Truck Scheduling in the Postal Service Industry

Nils Boysen, a Stefan Fedtke, a Felix Weidinger a

a Friedrich–Schiller–Universität Jena, Lehrstuhl für Operations Management, 07743 Jena, Germany

Contact: nils.boysen@uni-jena.de (NB); stefan.fedtke@uni-jena.de (SF); felix.weidinger@uni-jena.de (FW)

Received: May 12, 2015
Revised: January 2016
Accepted: June 27, 2016
Published Online in Articles in Advance: February 16, 2017

https://doi.org/10.1287/trsc.2016.0722

Abstract. The distribution networks of the postal service industry are organized according to the hub-and-spoke paradigm, so that parcel distribution centers play a crucial role to consolidate the parcel flows to full truckloads. In these terminals, inbound trucks are unloaded at gates, shipments are identified, sorted by the central sortation conveyor system, and loaded into outbound trailers, in which they are moved toward their next destination. In this context, the scheduling of inbound trucks, which assigns a gate and a processing interval to each truck, is an essential operational decision problem. We formalize the resulting optimization problem and provide suited solution procedures. Furthermore, we test the impact of truck scheduling on the sortation performance of the central conveyor system with the help of a comprehensive terminal simulation.

Funding: This research has been supported by the German Science Foundation (DFG) through the grant “Planning and operating sortation conveyor systems” [BO 3148/5-1].

Supplemental Material: The online appendix is available at https://doi.org/10.1287/trsc.2016.0722.

Keywords: postal service industry • hub terminals • truck scheduling • parcel sortation

1. Introduction

The tremendous growth of e-commerce has made the parcel service industry a backbone of modern society and one of the fastest rising segments of logistics services (e.g., Garcia-Romeu-Martinez et al. 2007). A large part of this success story is based on the distribution networks being organized according to the hub-and-spoke paradigm. On one hand, the consolidation of parcel flows in central hub terminals enables economies of scale in transportation by unifying multiple smaller shipments to full truckloads. This way, the costs for moving a truck are shared among more shipments, so that lower costs per unit can be handed down to the customers in terms of lower postal charges. On the other hand, centralization allows the application of fully automated sortation processes. In modern postal distribution centers, closed-loop tilt tray conveyors are applied to ensure a fast and reliable sortation of parcels. For instance, Hermes Logistik—Germany’s second largest postal service provider—erected its central hub terminal in Friedewald in 2009 for €45 million (Fedtke and Boysen 2014). The largest part of this investment, i.e., €25 million, was consumed by the central sortation system, which consists of two closed-loop tilt-tray conveyors (on top of each other) each having a length of 700 meters (m) and 766 trays. With this system, up to 400,000 parcels can be sorted per day. It can be concluded that efficient sortation processes are highly relevant for the postal service industry.

1.1. Operations in a Parcel Distribution Center

Once a truck loaded with mail arrives at a parcel distribution center, it is registered at the entry gate. Then, there are two alternative process flows:

- The truck is assigned a parking space where the truck and trailer have to wait until they are called up and directed to a dock door by the yard scheduler, e.g., via a pager handed over at the entry gate. Then, the truck directly transports the trailer to the terminal, where it is docked at its designated door and unloading starts. The truck is released from its duty once the trailer is docked and standing on its own base frames.
- Alternatively, the trailer is uncoupled from its delivery truck at a specific location in the trailer yard, which the yard scheduler has instructed. In this scenario, the truck may then directly leave the terminal area, whereas the trailer waits for processing. The final delivery to the terminal is executed by a special trailer truck, which is also denoted as a switcher or spotter (see Yano, Bozer, and Kamoun 1998).

While the latter alternative requires additional investment into the trailer trucks, they are often much faster in maneuvering the trailers into their docking positions and allow for an earlier release of the delivery trucks. In both settings, the decision of the yard manager where, i.e., at what dock door, and when, i.e., in what time interval, an inbound trailer should be unloaded, is supported by the truck scheduling problem investigated in this paper. Specifically, we aim to process trucks at docks, such that the parcels loaded onto the main conveyor quickly reach the chutes connected with their
designated outbound trucks. This way, multiple tray loads during a single rotation of the conveyor belt are enabled and sortation performance increases.

Inside the terminal, whose schematic layout is depicted in Figure 1, a logistics worker pulls a telescope conveyor directly in front of (later into) the trailer and unloads the parcels by putting them on a conveyor belt. The belts of neighboring doors belonging to the same inbound segment are unified and transport their parcels toward a loading station. Here, the addressee of each parcel is automatically identified by a camera system and—via a switch system—parcels are loaded onto the main closed-loop tilt tray conveyor. Each tray receives at most one parcel, so that the background information system precisely knows the destination of each tray’s parcel. Then, a parcel travels along the closed-loop conveyor until it reaches the chute of its outbound destination where the tray is tilted and the parcel slides down the chute onto another conveyor belt. This conveyor leads to the outbound trailer dedicated to a specific destination, where the parcel is loaded into an outbound trailer by another logistics worker. Once an outbound trailer is completely filled, this status is communicated to the background information system, so that the yard manager can request a switcher to park the trailer on a specific position in the trailer yard. The trailer is then collected by a truck transporting it to the respective destination. Outbound docks are typically dedicated to a fixed destination over a longer time period, e.g., three months (see Boysen and Fliedner 2010), so that another empty trailer dedicated to the same destination is docked at the respective outbound door once its predecessor has left.

1.2. Decision Problems and Literature Review
From a network perspective, the structure of the hub-and-spoke network and the specific terminal locations have to be determined. At a specific postal distribution center, the following main decision problems—ordered according to decreasing time frame—can be differentiated (Fedtke and Boysen 2014):

- **Layout planning** mainly decides on the shape of the terminal (e.g., I-, L-, T-, or X-shaped, see Bartholdi and Gue 2004), the number of inbound and outbound docking doors, their arrangement around the perimeter of the terminal (see Stephan and Boysen 2011b), and the layout of the sortation conveyor (see Fedtke and Boysen 2014). Sortation conveyors necessitate a high investment, so that they often constitute the bottleneck of a terminal. Thus, determining a proper conveyor layout is a very critical task and, in this context especially, the number of inbound segments considerably influences the sortation throughput. If only a single inbound segment is available, then each tray can only be loaded once during its complete cycle through the terminal. If, however, multiple inbound segments are available (separated by intermediate outbound segments), then multiple tray loads per cycle are enabled whenever a tray is emptied before reaching a successive inbound segment. In the worst case, however, an occupied tray may still travel through the complete terminal, so that again only a single tray load arises in a cycle. Thus, for exploiting the additional flexibility promised by multiple inbound segments, a careful planning where parcels enter and leave the system is essential.

