Complexity of Routing Problems with Release Dates and Deadlines

Alan Erera, Damian Reyes, and Martin Savelsbergh

H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Abstract

The desire of companies to offer same-day delivery leads to interesting new routing problems. We study the complexity of a setting in which a delivery to a customer is guaranteed to take place within a pre-specified time after the customer places the order. Thus, an order has a release date (when the order is placed) and a deadline (when the order needs to be delivered). A vehicle delivering an order cannot depart the depot before the order is released and has to arrive at the customer at or before the order's deadline. We show that the variant with a single delivery vehicle and customer locations on a half-line can be solved in polynomial time. This setting as well as our results generalize those found in Archetti et al. (2015).

1 Introduction

Even though dynamic vehicle routing problems have received a significant amount of attention, this attention has focused primarily on situations in which vehicle routes can be and have to be updated to accommodate newly arriving requests, e.g., new pick up requests, new pick up and delivery requests, or new service requests. For more information, see the excellent surveys by Berbeglia et al. (2010), Pillac et al. (2013), and Psaraftis et al. (2016).

More recently, dynamic vehicle routing problems in which routes cannot be updated once a vehicle departs from the depot have emerged as a research topic of interest, e.g., Azi et al. (2012). Archetti et al. (2015) study the complexity of such a problem setting arising when distribution centers, established in the vicinity of final customers to support same-day delivery, receive goods to be delivered to final customers throughout the day, which restricts when delivery routes from the distribution can commence. What makes these problems interesting and challenging is the trade-off between waiting for more goods to arrive, in order to build more effective delivery routes serving many customers, and dispatching a delivery vehicle on a less effective delivery route serving few customers, in order to avoid having to serve many customers towards the end of the day with limited time to do so (or worse, being unable to serve all customers by the end of the day due to the lack of resources). In general, not surprisingly, such problems are NP-hard, but some single-vehicle variants with customer locations on a line can be solved in polynomial time. More specifically, Archetti et al. (2015) show that the variant in which one seeks to minimize the total distance traveled while satisfying customer demand before a common deadline (the end of the day) and the variant in which one seeks to minimize the return time of the vehicle after satisfying

the demand of all customers can be solved by dynamic programming algorithms that run in $O(n^3)$ time, where n is the number of customers.

Archetti et al. (2015) refer to such problems as vehicle routing problems with release dates to reflect the fact that a customer order is only "released" for delivery after the associated goods have arrived at the distribution center. In this technical note, we extend their setting by considering a service guarantee, which guarantees that a delivery to a customer occurs within a pre-specified time after the goods have arrived at the distribution center. Thus, we not only consider release dates, but also "deadlines": a vehicle delivering a customer order cannot depart from the distribution center before the order is released and has to arrive at the customer at or before the order's deadline.

A service guarantee is often seen in dynamic delivery contexts as retailers seek to offer instant gratification to consumers. For example, a retailer may guarantee online shoppers that an order will be delivered within two hours after the order is placed. An environment where such a service guarantee arises naturally is meal-delivery: "if your pizza is not delivered within 30 minutes after you place your order, it will be free."

We develop dynamic programming algorithms that run in $O(n^2)$ time for the two variants considered by Archetti et al. (2015), i.e., with a common deadline, and dynamic programs that run in $O(n^2)$ and $O(n^3)$ time, respectively, for variants with customer-dependent deadlines and minimization of the return time and minimization of the distance traveled. Thus, we can handle the additional complexity without increasing the run time (and improve the run time for the common deadline case).

The remainder of the technical note is organized as follows. In Section 2, we formally introduce the problems studied and the notation used throughout. In Section 3, we prove a number of structural properties of optimal delivery schedules. In Section 4, we describe the dynamic programming algorithms and their analysis. Finally, in Section 5, we present some final remarks.

