Working Paper

Layout planning of sortation conveyors in parcel distribution centers

Stefan Fedtke and Nils Boysen
09/2012

Working Papers in Supply Chain Management
Friedrich-Schiller-University of Jena

Prof. Dr. Nils Boysen
Chair of Operations Management
School of Business and Economics
Friedrich-Schiller-University Jena
Carl-Zeiß-Str. 3, D-07743 Jena
Phone ++49 (0)3641 943100
e-Mail: nils.boysen@uni-jena.de

Prof. Dr. Armin Scholl
Chair of Management Science
School of Business and Economics
Friedrich-Schiller-University Jena
Carl-Zeiß-Str. 3, D-07743 Jena
Phone ++49 (0)3641 943170
e-Mail: armin.scholl@uni-jena.de

http://pubdb.wiwi.uni-jena.de
Layout planning of sortation conveyors in parcel distribution centers

Stefan Fedtke, Nils Boyesen

Friedrich-Schiller-Universität Jena, Lehrstuhl für ABWL/ Operations Management, Carl-Zeiß-Str. 3, D-07743 Jena, Germany, {stefan.fedtke,nils.boysen}@uni-jena.de

*Corresponding author, phone +49 3641 943-100.

Abstract
Fully automated sortation processes play a crucial role in modern distribution networks of the parcel service industry. In this context, the paper on hand systematically investigates different design alternatives of closed-loop tilt tray sortation conveyors. Especially, the number of loading stations for feeding parcels into a conveyor system and the relation to the subordinate decision problem of assigning inbound and outbound destinations to door doors are shown to considerably influence the throughput of a hub terminal. A novel destination assignment problem is formalized, suited solution procedures are presented and, by means of simulation, the resulting throughput is quantified, so that an experienced terminal manager can weigh the operational gains against the investment cost of different conveyor layouts.

Keywords: Logistics; Parcel service industry; Sortation conveyor; Layout planning; Door assignment

1 Introduction
In the wake of e-commerce, the parcel service industry became one of the fastest rising segments of logistics services (e.g., Garcia-Romeu-Martinez et al., 2007). A good part of this success is based on fast delivery processes, e.g., postal service provider DHL aims to deliver 80% of all shipments sent in Germany within 24 hours (Werners and Wülfling, 2010), provided at comparatively low costs. The latter aspect is especially enabled by the logistics networks being organized according to the hub-and-spoke paradigm. In these delivery networks, parcel distribution centers serve as hubs where multiple smaller shipments are consolidated to full-truckloads, so that economies of transportation are
gained. Compared to a point-to-point delivery, an additional transshipment within a hub slows down the distribution process. Thus, in order to not jeopardize the ambitious delivery targets rapid sortation processes are to be ensured within parcel distribution centers.

In modern parcel hubs, the sortation process is fully automated, i.e., by closed-loop tilt tray conveyor systems. In such a conveyor system, parcels being unloaded from an inbound trailer and isolated onto separate trays of the conveyor system circle through the distribution center until being tilted into the respective gravity chute serving the outbound truck dedicated to a parcel’s destination. Typically, these conveyor systems are the bottleneck of the transshipment process and consume a large part of the investment into a distribution center. For instance, at the airport hub in Leipzig (Germany), which is one of worldwide three airport hubs in DHL’s express logistics network, €70 million from a total investment of €300 million are allotted to the conveyor system. Business rival Hermes Logistik, Germany’s second largest parcel distributor, invested €45 million into its central hub at Friedewald including a sortation conveyor with a total length of 1.4 kilometer for €25 million. Thus, determining the layout of a sortation conveyor system when newly erecting or redimensioning a parcel distribution center is a very critical task.

1.1 Operations in a parcel distribution center

At a hub terminal, inbound trucks arrive delivering shipments either from other hubs, depots or directly from nearby customers, e.g., large e-commerce retailers. After being registered at the entry gate, a truck parks its trailer in a parking lot. Then, a special trailer truck picks up a trailer and moves it to its dedicated receiving dock. Here, a logistics worker opens the trailer, pulls a telescope conveyor directly in front of (later into) the trailer and successively unloads the parcels onto a belt conveyor. After unifying the loading segments of multiple inbound docks to a single conveyor line, parcels are automatically measured according to weight, size and addressee. If the receiver of a parcel was not recognized by the camera system, the parcel is discharged from the conveyor to one of multiple manual stations, where once identified – parcels are reloaded onto the conveyor. After identification, a switch system successively loads the parcels on free tilt trays of the main closed-loop sortation conveyor, so that a one-to-one mapping between parcels and trays is achieved and communicated to the background information system. On their trays, parcels travel along the conveyor until reaching their respective gravity chutes. Then, the tray is tilted and the parcel slides onto another telescope conveyor to be loaded into a trailer dedicated to the parcel’s destination. Once a trailer is fully loaded, a logistics worker closes the trailer and communicates this status to the information system. Finally, the trailer is picked up (including a potential interim stay at a parking lot) by an outbound truck to transport the trailer to its next destination in the distribution process. Figure 1 depicts a schematic layout of a parcel distribution center.

Note that the operational process described above is the one proceeding at the aforementioned central Hermes Hub in Friedewald. At other hub terminals some process steps might slightly deviate. For instance, at the DHL hub in Leipzig shipments arrive by airplane and some industrial towing vehicles deliver the parcels from the apron into
the terminal, where they are loaded on to the sorting conveyor. However, the general sortation procedure on the tilt tray conveyor system is the typical advancement in the parcel service industry.

1.2 Decision problems and literature review

From an overall network perspective, a suited structure of the distribution network, e.g., single- versus multi-stage hub-and-spoke networks, has to be identified and within the chosen structure location planning of hubs and depots has to executed (e.g., see the survey papers of Campbell, 1994; Klose and Drexl, 2005). The decision problems arising within a single node of the network are summarized by Figure 2.

When deciding on the layout of a distribution terminal, especially, the shape of the terminal (e.g., I, L, T or X-shaped, see Bartholdi and Gue, 2004), the number of inbound and outbound dock doors, and their arrangement around the perimeter of the terminal (see Stephan and Boysen, 2011) have to be determined. Also, the decision problem treated in this paper, the layout of the sortation conveyor, falls into this category. One
of the most important design aspects with regard to the performance of the conveyor system is whether only a single loading station or multiple ones should be utilized. With a single loading station (as is depicted in Figure 1), each tilt tray is only filled (at most) once per cycle of the conveyor system. If multiple loading stations are available, then a tray can repeatedly be loaded per cycle provided that the tray could intermediatively be emptied, i.e., by tilting the parcel to its respective outbound destination, before reaching a successive loading station. Whether or not such a repetitive loading of a tilt tray is possible heavily depends on the assignment of inbound trailers (and the parcels contained) to loading stations and outbound destinations to dock doors (and their associated chutes).

In the literature, the assignment of inbound and outbound destinations to dock doors, so that a rapid transshipment within a hub terminal is enabled, is known as the destination assignment problem (see Boysen and Fliedner, 2010). This decision problem assigns destinations to docks over a mid-term horizon, e.g., a month, so that all trucks serving a specific destination are processed at a unique dock door. Important contributions in the context of general cross docking terminals, where the movement of shipments inside a terminal is executed by forklifts, e.g., stem from Tsui and Chang (1990, 1992), Gue (1999), and Bozer and Carlo (2008). For fully-automated sortation processes with a closed-loop conveyor system and multiple loading stations, this problem has not been treated in literature. Only some related studies on destination assignment within parcel distribution centers exist where parcels are manually moved on wheeled pallets (Oh et al. 2006) or deviating conveyor layouts are applied (McWilliams et al., 2005; Werners and Wülfing, 2010; McWilliams 2010). Thus, for evaluating different design alternatives of a closed-loop sortation conveyor system we formalize a novel destination assignment problem.

Another subordinate decision problem is the truck scheduling problem. Here, the processing of individual inbound and outbound trucks at the dock doors is scheduled in detail. For operationally executing this problem in relation to the destination assignment problem three alternatives exist (see Bozer and Carlo, 2008; Boysen and Fliedner, 2010):

- With given inbound and outbound destinations as defined by the destination assignment problem, the truck scheduling problem reduces to determining the sequence of truck processing at any dock door. In the real-world, the sequencing decision is often not optimized but simply determined by a rule of thumb, e.g., first-come-first-served (FCFS). In addition to not requiring the solution of a sophisticated optimization problem, a fixed assignment of destinations has the advantage, that logistics workers can learn the topology of the docks to quickly address each shipment.