- The **destination assignment** problem assigns inbound and/or outbound destinations to dock doors over a midterm horizon. During this time span, all trucks serving a specific destination are processed at the same designated dock door. Important contributions in the context of general cross docking terminals, where the movement of shipments inside a terminal...
is executed by forklifts, are, e.g., provided by Tsui and Chang (1990, 1992), Gue (1999), Bartholdi and Gue (2000), and Bozer and Carlo (2008). Postal distribution centers, in particular, are addressed by McAree, Bodin, and Ball (2002) and Fedtke and Boysen (2014). Obviously, more flexibility is added if the assignment of trucks is not fixed, but can be adapted according to the current workload. However, frequently changing assignments aggravate the orientation of logistics workers and, thus, complicate the organization of the loading processes, so that a widespread solution especially applied in the postal service industry is to only fix the assignment of outbound destinations over a midterm horizon, whereas inbound trucks are flexibly processed at facultative inbound doors (see Bozer and Carlo 2008, Boysen and Fliedner 2010).

- **Truck scheduling** is an operational problem, which assigns a dock door to each individual truck and decides on the processing sequence per door. A detailed review and classification of this planning task is provided by Boysen and Fliedner (2010) and important contributions are provided, e.g., by Yu and Egbelu (2008), Boysen, Fliedner, and Scholl (2010), and Miao, Lim, and Ma (2009). All of these contributions, however, treat cross docking terminals and aim to synchronize incoming shipments with outbound trucks. In the context of postal distribution centers where sortation performance is a basic aim, this problem has not been investigated yet. Given the aforementioned peculiarities of this logistics segment, we only schedule inbound trucks, whereas on the outbound side, a fixed outbound dock is assigned to each destination and trailers depart once they are fully loaded. Afterwards, they are instantaneously replaced by another trailer dedicated to the same destination. In this setting, we aim to increase the sortation performance by preferring a truck processing at nearby inbound segments, so that trays are quickly unloaded in neighboring outbound segments and multiple tray loads per cycle are enabled.

- **Conveyor control**: Finally, with a given arrival pattern of parcels, the flow of parcels through the sortation system needs to be organized and controlled. There are some studies, which derive analytical equations for predicting the throughput of a closed-loop conveyor system (e.g., see Johnson and Meller 2002, Bozer and Hsieh 2004, 2005). However, our study evaluates a more complicated conveyor layout with multiple inbound segments. For this layout no analytical equations exist. Consequently, we apply a simulation study to evaluate the impact of truck scheduling on sortation performance.

A detailed literature review on these decision problems and their interdependencies in the context of general cross docking terminals is provided by Stephan and Boysen (2011a) and Van Belle, Valckeniers, and Cattrysse (2012). Note that with regard to our methodology there is some overlap with parallel machine scheduling and interval scheduling. However, we refer to the related literature later on (see Section 3) when the problem is defined in detail.

1.3. Research Question

This paper studies the truck scheduling problem at postal distribution centers. At the centers, a dock door and a feasible processing interval are to be assigned to inbound trucks, such that the number of outbound segments passed by the trucks’ shipments on the sortation conveyor is minimized. By pursuing this surrogate objective, we aim to enable multiple tray loads per cycle and, thus, a higher sortation performance. Measuring the detailed parcel throughput requires information on the unloading sequences of parcels from each truck and each parcel’s exact arrival at its loading station. As this information is typically not available before having unloaded a truck, we evaluate whether our surrogate objective indeed increases sortation performance by a detailed terminal simulation. This way, we can also provide yard managers with insights regarding the impact of forecast errors, truckload variability, and increasing the number of inbound segments.

The remainder of the paper is structured as follows. Section 2 defines our truck scheduling problem in detail and Section 3 provides suited solution procedures. Then, Section 4 details the setup of our computational study and presents the results. Finally, Section 5 concludes the paper.

2. The Truck Scheduling Problem in the Postal Service Industry

2.1. Problem Description

Consider a given set $J = \{1, \ldots, n\}$ of inbound trucks to be unloaded at the dock doors of a postal distribution center. Unloading truck $j \in J$ requires a deterministic processing time $p_j$ and is bound to arrival time $a_j$ and deadline $d_j$. A truck cannot be scheduled prior to its arrival at the distribution center, which in times of GPS-based navigation systems can continuously be announced to the terminal’s scheduling system. Furthermore, each truck is typically assigned a deadline restricting its completion time. Tight deadlines are a general characteristic of the postal service industry to avoid unsatisfied customers and fines in the money-back programs of express logistics services (see Boysen, Briskorn, and Tschöke 2013). A truck’s deadline may, for instance, originate from a general management regulation that a trailer must not wait in the yard for more than a predefined amount of time, a delivery time agreed with a customer of express services, or a successive appointment of the delivery truck.

The docks of the terminal are separated into multiple inbound and outbound segments. An inbound
segment \( s \in S \) consists of a set \( G_s \) (with \( \bigcup_{s \in S} G_s = G = \{1, \ldots, |G|\} \)) of successive inbound doors, whose incoming shipments are all loaded onto the sortation conveyor via the same loading station (see Figure 1). All outbound doors in between two inbound segments form an outbound segment. For the outbound side, we presuppose that outbound destinations have already been fixed in a previous planning step, i.e., when solving the destination assignment problem. Thus, our truck scheduling problem only schedules inbound trucks at the inbound segments, which are strictly separated from outbound operations.

Our aim is to enable multiple tray loads per cycle of the sortation system. Unfortunately, an exact calculation of tray loads depends on the unloading sequences of shipments from trailers and their exact arrival times at the loading stations. This information is barely predictable and cannot be forecasted accurately prior to the actual unloading process. Therefore, we apply a surrogate objective. Note that a surrogate objective is defined as “a function that is correlated to the true objective, but is less computationally demanding” (Gendreau and Potvin 2010, p. 51). We, thus, have to check whether our surrogate objective is indeed a good proxy, which we test with the help of a detailed terminal simulation. We aim to minimize the total number of loading stations passed by shipments before being tilted, because each loaded tray passing by a loading station postpones the loading of the items queuing there. What considerably complicates the problem is that each truck contains shipments for many outbound destinations and that, most probably, multiple trucks with overlapping processing time windows will compete for the same docks.