2 A vehicle routing problem with release dates and deadlines

Let $N = \{1, \ldots, n\}$ be a set of customers located on a half-line. Let each customer $i \in N$ place an order at time e_i, which implies the earliest possible dispatch time, and let the travel time from the depot to customer $i \in N$ be τ_i. Furthermore, let the planning horizon be T and the service guarantee be S, i.e., delivery to customer i is guaranteed to take place at or before $e_i + S$. In the remainder, we will use "customer" and "order" interchangeably. Furthermore, we assume that deliveries are instantaneous, and that all deliveries are made on the outbound journey from the depot. As a consequence, the latest possible dispatch time l_i or order i to serve customer i is $l_i = e_i + S - \tau_i$. We restrict our attention to instances in which $\tau_i \leqslant S$ and $e_i \leqslant T - 2\tau_i$ for $i = 1, \ldots, n$, because otherwise the instance is guaranteed to be infeasible. Without loss of generality, we also assume $e_i < e_{i+1}$ for $i = 1, \ldots, n-1$ (because customer locations are on a half-line, there is no need to consider customers that place an order at the same time; it suffices to consider only the customer furthest away from the depot). The feasibility problem we study in this technical note is:

Problem 1. *Is there a sequence of delivery routes that can be executed by a single driver, each starting and ending at the depot, such that each order $i \in N$ is dispatched at or after e_i and delivered at or before $e_i + S$ and the last route is completed at or before T?*

If customers $i_1, i_2, \ldots, i_k$ are served together on a delivery route r, then the earliest dispatch time of the route, $e(r)$, is $\max\{e_{i_1}, e_{i_2}, \ldots, e_{i_k}\}$, the latest dispatch time of the route, $l(r)$, is $\min\{l_{i_1}, l_{i_2}, \ldots, l_{i_k}\}$, and the furthest point visited on the route, $\tau(r)$, is $\max\{\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_k}\}$. Observe that $e(r) \leq l(r)$ is necessary for feasibility, and that if the driver is at the depot at time $e(r)$, there is no reason to delay the dispatch of the route.

Before proceeding with the analysis of the problem, we introduce the notion of "non-overlapping" routes.

Definition 1. Routes r_1 and r_2 are non-overlapping if for any pair of individual orders $i \in r_1$ and $j \in r_2$, we have $e_i \leq e_j$. Otherwise, r_1 and r_2 are overlapping routes.

Definition 2. A delivery schedule $\mathcal{S}$ of non-overlapping routes is a partition of N that can be characterized by the set of last customers in each route, i.e., $\mathcal{S} = \{i_1, i_2, \ldots, i_k, n\}$ with $1 \leq i_1 \leq i_2 \leq \cdots \leq i_k \leq n$, indicates that orders $\{1, \ldots, i_1\}$ are delivered on the first route, orders $\{i_1 + 1, \ldots, i_2\}$ are delivered on the second route, etc.

3 Structural properties of an optimal delivery schedule

The following properties hold for any feasible delivery schedule (also those that require more than one driver):

- if $i < j$ and $e_i + 2\tau_i > l_j$, then orders i and j must be dispatched together, and
- if $i < j$ and $l_i < e_j$, then orders i and j cannot be dispatched together.

Lemma 1. *Let $i < j < k$ be orders with ready times $e_i < e_j < e_k$. If the orders i and k must be dispatched together, i.e., $e_i + 2\tau_i > l_k$, then we can assume, without loss of generality, that i, j and k are dispatched together.*

Proof. Consider the following cases:

- If $\tau_j > \tau_i$, then $e_j + 2\tau_j > e_i + 2\tau_i > l_k$. Thus j must be dispatched together with k.
- If $\tau_j \leqslant \tau_i$, then $l_j > l_i$ and j can be dispatched together with i and thus with k and it will not impact the return time of the driver. Furthermore, there can be no advantage to dispatching j before i as it can only postpone the dispatch of i and k. □

Proposition 2. *Any feasible delivery schedule can be transformed into a feasible delivery schedule with non-overlapping routes.*

Proof. Suppose a feasible schedule contains overlapping routes $a = (i \cup k)$ and $b = (j \cup h)$ with $e_i < e_j < e_k < e_h$. Let a and b be the first pair of overlapping routes. Consider the following cases:

- If $\tau_j \leqslant \tau(a)$, then routes $a' = (i \cup j \cup k)$ and $b' = (h)$ are feasible (if j could be dispatched together with h, it can be dispatched together with k), the return time of the driver after executing a' is the same as after executing a, and the return of the driver after executing b' cannot be more than the return of the driver after executing b.