- Alternatively, the mid-term destination assignment problem can be skipped, so that assigning trucks to dock doors becomes part of the short-term truck scheduling problem. Clearly, this leaves more flexibility for quickly adopting to unforeseen events. On the negative side, such an advancement requires an information system, so that logistics workers can address a shipment among the continuously changing destinations of docks. A detailed review on truck scheduling is provided by Boysen and Fliedner (2010).
• As a compromise solution, only outbound destinations can be fixed over a longer planning horizon and inbound trucks are freely assigned to inbound dock doors during the short-term truck scheduling problem (see Bozer and Carlo, 2008).

We restrict our study to the first alternative and presuppose fixed inbound and outbound destinations, because this is the typical advancement in the parcel service industry. For instance, our two example hubs of DHL and Hermes are part of a multi-stage hub-and-spoke network where reliable time tables for the transport processes between hubs exist. Therefore, the managers of these terminals prefer a stable door layout (fixed over multiple months in order to ease process control) over additional flexibility offered by varying truck assignments.

Finally, with a given conveyor layout and a given arrival pattern of parcels, the flow of parcels through the sortation system is to be organized and controlled. There exist some studies, which derive analytical equations for predicting the throughput of a closed-loop conveyor system (e.g., see Bozer and Hsieh, 2004, 2005). However, our study evaluates more complicated conveyor layouts, e.g., with multiple loading station and multiple parallel conveyors on top of each other rotating in opposite directions. For such complex layouts no analytical equations exist, so that we apply a simulation study to finally evaluate the performance of specific conveyor settings.

1.3 Research question and paper structure

This paper evaluates different design alternatives of closed-loop tilt tray sortation conveyors applied in parcel distribution centers. Namely, we investigate the impact of multiple loading stations (and their effect on multiple tray loads per cycle), multiple parallel conveyors arranged on top of each other rotating unidirectional and in opposite directions. For finally exploring the resulting operational performance, i.e., the throughput, we formulate and solve the subordinate destination assignment problem with multiple loading stations for assigning inbound and outbound destinations to dock doors and simulate the resulting parcel flow through the terminal.

The remainder of the paper is structured as follows. Section 2 precisely defines the layout alternatives investigated. Then, Section 3 formalizes the subordinate destination assignment problem with multiple loading stations and presents suited solution procedures, which are tested with regard to their computational performance in a comprehensive computational study. The simulation study, which tests the layout alternatives of the conveyor system by integrating the subordinate destination assignment and emulating the resulting parcel flow through the terminal, is presented in Section 4. Finally, Section 5 concludes the paper.

2 Design alternatives for sortation conveyor systems

There exist different layout alternatives for closed-loop sortation conveyors, which are described in the following (and later on tested with regard to their impact on operational performance):
If only a single loading station is available (see Figure 1), then each tray can be loaded with a parcel (at most) once per cycle of the conveyor. Multiple tray loads are rendered possible if additional loading stations are available. Figure 3 depicts a terminal layout with two conveyor segments A and B, where each segment is assigned a set of inbound doors feeding a loading station and a set of outbound doors passed by the conveyor before reaching the next segment. With multiple segments, a loaded tray can be tilted and reloaded multiple times per cycle. This, however, heavily depends on the assignment of inbound and outbound destinations to dock doors (and their respective chutes). In the best case, each tray of a conveyor with $n$ segments can be loaded $n$ times per cycle, provided that each parcel is loaded and tilted in the same segment. However in the worst case, a parcel travels all along the conveyor to be tilted in the segment preceding its loading station, so that again only a single load per cycle is possible. It is the objective of our destination assignment problem described in Section 3 to ensure that each parcel quickly reaches its chute and multiple tray loads per cycle are enabled.

Another design alternative is to increase the capacity of a conveyor system by installing multiple parallel conveyors on top of each other. Typically, only two parallel conveyors are installed, but even three or more may be possible. A parallelization requires an additional switch at a loading station where for each parcel one of the parallel conveyors is chosen. A typical real-world decision rule is to select the conveyor where the next free tray can be reached.

With parallel systems, another design parameter to be determined is the flow direction of conveyors, which can either flow unidirectional or rotate in opposite directions. The latter case seems especially promising if parallel conveyors are combined with multiple loading stations. Then, the neighborhood of a loading station is increased and there exist more possibilities for tilting parcels in nearby segments. With opposing flow directions an alternative decision rule for determining a parcel’s conveyor is to select the next free tray on that conveyor, which provides the shorter path to a parcel’s designated chute.
The most obvious design parameter for increasing the throughput of a conveyor is to vary the number of trays. Over a short-term horizon another lever is the speed of a conveyor. However, in the real-world the tolerance for accelerating the conveyor is low, because the risk of shipments not being properly identified by the recognition system and parcels slipping off their trays quickly increases.

The aforementioned Hermes hub at Friedewald has a single loading station and two parallel (unidirectional) conveyors each equipped with 766 trays. The conveyors each have a length of 700 meters and travel with a speed of 2.3 meters per second, which can be increased up to 2.5 meters per second. The DHL sorter at the airhub in Leipzig has a length of 6,500 meters, four loading stations and two parallel conveyors which travel in opposite directions. For choosing the conveyor at a loading station the next-free-tray-on-the-shortest-path-rule (denoted as SD rule in Section 4.1) is applied.

Evaluating the impact of the aforementioned design alternatives on the operational performance of a terminal is investigated in Section 4. However, performance and especially the ability of realizing multiple tray loads per cycle are heavily influenced by the assignment of inbound and outbound destinations to conveyor segments. This decision problem and suited solution procedures are described in the following.

3 Destination assignment problem

3.1 Problem description

For a given conveyor layout, the destination assignment problem decides on the assignment of inbound and outbound destinations to dock doors. Because each loading station receives parcels from a given set of inbound doors and each outbound door is served by a specific chute, this assignment defines the travel distance on the closed-loop conveyor for each parcel. Clearly, if only a single loading station is available each tilt tray can only be loaded (at most) once per cycle of the conveyor, so that a reduction of the parcels’ total travel distance has no influence on the potential number of tray loads per cycle and, thus, operational performance. However, if multiple line segments are available, then inbound and outbound destinations should be assigned to nearby segments, so that multiple tray loads per cycle are enabled. In this context, the destination assignment problem with multiple line segments (DAP-MLS) is defined as follows:

**DAP-MLS:** Given a conveyor layout with \( s = 1, \ldots, n \) conveyor segments (numbered according to their succession in the loop) each consisting of a loading station, which receives parcels from a given number \( D_{s}^{\text{in}} \) of inbound docks, and a given number \( D_{s}^{\text{out}} \) of outbound docks (connected via chutes) passed before reaching the next segment. Furthermore, consider the sets \( I, O \) and \( V (I \cup O = V) \) of destinations, where set \( I \) \( (O) \) contains all inbound (outbound) destinations, and a flow matrix \( b \) with \( b_{io} \) defining the number of parcels to be transshipped between destinations \( i \in I \) and \( o \in O \). We presuppose \( |I| = \sum_{s=1}^{n} D_{s}^{\text{in}} \) and \( |O| = \sum_{s=1}^{n} D_{s}^{\text{out}} \), which can either be ensured by unifying destinations to joint destinations served at the same dock or by inserting virtual desti-
nations without any shipments. The objective is to assign destinations to line segments, such that

\[
Z(\phi) = \sum_{i \in I} \sum_{o \in O} b_{io} \cdot [((\phi(o) - \phi(i) \mod n) + 1] \rightarrow \min,
\]

(1)

with \( \phi \) being a surjective mapping from destinations \( V \) to segments \( \{1, \ldots, n\} \) defining the assignment of destination \( j \) to segment \( \phi(j) \). The objective aims to minimize the total number of segments passed by filled tilt trays, where the modulo-division defines the distance (measured in number of segments) between the doors assigned to any inbound-outbound-relation weighted by the number \( b_{io} \) of parcels exchanged. Furthermore, it is to be ensured, that each segment receives no more inbound and outbound destinations than respective doors available:

\[
|\{i \in I \mid \phi(i) = s\}| \leq D_{s}^{in} \quad \forall s = 1, \ldots, n \quad (2)
\]

\[
|\{o \in O \mid \phi(o) = s\}| \leq D_{s}^{out} \quad \forall s = 1, \ldots, n. \quad (3)
\]

**Theorem.** DAP-MLS is NP-hard in the strong sense.

**Proof.** See Appendix A.

Note that minimizing the shipments’ total travel distance on the sortation conveyor is only a surrogate objective for widespread real-world performance measures, e.g., maximizing the number of parcels processed per day. However, a throughput based measure depends on the detailed arrival patterns of parcels – an information typically not available when solving the DAP and especially not when deciding on a terminal’s layout. A detailed test on the suitability of our surrogate objective is conducted with the help of the simulation study in Section 4.

