Route Feasibility Testing and Forward Time Slack for the Synchronized Pickup and Delivery Problem

Timo Gschwind[a]

[a] *Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*

Abstract

The Synchronized Pickup and Delivery Problem (SPDP) consists of finding a set of minimum-cost routes servicing user-specified transportation requests from pickup to delivery locations subject to pairing and precedence, capacity, time-window, and minimum and maximum time-lag constraints. The temporal constraints of the SPDP impose a complex scheduling problem for the service times at the customer locations which makes the efficient feasibility checking of routes intricate. We present two different route feasibility tests for the SPDP and compare their practical runtime on a huge number of randomly generated routes. Furthermore, we generalize to the SPDP the concept of forward time slack, which has proven a versatile tool for feasibility testing of customer or request insertions into a given (feasible) route for many VRP variants.

Key words: Vehicle routing, temporal synchronization, feasibility testing, forward time slack

1. Introduction

The Synchronized Pickup and Delivery Problem (SPDP, Gschwind, 2015) is the prototypical Vehicle Routing Problem (VRP) with *temporal intra-route synchronization* constraints. It seeks to find a set of minimum-cost routes servicing n user-specified transportation requests from origin (or pickup) to destination (or delivery) points subject to pairing and precedence, capacity, and time-window constraints. Moreover, the service times at the pickup and delivery locations of the customer requests are synchronized in the following way: After completing the service at a pickup point, the corresponding delivery has to be performed within prespecified *minimum* and *maximum time lags* (or *Ride Times*, RTs). From a modeling point of view, the SPDP generalizes the Dial-A-Ride Problem (DARP, see Cordeau and Laporte, 2007, for a survey) in which no minimum RTs are present.

Because of the complex temporal constraints of the SPDP, deciding whether or not a given route is feasible is a non-trivial task. The efficient feasibility testing of routes, however, is a crucial part in many exact and heuristic algorithms for VRPs. Sophisticated feasibility tests for a DARP with an additional constraint on the maximum waiting time at the customer nodes have been proposed by Tang *et al.* (2010) and Firat and Woeginger (2011). The time complexity of these approaches is $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$, respectively. In the SPDP, the additional presence of minimum RT constraints further complicates the route feasibility problem.

Another crucial aspect for some solution approaches to VRP variants is the ability to quickly evaluate the feasibility of insertions of single nodes or requests into a given (feasible) route. The concept of *Forward Time Slack* (FTS) originally introduced by Savelsbergh (1992) for the VRP with Time Windows (VRPTW) can be a useful tool for this kind of evaluation. Generalized versions of the FTS have been used to asses the feasibility of insertions in heuristic algorithms, e.g., for the DARP (Cordeau and Laporte, 2003) and the Pickup and Delivery Problem with Transfers (PDPT) (Masson *et al.*, 2013). In a companion paper

Email address: gschwind@uni-mainz.de (Timo Gschwind)

(Gschwind, 2015), the FTS principle is used within a dynamic-programming labeling algorithm for the solution of the column-generation subproblem of the SPDP.

The contribution of this paper is twofold. First, we derive two different route feasibility checks for the SPDP by adapting the approaches for the DARP presented in (Firat and Woeginger, 2011) and (Tang *et al.*, 2010) and conduct a computational study over a large number of randomly generated routes to compare the practical runtime of the two procedures. Our results indicate that the algorithm with the greater worst-case complexity shows on average a better practical performance even on very long routes. Second, we generalize the concept of FTS to the SPDP and demonstrate why the definition of the FTS is not unique for problems with maximum RT constraints.

The remainder of the paper is structured as follows: Section 2 defines the SPDP. Section 3 derives two feasibility tests for individual routes and presents aggregated results comparing their practical runtime. The adaptation of the FTS concept to the SPDP is described in Section 4. Short conclusions are drawn in Section 5.

2. Problem Definition

The SPDP is defined on a complete digraph $G = (V, A)$ with node set V and arc set A. The node set V comprises the origin and destination depots 0 and $2n+1$, the set of pickup nodes $P = \{1, ..., n\}$, and the set of delivery nodes $D = \{n+1, ..., 2n\}$. For each node $i \in V$ a time window $[a_i, b_i]$, a service duration s_i, and a demand q_i with $q_i = -q_{i+n}$ are given. A travel time t_{ij} and a routing cost c_{ij} are associated with each arc $(i, j) \in A$. Both travel times and routing costs are assumed to satisfy the triangle inequality.

Each transportation request $i = 1, ..., n$ consists of transporting a specific good from the pickup node $i \in P$ to the delivery node $i+n \in D$. A minimum RT $\underline{L}_i$ and a maximum RT $\overline{L}_i$ are associated with each transportation request i.

A fleet K of homogeneous vehicles each with a capacity of Q is located at the origin depot 0 to serve the transportation requests. The task of the SPDP is to find a set of $|K|$ routes starting and ending at the depot nodes 0 and $2n+1$ such that each transportation request is performed exactly once. Thereby, vehicle capacities have to be respected and the service at each node has to be started within its time window. If a vehicle arrives prior to a_i at node i, it has to wait until the time window opens. Moreover, waiting, i.e., voluntarily delaying the start of service, is allowed at any node at any time. For each transportation request i, the pickup and delivery nodes have to be served on the same route and the pickup has to be serviced before. Furthermore, the service at the delivery node $i+n$ has to be started at least $\underline{L}_i$ and at most $\overline{L}_i$ units of time after the service at the pickup has been completed.

