routes in a partial solution. Therefore, such customers have to be selected at a later stage in which they may not combine well and extra vehicles are needed.

The duty times are substantially smaller with DP^{duty} than with DP^{dist} (-3.5%, on average). This is of particular interest, since the total duty time defines the total amount of vehicle hours that is needed to serve all customers. Since transport costs are directly related to this amount of vehicle hours, any reduction in duty time leads to cost savings. Note that the computation times are much smaller for the TD0 speed pattern, since speeds are constant with this speed pattern, such that the number of duty time record entries is substantially smaller with this speed pattern (specifically, this number is either 1 in case there is forced waiting time along the route, or 2: the earliest and latest feasible departure time without introducing unforced waiting time).

Table 2 presents the results for Set 2. Allowing travels before the morning peak and after the evening peak substantially reduces the number of vehicles needed (-4.4% and -3.1% for DP^{dist} and DP^{duty}, respectively). The total travel distance (2.5 and 3.5%, respectively) and total duty time (2.0 and 0.8%, respectively), however, increase.

Computation times are a bit larger for Set 2 than for Set 1. This difference can be explained by the average number of duty time record entries, which is larger for Set 2 than for Set 1. The longer planning horizon in Set 2 allows for more possible

Table 2 Results Set 2

Speed pattern	DP^{dist}				DP^{duty}			
	# Veh	Dist	Duty	Cpu(s)	# Veh	Dist	Duty	Cpu(s)
TD0	8.68	1,297	5,096	161	9.00	1,340	4,902	160
TD1	7.96	1,304	4,847	645	8.55	1,369	4,575	582
TD2	7.45	1,298	4,556	584	8.11	1,370	4,389	592
TD3	7.13	1,304	4,515	612	7.79	1,394	4,261	618
Average	7.80	1,301	4,753	500	8.36	1,368	4,532	488

Table 3 Optimality gaps VDO

Speed pattern	DP^{dist}	DP^{duty}
TD0	0.29%	0.11%
TD1	0.50%	0.43%
TD2	0.61%	0.34%
TD3	0.28%	0.19%
Average	0.42%	0.27%

departure times for each partial vehicle route. In addition, longer routes are allowed, such that more breaks have to be scheduled.

We also tested the quality of the VDO algorithm, which does not consider unforced waiting time and early breaks, by optimizing the departure times of the vehicle routes in the VRP solutions of Set 2 using the ILP model of Kok et al. (2010a), which *includes* unforced waiting times and early breaks. We solved the ILP model with CPLEX 11.0 for each vehicle route and compared the minimum duty times with the duty times found by our VDO algorithm. Table 3 presents the average optimality gaps in duty time.

We observe that the optimality gaps are very small (smaller than 0.5%, on average). The optimality gaps are slightly larger for DP^{dist}. This can be explained by less tight routes when travel distance is the secondary objective than routes when duty time is the secondary objective. For less tight routes it is more likely that there is room for improvement by introducing unforced waiting time. Although the optimality gaps are small on average, there are problem instances for which the average optimality gap over all routes is more than 3.7%. Therefore, optimizing departure times with the exact approach for the VDO of Kok et al. (2010a) as a post-processing step of solving a TDVRP-EC may lead to substantial cost savings.

6 Conclusions

We proposed a DP heuristic for the TDVRP-EC. To the best of our knowledge, this is the first solution approach that considers both time-dependent travel times and driving hours regulations within one vehicle routing model. Since the US

Hours-Of-Service Regulations are less restrictive than the EC social legislation, our DP heuristic can also solve the TDVRP with the US Hours-Of-Service Regulations.

We proposed a heuristic for the VDO to estimate the minimum duty time of partial vehicle routes. This heuristic is an efficient exact approach for the VDO without unforced waiting time and early breaks. Computational results show that this heuristic finds close to optimal solutions for the VDO.

The DP heuristic is flexible with respect to various extensions of the VRP. Therefore, the solution approach proposed in this paper can also be applied to those extensions of the VRP. The DP heuristic is also flexible with respect to different objective functions, as demonstrated with the computational experiments in which duty time minimization as the secondary objective, which is often considered in practice, is compared with travel distance minimization as the secondary objective, which is often considered in the VRP literature. Therefore, this solution approach is very promising for real-life vehicle routing problems.

The computational results show that duty time minimization as the secondary objective leads to substantial reductions of duty times, but at the cost of more vehicle routes and longer travel distances. Moreover, the results show that extending the depot opening hours, such that traveling before the morning peak and after the evening peak becomes possible, may result in substantial cost savings.