- If $\tau_j > \tau(a)$, then routes $a' = (i)$ and $b' = (j \cup k \cup h)$ are feasible (if j could be dispatched together with h, then k can be dispatched together with h), the return time of the driver after executing b' is the same as after executing b, and the return of the driver after executing a' cannot be more than the return of the driver after executing a.

Repeatedly applying these transformations gives the desired result. □

Proposition 3. *Any feasible delivery schedule can be transformed into a feasible delivery schedule with non-overlapping routes and no additional travel distance.*

Proof. The proof proceeds along the same lines as the proof of Proposition 2. Suppose that an optimal schedule contains overlapping trips $a = (i \cup k)$ and $b = (j \cup h)$ with $e_i < e_j < e_k < e_h$. Let a and b be the first pair of overlapping trips. Two cases may occur:

- If $\tau_j \leqslant \tau(a)$, then trips $a' = (i \cup j \cup k)$ and $b' = (h)$ are feasible (if j could dispatched together with h, it can be dispatched together with k). Moreover, $\tau(a') = \tau(a)$ and $\tau(b') \leqslant \tau(b)$, so the total travel time has not increased after the exchange.

- If $\tau_j > \tau(a)$, then trips $a' = (i)$ and $b' = (j \cup k \cup h)$ are feasible (if j could dispatched together with h, then k can be dispatched together with h). And $\tau(a') \leqslant \tau(a)$ while $\tau(b') = \tau(b)$, so the total travel time has not increased after the exchange.

Once again, repeatedly applying these transformations gives the desired result. □

Next, we observe that Problem 1 can be decided if the following optimization problem can be solved:

Problem 2. *Determine the minimum possible completion time, c^*, of a sequence of delivery routes that can be executed by a single driver, each starting and ending at the depot, such that each order $i \in N$ is dispatched at or after e_i and delivered at or before $e_i + S$.*

A dynamic programming algorithm that determines this minimum completion time will be given in the next section, but first we state and prove a proposition on which the algorithm relies:

Proposition 4. *If there exists a feasible delivery schedule, then there exists an optimal non-overlapping delivery schedule $\mathcal{S} = \{s_1, s_2, \ldots, s_k, n\}$ with the property that each partial schedule $\mathcal{S}_{[1,j]} = \{s_1, s_2 \ldots, s_j\}$, delivering orders $\{1, \ldots, s_j\}$, for $j = 1, \ldots, k$, completes the delivery of these orders as early as possible, i.e., it is an optimal delivery schedule for the (sub)problem defined by orders $\{1, 2, \ldots, s_j\}$.*

Proof. Let $\mathcal{Q} = \{q_1, q_2, \ldots, q_k, n\}$ be an optimal schedule, i.e., a feasible schedule with minimum completion time $c^*(\mathcal{Q})$. Let $j \leq k$ be the largest index of a route in $\mathcal{Q}$ for which the partial schedule $\mathcal{Q}_{[1,j]}$ does not have minimum completion time $c^*(\mathcal{Q}_{[1,j]})$ for the (sub)problem defined by orders $\{1, \ldots, q_j\}$. Since $\mathcal{Q}_{[1,j]}$ is a feasible schedule for this (sub)problem, there must exist a schedule $\mathcal{Q}'_{[1,j]}$ with minimum completion time $c^*(\mathcal{Q}_{[1,j]})$. We can replace $\mathcal{Q}$ by the schedule that concatenates $\mathcal{Q}'_{[1,j]}$ and $\mathcal{Q}_{[j+1,n]}$ (the routes in $\mathcal{Q}$ delivering orders $\{q_j + 1, \ldots, n\}$). Repeatedly applying this transformation gives the desired delivery schedule. □