Clearly, DAP-MLS is independent of the number of trays rotating or the chosen speed of the conveyor, so that no model modifications are required when varying these design parameters. The same holds true for multiple parallel conveyors rotating in identical direction. In this case, a parcel’s travel distance remains unaffected by the conveyor chosen, so that all inbound and outbound docks belonging to either conveyor and fed by the same loading station can be unified to a unique conveyor segment.

If two parallel conveyors rotate in opposite directions, then the conveyor choice directly influences each shipment’s travel distance. However, only reducing distances may lead to one conveyor being fully loaded, so that shipments queuing at the loading station cannot be fed into the congested loop conveyor, while the other conveyor runs idle. Under these prerequisites, it may be advantageous to sometimes accept longer travel distances and to load parcels onto empty trays of the opposite direction. In order to account for this additional requirement, the destination assignment problem with multiple parcel conveyors (DAP-MPC) aims to reduce the total travel distance and to evenly share the parcel loads among both rotation directions.
DAP-MPC: Instead of reducing the shipments’ total travel distance, which is the objective of DAP-MLS, we, now, aim to minimize the maximum total travel distance of both rotation directions. For this purpose, we introduce decision variables \( u_{io} \in \{0, 1, \ldots, b_{io}\} \forall i \in I, o \in O \) defining the number of parcels per inbound-outbound-relation dedicated to the upper conveyor, whereas \( b_{io} - u_{io} \) parcels are assigned to the lower conveyor. DAP-MPC consists of objective function (4) and constraints (2) and (3):

\[
Z(\phi) = \max \left\{ \sum_{i \in I} \sum_{o \in O} u_{io} \cdot \left[ \left( (\phi(i) - \phi(o) - 1) \mod n \right) + 1 \right]; \right. \\
\sum_{i \in I} \sum_{o \in O} (b_{io} - u_{io}) \cdot \left[ \left( (\phi(o) - \phi(i)) \mod n \right) + 1 \right] \left. \right\} \rightarrow \min.
\]  

Clearly, DAP-MPC is also NP-hard in the strong sense, which can easily be proven by a transformation from the DAP-MLS. Again, the objective of DAP-MPC is only a surrogate objective, whose suitability is to be investigated in our simulation study.

In case of more than two conveyors subdivided into two subsets both rotating in different directions, again, all conveyors of the same direction can be unified to unique conveyor segments, so that further model modification are only required if both subsets show non-identical cardinality. Then, both directions have a varying capacity of trays, which is to be considered when leveling the parcel load in objective function (4), e.g., by adding a weight according to the number of conveyors available per direction.

Note that integer linear programming formulations for both DAP problems solvable by an off-shelf-solver are given in the appendix.

3.2 Solution procedures

Due to their NP-hardness, heuristic solution procedures are required for solving destination assignment problems of real-world size. Thus, we introduce two meta-heuristics, namely greedy randomized adaptive search procedure (GRASP) and simulated annealing (SA), for solving both problem versions. We start the description of both procedures for the DAP-MLS.

3.2.1 Solving DAP-MLS

Both solution procedures apply the basic insight already introduced by Tsai and Chang (1990) and Bozer and Carlo (2008) for related destination assignment problems: Once all inbound destinations are fixed and assigned to conveyor segments, optimizing the remaining assignment of the outbound side can be solved by the linear assignment problem in polynomial time, and vice versa.

GRASP is an iterative solution process, where each iteration consists of a construction phase and a local search phase (e.g., see Feo and Resende, 1995). During the construction phase, we, first, randomly partition the inbound destinations among segments according to each segment’s number of available inbound doors. For this given assignment the optimal distribution of outbound destinations among segments is solved by the linear
assignment problem. In the second phase, the initial solution is, then, improved by rotationally optimizing inbound and outbound side for the currently fixed assignment of the other side, until the solution value does not improve anymore. Then, we restart the solution process with a novel random assignment of inbound destinations, until a given maximum number of \( R \) restarts or a time limit is reached.

SA is a stochastic meta-heuristic that is able to overcome local optima. It is based on the probabilistic acceptance of modified neighboring solutions inspired by thermal processes for obtaining low-energy states in heat baths (e.g., Kirkpatrick et al., 1983; Aarts et al., 1997). Our SA operates on two integer arrays \( \pi \) (with elements \( \pi(i) \in \{1, \ldots, n\} \forall i \in I \)) and \( \mu \) (with elements \( \mu(o) \in \{1, \ldots, n\} \forall o \in O \)) defining the door segments to which the respective destinations are assigned. The initial solution vectors are randomly determined where the maximum numbers of inbound and outbound doors of each segment are to be considered. For obtaining a neighboring solution, two randomly chosen elements of either array \( \pi \) or \( \mu \) are swapped. Given a new vector, the other array can be optimally determined, i.e., assigned to segments, by solving the linear assignment problem. The objective value associated with a solution can simply be determined by applying Equations (1).

Whether or not a neighboring solution \( (\pi', \mu') \) obtained by a swap move and the subsequent linear assignment problem is accepted is decided according to traditional probability schemes (Aarts et al., 1997):

\[
Prob((\pi', \mu') \text{ replacing } (\pi, \mu)) = \begin{cases} 
1, & \text{if } Z((\pi', \mu')) \leq Z((\pi, \mu)) \\
exp\left(\frac{Z((\pi, \mu)) - Z((\pi', \mu'))}{C}\right), & \text{otherwise}
\end{cases}
\]

If accepted, the current solution \( (\pi, \mu) \) is replaced by \( (\pi', \mu') \) as the starting point for further local search moves.

Our SA is steered by a simple static cooling schedule (see Kirkpatrick et al., 1983). The initial value for control parameter \( C \) is calculated as \( C = Z((\pi, \mu)^{\text{start}}) \), where \( (\pi, \mu)^{\text{start}} \) denotes a randomly determined destination assignment. Subsequently, this value \( C \) is continuously decreased in the course of the procedure by multiplying it with factor 0.9999 in each iteration. If within the last 100 iterations the global best solution does not improve, we restart SA with a new random solution and the initial temperature. Within our computational study, we have invariably used control parameter values as described above. Note that preliminary studies have indicated that this parameter constellation outperforms other settings and delivers a reasonable compromise between solution quality and time.

### 3.2.2 Solving DAP-MPC

In addition to the destination assignment, DAP-MPC requires a partition of parcels among the conveyors rotating in opposing directions. Once a destination assignment \( \phi \) is given, the remaining subproblem – the parcel partition problem (PPP) – can be defined
PPP: Given \( C = |I| \cdot |O| \) inbound-outbound-relations, each defined by a number \( b_c \) of parcels to be transported between two inbound and outbound destinations assigned to specific segments, and two loop conveyors rotating in opposite directions. Considering the objective function (4) of the DAP-MPC, the distances of parcels on the two conveyors are given by:

\[
\begin{align*}
    d^l_c &= (\phi(o) - \phi(i)) \mod n + 1 \quad \forall c = (i, o), i \in I, o \in O, \\
    d^u_c &= (\phi(i) - \phi(o) - 1) \mod n + 1 \quad \forall c = (i, o), i \in I, o \in O.
\end{align*}
\]

Because of the conveyors rotating in a closed loop, it holds that if \( d^l_c \) increases by one unit, \( d^u_c \) decreases by one unit, and vice versa. Also, \( 1 \leq d^l_c, d^u_c \leq n \), so that \( d^l_c + d^u_c = n + 1 \). For the sake of simplicity, we assume that all relations are sorted in non-decreasing order with regard to \( d^u_c \), so that \( d^u_c \leq d^u_k \forall c, k = 1, \ldots, C, c < k \). If \( 0 \leq l_c \leq b_c \) and \( 0 \leq u_c \leq b_c \) define the number of parcels of relation \( c \) transported on the lower and upper conveyor, respectively, we aim to partition the parcels among both directions such that \( \max \{ \sum_{c=1}^C l_c; \sum_{c=1}^C u_c \} \) is minimized.