Acknowledgment This work was financially supported by Stichting Transumo through the project ketensynchronisatie. We thank the anonymous referees for their helpful comments to improve this paper.

Appendix A

In this appendix, we show that the maximum number of breaks required for a certain departure time for one vehicle route and one-day planning equals 4. We first construct an example where exactly 4 breaks are required and next, we show that there cannot exist departure times which require more than 4 breaks.

Suppose that the first break, say at customer i, must be scheduled after a very small amount of accumulated driving time, say $\epsilon > 0$. This happens if the driving time to the next customer j equals 4.5 (see Fig. 6a). Next, assume that the driving time from i to j reduces to $3.75 + \epsilon$ if a break of 0.75 is taken at customer i (see Fig. 6b). This is possible under the non-passing property. Then, after $3.75 + 2\epsilon$ of total driving time, and $3.75 + \epsilon$ of accumulated driving time since the last break, we are at customer j. If the driving time to the next customer k equals 0.75, then we also have to schedule a break at customer j. Under the non-passing property, it is possible that after the break of 0.75, the driving time to customer k has reduced to ϵ (see Fig. 6c). Therefore, when arriving at customer k, $3.75 + 3\epsilon$ of total driving time has passed. Furthermore, the accumulated driving time is ϵ, which is the same as at

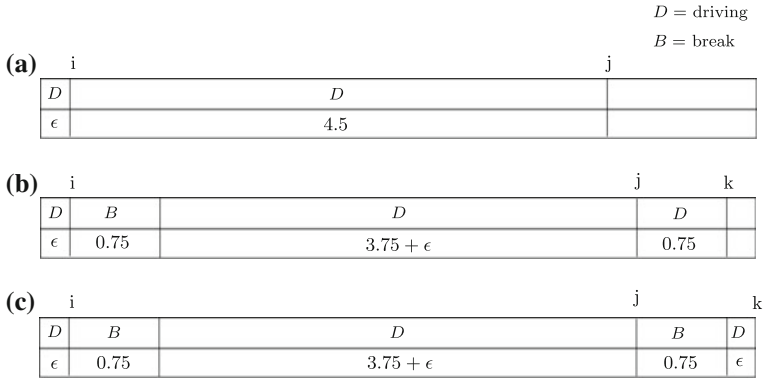

Fig. 6 **a** Route $0 \rightarrow i \rightarrow j$ with no break at j, **b** route $0 \rightarrow i \rightarrow j \rightarrow k$ with break at i and no break at j, **c** route $0 \rightarrow i \rightarrow j \rightarrow k$ with breaks at i and j

customer i. Next, we repeat the procedure to schedule two other breaks. By making ϵ arbitrarily small, the fourth break is required after 7.5 of total driving time.

A fifth break is never required because of the following. Observe that when the second break is scheduled, at least 3.75 of total accumulated driving time must have passed. This is, because the accumulated driving time before scheduling the first break at customer i, added to the driving time of the next travel, say to customer j, must exceed 4.5 (otherwise no break would be required). The non-passing property allows this total driving time to reduce by at most 0.75 during the first break. Therefore, before the second break is scheduled, at least 3.75 of total driving time must have passed. Next, after the second break is scheduled, the accumulated driving time is 0 again. With the same reasoning, we can derive that before the fourth break is scheduled, at least 7.5 of total driving time must have passed. Since the total driving time per day may not exceed 9 h, the remaining driving time after the fourth break is 1.5, while the accumulated driving time directly after the fourth break is 0. Therefore, a fifth break is never required.

References

ANWB Reisinformatie (2010) Dagelijks drukke trajecten (In Dutch). World Wide Web, last checked on 14 Jan 2010. http://www.anwb.nl/verkeer/nederland/verkeersinformatie/verkeersverwachting/Dagelijkse-drukke-trajecten.html

Archetti C, Savelsbergh MWP (2009) The trip scheduling problem. Trans Sci 43(1):417–431

Donati AV, Montemanni R, Casagrande N, Rizzoli AE, Gambardella LM (2008) Time dependent vehicle routing problem with a multi ant colony system. Eu J Operat Res 185(3):1174–1191

Ehmke JF, Meisel S, Engelmann S, Mattfeld DC (2009) Data chain management for planning in city logistics. Int J Data Mining Mode Manag 1(4):335–356

Ehmke JF, Meisel S, Mattfeld DC (2010) Floating car data based analysis of urban travel times for the provision of traffic quality. In: Hillier FS, Barceló J, Kuwahara M (eds) Traffic data collection and its standardization. vol 144, Springer, New York, pp 129–149

