Jena Research Papers in Business and Economics

Truck Scheduling in Cross Docking Terminals with fixed Outbound Departures

Nils Boysen, Dirk Briskorn, Martin Tschöke

10/2010

Jenaer Schriften zur Wirtschaftswissenschaft

Working and Discussion Paper Series
School of Economics and Business Administration
Friedrich-Schiller-University Jena

ISSN 1864-3108

Publisher:

Wirtschaftswissenschaftliche Fakultät
Friedrich-Schiller-Universität Jena
Carl-Zeiß-Str. 3, D-07743 Jena

www.jbe.uni-jena.de

Editor:

Prof. Dr. Hans-Walter Lorenz
h.w.lorenz@wiwi.uni-jena.de

Prof. Dr. Armin Scholl
armin.scholl@wiwi.uni-jena.de

www.jbe.uni-jena.de

Working Paper

Truck Scheduling in Cross Docking Terminals with fixed Outbound Departures

Nils Boysen[1], Dirk Briskorn[2], Martin Tschöke[3]

September 2010

[1]: Friedrich-Schiller-Universität Jena
Lehrstuhl für Operations Management
Carl-Zeiß-Straße 3, 07743 Jena, Germany
http://www.wiwi.uni-jena.de/pil/
nils.boysen@uni-jena.de

[2]: Universität zu Köln
Wirtschafts- und Sozialwissenschaftliche Fakultät
Albertus-Magnus-Platz, 50923 Köln, Germany
http://www.scmp.uni-koeln.de/mitarbeiter/briskorn.htm
briskorn@wiso.uni-koeln.de

[3]: Universität Hamburg
Institut für Industriebetriebslehre und Organisation
Von-Melle-Park 5, 20146 Hamburg, Germany
http://www.ibl-unihh.de/tea_tsc.htm
tschoeke@econ.uni-hamburg.de

Abstract

At a cross docking terminal, inbound shipments are directly transshipped across the terminal to designated outbound trucks, so that delays and inventories are kept as low as possible. We consider an operational truck scheduling problem, where a dock door and a start time has to be assigned to each inbound truck. A set of outbound trucks is scheduled beforehand and, therefore, departure times are fixed. If a shipment is not unloaded, transshipped to the outbound gate and loaded onto the designated outbound truck before its departure we consider the shipments's value as lost profit. The objective is to minimize total lost profit. The paper at hand formalizes the resulting truck scheduling problem. We settle its computational complexity and develop heuristics in order to tackle the problem. We show the efficiency of these heuristics by means of a computational study. Last but not least, a case study is presented.

Keywords: Cross docking terminal, truck scheduling, due dates, heuristics.

1 Introduction

Cross docking terminals are important nodes in modern distribution networks as they render possible a consolidation of multiple smaller shipments with equal destination to full truck loads, so that economies of scale in transportation can be gained. In these intermediate nodes inbound trucks are docked and shipments are unloaded. Then, all incoming shipments are registered, moved across the dock and directly loaded into their designated outbound trucks in order to leave the terminal towards their next destinations without any waiting time. Thus, cross docks merely carry intermediate stock during the transshipment process, which distinguishes them from traditional warehouses. Success stories on cross docking which resulted in considerable competitive advantages are reported for several industries with high proportions of distribution cost such as retail chains (Wal Mart; Stalk et al. [20]), mailing companies (UPS; Forger [9]), automobile manufacturers (Toyota; Witt [22]) and less-than-truckload logistics providers (Gue [11]).

However, compared to traditional point-to-point deliveries the consolidation in cross docks comes at the price of double-handling, which slows down the distribution process and jeopardizes on-time deliveries. Consequently, optimization procedures which aim to synchronize inbound and outbound flows by determining appropriate truck schedules at the dock doors receive more and more attention within recent years (see Boysen and Fliedner [2] for a detailed review). Truck scheduling assigns each truck to be processed a door and decides on the succession of trucks per door, so that it is the decision on the *where* and *when* of (un-)loading trucks which is supported by a truck scheduling procedure.

Existing research invariably presupposes that loads of outbound trucks are predetermined prior to truck scheduling, so that outbound trucks leave the cross dock only after all defined shipments have arrived at the terminal and are finally stowed in the respective truck. However, industries like postal services and less-than-truckload logistics, where firms mainly transport comparatively small and low-valued shipments of multiple senders, typically do not know the composition of inbound loads prior to opening the trailer. Thus, an outbound organization, where outbound trucks depart not before all dedicated shipments are loaded, is impossible. Instead, fixed outbound schedules are applied, which exactly define the points in time a truck serving a specific destination leaves the terminal. All shipments arriving in time to be stowed

in the respective truck before its departure are shipped the same day, whereas late shipments are delayed until the next day. Fixed schedules are especially important in larger hub-and-spoke networks where multi-stage cross docking is applied, because these networks depend on a reliable and steady flow of trucks.

A recent review article on truck scheduling at cross docking terminals is provided by Boysen and Fliedner [2]. Different truck scheduling procedures which vary, e.g., with regard to the number of dock doors, the objective function applied, whether or not transportation times inside the terminal are considered and whether or not commodities are interchangeable between outbound trucks, are presented by Boysen [1], Boysen et al. [3], Briskorn et al. [4], Chen and Lee [6], Chen and Song [7], Miao et al. [17] as well as Yu and Egbelu [23]. However, all of these papers concur in their assumption on the organization of outbound trucks' departures. They assume that any outbound truck leaves the terminal only after all predefined shipments are stowed, so that this paper is the first to treat truck scheduling with fixed outbound schedules.

As outbound schedules are assumed as given in this paper, the problem reduces to scheduling inbound trucks at a given set of inbound doors. If unloading a truck is interpreted as a job and a dock door is interpreted as a processor, our truck scheduling problem resembles a traditional scheduling problem with parallel processors, where the weighted number of late jobs is to be minimized. However, our problem is a generalization of this problem, as each inbound truck may contain multiple shipments with varying destinations and thus due dates. In our problem any job may have multiple due dates with varying weights, so that existing procedures for parallel machine scheduling, provided by Van den Akker et al. [21], Chen and Powell [8] as well as M'Hallah and Bulfin [16], cannot be applied. Furthermore, varying transshipment times for moving goods inside the terminal from inbound gates to outbound trucks are considered.

The paper at hand investigates a truck scheduling problem for cross docking terminals with fixed outbound schedules. For given departure times defined for any outbound destination we aim to schedule inbound trucks at multiple doors of a terminal, so that the total value of delayed shipments is minimized. For this purpose the remainder of the paper is organized as follows. We formally define the problem, develop a mathematical model, and settle the problem's complexity in Section 2. Two solution procedures being based on the separation of decisions about the where and when of truck processing are developed in Sections 3 and 4. Section 5 contains a computational study evaluating the suitability of our decomposition approaches. Then, Section 6 presents a real-world case for applying our truck scheduling problem at a real world express logistics provider and discusses appropriate short-term reactions on delayed shipments. Finally, Section 7 concludes the paper.

2 The truck scheduling problem with fixed outbound departures

2.1 Formal problem description

The truck scheduling problem with fixed outbound departures (TSFD) can be defined as follows. A set of outbound trucks O is given and for each outbound truck $o \in O$, a departure time d_o as well as the outbound door it is loaded at is determined beforehand. Furthermore, we are given a set I, $|I| = n$, of inbound trucks where each truck $i \in I$ is specified by an unloading time p_i and a value $w_{i,o}$ of commodities it delivers for outbound truck $o \in O$. The

cross docking terminal has a set G of inbound gates. Without loss of generality we assume all parameters to be integer.