As outbound destinations are already fixed, the contribution \( w_{j,s} \) of each inbound truck \( j \) processed at a door of a specific inbound segment \( s \) to the objective value, i.e., the number of passed inbound segments, can be calculated upfront by weighting the number of shipments per destination with the number of segments passed between \( s \) and the respective outbound dock. The following example shows how to calculate these penalty values.

**Example:** Consider a single inbound truck \( j = 1 \) loaded with shipments for four different outbound destinations \( O_1, \ldots, O_4 \) as depicted in Figure 2. The penalty values for processing \( j \) at the four alternative inbound segments \( s = 1, \ldots, 4 \) are calculated as follows: \( w_{1,1} = 2 \cdot 3 + 1 \cdot 2 = 8 \), i.e., the three shipments for \( O_1 \) are instantaneously tilted in the successive outbound segment, the two shipments dedicated to \( O_3 \) pass loading stations 2, 3, and 4, and the single shipment for \( O_4 \) passes two stations, \( w_{1,2} = 3 \cdot 3 + 2 \cdot 2 + 1 = 14, w_{1,3} = 3 \cdot 2 + 2 \cdot 1 = 8 \), and \( w_{1,4} = 3 \cdot 1 + 1 \cdot 3 = 6 \).

Given these weights, our truck scheduling problem aims to assign each truck a door of an inbound segment and a processing interval, which neither violates the truck’s given time window nor overlaps with other unloading intervals of trucks assigned to the same door. Our optimization objective is to minimize the total penalty value.

Note that the shipments arriving on a truck typically have already been identified and registered at the previous step in the distribution chain where the respective truck has been loaded, so that the load information can be announced to the successive terminal prior to a truck’s arrival. In such a setting (typical for the postal service industry), precise knowledge on the inbound-outbound relations of shipments is a realistic premise. Furthermore, note that the knowledge on truck loads also eases the forecast of each truck’s processing time, so that again deterministic values do not seem to be a very limiting assumption. However, in our computational study we investigate the robustness of our problem, if forecast errors occur. Without loss of generality, we, furthermore, assume that all parameters have integer values.

Given these explanations, our truck scheduling problem in the postal service industry (denoted as TSPS) can be formally defined as follows. A schedule \( \Theta \) consists of a set of triples \((j, g, C_j)\) \( \in \Theta \) defining completion time \( C_j \in \mathbb{N} \) of processing inbound truck \( j \in J \) at gate \( g \in G \). We say schedule \( \Theta \) is feasible if

1. for each \( j \in J \) there is exactly one triple \((j, g, C_j)\) \( \in \Theta \), that is each truck is scheduled exactly once;
2. for each \( j \in J \) we have \( a_j + p_j \leq C_j \leq d_j \), that is each truck is processed in its allowed time window;
3. for each pair of inbound trucks \( i, j \in J, i \neq j \), with \((i, g, C_i) \in \Theta \) and \((j, g', C_j) \in \Theta \) we have \( g \neq g', C_i < C_i - p_i + 1, \) or \( C_i < C_i - p_i + 1, \) that is trucks are assigned to different gates or the trucks’ unloading intervals do not overlap.

Given \( \gamma(g) \in S \) defining the inbound segment dock \( g \) belongs to, TSPS seeks a feasible schedule \( \Theta \) minimizing \( Z(\Theta) = \sum_{(j, g, C, \gamma) \in \Theta} w_{j, \gamma g} \).

The resulting TSPS equals a parallel machine scheduling problem with time windows and a machine-dependent weight. The aim is to timely execute all jobs, such that the sum of priority weights of the realized job-machine-assignments is minimized. In-depth
Finding a feasible solution to TSPS is strongly NP-complete for \(|G| \geq 1\).

The complexity status of TSPS, i.e., even finding a feasible solution turns out as a complex task, directly follows from the TSPS being a generalization of the single machine scheduling problem with given arrival times and deadlines, which is well known to be strongly NP-complete (Garey and Johnson 1979).

2.2. Mathematical Model

Applying the notation summarized in Table 1, TSPS can be represented as a linear mixed-integer program [TSPS-MIP] consisting of objective function (1) and constraints (2) to (8). Note that this model eliminates the symmetry resulting from the identical docks in each inbound segment. We do not explicitly assign a job to a specific door but to some door of an inbound segment.

\[
\text{Minimize } \quad Z(C, X) = \sum_{s \in S} \sum_{i \in J} w_{j,s} \cdot \left( \sum_{i \in \{0\}} x_{i,j}^s \right) \quad (1)
\]

subject to

\[
\sum_{s \in S} \sum_{i \in \{0\}} x_{i,j}^s = 1 \quad \forall j \in J, \quad (2)
\]

\[
\sum_{i \in \{j\}} x_{0,i}^s \leq |G_s| \quad \forall s \in S, \quad (3)
\]

\[
\sum_{i \in \{j\} \cup \{n+1\}} x_{i,i}^s = \sum_{i \in \{j\}} x_{i,i}^s \quad \forall j \in J; s \in S, \quad (4)
\]

\[C_i \geq C_j + p_j - B \cdot (1 - x_{i,j}^s) \quad \forall j \in J; i \in J \cup \{0\}; s \in S, \quad (5)\]

\[a_j + p_j \leq C_j \leq d_j \quad \forall j \in J, \quad (6)\]

\[C_0 = 0, \quad (7)\]

\[x_{i,j}^s \in \{0,1\} \quad \forall i, j \in J \cup \{0, n + 1\}; s \in S. \quad (8)\]

Objective function (1) seeks to minimize the total penalty value. Constraints (2) ensure that each inbound truck is processed at some dock. Inequalities (3) guarantee that for each inbound segment the number of truck sequences composed does not exceed the number of available docks. Constraints (4) ensure that the sequences of inbound trucks are well defined. Constraints (5)–(7) define the completion time \(C_j\) for each inbound truck \(j\). It has to be later than that of its predecessor truck \(5\), range in between arrival time and deadline \(6\), and virtual truck 0 starts at the beginning of the planning horizon \(7\). Note that \(B = \max_{j \in J} \{d_j\}\) is sufficiently large. Finally, constraints \((8)\) represent the binary integrality requirement of 0-1 variables.

Unfortunately, the complexity status of TSPS hinders instances of real-world size to be solvable to optimality by simply feeding [TSPS-MIP] into an off-the-shelf solver. Therefore, we introduce different decomposition approaches for solving TSPS heuristically.