The PPP resembles some traditional resource allocation problem (e.g., see Ibaraki and Katoh, 1988) with the special parameter constellation of \( d^l_c + d^u_c = n + 1 \). It can be solved to optimality in polynomial time by applying Algorithm 1. The basic idea of this procedure is to initially assign all parcels to the lower conveyor and, then, to reassign parcels according to decreasing savings in travel distance until the workload of both directions is leveled and a critical parcel is reached.

The runtime complexity of Algorithm 1 is clearly in \( \mathcal{O}(C) \). Because sorting the relations requires \( \mathcal{O}(C \log C) \), it follows that the PPP is solvable in \( \mathcal{O}(C \log C) \).

**Theorem.** For solving the PPP Algorithm 1 is optimal.

**Proof.** See appendix.

When solving the overall DAP-MPC, Algorithm 1 can straightforwardly be integrated into the two meta-heuristics (GRASP and SA) already described for solving the DAP-MLS. We simply solve the respective linear assignment problem as was described above by assuming that all parcels travel on the one conveyor having the shorter travel distance. For the resulting overall assignment of all inbound and outbound destinations, we relax this assumption and partition the parcels optimally between both rotation directions according to Algorithm 1, which finally leads to the overall objective value of an iteration.

### 3.3 Computational performance

In order to show that our SA and GRASP procedures are well suited for delivering near optimal solutions for DAP-MLS and DAP-MLC, respectively, we first elaborate
Algorithm 1: Solving the parcel partition problem

input: requires parameters $n$ and $d^l_c$, $d^u_c$, $b_c$ for each connection $c$

1. $cumLow := 0$
2. for $c = 1$ to $C$
3.   $cumLow := cumLow + d^l_c \cdot b_c$
4. end
5. $k := 1$
6. $cumUp := 0$
7. while $(n+1) \cdot b_k < cumLow - cumUp$
8.   $u_k := b_k$
9.   $l_k := 0$
10. $cumUp := cumUp + d^u_k \cdot b_k$
11. $cumLow := cumLow - d^l_k \cdot b_k$
12. $k := k + 1$
13. end
14. if $d^u_k \leq (cumLow - cumUp) \mod (n+1)$ then
15.   $u_k := \left\lfloor \frac{cumLow - cumUp}{n+1} \right\rfloor$
16. else
17.   $u_k := \left\lceil \frac{cumLow - cumUp}{n+1} \right\rceil$
18. end
19. $l_k := b_k - u_k$
20. for $c = k + 1$ to $C$
21.   $u_c := 0$
22.   $l_c := b_c$
23. end

return: optimal $u_c$ and $l_c$ values

on instance generation. Two cases of instances are distinguished: small instances are small enough to be solved to optimality, whereas large instances represent instances of real-world size and, thus, have to be solved heuristically. The parameters for generating instances are listed in Table 1.

For each combination of parameter values instance generation was repeated 100 times (small instances) and 25 times (large instances), so that in total $100 \cdot 3 \cdot 3 \cdot 2 = 1,800$ small instances and $25 \cdot 3 \cdot 3 \cdot 2 = 450$ large instances were obtained, respectively. For a given set of parameter values, each instance is generated as follows: First, the number of line segments is fixed to $n = Rd(\frac{2}{3} \cdot \min\{|I|,|O|\})$, where $Rd(x)$ is the nearest integer to $x$. We, then, determine the amount of outbound destinations $g_i$ actually served by each inbound destination $i$ by $g_i = Rd(\alpha_i \cdot |O|)$, where $\alpha_i$ is randomly drawn from the given interval $[\alpha_{a1}, \alpha_{a2}]$. Then, we iterate through all outbound destination of $i$ in a random sequence and for the first $g_i$ outbound destinations the number of parcels $b_{io}$ shipped is randomly drawn according to a normal distribution with $\mu = \frac{5000}{Rd(\alpha_i \cdot |O|)}$ and
The algorithms were coded in C# (Visual Studio.NET 2010) and all tests have been performed on an Intel Core 2.5 GHz i5-25205M notebook with 4GB memory.

At first, the solution performance of the SA and GRASP algorithms is tested for the small instances, for which optimal solutions are obtained by a complete enumeration (EN), which evaluates all possible assignments of destinations to docks. Furthermore, the optimization software “Xpress MP” (XP) was applied in order to compare the heuristics with an off-shelf-solver solving the integer programs given in the appendix. Table 2 shows the optimality gap as well as the cpu time for each combination of parameters and both problems, DAP-MLS and DAP-MPC. Our heuristics have been executed without time limit, and a maximum of $R = 3$ restarts.

With in the given time frame of 300 cpu seconds, off-the-shelf solver Xpress was only able to solve 97% of the small instances to optimality, which again highlights the necessity of solving real-world instances heuristically. Our heuristic procedures require less than 0.005 (GRASP) and 0.054 (SA) cpu seconds, respectively. However, SA leads to a much better solution quality than GRASP with an optimality gap considerably lower than one percent. This superiority can also be stated for the large instances (see Table 3). Here, both procedures are executed with an identical time limit of 30 cpu seconds and a prohibitive high number of restarts ($R = 999$). Note that within Table 3 performance measure '# best' denotes the number of instances the respective heuristic found a better solution value than its competitor. The subsequent measures summarizing different relative and absolute deviations are calculated in relation to the best value found per instance. Again, SA clearly outperforms GRASP with regard to the solution quality, so that it will be applied for the evaluation of the design alternatives in the following section.

### 4 Simulation study

This section aims to answer two questions: First, we investigate whether the objective functions proposed for DAP-MLS and DAP-MPC, respectively, are suited surrogate objectives. For this purpose, we measure the correlation between our objective functions and the results of a simulation study for random destination assignments. Furthermore,
| $|I|$ | $|O|$ | $[\alpha_1, \alpha_2]$ |            |            | $\alpha_1$ | $\alpha_2$ | $\text{gap} [%]$ | $\text{cpu time} [s]$ | $\text{gap} [%]$ | $\text{cpu time} [s]$ | $\text{gap} [%]$ | $\text{cpu time} [s]$ | $\text{gap} [%]$ | $\text{cpu time} [s]$ |
|------|------|-----------------|------------|------------|------------|------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 3    | 3    | [0.1,0.5]      | 0.00       | 2.53       | 0.00       | <0.01      | 0.01           | <0.01           | 0.00           | 5.09            | 0.33           | <0.01           | 0.07            | <0.01           | 0.01           |
| 3    | 3    | [0.6,0.9]      | 0.00       | 0.21       | 0.00       | <0.01      | 0.01           | <0.01           | 0.01           | 3.03            | 0.20           | <0.01           | 0.09            | <0.01           | 0.01           |
| 3    | 5    | [0.1,0.5]      | 0.00       | 1.15       | 0.00       | <0.01      | 0.02           | <0.01           | 0.00           | 4.90            | 0.28           | <0.01           | 0.11            | <0.01           | 0.01           |
| 3    | 5    | [0.6,0.9]      | 0.00       | 0.66       | 0.00       | <0.01      | 0.02           | <0.01           | 0.00           | 3.02            | 0.31           | <0.01           | 0.17            | <0.01           | 0.01           |
| 3    | 7    | [0.1,0.5]      | 0.00       | 1.00       | 0.00       | <0.01      | 0.02           | <0.01           | 0.00           | 4.98            | 0.39           | <0.01           | 0.17            | <0.01           | 0.01           |
| 3    | 7    | [0.6,0.9]      | 0.00       | 0.32       | 0.00       | <0.01      | 0.04           | <0.01           | 0.00           | 2.26            | 0.12           | <0.01           | 0.26            | <0.01           | 0.02           |
| 5    | 3    | [0.1,0.5]      | 0.00       | 0.65       | 0.00       | <0.01      | 0.02           | <0.01           | 0.00           | 5.06            | 0.33           | <0.01           | 0.12            | <0.01           | 0.01           |
| 5    | 3    | [0.6,0.9]      | 0.00       | 0.60       | 0.00       | <0.01      | 0.03           | <0.01           | 0.01           | 2.79            | 0.41           | <0.01           | 0.17            | <0.01           | 0.01           |
| 5    | 5    | [0.1,0.5]      | 0.00       | 3.52       | 0.00       | <0.01      | 0.25           | <0.01           | 0.00           | 3.91            | 0.12           | <0.01           | 0.84            | <0.01           | 0.02           |
| 5    | 5    | [0.6,0.9]      | 0.00       | 1.30       | 0.00       | <0.01      | 0.34           | <0.01           | 0.03           | 1.35            | 0.00           | 0.01           | 3.61            | <0.01           | 0.02           |
| 5    | 7    | [0.1,0.5]      | 0.00       | 3.99       | 0.00       | <0.01      | 0.45           | <0.01           | 0.02           | 3.77            | 0.16           | 0.06           | 1.21            | <0.01           | 0.02           |
| 5    | 7    | [0.6,0.9]      | 0.00       | 1.07       | 0.00       | <0.01      | 0.67           | <0.01           | 0.03           | 0.91            | 0.00           | 0.06           | 6.50            | <0.01           | 0.02           |
| 7    | 3    | [0.1,0.5]      | 0.00       | 1.00       | 0.00       | <0.01      | 0.03           | <0.01           | 0.00           | 4.93            | 0.30           | <0.01           | 0.19            | <0.01           | 0.01           |
| 7    | 3    | [0.6,0.9]      | 0.00       | 0.97       | 0.00       | <0.01      | 0.06           | <0.01           | 0.01           | 2.31            | 0.18           | <0.01           | 0.26            | <0.01           | 0.02           |
| 7    | 5    | [0.1,0.5]      | 0.00       | 4.45       | 0.00       | 0.03       | 0.45           | <0.01           | 0.02           | 2.69            | 0.09           | 0.06           | 1.53            | <0.01           | 0.02           |
| 7    | 5    | [0.6,0.9]      | 0.00       | 1.34       | 0.00       | 0.03       | 0.70           | <0.01           | 0.02           | 1.16            | 0.00           | 0.06           | 16.88           | <0.01           | 0.02           |
| 7    | 7    | [0.1,0.5]      | 0.00       | 8.79       | 0.58       | 7.85       | 10.36          | <0.01           | 0.03           | 4.94            | 0.50           | 18.79          | 69.60           | <0.01           | 0.04           |
| 7    | 7    | [0.6,0.9]      | 0.00       | 2.71       | 0.20       | 7.86       | 116.29         | <0.01           | 0.03           | 1.61            | 0.22           | 18.82          | 298.62          | <0.01           | 0.04           |