A schedule is a subset $S \subset I \times G \times \mathbb{N}$. A triple $(i, g, C_i) \in S$ specifies that inbound truck i is assigned gate g and it is scheduled to complete unloading at exactly C_i. We say S is feasible if

1. for each $i \in I$ there is exactly one $(i, g, C_i) \in S$, that is each truck is scheduled exactly once and
2. for each pair of trucks $i, j \in I$, $i \neq j$, with $(i, g, C_i) \in S$ and $(j, g', C_j) \in S$ we have $g \neq g'$, $C_j < C_i - p_i + 1$ or $C_i < C_j - p_j + 1$, that is trucks are assigned to different gates or trucks' unloading intervals do not overlap.

For a feasible schedule S the performance is defined to be the total value of commodities unloaded from inbound trucks too late to be loaded onto the corresponding outbound truck before its departure. Turnaround time for commodities delivered by $i \in I$ and picked up by $o \in O$, that is a time lag for moving a shipment inside a terminal, e.g., by forklift, depends on the doors i and o are assigned to. Clearly, this transshipment time is much larger if trucks i and o are docked at far distant doors instead of neighboring ones. Let $t_{g,o}$ represent this time lag for each $g \in G$ and $o \in O$. We say that $i \in I$ is tardy with supplying $o \in O$ according to S, if

$$(i, g, C_i) \in S \text{ and } C_i + t_{g,o} > d_o.$$

Let $U(S) \subseteq I \times O$ denote the set of missed connections, that is $(i, o) \in U(S)$ if and only if $i \in I$ is tardy with supplying $o \in O$ according to S. Then,

$$f(S) = \sum_{(i,o) \in U(S)} w_{i,o}$$

reflects the total value of delayed shipments according to S. The TSFD is to find a feasible schedule S such that $f(S)$ is minimal among feasible schedules.

In order to enable a more intuitive understanding we provide Figure 1. It depicts a Gantt chart representing a schedule S for 14 inbound trucks and three gates. Six outbound trucks are scheduled in advance. Their due dates are given in increasing order as d_1 to d_6. Since $C_1 = C_{10} < d_1$ both inbound trucks 1 and 10 provide all outbound trucks non-tardily. Trucks 2, 3, 7, and 11 provide all outbound trucks non-tardily but the one leaving first. Since $\max\{C_i \mid i \in I\} < d_6$ outbound truck 6 is provided non-tardily by all inbound trucks. Summarizing, we have

$$\begin{aligned} U(S) \;=\; \{&(2,1),(3,1),(4,1),(4,2),(5,1),(5,2),(5,3),(6,1),(6,2),(6,3),(6,4),(6,5),\\ &(7,1),(8,1),(8,2),(9,1),(9,2),(9,3),\\ &(11,1),(12,1),(12,2),(13,1),(13,2),(13,3),(14,1),(14,2),(14,3),(14,4)\}. \end{aligned}$$

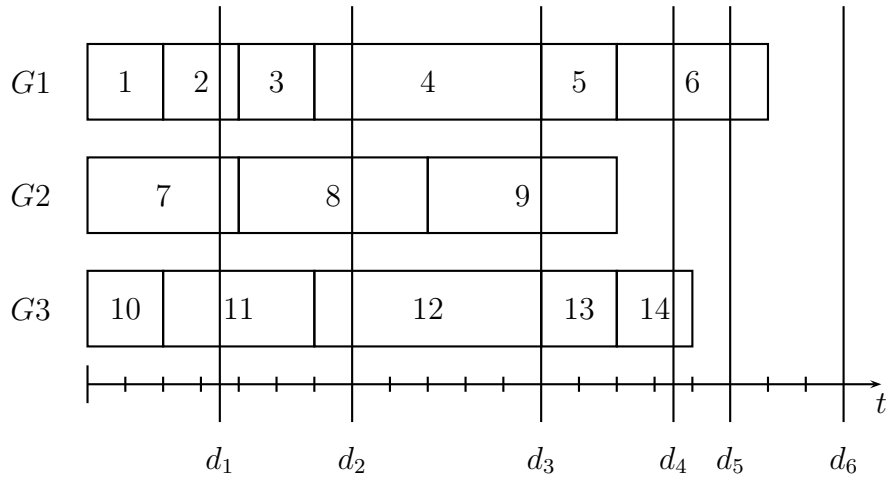

Figure 1: An exemplary schedule

2.2 Mathematical model

Applying the notation summarized in Table 1, TSFD can be represented as a linear mixed-integer program (TSFD-MIP) consisting of objective function (1) and constraints (2) to (9).

Objective function (1) seeks to minimize the total value of delayed shipments. Constraints (2) ensure that each inbound truck is scheduled for unloading. Inequalities (3) guarantee that for each gate no more than one sequence of trucks is composed. Constraints (4) ensure that the sequence of inbound trucks at each gate is well-defined. Constraints (5) define completion time C_i for each inbound truck i. Inequalities (6) ensure that $y_{i,o}$ equals one if the shipment delivered by inbound trucks i cannot reach a respective outbound truck o, that is if C_i plus movement time $t_{g,o}$ exceeds departure time d_o. Note that

$$M = \sum_{i \in I} p_i + \max \{t_{g,o} \mid g \in G, o \in O\}$$

is sufficiently large. Finally, constraint (7) fixes the completion time of virtual start truck 0 to zero and constraints (8) and (9) represent the binary integrality requirement of 0-1 variables.

2.3 Computational complexity

In the section at hand we settle the computational complexity of the TSFD and some special cases. First, we show that TSFD is strongly NP-hard even for one gate. Then, we briefly consider two special cases.

We introduce a problem which helps us to settle NP-hardness of TSFD. The problem is to schedule jobs on a single machine aiming to minimize total weighted tardiness. More formally, given

- a set $J = \{1, \ldots, n'\}$ of jobs and
- processing time p'_j, due date d'_j, and weight w'_j for each job $j \in J$,

I	set of inbound trucks with $I = \{1, 2, \ldots, n\}$
O	set of outbound trucks
G	set of inbound doors available for processing inbound trucks
p_i	processing time for unloading inbound truck i
d_o	departure time of outbound truck o
$t_{g,o}$	transshipment time from inbound dock g to the dock where outbound truck o is processed
$w_{i,o}$	weight, e.g., the number of products, of a shipment delivered by inbound truck i dedicated to outbound truck o
M	big integer
C_i	continuous variable: completion time of inbound truck i
$x^g_{i,j}$	binary variable: 1, if inbound truck j is processed directly after inbound truck i at gate g; 0, otherwise
$x^g_{0,i}$	binary variable: 1, if inbound truck i is processed first at gate g; 0, otherwise
$x^g_{i,n+1}$	binary variable: 1, if inbound truck i is processed last at gate g; 0, otherwise
$y_{i,o}$	binary variable: 1, if shipments delivered by inbound truck i are too late to reach outbound truck o; 0, otherwise