3. Decomposition Procedures for TSPS

The first step we take to solve our TSPS heuristically is to reduce the problem to an interval scheduling problem. It is a well-known result that any time-indexed scheduling problem can be formulated as a discrete interval scheduling problem (e.g., Kolen et al. 2007). However, we do not consider all feasible start times as a potential start of an additional processing interval, but generate just a subset of alternative intervals per truck. We do not allow each inbound truck to be processed anytime during its time window, but generate a finite set \(M_j\) of alternative fixed processing intervals, which we also denote as modes of execution. A mode \(m \in M_j\) is defined by its fixed completion time \(C_{jm} \in [a_j + p_j; d_j]\) \(\forall j \in J; m \in M_j\). This way, the problem reduces to selecting one mode of execution for each truck among all \(M_j\) alternatives and the interval scheduling version of our TSPS becomes easier to solve as fewer modes are generated. On the other hand, fewer modes enlarge the risk that actually optimal intervals are not contained in \(M_j\), optimal TSPS solutions are missed, and the optimality gap increases. We further investigate this tradeoff in our computational study (Section 4).

We apply a very basic approach for generating the modes of execution and simply draw a predefined number \(|M_j| = \mu\) of modes from interval \([a_j + p_j; d_j]\) randomly (with uniform distribution) for each truck \(j\). Note that experiments with more complicated generation schemes have not led to noteworthy improvement,
so we decided to keep this step as simple as possible. Once these intervals are available, the following three decisions remain:

- For each truck \( j \) exactly one mode of execution has to be selected from \( M_j \).
- The total set of trucks has to be partitioned into subsets of trucks that are processed at the same door.
- Any of these subsets, which we also denote as a chain of intervals, has to be assigned to a separate door of a specific segment.

These decisions can be distributed among an off-the-shelf solver and tailored solution algorithms in a variety of setups. We investigate three different decomposition approaches denoted as D1, D2, and D3 that stepwise decrease the decision tasks delegated to the standard solver (light grey tasks in Figure 3). The exact division of labor of our three decomposition approaches is summarized in Figure 3 and elaborated in the following sections. Note that we have also experimented with a fourth approach, which does not apply a standard solver and solves the complete interval scheduling problem with dynamic programming and beam search. This approach, however, was non-competitive.

### 3.1. D1: An Integer Program for Solving the Interval Scheduling Version of TSPS

Once finite sets \( M_j \) of processing intervals are available for each truck \( j \), preprocessing parameters \( \lambda_{j,m,t} \), which define whether \( \lambda_{j,m,t} = 1 \) or not \( \lambda_{j,m,t} = 0 \) truck \( j \) executed in mode \( m \) is active at time \( t \), is straightforward. This information allows to easily check whether two modes of different trucks can be assigned the same dock door. The assignment of a truck \( j \) executed in mode \( m \) to dock door \( g \) is represented by binary variables \( y_{j,g,m} = 1 \) (0 otherwise) and parameters \( \gamma(g) \) denote the inbound segment of dock \( g \). Given this notation, solving the interval scheduling version of TSPS (denoted as iTSPS), which determines all three remaining decision tasks in a holistic approach, can be formulated as integer program \([iTSPS-IP]\) consisting of objective function (9) and constraints (10) to (12).

\[
\begin{align*}
\text{Minimize } & \quad Z(Y) = \sum_{j \in J} \sum_{g \in G} \sum_{m \in M_j} w_{j,g} \cdot y_{j,g,m} \\
\text{subject to } & \quad \sum_{g \in G} y_{j,g,m} = 1 \quad \forall j \in J, \\
& \quad \sum_{j \in J} y_{j,g,m} \cdot \lambda_{j,m,t} \leq 1 \quad \forall t = 1 \ldots T, \ g \in G, \\
& \quad y_{j,g,m} \in \{0,1\} \quad \forall j \in J, \ g \in G, \ m \in M_j.
\end{align*}
\]

Objective function (9) minimizes the total penalty value. Constraints (10) ensure that exactly one mode of execution is selected per truck and (11) avoids that modes with time overlap are assigned the same door. Finally, the domain of the binary variables is defined by (12).

**Theorem 2.** Finding a feasible solution to iTSPS is strongly NP-complete.

**Proof.** See the online appendix. \( \square \)

### 3.2. D2: The Feasibility Version of iTSPS Plus Assignment Problem

Another possible decomposition is to apply an off-the-shelf solver merely for the selection of an interval per truck and the clustering of trucks that are served at the same door. The resulting chains of nonoverlapping intervals can finally be assigned to specific dock doors with the help of the linear assignment problem. The former—the solver part—consists of solving the aforementioned iTSPS (see Section 3.1) without an objective function. We denote this feasibility version as the iTSPS and its formulation as an integer program as \([iTSPS-IP]\), which is defined by constraints (10) to (12). Although doing without objective function facilitates the task for the standard solver, the iTSPS is still strongly NP-complete, which has been shown by Theorem 2.

**Figure 3.** Overview of the Decision Tasks and Our Decomposition Procedures
For the given chains of selected intervals, the decision at which specific door (and, thus, in which inbound segment) each chain should be processed can be solved to optimality in polynomial time by the linear assignment problem. On one hand, we have \( |TSPS-IP| \) and, on the other hand, we have the Boysen, Fedtke, and Weidinger:

associated with assigning a chain 

that trucks assigned to the same dock are processed in increasing order of start times. Accordingly, we consider trucks to be numbered in nondecreasing start times in a single gate.

It is easy to see that the selected intervals can somehow be assigned to docks, such that a feasible solution without overlap exists. Such a selection of intervals can be found by solving the aforementioned TSPS (see Section 3.2). From the resulting solution we only fix the set of intervals selected, but neglect the additional information on the assignment of intervals to chains. Therefore, the remaining decision task is to assign the single processing interval with given completion time \( C_j \) that has been fixed for each truck to a specific dock door, which also determines the chain of intervals at each single door. We denote this remaining decision task as the fixed-interval-TSPS that has the following complexity status.

Theorem 3. The fixed-interval-TSPS is strongly NP-hard.

Proof. See the online appendix. □

For solving the fixed-interval TSPS we now describe a dynamic programming (DP) scheme, which is then extended to a beam search heuristic. It is easy to see that trucks assigned to the same dock are processed in increasing order of start times. Accordingly, we consider trucks to be numbered in nondecreasing start times \( C_j - p_j \) in the following.