3. Route Feasibility Testing

Consider a given route $R = (v_1, ..., v_q)$ with $v_1 = 0$ and $v_q = 2n+1$. We use the notation $v_i \in R$ to indicate that a node v_i is part of route R. Furthermore, given a pickup (delivery) node $v_i \in P$ ($v_i \in D$), we indicate by v_{i^-} and i^- (v_{i^+} and i^+) the corresponding delivery (pickup) node and its associated notation.

Testing the feasibility of a route R means checking its consistency with all constraints of the SPDP that relate to individual routes. The verification of pairing and precedence and the capacity constraint is independent of each other and of the temporal constraints and can be done straightforwardly in linear time. We assume in the following that these constraints are respected. Checking whether or not the temporal constraints are satisfied is intricate. Indeed, one has to verify if there exists a time schedule $\mathcal{T}_R = (\tau_1, ..., \tau_q)$ satisfying

$$a_{v_i} \leq \tau_i \leq b_{v_i} \quad \forall v_i \in R, \tag{1}$$

$$\tau_i + s_{v_i} + t_{v_i v_{i+1}} \leq \tau_{i+1} \quad \forall v_i \in R, i \neq q \tag{2}$$

$$\tau_i + s_{v_i} + \overline{L}_{v_i} \geq \tau_{i^-} \quad \forall v_i \in R \cap P, \tag{3}$$

$$\tau_i + s_{v_i} + \underline{L}_{v_i} \leq \tau_{i^-} \quad \forall v_i \in R \cap P, \tag{4}$$

where τ_i denotes the start of service at node v_i. This is called the *scheduling problem* of the SPDP. Constraints (1) are time-window constraints. Consistency of the service times along the route is ensured by constraints (2), while (3) and (4) are maximum and minimum RT constraints, respectively. For ease of notation, we assume service durations of zero and omit them in all formulae in the following. The extension of all concepts to non-zero service durations is straightforward.

In (Tang *et al.*, 2010) and (Firat and Woeginger, 2011), feasibility tests for a similar scheduling problem, i.e., with constraints (1)–(3) and an additional constraint on the waiting time at each node $v_i \in R$, were presented. In the following, we sketch both algorithms and extend them to solve the scheduling problem of the SPDP. Also, we highlight the increased complexity coming from the additional presence of the minimum RT constraints that leads to increased worst-case running times of the adapted procedures.

3.1. Adapted Feasibility Test of Firat and Woeginger

The basic idea of Firat and Woeginger (2011) is to rewrite the considered scheduling problem as a system of difference constraints. It is well-known (see, e.g., Cormen *et al.*, 2001, Section 24.4) that such a system has a solution if and only if an associated digraph (called *constraint graph*) has no negative-weight cycle. Moreover, they formulate the difference-constraint system over an appropriate set of variables allowing the cycle-detection test to be performed in linear time.

Before we sketch their algorithm, some additional notation is necessary. For each node $v_i \in R$, the constant T_i denotes the sum of the travel times along the route R up to node v_i. The total waiting time up to node $v_i \in R$ is given by x_i.

Original Algorithm for the DARP. The first step of Firat and Woeginger (2011) is to rewrite constraints (1)–(3) of the considered scheduling problem in terms of T_i and x_i yielding

$$a_{v_i} \leq T_i + x_i \leq b_{v_i} \quad \forall v_i \in R, \tag{5}$$

$$x_i - x_{i+1} \leq 0 \quad \forall v_i \in R, i \neq q \tag{6}$$

$$x_{i^-} - x_i \leq T_i + \overline{L}_{v_i} - T_{i^-} \quad \forall v_i \in R \cap P. \tag{7}$$

Next, two additional dummy variables x_0 and x_{q+1} are introduced representing values $x_0 = 0$ and $x_{q+1} = b_{2n+1} - a_0$, i.e., they constitute lower and upper bounds for all $x_i, i = 1, ..., q$. Using x_0, x_{q+1}, and defining $M = b_{2n+1} - a_0$ constraints (5) can be written as the difference constraints

$$x_{q+1} - x_i \leq M + T_i - a_{v_i} \quad \forall v_i \in R, \tag{8}$$

$$x_i - x_0 \leq b_{v_i} - T_i \quad \forall v_i \in R. \tag{9}$$

The final set of difference constraints is (6)–(9) together with

$$x_{q+1} - x_0 \leq M, \tag{10}$$

$$x_0 - x_{q+1} \leq -M, \tag{11}$$

to enforce $x_{q+1} - x_0 = M$. The constraint graph associated with this system has a node for each $x_i, i = 0, ..., q+1$ and an arc with weight d_{ij} from node x_j to node x_i for each constraint $x_i - x_j \leq d_{ij}$ of the system (6)–(11). It has $\mathcal{O}(n)$ nodes and $\mathcal{O}(n)$ arcs. Thus, negative-weight cycles can be detected in $\mathcal{O}(n^2)$ using the Bellman-Ford algorithm.

The key observation of Firat and Woeginger (2011) to obtain a linear-time feasibility check is the following: They call an arc from x_j to x_i a forward arc if $j < i$, otherwise it is a backward arc. All arcs in the constraint graph are either forward arcs with non-negative weights (otherwise the scheduling problem would be trivially infeasible) or backward arcs with weight zero. The only exception is the arc corresponding to constraint (11) which is a backward arc with negative weight. Using this structure, they are able to transform the graph into a specific interval graph for which the cycle-detection test can be done in linear time.

Adapted Algorithm for the SPDP. The adaptation of the approach to the SPDP requires to perform the same transformation described above also to constraints (4) yielding:

$$x_i - x_{i^-} \leq T_{i^-} - T_i - \underline{L}_{v_i} \quad \forall v_i \in R \cap P. \tag{12}$$

The additional difference constraints (12) correspond to additional arcs in the constraint graph. Still, the total number of arcs is $\mathcal{O}(n)$ and the application of the Bellman-Ford algorithm gives a $\mathcal{O}(n^2)$ feasibility test.