Table 2: Performance of SA and GRASP for small instances
Table 3: Performance of SA and GRASP for large instances

<table>
<thead>
<tr>
<th></th>
<th>DAP-MLS</th>
<th>DAP-MPC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GRASP</td>
<td>SA</td>
</tr>
<tr>
<td># best</td>
<td>138</td>
<td>312</td>
</tr>
<tr>
<td>avg gap [%]</td>
<td>0.68</td>
<td>0.46</td>
</tr>
<tr>
<td>max gap [%]</td>
<td>2.95</td>
<td>4.42</td>
</tr>
<tr>
<td>avg abs</td>
<td>31237</td>
<td>5855</td>
</tr>
<tr>
<td>max abs</td>
<td>226143</td>
<td>87586</td>
</tr>
<tr>
<td>avg cpu time</td>
<td>20.27</td>
<td>29.51</td>
</tr>
</tbody>
</table>

we test the operational gains of the layout alternatives (see Section 2) compared to a basic setting with only a single conveyor and a single loading station. However, before presenting the respective results we elaborate on the setup of our simulation study.

4.1 Setup of simulation study

The aforementioned hub in Friedewald serves as the prototype for our study. Around the perimeter of the building (with dimension 216 × 112 meters) there are 50 inbound and 100 outbound doors. Within the basic setting, the sorter consists of 766 tilt trays each having a length of 0.85 meters. We define any movement of the conveyor by the length of one tray as a tray move. Thus, at a maximum conveyor speed of 2.5 meters per second 84,705 tray moves accumulate during an eight hours shift. Within a single simulation run, these 84,705 tray moves are executed and as the elementary performance measure we record the total number of parcels sorted per shift, i.e., the number of parcels tilted into their dedicated chutes. However, for being able to do so, we have to (i) define the inbound and outbound parcel flows to be sorted during a simulation run, (ii) introduce some assumptions with regard to the sortation process, and (iii) differentiate the layout alternatives of the sortation conveyor tested per instance.

(i) In order to keep our simulation as clear and manageable as possible, we abstract from multiple inbound trucks per destination arriving at different points in time, but presupposes a steady stream of parcels per loading station. The number $b_\text{io}$ of parcels per inbound-outbound-relation is generated as was already described in Section 3.3. Once the number of outbound destinations served per inbound destination is determined, we draw $b_\text{io}$ according to a normal distribution with $\mu = \frac{5000}{\text{I}(\alpha_i, |O|)}$ and $\sigma = \beta \cdot \frac{5000}{\text{I}(\alpha_i, |O|)}$.

The different parameter values introduced to this formulae are listed in Table 4.

Once inbound trucks are assigned to inbound docks and, thus, loading stations (which is either done randomly or by solving a destination assignment problem), we have to unify these parcel streams to a single one queueing in front of the loading station of the respective segment. If $I_s$ defines the set of inbound destinations assigned to the inbound docks of segment $s$, the probability $p_{\text{io}}^{i,o}$ of the next parcel at segment $s$ being shipped from destination $i$ to $o$ is given by:
\[ [\alpha_a, \alpha_b] \text{ interval for drawing of percentage of outbound destinations served by each inbound destination} \]
\[ \beta \text{ deviation of number of parcels for each outbound station from expected value} \]

<table>
<thead>
<tr>
<th>symbol</th>
<th>description</th>
<th>values</th>
</tr>
</thead>
<tbody>
<tr>
<td>([\alpha_a, \alpha_b])</td>
<td>interval for drawing of percentage of outbound destinations served by each inbound destination</td>
<td>([0.1;0.3],[0.4;0.6],[0.7;0.9])</td>
</tr>
<tr>
<td>(\beta)</td>
<td>deviation of number of parcels for each outbound station from expected value</td>
<td>(0.1,0.2,0.5)</td>
</tr>
</tbody>
</table>

Table 4: Parameters for instance generation

\[ p_{io}^s = \frac{b_{io}}{\sum_{\tau \in I_s} \sum_{o' \in O} b_{\tau o'}} \quad \forall s = 1, \ldots, n; \; i \in I_s; \; o \in O. \quad (6) \]

In a full-factorial design, instance generation per parameter constellation (see Table 4) is repeated three times, so that in total \(3 \cdot 3 \cdot 3 = 27\) basic simulation scenarios are generated.

(ii) The following simplifying assumptions with regard to the sortation process are made:

- We only investigate 'normal' operations, so that technical failures (such as unrecognized or lost parcels slipping of the conveyor or a breakdown of conveyors and loading stations) are not considered within our simulation.

- Furthermore, we presuppose that no congestions of parcels at their designated outbound docks occur. Thus, we assume that enough workforce for loading the trailers is available and that filled outbound trailers are changed early enough to proceed the loading process before congestions of a chute hindering parcels to be tilted can occur.

- As is typically the case in real-world applications, docks and loading stations are evenly allocated around the loop conveyors within our simulation. The distances between any two successive loading stations are assumed to be equispaced and all segments receive the same number of inbound and outbound docks (except for rounding differences), where outbound docks and their respective chutes are again located equispaced within each segment.