Table 1: Notation

TSFD-MIP

$$\text{Minimize } Z(C, X, Y) = \sum_{i \in I} \sum_{o \in O} w_{i,o} \cdot y_{i,o} \tag{1}$$

subject to

$$\sum_{g \in G} \sum_{\substack{j \in I \cup \{0\} \\ i \neq j}} x^g_{j,i} = 1 \qquad \forall\, i \in I \tag{2}$$

$$\sum_{j \in I} x^g_{0,j} \leq 1 \qquad \forall\, g \in G \tag{3}$$

$$\sum_{\substack{j \in I \cup \{0\} \\ i \neq j}} x^g_{j,i} = \sum_{\substack{j \in I \cup \{n+1\} \\ i \neq j}} x^g_{i,j} \qquad \forall\, i \in I;\, g \in G \tag{4}$$

$$C_i \geq C_j + p_i - M \cdot (1 - x^g_{j,i}) \qquad \forall\, i \in I;\, j \in I \cup \{0\};\, g \in G \tag{5}$$

$$y_{i,o} \cdot M \geq C_i - d_o + \sum_{g \in G} t_{g,o} \cdot \left(\sum_{\substack{j \in I \cup \{0\} \\ i \neq j}} x^g_{j,i} \right) \qquad \forall\, i \in I;\, o \in O \tag{6}$$

$$C_0 = 0 \tag{7}$$

$$x^g_{i,j} \in \{0,\, 1\} \qquad \forall i, j \in I \cup \{0, n+1\};\, g \in G \tag{8}$$

$$y_{i,o} \in \{0,\, 1\} \qquad \forall\, i \in I;\, o \in O \tag{9}$$

find a one-machine schedule, that is a permutation of J, such that

$$\sum_{j \in J} w'_j \cdot \max\left\{0, C_j - d'_j\right\}$$

is minimized where C_j is the completion time of job j. Following well-known three field notation (see Graham et al. [10]) this problem can be represented as $1||\sum w_jT_j$. As shown in Lawler [14], $1||\sum w_jT_j$ is strongly NP-hard.

Theorem 1. *TSFD is strongly NP-hard even if* $|G| = 1$ *and* $t_{g,o} = 0$ *for each* $g \in G$ *and* $o \in O$.

Proof. We reduce $1||\sum w_jT_j$ to TSFD as follows. Given an instance of $1||\sum w_jT_j$ we create an instance of TSFD by setting

- $G = \{1\}$,
- $I = \{1, \ldots, n'\}$,
- $p_i = p'_i$ for each $i \in I$,
- $O = \left\{0, \ldots, \sum_{j \in J} p'_j\right\}$,
- $d_o = o$ for each $o \in O$,
- $t_{g,o} = 0$ for each $g \in G$ and $o \in O$, and
- $w_{i,o} = \begin{cases} w'_i & \text{if } o \geq d'_i \\ 0 & \text{otherwise} \end{cases}$ for each $i \in I$ and $o \in O$.

Clearly, the reduction can be done in pseudo-polynomial time. It is easy to see that for two corresponding sequences of jobs and trucks respective completion times are identical. Note that shipments arriving on truck i reach exactly C_i outbound trucks tardily. For $\max\{0, C_i - d'_i\}$ and $\min\{d'_i, C_i\}$ of these outbound trucks we have $w_{i,o} = w'_i$ and $w_{i,o} = 0$, respectively. Then, the total value of delayed shipments of inbound truck i is given as

$$0 \cdot \min\{d'_i, C_i\} + w'_i \cdot \max\{0, C_i - d'_i\} = w'_i \cdot \max\{0, C_i - d'_i\}.$$

Clearly, this equals the weighted tardiness of job i in a corresponding sequence of jobs. Since this holds for each job and its corresponding truck it is obvious that two corresponding sequences are equally evaluated according to both problems. This completes the proof. □

The proof of Theorem 1 relies on the number of outbound trucks as well as unloading times of inbound trucks to be arbitrary. A natural question is whether the problem becomes easier if one or the other is not true. We briefly discuss these issues in the following. For the sake of shortness we only give basic ideas of reductions.

Theorem 2. *TSFD is solvable in polynomial time if* $p_i = p$ *for each* $i \in I$.

It is not hard to see that the problem reduces to the linear assignment problem if inbound trucks' unloading times are equal to p. Assigning truck i to the sth slot at gate g yields a total value of delayed shipments by i of $c_{i,g,s} = \sum_{o \in O} w_{i,o} \cdot u_{i,o}$ where

$$u_{i,o} = \begin{cases} 1 & \text{if } k \cdot p + t_{g,o} > d_o \\ 0 & \text{otherwise.} \end{cases}$$

A detailed description of the method is provided in Section 4.1.

Theorem 3. *TSFD is strongly NP-hard even if* $|O| = 1$ *and* $t_{g,o} = 0$ *for each* $g \in G$ *and* $o \in O$.

Again, we introduce a problem in order to sketch the proof. An instance of 3-PARTITION is specified by a set of $3m$ integer numbers $a_1, \ldots, a_{3m}$ where $\sum_{i=1}^{3m} a_i = mB$ and $B/4 < a_i < B/2$ for each $i \in \{1, \ldots, 3m\}$ for $B \in \mathbb{N}$. The problem is to find out whether there are subsets $A_1, \ldots, A_m$ of numbers such that $\sum_{a \in A_k} a = B$ for each $k \in \{1, \ldots, m\}$.

Having a single outbound truck with departure time d_1 a zero cost schedule gives us a partition of I into $|G|$ subsets such that the total processing time in each subset does not exceed d_1. If we set $d_1 = B$, $I = \{1, \ldots, 3m\}$, and $p_i = a_i$ for each $i \in \{1, \ldots, 3m\}$ a zero cost schedule for $|G| = m$ doors corresponds to a yes-answer to the instance of 3-PARTITION.

In addition to the fact that TSFD is hard to solve, now we can see that there is no constant-factor approximation algorithm for TSFD unless P=NP. This follows from the fact that instances of TSFD in our reduction have optimum objective value of zero if the answer to the instance of 3-PARTITION is yes. Hence, any constant-factor approximation algorithm would yield objective value of zero in these cases, as well, and, thus, would separate yes-instances from no-instances in polynomial time.

Lemma 1. *TSFD is NP-hard even if* $|O| = 1$ *and* $|G| = 1$.

Once more, we introduce a problem in order to sketch the proof. An instance of PARTITION is specified by a set A of integer numbers. The problem is to find out whether there is a subset A' of A such that $\sum_{a \in A'} a = \sum_{a \in A} a/2$.

Having a single outbound truck with departure time d_1 a zero cost schedule gives us a partition of I into $|G|$ subsets such that the total processing time in each subset does not exceed d_1. We set $G = \{1\}$, $O = \{1\}$, $d_1 = \sum_{a \in A} a/2$, $I = \{1, \ldots, |A|\}$, and $p_i = w_{i,1} = a_i$ for each $i \in \{1, \ldots, |A|\}$. The optimum schedule has an objective value of $\sum_{a \in A} a/2$ if and only if there is a yes-answer to the instance of PARTITION.