To show that feasibility testing in linear time is not possible for the SPDP (with the technique in Firat and Woeginger, 2011), we have to analyze the new arcs: They are backward arcs and have non-negative weight if $T_{i^-} - T_i \geq \underline{L}_{v_i}$, i.e., if the minimum RT of request v_i is trivially satisfied on the given route because the travel time from v_i to v_{i^-} is already larger than $\underline{L}_{v_i}$. Otherwise, the arc weight is negative. As a result, the specific structure of the constraint graph that was exploited to obtain a linear time cycle-detection test is lost.

Note that a direct reformulation, i.e., using the original variables, of the scheduling problem (1)–(4) as a system of difference constraints also leads to a constraint graph with the same number of nodes and arcs as in the approach of Firat and Woeginger (2011). Thus in the additional presence of minimum RTs, both reformulations lead to equivalent feasibility tests. Moreover, the direct reformulation as a system of difference constraints is essentially identical to the modeling of the feasibility problem as a simple temporal problem as proposed in (Masson *et al.*, 2014) for a DARP with transfer possibilities between routes.

Summing up, the different transformations into cycle-detection problems yield $\mathcal{O}(n^2)$ feasibility tests for the SPDP. However, these algorithms require building a constraint graph for each route to be tested which might negatively affect the practical runtime.

3.2. Adapted Feasibility Test of Tang et al.

Tang *et al.* (2010) proposed a different feasibility test for the same scheduling problem as in (Firat and Woeginger, 2011), but with weaker, quadratic worst-case runtime. However, their algorithm seems more intuitive as it gradually constructs the schedule $\mathcal{T}_R$ directly on the original network. This also means that some information that is needed might already be available within an exact or heuristic approach in which the feasibility test is necessary. Therefore, their algorithm might be sufficiently fast (or even faster than the algorithm in Firat and Woeginger, 2011) in practice, especially if the given routes are not too long (see, e.g., Gschwind and Irnich, 2015)).

Again, we first sketch the original algorithm before we present our adaptations to the SPDP.

Original Algorithm for the DARP. The algorithm of Tang *et al.* (2010) tries to construct a feasible schedule $\mathcal{T}_R$ satisfying (1)–(3) by traversing the route twice: once forward and once backward. If no feasible schedule can be found the route is infeasible. Note that the third traversal of the original algorithm is redundant.

The whole procedure is described in Algorithm 1. The forward pass (Steps 1–4) builds a schedule of service times τ_i that satisfy constraints (1) and (2). Thereby, all times are scheduled as early as possible.

The backward pass (Steps 5–14) checks for consistency with the maximum RT constraints (3) adjusting some values τ_i if necessary: At each pickup node $v_i \in P$ it is checked if the current schedule satisfies the maximum RT of request v_i. If not, the algorithm tries to shift waiting time that occurs between pickup and delivery of request v_i before the pickup node v_i in order to decrease the ride time of request v_i. Thereto, the service at node v_i is delayed by as much as necessary to meet the maximum RT. This requires shifting the service times τ_k of all succeeding nodes $v_k, k = i+1, ..., q$ forward in time and is done in the same fashion as in the first pass, i.e., accounting for constraints (1) and (2). If there is not enough waiting time between pickup and delivery of request v_i, the maximum RT of v_i cannot be satisfied and the route is infeasible (Step 14). Note that this adjustment of the service times can never increase the RT of a request that is picked up later than v_i meaning that the maximum RT constraint of such a request never gets violated. Thus, after traversing a pickup node v_i in the backward pass we have a schedule that respects constraints (1) and (2), and the maximum RT constraints (3) of those requests for which the pickup node is not before v_i in the route. Moreover, all service times are still scheduled as early as possible with respect to the constraints

Algorithm 1: Algorithm of Tang *et al.* (2010)

Result: **true** if feasible schedule exists, **false** otherwise

`// Pass 1 (forward)`

1 $\tau_1 := a_{v_1}$

2 **for** $i = 2, ..., q$ **do**

3 $\quad \tau_i := \max\left\{\tau_{i-1} + t_{v_i v_{i-1}}, a_{v_i}\right\}$

4 $\quad$ **if** $\tau_i > b_{v_i}$ **then return** false

`// Pass 2 (backward)`

5 **for** $i = q-1, ..., 1$ **do**

6 $\quad$ **if** $v_i \in P$ **then**

7 $\quad\quad \Delta := \tau_{i^-} - \tau_i - \overline{L}_{v_i}$

8 $\quad\quad$ **if** $\Delta > 0$ **then**

9 $\quad\quad\quad \tau_i := \tau_i + \Delta$

10 $\quad\quad\quad$ **if** $\tau_i > b_{v_i}$ **then return** false

11 $\quad\quad\quad$ **for** $k = i+1, ..., q$ **do**

12 $\quad\quad\quad\quad \tau_k := \max\left\{\tau_{k-1} + t_{v_k v_{k-1}}, a_{v_k}\right\}$

13 $\quad\quad\quad\quad$ **if** $\tau_k > b_{v_k}$ **then return** false

14 $\quad\quad\quad$ **if** $\tau_{i^-} - \tau_i - \overline{L}_{v_i} > 0$ **then return** false

15 **return** true

that are satisfied at that point of the algorithm. Consequently, whenever a rescheduling results in a service time $\tau_j > b_{v_j}$ no feasible schedule exists and the route is infeasible.

Both forward and backward pass traverse the route once. Because of the inner forward loop (Steps 11–13), the backward pass has a quadratic worst-case running time and, hence, the overall algorithm also has time complexity $\mathcal{O}(n^2)$.