European Union (2006) Regulation (EC) no 561/2006 of the European parliament and of the council of 15 March 2006 on the harmonisation of certain social legislation relating to road transport and

amending council regulations (EEC) no 3821/85 and (EC) no 2135/98 and repealing council regulation (EEC) no 3820/85. Official J Eu Union L 102:1
Federal Motor Carrier Safety Administration (2008) Hours-of-service regulations. http://www.fmcsa.dot.gov/rules-regulations/topics/hos/index.htm
Figliozzi MA (2009) A route improvement algorithm for the vehicle routing problem with time dependent travel times. In: Proceedings of the 88th transportation research board annual meeting, Washington, DC
Fleischmann B, Gietz M, Gnutzmann S (2004) Time-varying travel times in vehicle routing. Trans Sci 38(2):160–173
Funke B, Grünert T, Irnich S (2005) Local search for vehicle routing and scheduling problems: Review and conceptual integration. J Heurist 11(4):267–306
Goel A (2009) Vehicle scheduling and routing with drivers' working hours. Tran Sci 43(1):17–26
Goel A, Kok AL (2009a) Efficient scheduling of team truck drivers in the European Union. Working paper. University of Leipzig, Leipzig
Goel A, Kok AL (2009b) Efficient truck driver scheduling in the United States. Working paper. University of Leipzig, Leipzig
Gromicho J, van Hoorn J, Kok AL, Schutten JMJ (2008) Restricted dynamic programming: a flexible framework for solving realistic VRPs. Beta working paper series 266. http://beta.ieis.tue.nl/node/1154
Hashimoto H, Yagiura M, Ibaraki T (2008) An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. Disc Opt 5:434–456
Ichoua S, Gendreau M, Potvin JY (2003) Vehicle dispatching with time-dependent travel times. Eu J Oper Res 144(2):379–396
Kok AL, Hans EW, Schutten JMJ (2009) Vehicle routing under time-dependent travel times: the impact of congestion avoidance. Beta working paper series 267. http://beta.ieis.tue.nl/node/1441
Kok AL, Hans EW, Schutten JMJ (2010a) Optimizing departure times in vehicle routes. Eu J Oper Res (In Press, corrected proof). doi:10.1016/j.ejor.2010.10.017. URL http://www.sciencedirect.com/science/article/B6VCT-51BYS9J-1/2/4f06f25b5e0acd2832f286882b4a5318
Kok AL, Meyer CM, Kopfer H, Schutten JMJ (2010) A dynamic programming heuristic for the vehicle routing problem with time windows and european community social legislation. Trans Sci 44(4):442–454
Kolesar P, Walker W, Hausner J (1975) Determining the relation between fire engine travel times and travel distances in new york city. Oper Res 23(4):614–627
Malandraki C, Daskin MS (1992) Time dependent vehicle routing problems: formulations, properties and heuristic algorithms. Trans Sci 26(3):185–200
Savelsbergh MWP (1992) The vehicle routing problem with time windows: minimizing route duration. ORSA J Comput 4(2):146–154
Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res 35(2):254–265
Toth P, Vigo D (2002) The vehicle routing problem. SIAM monographs on discrete mathematics and applications, Philadelphia
Van Woensel T, Kerbache L, Peremans H, Vandaele N (2008) Vehicle routing with dynamic travel times: a queueing approach. Eu J Oper Res 186(3):990–1007
Xu H, Chen ZL, Rajagopal S, Arunapuram S (2003) Solving a practical pickup and delivery problem. Trans Sci 37(3):347–364

Author Biographies

Dr. ir. A. L. Kok (Leendert) is an Operations Research Engineer at the Algorithmic Research and Development department of ORTEC, the Netherlands. In 2010, he received his Ph.D. from the University of Twente on the subject of time-dependent vehicle routing and break scheduling. His research focuses on designing and implementing algorithmic solutions to optimization problems in the field of logistics.

Erwin Hans is an Associate Professor Operations Management and Process Optimization in Healthcare within the Operational Methods in Production and Logistics department at the University of Twente, the

Netherlands. His research focuses on the area of healthcare process optimization using modeling and optimization techniques from Operations Research and Management Science.

Marco Schutten is an Assistant Professor at the School of Management and Governance of the University of Twente, The Netherlands. He received his Ph.D. in 1996 from this university on the subject of job shop scheduling. His current research interests include transportation management, production and project planning and scheduling, and material handling systems.

Prof. dr. W. H. M. (Henk) Zijm is a full professor in Manufacturing and Supply Chain Management, and chair of the Operational Methods in Production and Logistics department at the University of Twente, the Netherlands. His research interests include manufacturing and maintenance planning and control, supply chain management and service logistics.