4 Dynamic Programming Algorithms

4.1 Minimizing completion time

4.1.1 No service guarantee

Problem 3. *Determine the minimum possible completion time, c^*, of a sequence of delivery routes that can be executed by a single driver, each starting and ending at the depot, such that each order $i \in N$ is dispatched at or after e_i.*

When there is no service guarantee, we may assume, without loss of generality, that for $i < j$, we have $\tau_i \geqslant \tau_j$ (see also Archetti et al. (2015)). The problem can be solved by the following polynomial-time DP:

- states: i for $i \in \{0, 1, 2, \ldots, n\}$.
- costs: $c(i)$ for $i \in \{1, 2, \ldots, n\}$ indicating the minimum completion time of a non-overlapping schedule delivering orders $\{1, \ldots, i\}$.
- recursion:

$$c(0) = 0 \tag{1a}$$

$$c(i) = \min_{j \in \{0,\ldots,i-1\}} \left\{ \max\{c(j), e_i\} + 2\tau_{j+1} \right\} \tag{1b}$$

Order i is either included in the last route, in which case the earliest possible return time is $\max\{c(j), e_i\} + 2\max_{k \in \{j+1,\ldots,i\}} \tau_k = \max\{c(j), e_i\} + 2\tau_{j+1}$ when the last route includes orders $j+1, \ldots, i-1$, or in a new route by itself, in which case the earliest possible return time is $\max\{c(i-1), e_i\} + 2\tau_i$. Because $c(n)$ is the minimum time required for a single driver to deliver orders $\{1, 2, \ldots, n\}$ and return to the depot after completing all deliveries, there is a feasible delivery schedule if and only if $c(n) \leqslant T$.

Solving the forward recursion requires the evaluation of i possible actions at the i^{th} stage, and evaluating each possible action takes a constant number of operations. Therefore, $c(n)$ is found in $\mathcal{O}(n^2)$ time.

4.1.2 With Service Guarantee

Building on the insights from the "no service guarantee" setting, a polynomial-time DP is given by:

- states: i for $i \in \{0, 1, 2, \ldots, n\}$.
- costs: $c(i)$ for $i \in \{1, 2, \ldots, n\}$ indicating the minimum completion time of a non-overlapping schedule delivering orders $\{1, \ldots, i\}$ or ∞ if it is not possible to serve $\{1, 2, \ldots, i\}$ feasibly with a single driver.
- recursion:

$$c(0) = 0 \tag{2a}$$

$$c(i) = \min\left\{ \min_{j\in\{0,\dots,i-1\}} \left\{ \max\{c(j), e_i\} + 2\max_{j<k\le i}\{\tau_k\} \;\middle|\; \max\{c(j), e_i\} \le \min_{j<k\le i}\{l_k\} \right\}, \infty \right\} \tag{2b}$$

Order i is either included in the last route, in which case the earliest possible return time is $\max\{c(j), e_i\} + 2\max_{k\in\{j+1,\dots,i\}} \tau_k$ when the last route includes orders $j+1,\dots,i-1$, or in a new route by itself, in which case the earliest possible return time is $\max\{c(i-1), e_i\} + 2\tau_i$. Because $c(n)$ is the minimum time required for a single driver to deliver orders $\{1,2,\dots,n\}$ and return to the depot after completing all deliveries, the problem is feasible if and only if $c(i) < \infty$ for $i = 1,\dots,n-1$ and $c(n) < T$.