Adapted Algorithm for the SPDP. Solving the scheduling problem of the SPDP with the technique in (Tang *et al.*, 2010) requires the integration of the minimum RT constraints into the scheduling process. This can be done as follows: In the forward pass, Step 3 changes to

$$\tau_i := \begin{cases} \max\left\{\tau_{i-1} + t_{v_i v_{i-1}}, a_{v_i}, \tau_{i^+} + \underline{L}_{v_{i^+}}\right\} & \text{if } v_i \in D, \\ \max\left\{\tau_{i-1} + t_{v_i v_{i-1}}, a_{v_i}\right\} & \text{otherwise.} \end{cases}$$

The resulting service times after the first pass satisfy constraints (1), (2), and (4) and they are scheduled in an early-as-possible fashion.

In the backward pass, whenever a shifting of service times is necessary because of some violated maximum RT (Steps 8–14), we need to change Step 12 in the same way as Step 3 in order to maintain feasibility with respect to constraints (1), (2), and (4). In contrast to the original algorithm, however, the shifting of waiting times can increase the RT of requests that are picked up later in the route. Decisive is that we might be forced to re-introduce waiting time somewhere in the route due to the minimum RT constraints of other requests. As a result, the property that after traversing a pickup node v_i in the backward pass all maximum RTs of requests which are not picked before v_i are respected, is lost.

Consider the example given in Table 1. The travel times between all nodes are assumed to be 10. The maximum RTs of requests i and m are 30. The minimum RT of request j is 41. The time window of each node is specified in Table 1. All other constraints are assumed to be never binding.

Table 1 gives the service times at each node at different stages of the algorithm. Waiting times at nodes are in brackets. Pairs of service times that do not satisfy the RT constraints are highlighted in italics.

	Node	0	i	j	k	i^-	m	j^-	k^-	m^-	$2n{+}1$
	$[a.,b.]$	$[0,100]$	$[10,20]$	$[20,30]$	$[31,41]$	$[41,51]$	$[51,61]$	$[61,71]$	$[71,81]$	$[81,91]$	$[0,100]$
	$\underline{L}./\overline{L}.$	-	$0/30$	$41/100$	$0/100$	-	$0/30$	-	-	-	-
After pass 1	$\tau.$	0	*10*	20	(1) 31	*41*	51	61	71	81	91
Pass 2, at node i	$\tau.$	0	(1) 11	21	31	41	*51*	(1) 62	72	*82*	92
After add. pass 2	$\tau.$	0	(1) 11	21	31	41	(1) 52	62	72	82	92

Table 1: Example showing the necessity to loop over second pass in Algorithm 1

	Node	0	i	j	m	i^-	j^-	m^-	$2n{+}1$
	$[a.,b.]$	$[0,100]$	$[10,20]$	$[20,30]$	$[31,41]$	$[41,51]$	$[51,61]$	$[61,71]$	$[0,100]$
	$\underline{L}./\overline{L}.$	-	$0/30$	$31/100$	$0/30$	-	-	-	-
After pass 1	$\tau.$	0	*10*	20	(1) 31	*41*	51	61	71
Pass 2, at node i	$\tau.$	0	(1) 11	21	*31*	41	(1) 52	*62*	72
2nd pass 2, at node m	$\tau.$	0	(1) *11*	21	(1) 32	*42*	52	62	72
2nd pass 2, at node i	$\tau.$	0	(2) 12	22	*32*	42	(1) 53	*63*	73

Table 2: Example showing nested requests causing a worst-case complexity greater than $\mathcal{O}(n^2)$

After the first pass of the algorithm, the maximum RT constraint of request i is violated by 1 unit of time. Therefore, the second pass delays the service τ_i at i by 1 unit trying to shift waiting time that occurs between i and i^- before i. However, the removed waiting time at node k decreases the RT of request j so that it has to be re-introduced before starting the service at node j^-. Otherwise, the minimum RT constraint of j would be violated. As a consequence, the RT of request m increases and the schedule after the backward pass is infeasible which, however, does not imply that the route is infeasible.

A straightforward way to fix this defect of the algorithm is to loop over the second pass as long as an adjustment of the service times was necessary because of a violated maximum RT. For the example above, one additional backward pass identifies a feasible schedule (see Table 1). Clearly, this change of the algorithm leads to a significant deterioration of the worst-case complexity, especially if the number of iterations of the backward pass cannot be bounded.

Consider the example as given in Table 2. The time windows of the nodes are specified in the table. All other data is equivalent to the previous example. Apparently, the three requests i, j, and m are nested in a way that the adapted Algorithm 1 alternatively shifts waiting times that occur in between pickup and delivery of the requests i and m. In general, this can lead to an unbounded number of iterations and, hence, a superlinear runtime of the algorithm. In case of integer inputs for the temporal data, the total number of iterations that can be caused by this behavior can be bounded by $M = \max_{i \in P \cup D}(b_i - a_i)$. The runtime of the algorithm is then given by $\mathcal{O}(Mn^2)$. Overall, the worst-case complexity of the adapted algorithm seems to be prohibitively large for frequent calls. However, in the computational tests of a companion paper (Gschwind, 2015), we found that the described algorithm can solve the scheduling problems arising within a proposed branch-and-cut-and-price approach to the SPDP sufficiently fast. Note that the paths arising in the instances considered in (Gschwind, 2015) are typically very short (not more than 10–15 requests). We, therefore, conduct a more comprehensive computational study with much longer routes in the following.