Our DP is subdivided into \( n + 1 \) stages where stage \( j = 0, 1, \ldots, n \) decides on the assignment of truck \( j \)'s interval to the dock doors. Additionally, we have stage zero as the start stage. Each stage \( j \) contains states \((j, t_1, \ldots, t_{|G|})\) where \( t_g \) defines the point in time when door \( g \) is available again. We consider a transition from one state to another when both states of successive stages differ only at a single dock and the availability time set by the last truck of this gate leaves enough time for processing the current truck at the same dock without overlap. Formally, there is a transition from \((j-1, t_1, \ldots, t_{|G|})\) to \((j, t'_1, \ldots, t'_{|G|})\) if and only if there is a single gate \( g' \in G \) such that

- \( t'_g \leq C_j - p_j \), and
- \( t'_g = t_g \) for each \( p \in G \setminus \{g'\} \).

Let \( T_{j, t_1, \ldots, t_{|G|}} \) be the set of states from which a transition to state \((j, t_1, \ldots, t_{|G|})\) exists, then the basic Bellman recursion calculating the penalty value \( W(j, t_1, \ldots, t_{|G|}) \) for state \((j, t_1, \ldots, t_{|G|})\) is defined as follows:

\[
W(j, t_1, \ldots, t_{|G|}) = \min_{(j-1, t'_1, \ldots, t'_{|G|}) \epsilon T_{j-1, t_1, \ldots, t_{|G|}}} \{ W(j-1, t'_1, \ldots, t'_{|G|}) + w_{j, t_{|G|}} \},
\]

with \( g' \) being the gate that the interval of the current stage's truck is assigned to. All states of the final stage, which can be reached by a sequence of transitions starting at the initial state \((0, 0, \ldots, 0)\) of stage \( j = 0 \), represent a feasible schedule. Among these, one having a minimum total penalty value \( W(n, t_1, \ldots, t_{|G|}) \) constitutes an optimal schedule.

The number of states is in \( \Theta(n^{|G|}) \), the number of transitions is in \( \Theta(|G| \cdot n^{|G|}) \), and as the calculations per transition are in \( \Theta(1) \), the fixed-interval TSPS can be solved to optimality in \( \Theta(|G| \cdot n^{|G|}) \).

To break the symmetry of the problem and to speed up DP (on average, but without altering the aforementioned worst-case runtime complexity), we introduce the following two extensions:

- First, we can exploit the ordering of intervals by nondecreasing start times to reduce the redundancy of states. We need not differentiate otherwise identical states of the same stage \( j \), which only vary in availability times prior to the current start \( C_j - p_j \). The current interval starts at its given start time anyway, so that we only need to know whether a dock is free but not the exact time when processing has ended. Thus, we can alter any state \((j, t_1, \ldots, t_{|G|})\) generated according to the previous description to \((j, \max\{t_1, C_j - p_j\}, \ldots, \max\{t_{|G|}, C_j - p_j\})\), which may reduce the number of states per stage.

- Furthermore, we break the symmetry resulting from the fact that we do not have to differentiate dock doors belonging to the same segment. To do so, we have to slightly adapt the states and their interpretation. A state \((j, t_1, \ldots, t_{|G|})\) now differentiates the assignment of doors to specific segments. This way, an ordering of each subset of gates belonging to the same segment \( s \) according to nondecreasing availability times \( t_{gs} \) allows us to eliminate redundant states that only represent permutations of gates. The reordering of the state information requires
a slight reinterpretation; availability time $t_g$ of a state
no longer refers to a specific door $g$ of segment $s$ (e.g.,
with number $g$), but to the door of segment $s$, which is
the $g$th becoming available again.

In spite of these alterations, the state space of DP
still grows exponentially. However, the proposed DP
scheme can also be used as the basis for a beam search
( BS) heuristic (see Lowerre 1976). Applying BS requires
the setting of a single steering parameter BW (the beam
width), which constrains the number of states that are
further explored in each stage to the BW most promis-
ing ones. For selecting the BW best states per stage we
simply rank them according to their current solution
value $W(j, t_j, \ldots, t_{|G|})$.

4. Computational Study

Our computational study has to answer quite a few
research questions. First, we explore the computa-
tional performance of our alternative decomposition
approaches when solving TSPS (Section 4.2). This way,
we identify the best performing heuristic of our tool
set. Unfortunately, no established testbed for TSPS
exists, so that we begin with a description on how
our test instances have been generated (Section 4.1).
Furthermore, we aim to answer whether our TSPS
objective of minimizing the passed loading stations
is indeed a suited surrogate objective. To do so, we
apply a terminal simulation, whose setup is described
in Section 4.3. Then, we investigate managerial aspects
(Section 4.4), i.e., the impact of additional loading sta-
tions and the heterogeneity of truckloads on sortation
performance as well as the robustness of truck schedul-
ing solutions in case of forecast errors.

4.1. Instance Generation

We have generated two differently sized data sets. The
larger instances are of real-world size and consider 50
inbound gates; the smaller data set contains instances
with 10 inbound gates. The input parameters specified
in Table 2 are applied to initialize our data generator,
which generates 10 instances for each parameter com-
bination as follows.

First, we derive a reference truck with a total truck-
load of exactly $v_j = 3,000$ parcels, which are randomly
distributed among 10 destinations. We draw 10 uni-
formly distributed numbers from interval $[0;1]$ and
proportionally assign the $v_j = 3,000$ shipments to the
10 destinations (as far as rounding differences allow).
All other trucks are derived from the reference truck
by copying the truckload and executing $\psi$ swaps each
adding $r \cdot 50$ shipments to a randomly selected destina-
tion for all swaps $r = 1, \ldots, \psi$. The resulting shipments
per destination are proportionally scaled down to $v_j = 
3,000$ parcels. This way, varying parameter $\psi$ allows
us to change the level of truckload heterogeneity, such
that an increasing number of swaps ensures a higher
deviation from the reference truck. Processing times
$p_j$ are standardized to $v_j/100$, which equals a typical
processing time of 50 minutes.

We aim at feasible TSPS instances, for which feasible
schedules with regard to the time windows and avail-
able docks are obtainable. Therefore, we first derive
a random no-wait truck schedule and generate the
time windows around the respective processing inter-
vals. To do so, we apply a random sequence of trucks,
successively assign them to the next gate available,
and fix preliminary start times $\eta_j$ to the earliest feasi-
ble start time at the chosen gate. Once all trucks are
assigned, we randomly determine each truck’s time
window length and its relative position within by ran-
don larger instances to build a hierarchy of solutions.