Lemma 2. *TSFD can be solved in pseudo-polynomial time if both,* $|G|$ *and* $|O|$*, are fixed.*

Proof. Let σ_g, $g \in G$, be the sequence of outbound trucks sorted in non-decreasing $d_o - t_{g,o}$. Let, furthermore, $I_{k,g}$, $k \in \{1, \ldots, |O| - 1\}$, be the subset of inbound trucks scheduled at gate g, $g \in G$, such that $i \in I_{k,g}$ if and only if $d_{\sigma_g(k)} - t_{g,\sigma_g(k)} < C_i \leq d_{\sigma_g(k+1)} - t_{g,\sigma_g(k+1)}$, that is each truck in $I_{k,g}$ provides outbound trucks $\sigma_g(1), \ldots, \sigma_g(k)$ tardily and outbound trucks $\sigma_g(k+1), \ldots, \sigma_g(|O|)$ non-tardily. Additionally, let $I_{0,g}$ be the subset of inbound trucks scheduled at gate g, $g \in G$, such that $i \in I_{0,g}$ if and only if $C_i \leq d_{\sigma_g(1)} - t_{g,\sigma_g(1)}$, that is each truck in $I_{0,g}$ provides no outbound truck tardily. Similarly, let $I_{|O|,g}$ be the subset of inbound trucks scheduled at gate g, $g \in G$, such that $i \in I_{|O|,g}$ if and only if $C_i > d_{\sigma_g(|O|)} - t_{g,\sigma_g(|O|)}$, that is each truck in $I_{|O|,g}$ provides each outbound truck tardily.

Consider an optimum schedule. The order of trucks in $I_{k,g}$, $k \in \{0, \ldots, |O|\}$, $g \in G$, can be determined according to non-decreasing truck index. This is due to the fact that reshuffling $I_{k,g}$ cannot increase the largest completion time of trucks in $I_{k,g}$.

Accordingly, we design a dynamic programming (DP) approach based on the idea of assigning trucks one by one to one of the aforementioned subsets. Consider state (i, P) where $P \in \mathbb{N}^{|G| \cdot (|O|+1)}$. State (i, P) describes a partial schedule where trucks $1, \ldots, i$ are assigned to

subsets and P is the vector of corresponding total processing times of each subset. Let $P(k,g)$ be the total processing time of the trucks assigned to $I_{k,g}$ for each $k \in \{0,\dots,|O|\}$ and $g \in G$. Then, we say a state (i,P) is feasible if

1. $i \in I$,
2. $\sum_{k=0}^{|O|} \sum_{g \in G} P(k,g) = \sum_{i'=1}^{i} p_{i'}$, and
3. $\sum_{k=0}^{l} P(k,g) \leq d_{\sigma_k(l+1)-t_{g,\sigma_g(l+1)}}$ for each $g \in G$ and $l \in \{0,\dots,|O|-1\}$.

Additionally, we consider the initial state $\left(0, 0^{|G|\cdot(|O|+1)}\right)$ where $0^{|G|\cdot(|O|+1)}$ is a vector consisting of $|G| \cdot (|O|+1)$ zeroes. The number of feasible states is in $O\left(nP^{|G|\cdot(|O|+1)}\right)$. We consider a transition from state (i,P) to $(i+1,P')$ for $0 \leq i < n$ if there is a $(k,g) \in \{0,\dots,|O|\} \times G$ such that

1. $P'(k',g') = P(k',g')$ for each (k',g') with $k' \neq k$ or $g' \neq g$ and
2. $P'(k,g) = P(k,g) + p_{i+1}$.

Such a transition represents truck $i+1$ being assigned to gate g such that it provides outbound trucks $\sigma_k(k+1),\dots,\sigma_k(|O|)$ non-tardily if $k < |O|$. If $k = |O|$ it represents truck $i+1$ being assigned to gate g such that it provides each outbound truck tardily. Due to condition 3, we can assign truck $i+1$ to a certain subset only if each outbound truck o can still be provided non-tardily by all inbound trucks assigned to subsets $I_{0,g},\dots,I_{o-1,g}$ for each $g \in G$.

The cost $c_{(i,P),(i+1,P')}$ of transition $((i,P),(i+1,P'))$ representing the assignment of $i+1$ to $I_{k,g}$ equals $\sum_{l=1}^{k} w_{i,\sigma_k(l)}$. Then, cost $c_{i,P}$ of (i,P) can be determined as

$$c_{i,P} = \min\{c_{i-1,P'} + c_{(i-1,P'),(i,P)} \mid \text{ transition } ((i-1,P'),(i,P)) \text{ exists}\}.$$

Solving TSFD, then, reduces to finding

$$\min\{c_{n,P} \mid (n,P) \text{ is feasible}\}.$$

In order to solve TSFD we have to evaluate $O\left(n \cdot |G| \cdot (|O|+1) \cdot P^{|G|\cdot(|O|+1)}\right)$ transitions. □

Theorem 4. *TSFD is ordinarily NP-hard if both $|G|$ and $|O|$ are fixed.*

Theorem 4 immediately follows from Lemmas 1 and 2.

Summarizing results in this section, TSFD is unary NP-hard even for rather restricted cases, e.g. for $|G| = 1$ or $|O| = 1$. For fixed $|G|$ and $|O|$, TSFD is binary NP-hard. However, this result is of theoretical interest mainly unless the run time complexity of the pseudo-polynomial algorithm outlined in the proof of Lemma 2 can be significantly reduced. Moreover, the problem is not only hard to solve but also hard to approximate. Thus, efficient heuristics are necessary in order to tackle real world problems. Last but not least, TSFD is solvable in polynomial time if unloading times for all inbound trucks are equal to each other. We use this result for the customization of a heuristic in Section 4.

3 First-sequence-then-assign

In this section we develop a heuristic based on the separation of two types of decisions specifying a solution to TSFD. In order to find a schedule we have to decide on

- the assignment of trucks to gates and
- the order of trucks assigned to the same gate.

First, we develop a DP based heuristic for the special case of TSFD with a single gate in Section 3.1. In this case, we have to consider the second decision type only. In order to consider instances with several gates, we generalize this approach in Section 3.2. Here, a sequence of trucks is generated in step one, which, then, serves as input for a list scheduling algorithm assigning trucks to gates in a second step.

3.1 The one gate case

Our approach is based on a DP method following the basic DP procedure for sequencing problems provided in Held and Karp [12]. Thus, the decision process is subdivided into n stages, where each stage $k = 1, \ldots, n$ represents a sequence position. Any stage k contains a set of states, where each state represents a possible subset $I' \subseteq I$, $|I'| = k$, of inbound trucks scheduled up to sequence position k. The optimum schedule for subset I' is found according to the recursive equations

$$h^*(I') = \min_{i \in I'} \left\{ h^*(I' \setminus \{i\}) + f_i \left(\sum_{j \in I'} p_j \right) \right\} \tag{10}$$

with $h^*(\emptyset) = 0$. Here, the objective value of the optimal schedule for trucks $I' \setminus \{i\}$ is added to the actual contribution $f_i(t)$ of finishing truck i at time t, where $f_i(t)$ is calculated according to

$$f_i(t) = \sum_{o \in O} w_{io} \cdot \gamma(t - d_o), \quad \text{with } \gamma(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise.} \end{cases}$$

The overall objective is to find optimal solution value $h^*(I)$ for the set of all inbound trucks being scheduled. We can determine $h^*(I)$ by a stage-wise forward recursion using (10). Finally, when the final stage is reached and optimal solution value $h^*(I)$ is determined, a simple backward recursion can be applied to determine the optimal sequence of truck processing. Note that without loss of generality this description is based on the assumption of all transshipment times being zero, that is $t_{1o} = 0$ for each $o \in O$. Otherwise, outbound departure times can simply be reduced by given transshipment times.