3.3. Computational Comparison

To compare the practical performance of the approaches of Sections 3.1 and 3.2, we evaluate their runtime over a huge number of randomly generated paths with lengths reaching from 15 to 200 requests. A separate analysis is conducted for integer and double precision input parameters. Details on the generation of the paths and more detailed results can be found in the online supplement of this paper. Note that in our analysis we do not include instances that are identified as infeasible already in the first pass of the adapted Algorithm 1.

Table 3 summarizes our results. Each row aggregates over a total of 50 000 paths. *Firat* and *Tang* denote the adapted feasibility checks of Firat and Woeginger (2011) and Tang *et al.* (2010), respectively.

Table 3 reports over all instances the average, maximum, and minimum times in milliseconds needed by the algorithms to solve an instance 10 000 consecutive times. Furthermore, we give the average, maximum, and minimum number of restarts of the second pass of the adapted Algorithm 1.

Table 3 reveals that there are paths, for which a huge number of iterations of the second pass of the adapted Algorithm 1 is necessary, especially for instances with double precision parameters. For these instances the runtime of *Tang* is very long. However, this seems to happen only very rarely so that the average number of iterations is small. Moreover and in contrast to the theoretical worst-case complexity, the average runtime of *Tang* is significantly better than that of *Firat*.

		Solution time			# Restarts		
Param.	Algo.	avg.	max	min	avg.	max	min
integer	*Firat*	2 160.3	19 081	15	-	-	-
	Tang	95.6	21 590	<1	3.1	420	0
double	*Firat*	1 937.4	21 684	15	-	-	-
	Tang	111.6	176 016	<1	6.0	12 425	0

Table 3: Aggregated computational results

4. Forward Time Slack

The concept of FTS was originally introduced by Savelsbergh (1992) in the context of the VRPTW. Let $\mathcal{T}_R = (\tau_1, ..., \tau_q)$ be a feasible schedule for route R. Savelsbergh (1992) defines the FTS $F_i^{\mathcal{T}}$ for a node $v_i \in R$ as the maximum value by which the start of service τ_i at v_i can be increased without causing the route to become infeasible. Different authors have generalized the concept of FTS to other problems, e.g., Cordeau and Laporte (2003) for the DARP or Masson *et al.* (2013) for the PDPT.

VRPTW. The slack at a node $v_j, j \geq i$ with respect to node v_i is given by the cumulative waiting time between v_i and v_j and the difference between the end of the time window and the start of service at node v_j. The FTS $F_i^{\mathcal{T}}$ at node v_i is the minimum of all slacks at nodes v_j with $i \leq j \leq q$. Denote by W_i the waiting time at node $v_i \in R$. Then (Savelsbergh, 1992):

$$F_i^{\mathcal{T}} = \min_{i \leq k \leq q} \left\{ \sum_{i < p \leq k} W_p + (b_{v_k} - \tau_k) \right\}. \tag{13}$$

Note that this definition of $F_i^{\mathcal{T}}$ is independent of the service times $\tau_k, k < i$, i.e., it is unique with respect to the route segment preceding v_i. This can be an important property for feasibility testing when inserting pairs of nodes as, e.g., in (Masson *et al.*, 2013) for the PDPT.

DARP. For problems with RTs, shifting the start of service at some node may cause infeasibilities also with respect to the RT constraints. This has to be incorporated in the definition of the FTS for such problems. For example, postponing the service at node v_i may increase the RT of a request v_k with $k < i$ and $k^- > i$. Consequently, Cordeau and Laporte (2003) define the following generalization of the FTS for the DARP:

$$F_i^{\mathcal{T}} = \min_{i \leq k \leq q} \left\{ \sum_{i < p \leq k} W_p + \min \left(b_{v_k} - \tau_k, S_{ik}^{\mathcal{T}} \right) \right\}, \tag{14}$$

where $S_{ik}^{\mathcal{T}}$ is given by $\overline{L}_{v_{k^+}} - (\tau_k - \tau_{k^+})$ if $v_k \in D; i > k^+$ and $+\infty$ otherwise.

In contrast to (13), the FTS (14) of a node v_i for the DARP (and similarly for other problems with maximum RTs) is not unique with respect to the route segment preceding v_i. Indeed, it may be possible for some $v_k \in D$ with $k^+ < i$ and $k > i$ to increase τ_{k^+} such that τ_i stays constant while $S_{ik}^{\mathcal{T}}$ and $F_i^{\mathcal{T}}$

Node		0	i	j	j^-	i^-	$2n+1$
$[a., b.]$		$[0, 100]$	$[10, 30]$	$[30, 50]$	$[40, 60]$	$[50, 70]$	$[0, 100]$
Earliest as possible schedule	$\tau_\cdot$	0	10	(10) 30	40	50	60
FTS for nodes after v_i with $\tau_i = 10$	$F_\cdot^{\mathcal{T}}$	-	-	0	0	0	40
FTS for nodes after v_i with $\tau_i = 20$	$F_\cdot^{\mathcal{T}}$	-	-	10	10	10	40

Table 4: Example instance for which feasibility checking of request insertions based on FTS is problematic

also increase. For that reason, feasibility checking of request insertions based on FTS (as, e.g., proposed in Masson *et al.*, 2013) is problematic.

Consider the route given in Table 4. Feasibility of the insertion of another request k is to be evaluated. Assume a travel time of ten between all nodes, $\overline{L}_i = 40$, and time window $[0, 20]$ at the pickup node k. Inserting node k before i and node k^- before j^- results in a feasible route. When using the FTS-values of the earliest-as-possible schedule (with $\tau_i = 10$), however, the insertion appears to be infeasible because τ_{j^-} is increased by $10 > F_{j^-}^{\mathcal{T}} = 0$. Considering FTS-values based on servicing node i at time 20 correctly determines feasibility of this route. On the other hand, using these FTS-values results in misjudging the insertion of k before j and k^- before j^- as feasible because $F_{j^-}^{\mathcal{T}} = 10$ relies on $\tau_i = 20$ which is not possible in the new route.