We can compute all values $\max_{j<k\le i}\{\tau_k\}$ and $\min_{j<k\le i}\{l_k\}$ in advance in $O(n^2)$ time. After this, solving the forward recursion requires the evaluation of i possible actions at the i^{th} stage, and evaluating each possible action requires a constant number of operations. Therefore, $c(n)$ is found in $\mathcal{O}(n^2)$ time.

4.2 Minimizing total distance traveled

4.2.1 No service guarantee

We assume that there exists a delivery schedule for a single driver in which all orders are delivered within the guaranteed service window, which can be established with the dynamic programming algorithm described above.

Problem 4. *Determine a sequence of delivery routes that can be executed by a single driver, each starting and ending at the depot, such that each order $i \in N$ is dispatched at or after e_i and that completes at or before time T with minimum total travel distance.*

The problem can be solved by the following polynomial-time backward DP:

- states: i for $i \in \{1,2,\dots,n,n+1\}$.
- values: $c(i)$ for $i \in \{1,2,\dots,n\}$ indicating the latest dispatch time of a non-overlapping schedule delivering orders $\{i+1,\dots,n\}$ that completes at or before time T.
- recursion:

$$c(n+1) = T \tag{3a}$$

$$c(i) = \max_{j\in\{i+1,\dots,n+1\}} \left\{ c(j) - 2\tau_i \mid c(j) - 2\tau_i \ge e_{j-1} \right\} \tag{3b}$$

It is feasible to serve order i together with orders $i+1,\dots,j-1$, if the dispatch time is greater than or equal to e_{j-1} and the return time to the depot is less than or equal to the latest possible dispatch time to serve the orders $j,\dots,n$, i.e., less than or equal to $c(j) - 2\tau_i$ (where we use that $i < j$ implies $\tau_i \geqslant \tau_j$). The backward recursion chooses the maximum dispatch time among the feasible options.

Observe that the non-overlapping delivery schedule associated with time $c(1)$ does not contain any waiting time (otherwise there would be a non-overlapping delivery schedule that departs later) and thus has minimum total distance $T - c(1)$.

Solving the backward recursion requires the evaluation of $n - i + 1$ possible actions at the i^{th} stage, and evaluating each possible action takes a constant number of operations. Therefore, $c(1)$ is found in $\mathcal{O}(n^2)$ time.

4.2.2 With service guarantee

Problem 5. *Determine a sequence of delivery routes that can be executed by a single driver, each starting and ending at the depot, such that each order $i \in N$ is dispatched at or after e_i and delivered at or before $e_i + S$ and that completes at or before time T with minimum total travel distance.*

In this case too, it helps to construct a schedule backwards, but, because of the presence of service guarantees and, thus, latest dispatch times for orders, a delivery schedule that is pushed forward in time until it cannot be pushed forward anymore may still contain waiting time. As a consequence, it is no longer true that dispatching as late as possible is equivalent to minimizing the total distance traveled. Therefore, the backward dynamic program below explicitly minimizes the total distance traveled for "every" possible time that a delivery schedule may begin.

We assume, without loss of generality, that $l_j \leqslant T - 2\tau_j$ for $j = 1, \ldots, n$, and we introduce the following notation: $\tau_{ij} = \max_{i \leqslant k \leqslant j} \tau_k$ and $l_{ij} = \min_{i \leqslant k \leqslant j} l_k$.

The backward dynamic program computes value function $f_j(t)$, the minimum distance required to serve orders $j, \ldots, n$ by a driver available at time t (letting $f_j(t)$ be ∞ if it is infeasible to serve these orders between t and T). Since the driver is available at the depot at time 0, the minimum distance required to serve all orders is given by $f_1(0)$.