Obviously, there are $2^{|I|}$ states to be evaluated, so that computational complexity of DP amounts to $O(2^{|I|})$, see Held and Karp [12]. Hence, although this techniques is known to be more efficient than full enumeration it is to be expected that solving instances to optimality will be very time consuming. Consequently, we extend the DP approach by beam search (BS).

BS is a truncated breadth-first search heuristic and was first applied in speech recognition systems by Lowerre [15]. Since then, BS has been applied within multiple fields of application and many extensions have been developed. A review is provided by Sabuncuoglu et al. [19]. Instead of exploring the whole state space outlined above BS excludes some states from further consideration. The general idea is to reduce the number of states which have to be evaluated and, thus, save run time. The price, however, is that we may miss the optimum solution due to exclusion of states. The subset of states to be considered in higher stages is determined by heuristic choice in a filtering process.

Within such a multi-pass filtering process multiple measures are successively applied with increasing complexity, e.g., simple priority values, lower bounds and even upper bounds (see Ow and Morton [18]), to reduce the set of states to be further considered. This way, after the final filtering stage only BW nodes are chosen and further branched, so that BW is a basic control parameter of BS called *beam width*. Such a restricted branching is executed until the final stage is reached, where the best solution out of the set of considered states and the corresponding path from the initial state represent the result of BS.

To apply the general procedure of BS in a specific domain special filtering mechanisms must be specified with respect to the problem. In the following, we describe these specifications for TSFD with a single gate.

A two-stage filtering is applied. First, a rough filtering is applied to identify FW, $FW \geq BW$, (filter width) promising states per stage with a fast to compute priority value $pv(I_k)$ defined as

$$pv(I_k) = \frac{\sum_{o \in O} \sum_{i \in I_k} w_{io}}{\sum_{i \in I_k} p_i}.$$

In this filtering we store the FW states having highest priority values on stage k, $k = 1, \ldots, n$.

This way, especially those nodes which ensure an early scheduling of those inbound trucks with comparatively high penalty weights and comparatively short processing times are preferred.

Then, remaining FW nodes are further reduced to a number of BW per stage to be further considered in higher stages by determining a more sophisticated lower bound. This lower bound $LB(I_k)$ comprises the partial objective value of scheduling subset I_k according to the current state and a lower bound for the total value of delayed shipments of trucks in $I_k^* = I \setminus I_k$ if the first truck in I_k^* starts not before $\sum_{i \in I_k} p_i$.

Such a lower bound $LB(I_k)$ can be calculated by treating each outbound truck o separately. The resulting problem for a given o can be seen as an instance of the knapsack problem. Here, putting items in the knapsack represents scheduling trucks to be completed before d_o bearing a profit of non-tardy supply. Then, the continuous knapsack problem, represented by (11) to (13), can be solved to optimality for any outbound truck o by step-wise filling knapsack capacity $d_o - \sum_{i \in I_k} p_i$ according to nonincreasing relative weights w_{io}/p_i, see, e.g., Kellerer et al. [13].

$$\text{Minimize } LB_o(I_k^*) = \sum_{i \in I_k^*} w_{io} - \sum_{i \in I_k^*} w_{io} \cdot x_i \tag{11}$$

subject to

$$\sum_{i \in I_k^*} p_i \cdot x_i \leq d_o - \sum_{i \in I_k} p_i \tag{12}$$

$$0 \leq x_i \leq 1 \quad \forall\, i \in I_k^* \tag{13}$$

Clearly, $LB_o(I_k^*)$ gives a lower bound on the total value of delayed shipments for truck o. With these values on hand the overall lower bound is given as

$$LB(I_k) = OV(I_k) + \sum_{o \in O} LB_o(I_k^*).$$

where $OV(I_k)$ is the partial objective value of I_k according to the actual state. Employing these customized components our BS approach stage-wise identifies a subset of no more than BW promising states. Finally, when the last stage is reached the best solution value is returned as the solution of the BS approach.

An interesting question is whether we can incorporate release dates in this approach. Note that we have to take idle time into account then. Extending states to additionally keep track of the total idle time Q inserted between trucks scheduled yet we can derive completion time $C(I_k, Q)$ of the truck scheduled last from each state. Then, the next truck's unloading process is started at this point of time or at its release date depending on which is larger. While not complicated to design the extended approach will have higher run time requirements since the state space grows. However, given the heuristic nature of the BS approach it seems tolerable to use an estimate of Q derived from I_k in order to obtain $C(I_k, Q)$ approximately. Then, the state space is not further increased.

3.2 The general case

In the general case, $|G| > 1$ gates are available for processing inbound trucks. However, our first-sequence-then-assign heuristic first translates the actual problem data (with $|G| > 1$) into a one gate problem. Therefore, any input data is directly taken over except for the number of inbound gates, which is set to $|G'| = 1$, and departure times d'_o of outbound trucks:

$$d'_o = \left(d_o - \frac{\sum_{g \in G} t_{g,o}}{|G|} \right) \cdot |G| \quad \forall\, o \in O.$$

Modified departure times d'_o account for merely one inbound door being available for truck processing, so that actual departure times d_o minus average transshipment times to o are multiplied with the actual number of inbound doors $|G|$. This modified problem instance is then solved by the aforementioned BS procedure for the one gate case (step one). The result is a sequence of inbound trucks, which serves as a priority list for a list scheduling procedure applied in the assignment phase (step two) of our heuristic. Here, inbound trucks are assigned to gates one after another according to the given sequence by choosing dock door g^*, which minimizes average availability time of shipments delivered on current truck i^*:

$$g^* = \text{argmin}_{g \in G} \left\{ C_g + \frac{\sum_{o \in O} t_{g^*,o}}{|O|} \right\},$$

with C_g being the cycle time after processing all trucks currently assigned to door g. Then, the current cycle time of the selected door is updated by $C_{g^*} := C_{g^*} + p_{i^*}$. Finally (step three), a separate run of our BS procedure is executed for each gate, where the respective data instances are gained by simply extracting all inbound trucks being assigned to the respective door.

4 First-assign-then-sequence

In this section we propose a heuristic based on the separation of two types of decisions mentioned above, as well. However, contrary to the method in Section 3, we first decide the assignment of trucks to gates, here.

In Section 4.1 we propose an exact method for the special case of TSFD where all trucks' processing times are equal. Based on this method we develop a heuristic method in order to assign trucks to gates for the general version of TSFD in Section 4.2. While it provides sequences for trucks already these may not be optimal even for a given assignment of trucks to gates. Accordingly, we aim to improve obtained schedules by resequencing trucks assigned to the same gate using a DP approach similar to the one developed in Section 3.1.

4.1 The case of equal processing times

It is well-known, see, e.g., Brucker [5], that several scheduling problems can be reduced to the linear assignment problem. In particular, this holds for many machine scheduling problems where

- jobs have equal processing times and
- the objective is to minimize $\sum_{j \in J} f_j(C_j)$, that is total processing costs of jobs in J where processing cost of job j is given as a non-decreasing job-dependent function of its completion time C_j.