While the adaptation of the FTS-based algorithm of Masson *et al.* (2013) does not lead to a valid feasibility check of request insertions in the presence of maximum RT constraints, the same technique can still be used as a constant-time verifiable sufficient condition: Choosing a specific feasible schedule for a given route, the insertion of new requests can be evaluated based on the considered schedule and its associated FTS-values. Additionally, the maximum RT constraint of the newly inserted request has to be checked. As shown in the example above, failing these tests does not imply infeasibility of the insertion. Complementing this strategy with other sufficient or necessary conditions might significantly reduce the number of calls to a computationally more expensive exact feasibility test in an insertion-based approach to the DARP.

SPDP. The presence of minimum RTs raises two additional issues compared to the DARP. First, shifting the service time at a pickup node is constrained by the minimum RT of this request. Second, not necessarily all of the cumulative waiting time between two nodes v_i and v_k is slack. Decisive is that some waiting time may have to be included between them because of the minimum RT of some requests. Denote by $UW_{ik}^{\mathcal{T}}$ the *usable waiting time* between v_i and v_k, i.e., the waiting time between v_i and v_k that can be incorporated into the slack. Then, the FTS for a node v_i is given by

$$F_i^{\mathcal{T}} = \min_{i \le k \le q} \left\{ UW_{ik}^{\mathcal{T}} + \min\left(b_{v_k} - \tau_k, S_{ik}^{\mathcal{T}}\right) \right\}, \tag{15}$$

with

$$S_{ik}^{\mathcal{T}} = \begin{cases} b_{v_{k^-}} - \left(\underline{L}_{v_k} + \tau_k\right) & \text{if } v_k \in P, \\ \overline{L}_{v_{k^+}} - (\tau_k - \tau_{k^+}) & \text{if } v_k \in D; i > k^+, \\ +\infty & \text{otherwise.} \end{cases}$$

To determine $UW_{ik}^{\mathcal{T}}$, we first define the *necessary waiting time* $NW_{k^-} = \left(\underline{L}_k - \sum_{k<j\le k^-} t_{j-1,j}\right)^+$ for delivery node v_{k^-} which is the amount of waiting time that inevitably occurs between pickup and delivery of request v_k and, hence, can never be part of the slack between nodes v_i and v_{k^-} if $k > i$. Note that NW_{k^-} is independent of the actual schedule $\mathcal{T}_R$ for which the FTS is determined. When there is another request v_j with positive NW_{j^-} in between v_i and v_k, it has to be ensured that the overlapping necessary waiting times NW_{j^-} and NW_{k^-} are not considered twice in the definition of $UW_{ik}^{\mathcal{T}}$. Therefore, we define for each delivery node v_j after v_i the already *included waiting time* between them, which is also independent of the

base schedule $\mathcal{T}_R$, as

$$IW_{ik} = \sum_{\substack{i<j<k \\ v_j \in D, j^+ \geq i, j > k^+}} (NW_j - IW_{ij})^+ .$$

The term $(NW_k - IW_{ik})^+$ then gives the amount of waiting time that will always be present at a delivery node $v_k \in D$ (because of the minimum RTs of the requests) if there is no other, unnecessary waiting time in between nodes v_k^+ and v_k on a schedule. Consequently, this needed amount of waiting time is never slack and has to be deducted from the cumulative waiting time of any considered schedule between nodes v_i and v_k when computing the usable waiting time $UW_{ik}^{\mathcal{T}}$.

In addition to these schedule-independent times, there may also be non-slack waiting times that are induced by the considered base schedule $\mathcal{T}_R$. This can happen if for a node v_k, $k \geq i$ in between pickup and delivery of a request v_j, i.e., $v_j \leq v_k < v_{j^-}$, and the considered schedule $\mathcal{T}_R$ there is a deficit in waiting times, meaning that the difference

$$\Delta_{ijk}^{\mathcal{T}} = \sum_{j<p\leq j^-} W_p - \sum_{\substack{k<p\leq j^- \\ v_p \in D, p^+ \geq i}} (NW_p - IW_{ip})^+$$

of the cumulative waiting time between v_j and v_j^- and the cumulative needed amount of waiting time at the delivery nodes $v_p \in D$ with $p^+ > i$ on the partial route between node v_k and the delivery node v_j^- of request v_j is negative. This is possible whenever request v_j is nested with another request $v_{p'}$ whose delivery node $v_{p'^-}$ lies in between v_k and v_{j^-} on route R and has a positive need for waiting time $NW_{p'^-} - IW_{ip'^-} > 0$.

If for $\mathcal{T}_R$ the deficit in waiting times $\left(\Delta_{ijk}^{\mathcal{T}}\right)^-$ is larger than $\delta_j = \overline{L}_j - (\tau_{j^-} - \tau_j)$, i.e., the slack with respect to the maximum RT constraint of v_j, the amount $\left(\Delta_{ijk}^{\mathcal{T}}\right)^- - \delta_j$ must not be incorporated into the usable waiting time of node v_k. The reason is that shifting this waiting time (by delaying the service at v_i) before nodes v_j, v_k, and $v_{p'}$ causes the reintroduction of waiting times right before the delivery node $v_{p'^-}$ to meet the associated minimum RT $\underline{L}_{v_{p'}}$ of request $v_{p'}$ due to $NW_{p'^-} - IW_{ip'^-} > 0$. This in turn increases the RT of request v_j compared to the initial schedule $\mathcal{T}_R$ so that the services at all nodes on the partial path from v_j to v_k have to be delayed by $\left(\Delta_{ijk}^{\mathcal{T}}\right)^- - \delta_j$.