(iii) For each basic simulation scenario, several design alternatives are combined in order to generate a specific conveyor layout. For generating these layouts, the design parameters defined in Table 5 are applied. The number of segments \(n\) is varied from the minimum of one segment to the maximum of 50 (\(= \min(|I|,|O|)\)) segments. For any given conveyor layout and parcel flow, two alternative ways of assigning destinations to dock doors are investigated: a random assignment and an optimized one with one of our two DAPs. Furthermore, the design alternatives of having one conveyor (\(CON = 1\)), two
conveyors with unidirectional flow (2u), and two conveyors rotating in opposite direction (2o) are differentiated. In case of two conveyors with different flow directions, the loading rule (denoted as LOAD) defines the conveyor, on which a parcel being first in the queue of a loading station is loaded per tray move. The following three loading rules are tested:

<table>
<thead>
<tr>
<th>symbol</th>
<th>description</th>
<th>values</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>number of line segments</td>
<td>1,2,3,4,5,10,15,25,50</td>
</tr>
<tr>
<td>CON</td>
<td>number of conveyors</td>
<td>1,2u,2o</td>
</tr>
<tr>
<td>LOAD</td>
<td>loading rule</td>
<td>FET,SD,MTT</td>
</tr>
<tr>
<td>t</td>
<td>number of tilt trays</td>
<td>666,766,866</td>
</tr>
</tbody>
</table>

Table 5: Parameters for design alternatives

**First-empty-tray (FET) rule:** Choose the flow direction where the first empty tray reaches the loading station.

**Shortest-distance (SD) rule:** Choose the first empty tray of that flow direction traveling the shortest distance to the parcel’s chute.

**Minimum-tilt-time (MTT) rule:** Choose the first empty tray of either flow direction which reaches the parcel’s chute first.

Note that for two conveyors rotating unidirectional all three loading rules lead to the same result, so that we simply consider the FET rule. Finally, also the number \( t \) of tilt trays is varied. This corresponds to a modification of the sorter length, because the tray length is predetermined by the parcel sizes to be sorted and can, thus, typically not be altered.

This leads to \( 9 \cdot (2 \cdot 1 + 1 \cdot 3) \cdot 3 = 135 \) different conveyor settings, where each setting is evaluated with and without optimizing the DAP (DAP-MLS for \( CON = 1 \) and 2u and DAP-MPC for \( CON = 2o \)), so that in total \( 27 \cdot 135 \cdot 2 = 7,290 \) simulation runs have been executed. Note that the DAPs are solved by the SA algorithms with a time limit of 60 cpu seconds and a prohibitive high number of restarts \((R = 999)\).

Within each simulation run, the conveyor flow is simulated by discrete tray moves. Each move consists of a loading and an unloading phase. During the loading phase, the first parcel queuing at a station is loaded onto the sorter provided that the current tray is empty and the given loading rule allows a loading. During the unloading phase, full trays are tilted, if they reach the designated chute of the respective parcel. Each unit leaving the sorter is marked as ‘delivered’ and the counter of sorted parcels per shift is increased. The simulation is stopped once all tray moves of an eight hours shift are executed and the total number of parcel sorted per shift is returned. Note that other performance measures, e.g., the average percentage of filled tilt trays, lead to very similar results, so that we only report the throughput.
4.2 Testing the appropriateness of DAP-MLS and DAP-MPC

In this section, we test the suitability of our two (surrogate) objective functions for the DAP (see Section 3), which minimize the total number of segments passed by all parcels (DAP-MLS) and the maximum number of segments passed over all conveyors (DAP-MPC), respectively. Recall that instead of directly tackling an actual performance measure, e.g., maximizing the throughput of parcels per day, a surrogate objective is required, because the exact arrival pattern of parcels is unknown when solving a DAP and especially when deciding on a terminal’s layout. In order to prove the suitability of our choice, we randomly draw five destination assignments per simulation scenario and conveyor setting (a total of 18,225 instances), evaluate these assignments with our (surrogate) objective functions (DAP-MLS for \( CON = 1 \) and \( 2u \) and DAP-MPC for \( CON = 2o \)), and compare these values with the results of the respective terminal simulation. If our objective functions are well suited, the correlation between objective value and simulation result should be close to -1. Table 6 summarizes the Pearson correlation coefficients per design alternative averaged over all specific layout settings applying the respective design alternative.

<table>
<thead>
<tr>
<th>CON LOAD</th>
<th>objective function</th>
<th>number of passed segments</th>
<th>even workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>–</td>
<td>-0.95901</td>
<td>-0.64724</td>
</tr>
<tr>
<td>2u</td>
<td>FET DAP-MLS</td>
<td>-0.89494</td>
<td>-0.60984</td>
</tr>
<tr>
<td>2o</td>
<td>FET DAP-MPC</td>
<td>-0.04770</td>
<td>-0.00991</td>
</tr>
<tr>
<td>2o</td>
<td>SD DAP-MPC</td>
<td>-0.80175</td>
<td>-0.28962</td>
</tr>
<tr>
<td>2o</td>
<td>MTT DAP-MPC</td>
<td>-0.81618</td>
<td>-0.29916</td>
</tr>
</tbody>
</table>

Table 6: Average correlation coefficients (Pearson) for each design alternative

For all design alternatives our objective functions seem well suited; only when steering parcel loading in a system of two conveyors rotating in opposite directions (\( CON = 2o \)) with the FET rule no correlation exists. This, however, is not astounding, because FET chooses the first empty tray passing a loading station and, thus, does not take advantage of an optimized destination assignment, but loads the sorter rather randomly.

To further strengthen our choice, we also tested an alternative surrogate objective function, which is to minimize the maximum number of parcels passing each segment. A precise definition of this objective is given in the appendix (see Appendix E). However, this objective function, whose results are summarized in column \textit{even workload} within Table 6, shows much lower correlations with the simulation results and is, thus, clearly outperformed by our DAP-objectives.

4.3 Performance impact of design alternatives

In order to evaluate the efficiency of the design alternatives, the very basic conveyor setting consisting of a single conveyor, a single segment, and \( t = 766 \) trays serves as a benchmark. In the basic setting, on average 84,325.74 parcels are sorted per eight hours
First, we investigate the impact of an increased conveyor speed. Clearly, this is not a design alternative, but an operational lever to increase parcel throughput. Figure 4a depicts the resulting improvement of throughput for four selected conveyor layouts in relation to our benchmark. Obviously, throughput increases linear when accelerating the conveyor, where the slope depends on the respective layout. However, recall that the conveyor speed can only be varied within narrow margins, e.g., between 2.3 and 2.5 meters per second at the Friedewald terminal. Such an acceleration of 0.2 meters per second leads to an increased throughput of 8% or 6745.66 parcels (for layout CON = 1 and n = 1).

The resulting performance when varying the number of tilt trays for four specific conveyor layouts is summarized in Figure 4b. Recall that tray sizes are unchangeably defined by the given parcel dimensions, so that additional trays enlarge the total length of the loop conveyor. The results show that a varying number of trays has no impact on sortation performance. This, however, must be the case within our simulation study, because the positive effect of an earlier parcel load into the sortation system enabled by additional trays is directly compensated by the enlarged distance to the respective outbound chute. The only positive impact from additional trays arises whenever congestions of outbound chutes occur, so that blocked parcels have to rotate multiple rounds on the loop conveyor before the congestion being dissolved. In this case, additional trays serve as storage space, so that the sortation process of successive parcels dedicated to unblocked chutes can proceed unaffected. This effect is, however, not explored within our simulation study.

Multiple segments: Additional loading stations promise multiple tray loads per cycle of the conveyor and, thus, a higher throughput. Figure 5a depicts the performance
gains for each design alternative when increasing the number of segments and assigning destinations to docks randomly. From these results, the following conclusions can be drawn:

- Multiple loading stations have considerable positive performance impact for all design alternatives. When comparing a single conveyor ($CON = 1$) with a layout of two unidirectional conveyors ($CON = 2u$), it is not astounding to see that (after a brief transitional phase) the average throughput is always about 100% higher for an equal number of segments, which exactly matches the doubled sortation capacity. However, two conveyors rotating in opposite directions ($CON = 2o$) profit disproportionately high from additional segments. This is due to the parcels’ reduced travel distances (enabled by the loading rules SD and MTT), so that excessive use of multiple tray loads per cycle is made.

- For all design alternatives, the positive impact of additional loading stations quickly diminishes. With a single conveyor (denoted as $1/-$ within Figure 5a), the second segment increases the number of sorted parcels by 33%, whereas an extra of 25 segments raising the number of loading stations from $n = 25$ to $n = 50$ only leads to an additional throughput of 4%. This is an important finding for the practitioner, because already just a few additional segments (and thus only moderately higher investment costs) promise considerable throughput gains.