The dynamic program uses values L_j, denoting the latest time that a driver can depart and feasibly serve orders $\{j, \ldots, n\}$ for $j = 1, \ldots, n$. We start by showing how to compute the values L_j. Let $L_{n+1} = T$, then

$$L_j = \max_{j \leq k \leq n} \left\{ \min\{l_{jk}, L_{k+1} - 2\tau_{jk}\} \;\middle|\; e_k \leqslant \min\{l_{jk}, L_{k+1} - 2\tau_{jk}\} \right\}$$

for $j = n, \ldots, 1$. Note that $\min\{l_{jk}, L_{k+1} - 2\tau_{jk}\}$ is the latest feasible departure time for the driver if he is to serve orders $j, \ldots, k$ together in a single trip before serving the remaining orders and e_k is the earliest feasible departure time if the driver is to serve orders $j, \ldots, k$ together.

The backward dynamic program for computing $f_j(t)$ is given below:

Initialization

$$f_{n+1}(t) = \begin{cases} 0 & \text{if } 0 \leqslant t \leqslant T \\ \infty & \text{otherwise} \end{cases}$$

Recursion

Given $j \in \{n, \ldots, 1\}$ compute for $k = j, \ldots, n$:

$$g_j^k(t) = \begin{cases} 2\tau_{jk} + f_{k+1}(\max\{e_k, t\} + 2\tau_{jk}) & \text{if } \max\{e_k, t\} \leqslant \min\{l_{jk}, L_{k+1} - 2\tau_{jk}\} \\ \infty & \text{otherwise} \end{cases}$$

Then

$$f_j(t) = \min_{j \leq k \leq n} \{g_j^k(t)\}$$

Note that $g_j^k(t)$ expresses the minimum distance traveled by the driver if he departs at or after time t and on his first trip delivers orders $j, \ldots, k$ (or ∞ if that is infeasible).

Proposition 5. *The dynamic program correctly computes the value function $f_j(t)$ for $j = 1, \ldots, n$ and $t \in [0, T]$.*

Proof. We will argue that the dynamic program correctly computes the value function for (any partial) minimum distance non-overlapping schedule. Proposition 3 implies that this suffices.

We proceed by induction. Clearly, the dynamic program constructs an optimal value function when there is a single order. Next, assume that the dynamic program correctly computes the value function for any set of k or fewer orders. Given a set of $k+1$ orders $\{1, 2, \ldots, k, k+1\}$, the dynamic program computes the value function $f_1(t)$ for $t \in [0, T]$ as the point-wise minimum of a number of functions of the form

$$g_1^i(t) = 2\tau_{1i} + f_{i+1}(\max\{e_i, t\} + 2\tau_{1i}).$$

For each i, with $1 \leq i \leq k+1$, the function $g_1^i(t)$ represents the minimum travel distance of any schedule starting at t and serving orders $\{1, \ldots, i\}$ on the first delivery route. The travel distance of the first delivery route is $2\tau_{1i}$ and the earliest return to the depot is at time $\max\{e_i, t\} + 2\tau_{1i}$. This partial schedule is extended with an optimal non-overlapping schedule serving orders $\{i+1, \ldots, k+1\}$, which adds $f_{i+1}(\max\{e_i, t\} + 2\tau_{1i})$ to the distance traveled. (By the induction hypothesis, the value function $f_{i+1}(\max\{e_i, t\} + 2\tau_{1i})$ yields the minimum travel distance for serving orders $\{i+1, \ldots, k+1\}$ as the set has at most k orders.) For a given t, this schedule is feasible, and therefore considered in the point-wise minimum operation, if and only if

a) the earliest departure time, $\max\{e_i, t\}$, of the first delivery route serving orders $\{1, \ldots, i\}$ is no later than l_{1i}, the latest possible departure that ensures that the service guarantee of the orders is satisfied, and

b) the earliest return to the depot, $\max\{e_i, t\} + 2\tau_{1i}$, of the first delivery route serving orders $\{1, \ldots, i\}$ is not too late for the timely delivery of orders $\{i+1, \ldots, k+1\}$ in subsequent routes, i.e., no later than L_{i+1}.