### Table 2. Parameters for Instance Generation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Small</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of gates $</td>
<td>G</td>
<td>$</td>
</tr>
<tr>
<td>Number of trucks $n$ (25, 50)</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Number of destinations $d$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of parcels per truck $v_j$</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>Maximum size of time window $\Omega_{max}$</td>
<td>(0.5, 2, 4)</td>
<td>(5, 50, 90)</td>
</tr>
<tr>
<td>Number of swaps $\psi$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The results for the small test instances are summarized in Table 3. Here, we list the fraction of instances where the respective decomposition procedure was able to find a feasible solution. Recall that all our instances are generated such that they have a feasible solution. Furthermore, we report the average optimality gap and the solution time for our three decomposition approaches D1 to D3 for all those instances XPress could solve to optimality. Within a given time frame of 300 CPU-seconds our standard solver solved 47.4% out of 540 small test instances to proven optimality. The beam search procedure applied within decomposition procedures D3 has been executed with a beam width of BW = 100.

The results indicate that all three decomposition approaches seem well suited for efficiently solving TSPS. Especially if $|M_i| = 10$ alternative modes of execution are drawn for each truck, any of them is able to find a feasible solution for all instances except for one. While the solution time is rather short for all three approaches, D1 has the lowest optimality gap and solves 237 of 256 instances (with $|M_i| = 10$) to proven optimality. The two competitors D2 and D3 are a bit faster, but produce a slightly larger optimality gap.

A better distinction between our decomposition procedures can be made by analyzing the results for the large instances (see Table 4). These instances cannot be solved to proven optimality, so that we measure the gap to the best solution found among XPress and our three heuristics instead. These results indicate that all our decomposition approaches outperform the standard solver. Especially with $|M_i| = 10$ they are able to find considerably more feasible solutions and produce a lower average gap. Among our decomposition approaches D1 is the most successful. It finds the best solutions and produces the lowest gap. D2 and D3 are faster but produce a larger gap, so they might be an appropriate choice if time is a pressing concern or instances become even larger.

The previous results indicate that the performance of our decomposition approaches depends on the right number $|M_i|$ of modes generated for each truck. The impact of the number of modes on the solution quality is detailed in Figure 4 for the 90 largest test instances with $|S| = 5$ and $n = 200$. The left panel shows the fraction of instances for which a feasible solution could be found. The next two panels highlight the increase of solution time if more modes are generated. The

### Table 3. Computational Performance for the Small Data Set

| $n$ | $|S|$ | $|M_i|$ | XPress | D1 | D2 | D3 |
|-----|-----|------|-------|----|----|----|
| 25  | 2   | 5    | 86.7%/41.55 | 88.9%/46.15 | 88.9%/0.15 | 88.9%/0.17 |
| 25  | 2   | 10   | 86.7%/41.55 | 100.0%/0.27 | 100.0%/0.26 | 100.0%/0.28 |
| 25  | 3   | 5    | 33.3%/199.69 | 92.2%/1.19 | 92.2%/1.15 | 92.2%/1.18 |
| 25  | 3   | 10   | 33.3%/199.69 | 100.0%/0.32 | 100.0%/0.26 | 100.0%/0.29 |
| 25  | 5   | 5    | 33.3%/199.61 | 92.2%/0.21 | 92.2%/0.15 | 92.2%/0.19 |
| 25  | 5   | 10   | 33.3%/199.61 | 100.0%/0.33 | 100.0%/0.26 | 100.0%/0.29 |
| 50  | 2   | 5    | 64.4%/112.61 | 72.2%/0.61 | 72.2%/0.42 | 72.2%/0.46 |
| 50  | 2   | 10   | 64.4%/112.61 | 100.0%/0.87 | 100.0%/0.72 | 100.0%/0.78 |
| 50  | 3   | 5    | 33.3%/199.76 | 70.1%/1.09 | 70.0%/0.42 | 70.0%/0.46 |
| 50  | 3   | 10   | 33.3%/199.76 | 98.9%/2.34 | 98.9%/2.02 | 98.9%/2.09 |
| 50  | 5   | 5    | 33.3%/199.82 | 67.8%/6.19 | 67.8%/3.61 | 67.8%/3.71 |
| 50  | 5   | 10   | 33.3%/199.82 | 100.0%/2.93 | 100.0%/0.74 | 100.0%/0.83 |
| Total: | 5 | 47.4%/-158.84 | 80.6%/0.0% | 80.6%/0.28 | 80.6%/0.32 |
| Total: | 10 | 47.4%/-158.84 | 99.8%/0.0% | 99.8%/0.49 | 99.8%/0.54 |

Note: Optimal (XPress) and feasible (D1 to D3) solutions in %/average gap to optimum/CPU seconds of solved instances.

### Table 4. Computational Performance for the Large Data Set

| $n$ | $|S|$ | $|M_i|$ | XPress | D1 | D2 | D3 |
|-----|-----|------|-------|----|----|----|
| 200 | 2   | 5    | 70.0%/172.57 | 66.7%/13.47 | 66.7%/11.17 | 66.7%/2.12 |
| 200 | 2   | 10   | 70.0%/172.57 | 81.1%/27.14 | 81.1%/6.2%/0.33 | 81.1%/1.5%/1.67 |
| 200 | 3   | 5    | 68.9%/111.07 | 66.7%/16.25 | 66.7%/8.3%/11.6 | 66.7%/5.0%/12.64 |
| 200 | 3   | 10   | 68.9%/111.07 | 85.6%/28.72 | 85.6%/6.3%/0.32 | 85.6%/3.6%/1.97 |
| 200 | 5   | 5    | 67.8%/13.2%/201.37 | 66.7%/3.41 | 66.7%/8.6%/12 | 66.7%/5.5%/13.42 |
| 200 | 5   | 10   | 67.8%/13.2%/201.37 | 92.2%/29.98 | 92.2%/6.1%/0.32 | 92.2%/3.8%/2.53 |
| Total | 5 | 68.9%/95.9%/191.67 | 66.7%/21.37 | 66.7%/7.9%/11.59 | 66.7%/4.2%/12.74 |
| Total | 10 | 68.9%/95.9%/191.67 | 86.3%/28.61 | 86.3%/6.2%/0.33 | 86.3%/3.0%/2.06 |

Note: Feasible solutions found in %/gap to best solution/CPU seconds of solved instances.
right panel shows the average solution time over all instances and the middle panel excludes all nonsolved instances. As expected, more modes increase the fraction of feasible solutions. However, we have no such clear coherency with regard to solution time. Apparently, just a few modes reduce the input data but increase the combinatorial effort for evaluating all combinations. The results indicate that $|M_j| = 10$ seems to be a good choice to balance both effects. Consequently, for all subsequent computational tests we apply decomposition approach D1 with $|M_j| = 10$ modes.