Being aware of this, it is not hard to see that we can create an instance of the linear assignment problem representing TSFD with $p_i = p$ for each $i \in I$. We consider the set of slots $S_g = \{1, \ldots, n\}$ at each gate $g \in G$. Assigning truck i to a certain slot $s \in S_g$ represents unloading i at gate g starting at $(s-1) \cdot p$ and finishing at $s \cdot p$. Note that we can reduce the set of slots such that

$$S_g = \{1, \ldots, \lceil n/|G| \rceil + \lfloor \max \{\max \{t_{g',o} \mid g' \in G\} - t_{g,o} \mid o \in O\} / p \rfloor\}$$

for each gate g since there is no need to schedule i as the last truck at gate g if all outbound trucks can be supplied earlier when scheduling i as last truck at another gate. Cost $c_{i,g,s}$ can be defined as shown in Section 2.3.

Now, we can formulate the TSFD with $p_i = p$ for each $i \in I$ as linear assignment problem as represented by (14) to (17).

Of course, an important question is whether the treated special case of TSFD has any practical meaning. In fact, this special case of TSFD is interesting on its own since often unloading times of trucks do not (significantly) differ from each other. For instance, in the case of only standardized trailers being fully loaded with standardized pallets being processed. Then, considering TSFD with $p_i = p$ seems to be an appropriate heuristic. However, even if we cannot allow to ignore differences in unloading times, then a solution to AP-MIP using modified unloading times may serve well as a start solution to be further improved. Based on this idea we design a heuristic for the general version of TSFD in Section 4.2.

AP-MIP

$$\text{Minimize } Z(Z) = \sum_{i \in I} \sum_{g \in G} \sum_{s \in S_g} c_{i,g,s} \cdot z_{i,g,s} \tag{14}$$

subject to

$$\sum_{g \in G} \sum_{s \in S_g} z_{i,g,s} = 1 \qquad \forall\, i \in I \tag{15}$$

$$\sum_{i \in I} z_{i,g,s} \leq 1 \qquad \forall\, g \in G;\, s \in S_g \tag{16}$$

$$z_{i,g,s} \in \{0,\, 1\} \qquad \forall\, i \in I;\, g \in G;\, s \in S_g \tag{17}$$

4.2 The general case

In this section we propose a heuristic for TSFD employing basic ideas developed in Sections 3.1 and 4.1. The basic scheme is as follows. First we obtain an assignment of trucks to gates and a temporary sequence of trucks for each gate by solving an instance of the linear assignment problem as developed in Section 4.1. Afterwards, we improve the sequencing decisions by employing the BS approach developed in Section 3.1 for each gate.

Given an instance of TSFD we derive an instance of the linear assignment problem by assuming that the unloading time of each truck equals (i) the minimum unloading time among trucks, (ii) the average unloading time of trucks, or (iii) the maximum unloading time among trucks. The solution to the linear assignment problem can be easily transformed into a feasible solution of TFSD without any gate idle time keeping the assignment of trucks to gates and sequences at each gate. We set the start time of each gate's first truck to zero and the start time of each other truck to the completion time of its immediate predecessor. Finally, for each door a separate BS run as developed in Section 3.1 is executed, so that the sequencing decision of all trucks assigned to the current gate is improved.

In the following, as in Section 3.1, we discuss possible extensions for incorporating release dates in this approach. There seems to be no way to incorporate release dates into the assignment problem structure without losing optimality for the case of equal processing times studied in Section 4.1. Note, however, that in the section at hand employing the assignment problem structure has a heuristic nature on its own since we manipulate processing times. Therefore, it seems appropriate to simply prohibit those slots starting before a truck's release date. Clearly, this will not slow down the approach.

5 Computational study

The computational study seeks to explore the solution performance of (i) dynamic programming (DP) and beam search (BS) for TSFD with one gate and (ii) our first-sequence-then-assign (FS) and first-assign-then-sequence (FA) procedures for the TSFD with multiple gates. Since no established test bed is available we first elaborate on instance generation. Note that the solution procedures described above have been implemented in C#.NET and run on a 2.1 GHz PC, with 2 GB of memory.

When we obtain optimal solutions in order to evaluate our approaches we do so by implementing optimization model TSFD-MIP in the off-the-shelf solver XPress-Optimizer (version 20.00.05).

5.1 Instance generation

Our instance generator receives the following initial input data to determine TSFD data instances:

- two sets of values specifying varying numbers of inbound and outbound trucks, $|I|$ and $|O|$
- a set of values specifying varying numbers of gates $|G|$
- a set of standard deviations for generating processing times
- the number of repetitions R per parameter constellation

With these data on hand, all parameter values are combined in a full-factorial design and per parameter constellation instance generation is repeated R times. Each single instance for a given parameter constellation is generated as follows:

For each inbound truck its processing time is randomly drawn following normal distribution $\mathcal{N}(\mu, \sigma^2)$ with $\mu = 30$ and σ as specified by the input parameter. Therefore, unloading a truck is assumed to last 30 minutes on average, which according to terminal managers is a typical real-world time span. Then, departure time d_o is determined by

$$d_o = \frac{\sum_{i \in I} p_i}{|G|} \cdot rnd[0.9; 0.5]$$

for each $o \in O$, where $rnd[x; y]$ represents an equally distributed random number between x and y. Weight w_{io} is chosen according to

$$w_{io} = \begin{cases} rnd[10; 1], & \text{if } rnd[1; 0] < 0.5 \\ 0, & \text{otherwise} \end{cases}$$

for each $i \in I$ and $o \in O$. Finally, transshipment times t_{go} are randomly drawn from interval $[1; 10]$ with equal distribution.

5.2 Solution performance for the one gate case

For testing the solution performance of DP and BS in the one gate case, our instance generator is initialized with parameter values $|I| \in \{9, 12, 15, 18\}$, $|O| \in \{3, 5, 7, 9\}$, $|G| = 1$, and $\sigma = 6$. Per parameter constellation instance generation is repeated $R = 100$ times, so that in total 1,600 data instances for the one door case of TSFD are generated.

First, Figure 2 depicts the runtime (measured by cpu seconds averaged over all instances per parameter value and labeled *avg cpu*) of DP in dependency of the number $|I|$ of inbound trucks. As was already proven with the theoretical runtime complexity, cpu seconds show

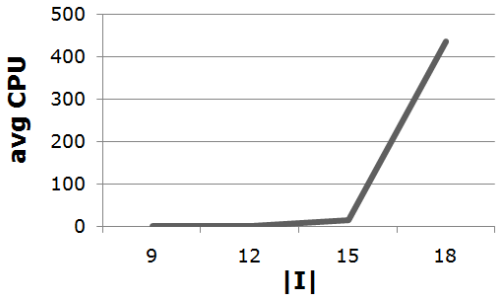

Figure 2: Runtime (avg cpu) of DP in dependency of the number $|I|$ of inbound trucks

an exponential increase. However, instances with up to 18 inbound trucks can be solved in reasonable time.

Thus, heuristic procedures are required to solve larger instances of real-world size. For this purpose, BS was introduced. The solution performance of BS depending on varying values of Filter Width (FW) and Beam Width (BW) is reported in Table 3, where % opt, avg gap and avg cpu denote the percentage of instances solved to optimality, the relative deviation from the optimum averaged over all instances and the average cpu seconds, respectively.