From the previous observations it follows that, for any given t, if $g_1^i(t)$ is included in the point-wise minimum comparison defining $f_1(t)$, then $g_1^i(t)$ represents the extension of a non-dominated schedule serving orders $\{i+1, \ldots, k+1\}$ into a schedule serving all orders $\{1, \ldots, k+1\}$. Since for any given t, all feasible extensions from non-dominated schedules are considered, it follows that the best among them must be optimal. Therefore, for any $t \geqslant 0$, $f_1(t)$ represents the minimum total travel time of a non-overlapping schedule serving all orders $\{1, \ldots, k+1\}$ starting no earlier than time t, as conjectured. □

Proposition 6. *The value $f_1(0)$ can be computed in $\mathcal{O}(n^3)$ time.*

Proof. First, we observe that any schedule that is feasible with a start at time t is also feasible with a start at time $s < t$. Therefore, the value function $f_j(t)$ is nondecreasing in t.

Next, we observe that $g_j^k(t)$ is a simple transformation of $f_{k+1}(t)$ involving a horizontal translation, with a "leveling-up" for values below a given threshold,

$$h_j^{k+1}(t) = \begin{cases} f_{k+1}(e_k + 2\tau_{jk}) & \text{if } 0 \leqslant t \leqslant e_k \\ f_{k+1}(t + 2\tau_{jk}) & \text{if } t > e_k \end{cases},$$

followed by a vertical translation

$$g_j^{k+1}(t) = h_j^{k+1}(t) + 2\tau_{jk}.$$

Since $f_n(t)$ is a nondecreasing step-function and the operations outlined above preserve the functional form, we have that $f_j(t)$ is a nondecreasing step-function for $j = 1, \ldots, n$.

Furthermore, if s is a breakpoint of $f_j(t)$, then the optimal schedule that starts at s and serves orders $\{j,\ldots,n\}$ – with a first route delivering orders $\{j,\ldots,k^*\}$ and with total travel distance $g_j^{k^*}(s)$ – cannot be started later. That is, if s is a breakpoint, then $s = \min\{l_{jk^*}, L_{k^*+1} - 2\tau_{jk^*}\}$, for some $k^* \geqslant j$. Thus, there can be no more than $n - j + 1$ breakpoints in $f_j(t)$.

Consequently, starting from the breakpoint description of $f_n(t)$, we can find a breakpoint description of $f_j(t)$ for $j = n-1,\ldots,1$ (in that order). At each step, we first find the breakpoint descriptions of g_j^k for $k = j,\ldots,n$, which takes $O(n^2)$ time, and then determine the breakpoint description of $f_j(t)$ by comparing the functions $g_j^k(t)$, which also takes $O(n^2)$ time. Thus, computing the breakpoint description of $f_1(t)$ takes $\mathcal{O}(n^3)$ time. □

5 Final remarks

As many companies are collecting massive amounts of data on their customers' order patterns, it becomes more and more realistic to assume that probabilistic information about future orders will be available. This suggests that the study of anticipatory routing algorithms (as done in Ghiani et al. (2012)) for dynamic delivery settings will be beneficial. An initial investigation along those lines can be found in Klapp et al. (2015), but much more can (and needs) to be done.

References

C. Archetti, D. Feillet, and M.G. Speranza. Complexity of routing problems with release dates. *European Journal of Operational Research*, 247(3):797–803, 2015.

N. Azi, M. Gendreau, and J.-Y. Potvin. A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research*, 199(1):103–112, 2012.

G. Berbeglia, J.F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010.

G. Ghiani, E. Manni, and B.W. Thomas. A comparison of anticipatory algorithms for the dynamic and stochastic traveling salesman problem. *Transportation Science*, 46(3):374–387, 2012.

M. Klapp, A.L. Erera, and A.Toriello. The one-dimensional dynamic dispatch waves problem. 2015. Optimization Online 2015-03-4826.

V. Pillac, M. Gendreau, C. Guéret, and A.L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

H.N. Psaraftis, M. Wen, and C.A. Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016.