### 4.3. Setup of Simulation Study

Our TSPS solves a surrogate objective, i.e., minimizing the number of passed inbound segments, so that the “true” impact of a truck schedule derived with TSPS can only be evaluated with a simulation study. In the following we describe the setup of our terminal simulation.

The aforementioned hub in Friedewald serves as the prototype of our terminal simulation. Around the terminal building (dimensions $216 \times 112$ m) there are 50 inbound doors. The central sorter consists of 766 tilt trays. Each of these trays is 0.85 m long and the conveyor moves with 2.5 m/s, so that (about) three trays pass by each loading station per second. We place the alternating in- and outbound sections equidistant around the conveyor and distribute the inbound and outbound docks equally among the given segments (see Section 4.1). The outbound destinations are randomly assigned to outbound docks. Furthermore, we presuppose $n = 200$ inbound trucks, $|S| = 5$ segments, and $d = 100$ outbound destinations as default values, if not explicitly stated otherwise.

The inbound stream of parcels at any loading station is generated as follows. First, we shuffle all given shipments per destination and generate a randomized unloading sequence for each truck. We assume that any of these sequences is unloaded with constant velocity during a truck’s docking time, so that in any time unit a truck is processed, another set of $\lfloor v_j/p_j \rfloor$ parcels (according to truck $j$’s unloading sequence) arrives at the respective inbound station. Note that rounding deviations are considered, so that in the last processing period all remaining shipments are unloaded. At a loading station, the arriving sets of all trucks currently docked at the gates of the respective inbound segment need to be merged. To emulate the zipper-like process we apply the well-known Monden algorithm, which is originally applied to equally distribute copies of different products over a production sequence (see Monden 2012). Each truck receives an ideal target rate for a leveled release of shipments from this truck per period. Then, we successively launch the next shipment in a greedy manner and choose the truck that increases the (squared) deviation from the target rates the least. The resulting (leveled) sequence is, then, appended to the current queue at a loading station. At the head of the queue, the next shipment is loaded whenever a free tray passes by (first-empty-tray rule, see Fedtke and Boysen 2014).

This way, we emulate the conveyor rotation with its loading and tilting operations in a discrete time simulation. To evaluate the quality of a truck schedule (and its subsequent simulation run), we record the following four performance measures:

- **Starting at the beginning of the planning horizon** the makespan is reached once the last parcel is tilted into its designated chute. Reducing the makespan frees all resources of a terminal as early as possible, so that either processing the subsequent wave of shipments can be initiated earlier or the closing time is reached sooner.

- The mean completion time averages the time spans from the arrival time $a_j$ of the respective truck to the tilting of each shipment. Given tight delivery times for each single parcel, reducing the mean completion time increases the fraction of deliveries on time.

- The in-terminal time calculates the mean time span each shipment remains in the terminal from the start of unloading the respective truck up to the tilting of each shipment. Minimizing the in-terminal time relieves all

### Figure 4. Influence of the Number of Modes $|M_j|$ on the Solution Quality of D1

| $|M_j|$ | Percentage of feasible instances | Avg. solution time over all feasible instances | Avg. solution time over all instances |
|-------|---------------------------------|-----------------------------------------|-----------------------------------|
| 0     | 0                               | 1.36                                    | 1.36                              |
| 1     | 33.3                            | 31.99                                   | 31.99                             |
| 2     | 66.7                            | 47.89                                   | 34.41                             |
| 5     | 92.2                            | 55.37                                   | 40.80                             |
| 7     | 100.0                           | 51.06                                   | 39.52                             |
| 8     | 68.9                            | 44.21                                   | 61.10                             |
| 10    | 100.0                           | 61.10                                   | 44.21                             |
| 15    | 66.7                            | 12.60                                   | 12.60                             |
| 20    | 100.0                           | 29.98                                   | 29.98                             |

Downloaded from informs.org by [130.102.82.80] on 21 February 2017, at 19:52. For personal use only, all rights reserved.
terminal resources, so that capacity is freed for additional shipments not reaching the sorting system via trucks but, for instance, from an interconnected warehouse or a storage area in the same terminal.

- The sorter time measures the average time span each shipment remains on the sorter. Reducing this measure frees sorting capacity for additional shipments, if the sorter is the unique bottleneck resource.

Note that each performance measure addresses a specific aspect of a terminal’s performance, so that it seems advantageous to not just report a single measure. An isolated measure may fail to properly assess performance. The makespan, for instance, is barely influenceable whenever a single truck with an exceptionally late arrival time arises.

### 4.4. Managerial Aspects

In the first test, we evaluate whether our surrogate objective applied by TSPS, i.e., minimizing the number of loading stations passed by all shipments, is indeed a good proxy for an efficient sortation process. Instead of directly tackling an actual performance measure, e.g., minimizing the makespan of processing all parcels, a surrogate objective is required, because the exact arrival sequence of parcels merged among all inbound trucks cannot be anticipated. To prove the suitability of our choice, we solve each data instance of TSPS with our decomposition approach D1 (with \(|M_j| = 10\) modes) and also generate a feasible random solution for the same instance. The random solution is the one we have initially produced during instance generation. Recall that already finding a feasible solution to TSPS is strongly NP-complete, so that simple decision rules like FCFS (first-come first-served), which are often applied in practical settings to solve truck scheduling problems (Boysen, Briskorn, and Tschöke 2013), cannot ensure feasible solutions. Therefore, we apply our random feasible solutions, which allows us to focus on sortation performance and exclude additional aspects like delayed trucks. We feed both solutions into our terminal simulation to compare the resulting performance measures of both solutions.

In preliminary tests two aspects have been shown to greatly impact the results of this comparison: the tightness of the trucks’ time windows and the heterogeneity of the truckloads. Thus, we generate large test instances as described in Section 4.3 with small \(\Omega_{\text{max}} = 0.5\) and large \(\Omega_{\text{max}} = 2\) time windows and \(\psi \in \{0, 5, 15, 30, 45, 60, 75, 90\}\) swaps. To classify the resulting 160 instances according to their heterogeneity we calculate the heterogeneity measure Var, which equals the weighted sum of variances of the number of shipments per truck over all segments

\[
\text{Var} = \sum_{s \in S} \frac{v_s}{v} \cdot \sum_{j \in J} \left( \frac{v_s}{|J|} - v_{s,j} \right)^2,
\]

with \(v, v_s, v_{s,j}\) defining the total number of shipments, the number of shipments for segment \(s\), and the number of shipments for \(s\) loaded on truck \(j\), respectively.