	BW				
FW	5	20	50	100	200
10	38/17.8/0.01	–	–	–	–
50	48/13.3/0.04	68/4.8/0.05	67/6.3/0.07	–	–
100	47/13.9/0.05	72/3.9/0.08	81/2.9/0.12	88/1.3/0.23	–
200	47/13.9/0.04	71/6.5/0.13	84/1.5/0.18	88/1.3/0.23	86/1.8/0.37
500	47/13.9/0.04	69/7.1/0.14	86/3.6/0.27	92/0.7/0.38	93/0.6/0.56

legend: % opt/avg gap/avg cpu

Table 2: Solution performance of BS for TSFD in the one gate case

The results indicate the elementary trade-off between runtime and solution quality, which can, in our case, be managed by an appropriate choice of control parameters FW and BW. On the one hand, BS can be applied as a very fast procedure with moderate solution quality, e.g., with $FW = 100$ and $BW = 20$ BS produces an average gap of 3.9% in less than 0.1 cpu seconds on average. On the other hand, increasing control parameters leads to much better solutions for the price of longer run times. However, also in this case BS seems to be an effective compromise. For instance, for the additional parameter constellation of $FW = 5000$ and $BW = 300$, BS shows an average gap of only 0.3% and still requires merely 1.38 cpu seconds on average. Thus, BS seems well suited for efficiently solving TSFD in the single gate case.

5.3 Solution performance for the general case

For testing solution performance if TSFD faces multiple doors, first small test instances are generated, for which optimal solutions can be gained by using the standard solver XPress-Optimizer. Therefore, we initialized the procedure for instance generation as follows: $|I| \in \{5, 6, 7, 8\}$, $|O| \in \{5, 7, 9\}$, $|G| \in \{2, 3, 4\}$, and $\sigma \in \{2, 4, 6, 8\}$. Along with $R = 5$ repetitions per parameter constellation, this resulted in 720 instances.

Aggregated results are reported in Table 3. Here, we compare optimal objective values gained by the standard solver XPress-Optimizer (labeled XPress) with the heuristic results of our first-assign-then-sequence (FA) procedures, where we differentiate between minimum (FA-MIN), average (FA-AVG), and maximum (FA-MAX) processing times being applied as intervals within the assignment step. Furthermore, we report the best results (FA-BEST) out of all three FA procedures, our first-sequence-then-assign procedure (FS) and the result of a simple first-come-first-serve (FCFS) policy. FCFS is emulated by simply applying the list scheduling approach of FS to a random inbound sequence. Note that FCFS is a widespread real-world policy for scheduling inbound trucks. Further note that control parameters, filter width (FW) and beam width (BW), for beam search (contained in FA and FS) are set to unrestrictive values, so that solution graphs are fully explored. As performance measures we report the number of instances, where an optimal solution was determined (#opt), the average absolute and relative deviation from optimal solutions value (avg abs and avg gap), as well as average cpu seconds per instance (avg cpu). Finally, we also report the solution time over all instances (sum cpu), since isolated solutions times of some approaches are barely measurable.

measure	XPress	FA-MIN	FA-AVG	FA-MAX	FA-BEST	FS	FCFS
#opt	720	423	462	447	509	130	21
avg abs	–	3.3	2.2	2.4	1.4	8.0	26.7
avg gap	–	11.6	7.8	8.4	4.8	20.3	78.5
avg cpu	61.5	<0.01	<0.01	<0.01	<0.01	0.02	<0.01
sum cpu	44.308	0.4	0.4	0.4	1.1	12.7	0.3

Table 3: Solution performance for small instances

The results reveal that XPress was able to solve all 720 instances to optimality. However, with 61.5 average cpu seconds solution time for these small instances is considerable. If the number of inbound trucks is further increased, XPress was not able to determine any optimal solution within a given time frame of 300 seconds. Thus, it can be concluded that XPress is not suited for solving TSFD instances of real-world size. Among our FA approaches applying the average processing time as the interval for the assignment problem (FA-AVG) performs best. However, this result does not hold for each single instance, so that taking the best out of all three FA procedures (FA-BEST) shows promising results with an average gap of merely 4.8 %. Note that successively applying all three intervals causes no additional implementation effort and still requires negligible solution time (avg cpu <0.01). On the other hand, FS and FCFS drop behind. With an average gap of 20.3% and 78.5% both approaches exhibit inferior results. Note that also the solution time of FS is longer then that of FA, since solving a single larger sequencing instance with beam search (FS) takes more time than several smaller ones (FA). However, while FS is inferior to all FA approaches it clearly yields better solutions than FCFS. Note that FCFS is an approach often employed in the real world since it is easy to implement and easy to understand. Therefore, managers may benefit from FS since it provides a similar intermediate result, that is, a sequence of trucks, which has to be scheduled by some kind of list scheduling algorithm. Obviously, our results imply that the sequence used in FS is better suited than the FCFS list.

Furthermore, we investigate the impact of the parameters for instance generation on the solution quality (avg gap) of our best performing procedure (FA-BEST) as depicted in Figure 3:

a) The smaller the standard deviation σ of processing times the better the performance of

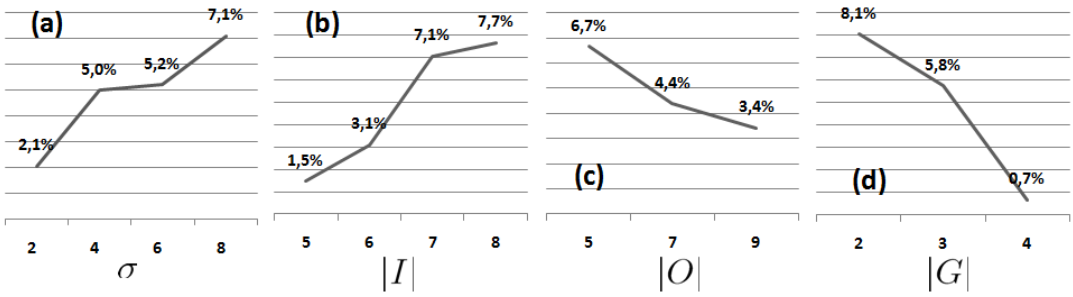

Figure 3: Solution performance of FA-BEST in dependence of parameters for instance generation

FA-BEST. With smaller variation the assumption of identical processing times applied within the assignment step of FA turns out as a much better approximation. Thus, FA seems especially suited if variations in processing time are small, which is a widespread phenomenon in real-world cross docking, since, typically, standardized trucks containing shipments being packed on standardized pallets are processed at a cross docking terminal.

b) With increasing number $|I|$ of inbound trucks solution performance decreases, because an increasing solution space impedes finding high quality solutions.

c) With increasing number $|O|$ of outbound trucks solution performance increases. If any inbound truck carries shipments for multiple destinations a heuristic scheduling decision leading to a late shipment can be outweighed by multiple other destinations, so that single (wrong) decisions lose influence on solution values.

d) The more gates $|G|$ are available at a terminal the better is the solution performance we observe. With decreasing gate number (ceteris paribus) the number of trucks assigned to each single gate increases and deviations of actual completion times compared to those assumed by the assignment problem (being based on equidistant processing times) increase with a longer queue of inbound truck per gate. As a typical cross docking terminal has a large number of dock doors this effect should balance the negative effect of increasing inbound number, when applying FA in real-world settings.