![Figure 5: Simulation results when varying the number of segments (with (b) and without (a) solving the DAP)](image)

**Impact of DAP:** Clearly, solving the DAP leads to better assignments of destination to docks, so that the parcels’ travel distances on the conveyor are further reduced. This positive effect becomes obvious for all design alternatives, when comparing Figures 5a and b. For instance, the design alternatives $CON = 1$, $CON = 2u/LOAD = FET$, and $CON = 2o/LOAD = SD$ with $n = 4$ loading stations lead to an additional increase of throughput over the benchmark by 27%, 32%, and 21% when optimizing the DAP.
instead of randomly assigning destinations to docks. These gains become the higher the more additional loading stations are introduced. The maximum gains of an additional 170% of throughput are realized for the (unrealistic case) of \( n = 50 \) loading stations in design alternative \( CON = 2o/LOAD = SD \). However, it must be noted that these (simulated) gains are only upper bounds for the gains actually realized in real-world terminals. In our simulation study, it is assumed that the parcel arrival patterns at the inbound docks actually realize as was anticipated when solving the DAP. In the real-world, the forecasted number of parcels exchanged per inbound-outbound-relation will deviate from each day’s arrival pattern, so that it depends on the forecast errors to what extend the gains approximated by our simulation study can finally be realized.

A loading rule steers the loading of parcels onto the trays if more than a single conveyor exists. The results depicted in Figures 5a and b allow for the following conclusions with regard to the performance impact of the loading rules:

- First, it can be stated that for two conveyors rotating in opposite directions (\( CON = 2o \)) the loading rules SD (shortest-distance) and MTT (minimum-tilt-time) lead to identical results and both considerably outperform the FET (first-empty-tray) rule. This finding is not astounding, because the latter loads parcels randomly into either direction, whereas the former two prefer shorter distances.

- Most striking seems the vast improvement achievable when changing two unidirectional conveyors (\( CON = 2u \)) to a setting where two conveyors rotate in opposite directions (\( CON = 2o \)) and tray loading is steered either by the SD or the MTT rule. For instance, an extra throughput of 136% (without DAP) and 124% (with DAP) for \( n = 10 \) loading stations is realized when comparing \( CON = 2o \) to \( CON = 2u \), respectively. This result is especially remarkable, because when newly erecting a sortation system both settings only deviate in the rotation direction of a single conveyor and should, thus, require very similar investment costs.

- For two conveyors loaded according to the FET rule and a randomly determined destination assignment, it makes no difference rotating them unidirectional (\( CON = 2u \)) or in opposite directions (\( CON = 2o \)). The random choice of the FET rule cannot exploit the additional flexibility of choosing the shorter direction offered by opposing rotation directions, so that on average the same total distance is taken as for \( CON = 2u \). Furthermore (for the case \( CON = 2o \)), performance cannot be improved by integrating the DAP. Minimized distances taken by chance by the FET rule are counterbalanced by even longer distances when choosing the “wrong” direction. However, if the two conveyors rotate unidirectional (\( CON = 2u \)) integrating the DAP and loading according to the FET rule leads to a much better performance. In this case, all parcels take the same flow direction, so the shorter distances reduce the number of segments passed, which in turn increases the throughput.

- Identical performance also results from the SD and the MTT rule, if two conveyors rotate in opposite directions (\( CON = 2o \)) irrespective of integrating the DAP or
not. Apparently, the trays are (nearly) never fully loaded, so that the MTT rule would have to send a parcel over the longer distance. If, however, always an empty tray can be found quickly enough and the MTT rule always chooses the shorter distance, then in fact it decides identical to the SD rule.

In summary, Table 7 specifies suited conveyor layouts, if the throughput is to be doubled, tripled or quadrupled (compared to the benchmark). Clearly, when generalizing these figures some caution seems recommendable, since the results depend on a specific number of inbound and outbound docks available at a terminal. However, at least roughly practitioners receive some advice on the operational performance of a specific conveyor layout when weighing operational gains against the respective investment costs.

<table>
<thead>
<tr>
<th>throughput</th>
<th>n</th>
<th>DAP</th>
<th>CON</th>
<th>LOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>×2</td>
<td>2</td>
<td>✓</td>
<td>2u,2o</td>
<td>FET, SD, MTT</td>
</tr>
<tr>
<td>×2</td>
<td>5</td>
<td>✓</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>×3</td>
<td>4</td>
<td>✓</td>
<td>2o</td>
<td>SD, MTT</td>
</tr>
<tr>
<td>×3</td>
<td>4</td>
<td>✓</td>
<td>2u</td>
<td>FET</td>
</tr>
<tr>
<td>×3</td>
<td>5</td>
<td>✓</td>
<td>2u</td>
<td>FET</td>
</tr>
<tr>
<td>×4</td>
<td>6</td>
<td>✓</td>
<td>2o</td>
<td>SD, MTT</td>
</tr>
<tr>
<td>×4</td>
<td>7</td>
<td>✓</td>
<td>2u</td>
<td>FET</td>
</tr>
<tr>
<td>×4</td>
<td>7</td>
<td>✓</td>
<td>2o</td>
<td>SD, MTT</td>
</tr>
</tbody>
</table>

Table 7: Profitable conveyor layouts

5 Conclusion

This paper tests different design alternatives of closed-loop tilt tray sortation conveyors applied in the parcel service industry. Especially, the impact of parallel conveyors, multiple loading stations, and different loading rules on parcel throughput are investigated. For this purpose, a novel destination assignment problem is defined in order to foster multiple tray loads per cycle of the conveyor at multiple loading stations. All design alternatives and their respective parcel flows are simulated in a comprehensive computational study, from which the following key findings can be extracted:

- The number of tilt trays rotating in the loop conveyor does not influence performance during “normal” operations (see Section 4.1). Only frequent congestions of outbound chutes may justify the integration of additional trays.

- The incremental benefit of additional loading stations quickly diminishes the more loading stations are added. Thus, already two, three, or four loading stations, which are typical choices in real-world sortation systems, lead to a considerable increase in throughput while keeping investment costs at a manageable level.
• Larger sortation systems with two conveyors and multiple loading stations should be rotated in opposite directions. Huge performance gains over a corresponding unidirectional layout can be realized by simply choosing the shorter flow direction for each parcel.

• Assigning inbound and outbound destinations according to the novel destination assignment problems (DAPs) introduced in this paper considerably improves parcel throughput in all layout settings with multiple segments. Furthermore, integrating the DAP requires merely a manageable investment into information technology for forecasting the number of parcels per inbound-outbound-relation from historical data (stored by the parcel distributors anyway).

Future research could investigate the performance gains of other design alternatives not yet realized by the parcel service industry. For instance, multiple conveyors need not take identical paths through the terminal or heavily loaded outbound destinations could be assigned to multiple outbound docks. Furthermore with regard to our novel DAPs, efficient exact solution procedures should be developed and other surrogate objective should be tested. Finally, due to their considerable impact on throughput further tests of the loading rules are desirable. These simple priority rules could be tested against exact solution procedures, which optimally load parcels according to the current status of tilt trays and the actual parcel streams at the loading stations.

References


Appendix

Appendix A: Complexity of DAP-MLS

We will prove NP hardness for the door assignment problem with multiple line segments (DAP-MLS) by a transformation from the 3-Partition problem, which is well known to be NP hard in the strong sense (Garey and Johnson, 1979).

**3-Partition:** Given a set of \( n = 3m \) positive integers \( A = \{a_1, \ldots, a_n\} \) and a positive integer \( R \) with \( \frac{R}{2} < a_i < \frac{R}{2} \) \( \forall i = 1, \ldots, n \) and \( \sum_{i=1}^{n} a_i = mR \), can \( S \) be divided into \( m \) subsets \( A_j \) with \( |A_j| = 3 \) and \( \sum_{a_i \in A_j} a_i = R \) \( \forall j = 1, \ldots, m \)?

**Transformation:** Given an instance of 3-Partition, we generate an instance of DAP-MLS with \( S = m \) door segments, \( mR \) inbound destinations and \( 3m \) outbound destinations. Each door segment \( s \) contains \( R \) inbound doors \( (D_{s}^{in} = R) \) and three outbound doors \( (D_{s}^{out} = 3) \). Furthermore, every outbound destination \( o \) receives a single shipment from \( a_o \) unique inbound destinations, where \( a_o \) corresponds to the integers of 3-Partition. The question we ask is whether we can find a solution of DAP-MLS with objective value \( F = mR \), i.e., every parcel is loaded and unloaded in the same segment.