Formally, we have for each $k > i$ the schedule-dependent additional amount of time

$$AW_{ik}^{\mathcal{T}} = \max_{\substack{i<j\leq k \\ v_j \in P, j^- > k}} \left(\left(\Delta_{ijk}^{\mathcal{T}}\right)^- - \delta_j\right)^+ \tag{16}$$

that has to be deducted from the cumulative waiting times at node v_k. For convenience, we define $AW_{ik}^{\mathcal{T}} = 0$ if the maximum in (16) is empty and for all $k \leq i$. Furthermore, if for a request v_k the times $AW_{ik}^{\mathcal{T}}$ and $AW_{ik^-}^{\mathcal{T}}$ differ they may imply additional, schedule-dependent waiting times that are necessary between the pickup and delivery nodes. Consequently, *schedule-dependent necessary waiting times* and *schedule-dependent already included waiting times* at delivery nodes $v_k, k > i$ are defined as

$$NW_{ik}^{\mathcal{T}} = NW_k + \left(AW_{ik^+}^{\mathcal{T}} - AW_{ik}^{\mathcal{T}}\right)^+$$

and

$$IW_{ik}^{\mathcal{T}} = \sum_{\substack{i<j<k \\ v_j \in D, j^+ \geq i, j > k^+}} \left(NW_{ij}^{\mathcal{T}} - IW_{ij}^{\mathcal{T}}\right)^+ ,$$

Node	0	i	j	k	m	m^-	i^-	j^-	k^-	$2n+1$
$[a.,b.]$	$[0,200]$	$[12,24]$	$[25,35]$	$[35,45]$	$[45,55]$	$[55,70]$	$[75,87]$	$[85,97]$	$[95,110]$	$[0,200]$
$\underline{L}_{.}/\overline{L}_{.}$	-	63/200	0/62	65/200	15/200	-	-	-	-	-
$\tau_{.}$	0	(2) 12	(3) 25	35	45	(5) 60	(5) 75	85	(5) 100	110
$NW_{.}$	-	-	-	-	-	5	13	0	15	-
$IW_{0.}$	-	-	-	-	-	0	5	13	13	-
$\Delta^{\mathcal{T}}_{0j.}$	-	-	-3	-3	-3	0	0	-	-	-
$AW^{\mathcal{T}}_{0.}$	0	0	1	1	1	0	0	0	0	0
$NW^{\mathcal{T}}_{0.}$	-	-	-	-	-	6	13	1	16	-
$IW^{\mathcal{T}}_{0.}$	-	-	-	-	-	0	6	13	13	-
$UW^{\mathcal{T}}_{0.}$	0	2	4	4	4	4	2	2	4	4

Table 5: Example demonstrating the computation of $UW^{\mathcal{T}}_{0.}$ in the case of nested requests. The travel times between all nodes are assumed to be 10.

respectively.

Then, the usable waiting time $UW^{\mathcal{T}}_{ik}, k \geq i$ is given by

$$UW^{\mathcal{T}}_{ik} = \sum_{i<j\leq k} W_j - \sum_{\substack{i<j\leq k \\ v_j\in D, j^+\geq i}} \left(NW^{\mathcal{T}}_{ij} - IW^{\mathcal{T}}_{ij}\right)^+ - AW^{\mathcal{T}}_{ik}.$$

The small example in Table 5 demonstrates the computation of values $UW^{\mathcal{T}}_{i.}$ given the as early-as-possible schedule of a route with nested requests leading to a deficit in waiting times for nodes j, k, and m in between pickup and delivery of request j.

As in the DARP, the FTS (15) for a node v_i is not unique with respect to the route segment preceding v_i and the technique of Masson *et al.* (2013) cannot be used for exact feasibility testing of request insertions. However, it enables a constant time verifiable sufficient condition similar to the one described above for the DARP that can be used within insertion-based approaches to the SPDP in order to reduce the number of calls to one of the computationally expensive feasibility tests of Section 3. To analyze the possible benefits of using such a sufficient condition, we test it on the instance set of Section 3.3 as follows. Given a feasible route we remove one of the requests, calculate the early-as-possible schedule and associated FTS-values for the resulting route and evaluate the reinsertion of the removed nodes at their original positions based on the sufficient condition. This is done for all routes and all requests on the routes (one at a time). Overall, the condition appears to be very effective as it was able to prove feasibility of the insertions in approximately 96% of the 3 900 000 calls. More detailed results are presented in the online supplement of this paper. They reveal that the condition is especially effective for routes with few nested requests.

5. Conclusions

In this paper, we derived two route feasibility checks for the SPDP and extended the concept of FTS to this problem. Because of the prototypical character of the SPDP, we believe that the presented concepts are also relevant for other routing problems with temporal synchronization constraints. For example, the feasibility test of Section 3.1 can be adapted to a SPDP with transfer possibilities between routes in a straightforward way. Likewise, the proposed definition of the FTS can be generalized to the problem with transfer possibilities using the technique in (Masson *et al.*, 2013).

References

Cordeau, J.-F. and Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B*, **37**(6), 579–594.

Cordeau, J.-F. and Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, **153**(1), 29–46.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction To Algorithms*. MIT Press.

n	Cust. loc.	Horizon	TW	Min RT	Max RT
15	$[10, 10] \times [10, 10]$	450	60	2	10
25	$[10, 10] \times [10, 10]$	750	100	2	10
50	$[10, 10] \times [10, 10]$	1500	200	2	10
100	$[10, 10] \times [10, 10]$	3000	400	2	15
200	$[10, 10] \times [10, 10]$	6000	500	2	30

Table 6: Details on the parameters used for instance generation

Firat, M. and Woeginger, G. J. (2011). Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. *Operations Research Letters*, **39**(1), 32–35.