Figure 5 displays the simulation results of our comparison between optimized and random truck schedules. For all four performance measures, the improvement of the optimized truck schedules over the random solutions is related to the level of heterogeneity. Note that an increasing value of Var indicates more heterogeneous truckloads. The results suggest the following findings:

- In line with the results obtained by Gue (1999) for less-than-truckload freight terminals, the advantage of optimized solutions increases with an increasing level of truckload heterogeneity. If all trucks have similar loads and equal our reference truck, then TPS aims to assign all trucks to the same inbound segment. Only the time windows may enforce a variational assignment. This leads to long queues of shipments in just a few inbound segments, so that optimized truck schedules are barely better or even worse than random solutions.

- The makespan and (less distinct) the mean completion time suffer, if large time windows (tw) are available. In these cases, TPS tends to postpone trucks, so that each of them can be processed at a preferred inbound segment. Random solutions do not delay trucks and always schedule a truck once a dock is available. The capacity related measures, i.e., in-terminal and sorter time, profit from optimized schedules irrespective of the tightness of time windows.

It can be concluded that parcel distribution centers profit from optimizing their truck schedules according to TSPS, if the truckloads arriving at a terminal are not too homogeneous and time windows are not too long. Note that the leftmost data points in each graph of Figure 5 represent the case where all inbound truckloads are identical. This is certainly a purely theoretical setting, which never occurs in the real world. Our Friedewald terminal, for instance, processes inbound trucks for many different supply chains ranging from online retailers for books and hunting equipment up to luggage services of Germany’s largest railway operator. All these supply chains focus on different customer groups, so that the heterogeneity level we observed falls in the upper range, i.e., \(\text{Var} \geq 7.5 \cdot 10^6\). In these relevant cases, sortation performance can be increased by up to 10% without a costly investment in technical equipment just by optimizing the processing of inbound trucks.

Sortation performance is also impacted by the number of inbound segments. The more segments available, the greater the flexibility for TSPS to reduce the number of passed segments. Figure 6 depicts the simulation results for optimized truck schedules in dependency of the number of segments \(|S|\). We report the
Improvement of our four performance measures over the single-segment solution. It can be concluded that sortation performance greatly profits from additional segments, but their positive impact quickly diminishes. This is good news for terminal managers. Large performance gains can already be realized with a moderate investment (e.g., for conveyors, switches, and camera systems) into two or three additional inbound segments. Among our four performance measures, the sorter time profits least from additional inbound segments. On the positive side, additional segments tend to decrease travel distance on the sorter once a parcel is loaded. Multiple segments, however, also increase the probability that some of them need to be passed, so that more shipments get blocked at their loading stations. The latter effect influences the other performance measures as well, but they greatly profit from the decreasing queue lengths, if additional inbound segments are available.

Finally, we explore the impact of forecast errors on the performance of TSPS. If the exact loads arriving on each inbound truck are not announced at a terminal in a timely fashion, the number of arriving shipments per inbound-outbound relation have to be forecasted. Forecasts are bound to errors and from a practitioner’s point of view it is important to know whether the plans gained with TSPS (obtained with inaccurate data) are still reliable. We first solve each of the large instances also applied for deriving Figure 5 (with Var \( \geq 7.5 \cdot 10^6 \)) with decomposition approach D1 and \( |M_j| = 10 \) modes. The resulting schedule (based on the inaccurate data) is then handed over to the simulation where we emulate increasing forecast errors by introducing probability \( \gamma \), which defines the percentage of parcels with a wrongly anticipated outbound destination. For each parcel of the inbound queue we randomly determine (with given probability \( \gamma \)) whether this parcel has a false destination. If so, we randomly alter its destination (by an equally distributed choice among all other destinations). The resulting performance measures of the simulation run...
are compared to that resulting from a randomly generated truck schedule. Figure 7 shows that with increasing forecast errors (indicated by an increasing \( \gamma \) value) the gap between the optimized TSPS and the randomized schedule becomes smaller. However, even if forecasts are unreliable, i.e., 30% of all addressees are wrongly anticipated, there are still considerable performance gains of more than 6%. Note that there exists other sources of forecast errors, i.e., the trucks’ time windows. However, exploring their impact is left to future research.

5. Conclusion

This paper investigates the scheduling of inbound trucks at hub terminals of the postal service industry. A processing interval and a dock door is assigned to each incoming truck loaded with mail, such that the performance of the fully-automated sortation conveyor inside the terminal is increased. On the main conveyor, connections to inbound and outbound dock segments are arranged in an alternating manner, so that inbound trucks are best processed at segments which allow quick access to each parcel’s designated outbound dock and do not occupy excessive conveyor capacity. We formalize the resulting truck scheduling problem and suggest three different decomposition procedures based on integer programs and dynamic programming. In a comprehensive computational study, we show the superiority of an interval scheduling approach solved with an off-the-shelf solver.

The derived solution procedure is, furthermore, applied to managerial questions and the following key insights are gained from these computational tests:

- A sophisticated optimization approach outperforms random solutions. Note that a solution being random with regard to the inbound segments may, for instance, be obtained by the widely applied FCFS rule (Boysen, Briskorn, and Tschöke 2013), which only focuses on arrival times. Our simulation study indicates that for heterogeneous truckloads and tight time windows up to 10% sortation performance can be gained without an investment in technical equipment just by optimizing truck processing.

- The incremental benefit of additional inbound segments quickly diminishes as more loading segments are added. Thus, two, three, or four segments, which can often be observed in real-world terminals, lead to a considerable increase of sortation performance while keeping investment costs for conveyors, switches, and camera systems at a manageable level.

- If the data fed into the optimization approach is prone to forecast errors, the sortation performance deteriorates. However, even if confronted with fairly large forecast errors, our optimization approach still leads to far better results than random truck schedules.

Some terminals apply parallel conveyors arranged on top of each other, which cycle in opposite directions. In such a setting, parcels queuing at a loading station can be sent into either direction. Adapting our approach to such a setting and exploring the resulting performance gains seems to be a valuable task for future research. Furthermore, integrating robustness measures to immunize planned truck schedules against wrongly anticipated shipments or untimely truck arrivals constitute challenging future research tasks.

References