A feasible solution for an instance of 3-Partition can be transformed to a feasible solution of the corresponding DAP-instance in polynomial time by assigning all \( R \) inbound and 3 outbound trucks corresponding to an integer set of the 3-Partition solution to a separate door segment. This setup guarantees that all shipments can be delivered within the same segment and objective value \( Z = mR \) is realized.

On the other hand, each feasible solution for any DAP-MLS instance is also a feasible solution for 3-Partition, because with \( Z = mR \) a deviating door assignment as the one described above is not possible. Any inbound destination \( i \) that delivers to an outbound destination \( o \) has to be assigned to the same door segment. The number of inbound destinations serving a certain outbound destination \( o \) matches an integer value of 3-Partition and there exists a direct mapping between the door segments and the sets of integers.

Appendix B: Integer linear programming formulation for DAP-MLS

Given the notation summarized in Table 8 the DAP-MLS can be formulated as an integer linear program as follows:

\[
Z(X, Y, Z) = \sum_{i \in I} \sum_{o \in O} b_{i,o} \cdot \left( \sum_{s=1}^{n} \sum_{s'=1}^{s-1} (n - (s - s')) \cdot z_{isos'} + \sum_{s=1}^{n} \sum_{s'=s}^{n} (s' - s) \cdot z_{isos'} + 1 \right) \rightarrow \min
\]

subject to
Table 8: Notation for DAP-MLS

\[ \sum_{s \in S} x_{is} = 1 \quad \forall i \in I \]  

(8)

\[ \sum_{s \in S} y_{os} = 1 \quad \forall o \in O \]  

(9)

\[ \sum_{i \in I} x_{is} \leq D_s^{in} \quad \forall s \in S \]  

(10)

\[ \sum_{o \in O} y_{os} \leq D_s^{out} \quad \forall s \in S \]  

(11)

\[ 2 \cdot z_{isos'} \leq x_{is} + y_{os'} \quad \forall i \in I; o \in O; s, s' \in S \]  

(12)

\[ z_{isos'} \geq x_{is} + y_{os'} - 1 \quad \forall i \in I; o \in O; s, s' \in S \]  

(13)

\[ x_{is}, y_{os}, z_{isos'} \in \{0, 1\} \quad \forall i \in I; o \in O; s, s' \in S \]  

(14)

Appendix C: Integer linear programming formulation for DAP-MPC

Table 9: Additional notation for DAP-MPC

<table>
<thead>
<tr>
<th>\text{notation}</th>
<th>\text{description}</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{lisos'}</td>
<td>number of parcels from destination (i \in I) to (o \in O) assigned to segments (s) and (s'), respectively, dedicated to the lower conveyor</td>
</tr>
<tr>
<td>\text{uisos'}</td>
<td>number of parcels from destination (i \in I) to (o \in O) assigned to segments (s) and (s'), respectively, dedicated to the upper conveyor</td>
</tr>
<tr>
<td>(w)</td>
<td>auxiliary variable: objective value</td>
</tr>
</tbody>
</table>

Given the notation summarized in Tables 8 and 9 the DAP-MPC can be formulated as the following integer linear program:
\[ Z(L, U, w, X, Y, Z) = w \rightarrow \min \]

subject to (8)-(14) and

\[
\sum_{i \in I} \sum_{o \in O} \left( \sum_{s=1}^{n} \sum_{s'=1}^{n} (n - (s - s') + 1) \cdot l_{isos'} + \sum_{s=1}^{n} \sum_{s'=n}^{n} (s' - s + 1) \cdot l_{isos'} \right) \leq w \tag{16}
\]

\[
\sum_{i \in I} \sum_{o \in O} \left( \sum_{s=1}^{n} \sum_{s'=1}^{n} (s - s') \cdot u_{isos'} + \sum_{s=1}^{n} \sum_{s'=n}^{n} (n - (s' - s)) \cdot u_{isos'} \right) \leq w \tag{17}
\]

\[
u_{isos'} + l_{isos'} = b_o \cdot z_{isos'} \quad \forall i \in I, o \in O, s, s' \in S \tag{18}
\]

\[
u_{isos'}, l_{isos'} \in \mathbb{N} \quad \forall i \in I, o \in O, s, s' \in S \tag{19}
\]

\[
w \in \mathbb{R} \tag{20}
\]

**Appendix D: Proof of optimality for Algorithm 1**

In order to show that Algorithm 1 solves the PPP to optimality, we need to prove the following three statements:

(i) Swapping the assignment of two parcels from different conveyors does not improve the solution,

(ii) swapping the assignment of a parcel dedicated to the upper conveyor does not improve the solution and

(iii) swapping the assignment of a parcel dedicated to the lower conveyor does not improve the solution.

Let \( L \) (\( U \)) be the total travel distance (in number of segments) on the lower (upper) conveyor and let \( Z \) be the objective function value of a solution:

\[
L = \sum_{c=1}^{C} d_c^n \cdot l_c, \quad U = \sum_{c=1}^{C} d_c^u \cdot u_c, \quad Z = \max\{U, L\}.
\]

Without loss of generality, let \( U \leq L \), so that \( Z = L \).

(i) Let \( p_c \) (\( p_k \)) be a parcel assigned to the lower (upper) conveyor with relation \( c \) (\( k \)). By swapping the assignment of both parcels, the total transportation of both conveyors changes:

\[
U' = U - d_k^u + d_c^u, \\
L' = L - d_c^u + d_k^u.
\]
Due to the sortation of inbound-outbound relations before applying Algorithm 1, we know that $d_k^b \leq d_k^a$ and $d_k^l \leq d_k^l$. Therefore, $U' \geq U$, $L' \geq L$ and

$$Z' = \max\{U', L'\} \geq \max\{U, L\} = Z.$$ 

(ii) Let $p_k$ be a parcel assigned to the upper conveyor with relation $k$. By changing the assignment of the parcel, the total transportation of both conveyors changes:

$$U' = U - d_k^a,$$

$$L' = L + d_k^l.$$

Hence, $U' \leq U$, $L' \geq L$ and

$$Z' = \max\{U', L'\} = L' \geq L = \max\{U, L\} = Z.$$

(iii) Let $p_c$ be a parcel assigned to the lower conveyor with relation $c$. By changing the assignment of the parcel, the total transportation of both conveyors changes:

$$U' = U + d_c^a,$$

$$L' = L - d_c^l.$$

We assume that the new assignment is an improvement to the former solution:

$$Z' < Z \Rightarrow \max\{U', L'\} < \max\{U, L\} = L \Rightarrow U' < L \Rightarrow d_c^a < L - U.$$

But considering lines 14-19 of Algorithm 1 there is no parcel on the lower conveyor with $d_c^a < L - U$. This contradiction to our assumption shows that $Z' \geq Z$, which completes the proof.

**Appendix E: Alternative DAP-model**

The alternative DAP-model minimizes the maximum number of parcels passing some segment $s \in S$:

$$\bar{Z}(\phi) = \max\{a_s(\phi) | s \in S\} + \max\{b_s(\phi) | s \in S\} \rightarrow \min \quad (21)$$

subject to (2), (3) and

$$a_s(\phi) = \sum_{i \in I, o \in O: \phi(i) \leq \phi(o), \phi(o) \leq s \leq \phi(i)} b_{io} - u_{io} + \sum_{i \in I, o \in O: \phi(i) > \phi(o), \phi(o) \leq s \leq \phi(i)} b_{io} - u_{io} + \sum_{i \in I, o \in O: \phi(i) > \phi(o), 1 \leq s \leq \phi(i)} b_{io} - u_{io}, \quad (22)$$

$$b_s(\phi) = \sum_{i \in I, o \in O: \phi(i) > \phi(o), \phi(o) \leq s < \phi(i)} u_{io} + \sum_{i \in I, o \in O: \phi(i) \leq \phi(o), \phi(o) \leq s \leq \phi(i)} u_{io} + \sum_{i \in I, o \in O: \phi(i) \leq \phi(o), 1 \leq s < \phi(i)} u_{io}, \quad (23)$$

28
where $a_s(\phi)$ and $b_s(\phi)$ define the number of parcels on the lower and upper conveyor passing segment $s$ for a given assignment $\phi$, respectively. Note that $b_s(\phi) = 0$, whenever only a single conveyor exists ($CON = 1$) or both conveyors run in the same direction ($CON = 2u$). Clearly, this problem is also NP-hard in the strong sense, which could be proven by a transformation from the 3-Partition problem very similar to that for DAP-MLS (see Appendix C).