Gschwind, T. (2015). A comparison of column-generation approaches to the synchronized pickup and delivery problem. *European Journal of Operational Research*, **247**(1), 60–71.

Gschwind, T. and Irnich, S. (2015). Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. *Transportation Science*, **49**(2), 335–354.

Masson, R., Lehuédé, F., and Péton, O. (2013). Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. *Operations Research Letters*, **41**(3), 211–215.

Masson, R., Lehuédé, F., and Péton, O. (2014). The dial-a-ride problem with transfers. *Computers & Operations Research*, **41**, 12–23.

Savelsbergh, M. W. P. (1992). The vehicle routing problem with time windows: Minimizing route duration. *INFORMS Journal on Computing*, **4**(2), 146–154.

Tang, J., Kong, Y., Lau, H., and Ip, A. W. H. (2010). A note on "efficient feasibility testing for dial-a-ride problems". *Operations Research Letters*, **38**(5), 405–407.

Online Supplement

A. Detailed Computational Results

In this section, we give additional details on the computational comparison of the two feasibility checks of Sections 3.1 and 3.2 and on the performance of the FTS-based sufficient condition for evaluating the feasibility of request insertions as described in Section 4. The algorithms are tested on a huge number of randomly generated paths using Euclidean distances (rounded down in the case of instances with integer inputs) as travel times between the nodes. Minimum and maximum RTs are specified proportional to the Euclidean distances between pickup and delivery location. Details on the parameters used for the generation of the paths are given in Table 6. The columns have the following meaning:

n	Number of requests
Cust. loc.	Intervals from which the customer locations are drawn. The depots are located at $(0, 0)$.
Horizon	Time horizon
TW	Mean value for the length of the time windows
Min RT	Mean value for the factor specifying the minimum RT
Max RT	Mean value for the factor specifying the maximum RT

For each value of n we generate paths with different characteristics regarding the number of requests that are open at the nodes in the following way: At each node another request is picked up with probability p, otherwise one of the open requests is delivered. The considered values of p reach from 0.15 to 0.50 in steps of 0.05. For each probability p, a pretest filters 625 feasible and 625 infeasible instances so that a total of 10 000 paths is considered for each value of n. Note that we consider only instances that are not identified as infeasible already in the first pass of the adapted Algorithm 1.

Table 7 summarizes our results on the comparison of the two feasibility checks of Sections 3.1 and 3.2. The columns have the following meaning:

Param.	Type of input data (integer or double precision) of the instances
n	Number of requests

					Firat			*Tang*					
		Max # open			Solution time			Solution time			# Restarts		
Param.	n	avg.	max	min	avg.	max	min	avg.	max	min	avg.	max	min
integer	15	2.7	6	1	57.2	146	15	5.8	115	<1	1.0	45	0
	25	3.3	8	1	137.9	312	31	11.4	319	<1	1.6	76	0
	50	4.2	11	2	513.9	1 046	78	33.0	1 810	<1	2.2	162	0
	100	5.5	19	2	2 077.7	4 477	281	125.0	10 936	15	4.4	364	0
	200	6.8	23	2	8 014.9	19 081	726	302.7	21 590	31	6.4	420	0
double	15	2.7	6	1	49.8	125	15	8.9	15 459	<1	3.3	7 596	0
	25	3.3	6	1	115.8	312	31	18.2	14 492	<1	4.2	3 628	0
	50	4.2	11	1	438.4	1 123	78	37.8	15 116	<1	4.1	4 979	0
	100	5.5	19	2	1 787.5	4 556	203	161.6	140 994	<1	9.0	12 425	0
	200	6.8	23	2	7 295.4	21 684	624	331.5	176 016	15	9.7	9 548	0

Table 7: Detailed computational results for the feasibility tests of Sections 3.1 and 3.2

		Max # open			Insertion tests		
Param.	p	avg.	max	min	total	# pos.	% pos.
integer	0.15	2.8	7	1	243750	242540	99.50
	0.20	3.1	8	1	243750	241784	99.19
	0.25	3.4	10	1	243750	240771	98.78
	0.30	3.8	10	1	243750	239016	98.06
	0.35	4.1	11	1	243750	236936	97.20
	0.40	4.5	15	1	243750	234274	96.11
	0.45	4.8	13	2	243750	230505	94.57
	0.50	5.0	16	2	243750	226653	92.99
double	0.15	2.8	7	1	243750	242192	99.36
	0.20	3.2	8	1	243750	241354	99.02
	0.25	3.5	9	1	243750	239739	98.35
	0.30	3.8	12	1	243750	237747	97.54
	0.35	4.2	11	2	243750	234592	96.24
	0.40	4.5	13	1	243750	230732	94.66
	0.45	4.8	14	2	243750	226077	92.75
	0.50	5.1	13	2	243750	220482	90.45

Table 8: Detailed computational results for FTS-based sufficient condition

Max # open	The maximum number of open requests at a node; we give average, maximum, and minimum values over the instances
Firat	The adapted feasibility check of Firat and Woeginger (Section 3.1)
Tang	The adapted feasibility check of Tang *et al.* (Section 3.2)
Solution time	The time in milliseconds needed by the respective algorithm to solve an instance 10 000 consecutive times; we present average, maximum, and minimum values over the instances
# Restarts	The number of restarts of the second pass of Algorithm 1

Table 8 presents details on the performance of the FTS-based sufficient condition proposed in Section 4 for evaluating the feasibility of request insertions into given feasible routes. For each value of parameter p, we present the total number of evaluated insertions together with the number and fraction (in percent) of insertions which are proven to be feasible by the condition.