Finally, large test instance of real-world size are investigated. Instance generation is executed applying the following parameter values: $|I| \in \{40, 60, 80, 100\}$, $|O| \in \{5, 7, 9\}$, $|G| \in \{10, 20, 30\}$, and $\sigma \in \{2, 4, 6, 8\}$. Again, instance generation is repeated $R = 5$ times, so that 720 instances are generated. Results are summarized in Table 4. As optimal solution values can not be gained, we report the improvement of our heuristic procedures compared to real-world policy FCFS. Therefore, the average gap of each single instance is determined as follows: $\frac{Z(FCFS)-Z(H)}{Z(FCFS)}$, where $H \in$ {FA-MIN, FA-AVG, FA-MAX, FA-BEST, FS}. Note that beam search (BS) was executed with $FW = 100$ and $BW = 50$.

The results of Table 4 reveal that again FA-AVG performs best among our three first-assign-then-sequence (FA) procedures. All three approaches (and obviously the best of all three (FA-BEST), too) impressively improve FCFS solution values and clearly outperform the FS procedure. The runtime of all FA procedures is still negligible. Even for most the challenging instances with 100 inbound trucks and 10 gates solution time never surmounts 0.6 cpu seconds. Note that FA instances with $|I| = 100$ and $|G| = 10$ require the most solution time, since the single door TSFD problems to be solved per door receive most inbound trucks. Again, the FS

measure	FA-MIN	FA-AVG	FA-MAX	FA-BEST	FS	FCFS
avg abs	265.8	295.3	286.0	298.0	255.9	–
avg gap	42.5	47.1	42.3	47.6	40.1	–
avg cpu	0.06	0.06	0.06	0.17	37.5	$<$0.01
sum cpu	39.5	42.3	41.2	123.0	27,023	0.2

Table 4: Solution performance for large instances

procedure proves inferior, since solving a single TSFD with one gate problem with all inbound trucks raises maximum solution times to up to 164 cpu seconds ($|I| = 100$ and $|G| = 30$). Comparing FS to FCFS we, again, observe that FS provides much better results. Over all computational tests we conclude that especially our FA-BEST procedure finds solutions with satisfying solution values in fractions of a second even for large instances of real-world size.

6 Managerial implications

In this section we give an example on how to apply TSFD in a real-world cross dock setting. Therefore, we were in close contact with two yard managers of the DHL air-hub in Leipzig (Germany). Leipzig airport is one of three major hubs in DHL's world-wide logistics network for express postal services. Each night about 50 airplanes arrive from all over the world, express shipments are unloaded, sorted by a huge fully automated sorting system and re-loaded onto re-starting aircraft heading for their next destinations the same night. In addition to shipments arriving by aircraft, express goods forwarded in Germany (within a radius of a few hundred kilometers) enter the hub terminal by truck via one of a few dozens of dock doors available. DHL aims at fast and reliable deliveries all around the world and, thus, faces very tight schedules. Moreover, dock doors at the hub are typically scarce, so that the truck scheduling problem turns out to be a crucial decision problem, whose structure exactly resembles that of TSFD.

Each airplane has a given landing and start time as defined by the flight schedule, so that each inbound shipment arriving by truck faces a fixed departure time of its dedicated outbound aircraft. Any outbound destination is assigned a packing station ("reload section"), where special air containers are packed with goods arriving via the sorting system. Any shipment reaching the packing station prior to the "cut-off-time" (start time of outbound plane minus container loading time minus transportation time via industrial truck from packing station to aircraft parked on apron) is packed in a container and loaded the same day, whereas shipments arriving late remain at packing stations and are loaded only the next day. Therefore, each real-world aircraft can be represented by an outbound truck within TSFD and cut-off-times at the packing stations can directly be applied as their respective departure times d_o.

To offer track-and-trace services to its customers, DHL continuously updates the current location of any shipment. This information is used to communicate the number and properties of incoming shipments to the hub prior to any inbound truck's arrival. Therefore, anticipating unloading times of any truck from historical data is uncritical and deterministic values of p_i in TSFD are no shortcoming in the DHL-case. The transshipment time of a shipment is the time span from the dock door to its packing station, so that indeed transshipment times vary from gate to gate and outbound destination to outbound destination. As distances are bridged by a system of fully automated conveyor belts and sorting devices directing shipments via defined

routes, transshipment times $t_{g,o}$ can also be anticipated with great precision. Finally, weights $w_{i,o}$ for penalizing delayed shipments need to be determined. DHL offers its customers a money-back program if shipments arrive late, where the promised due date depends on the service booked by the customer. Thus, each weight can be set to the fine falling due, when missing the cut-off-date at the hub causes a late arrival at the customer. If despite delaying a shipment up to the next day, time is still sufficient to meet the appointed due date, then the weight can either be set to zero, or the money-back amount could be weighted with a scaling factor $0 < \alpha < 1$, so that the increasing risk of reducing the time buffer for additional delays in later distribution steps is penalized.

With this data on hand, TSFD can be applied at the air-hub in a rolling planning horizon, as is described by Boysen [1] for another truck scheduling problem in the food industry: Once trucks are completely unloaded and free gates, the TSFD is solved with all inbound trucks currently waiting on the yard. Typically, expected inbound trucks which have not yet arrived are not considered, because their actual arrival time heavily depends on unforeseeable traffic situations. Then, basic input data is adopted. All inbound trucks still under processing need to be fixed up to their final completion, so that these gates are only available after the current trucks left the terminal. The inbound trucks assigned to the free gates of period one in the solution to TSFD are called up to occupy the unassigned gates. This procedure is repeated as soon as the next inbound door is freed.

Recall that a typical real-world policy for truck scheduling is the first-come-first-serve (FCFS) policy. This policy is also applied at the DHL air-hub for scheduling inbound trucks. However, we have already shown the considerable disadvantage of the FCFS policy compared to our heuristic procedures (see Section 5.3), so that we abstain from additional experiments. Instead, we investigate a question DHL managers identified as most relevant, which is the question for the right reaction if TSFD detects late shipments. Typically, if delays occur they cause not a single but several shipments to be late, so that comparatively high contractual penalties threaten. Moreover, it violates the general philosophy of an express logistic provider to simply accept late shipments. Therefore, one of the following reactions can be accomplished if delays impend or repeatedly occur during daily operations:

- At least over a mid-term horizon a terminal can be extended, so that *additional gates* are available for truck scheduling. Cross docking terminals are typically light-weight built, so that (re-)construction cost are comparatively low. However, in the DHL-case an additional gate also requires connection to the sorting system, so that extending the hub-terminal requires considerable effort.

- Alternatively, *departure times* can be postponed. Such a manipulation of departure times might not always be possible, if succeeding stages of a distribution network rely on unchanged time tables. However, in the DHL-case cut-off-time can be postponed, if additional "ramp agents" are applied to speed-up the loading process of aircraft.

- Finally, *transshipment times* can be reduced. In a conventional terminal this would require additional workforce, e.g., forklift drivers, whereas DHL can (up to a certain maximum) increase conveyor speed.

To gather a general idea about the impact of these reactions with regard to reduced penalty cost, the following additional experiment was conducted. 10 basic instances of TSFD are

generated with our instance generator (see Section 5.1) by applying the following parameters: $|I| = 48$, $|O| = 15$, $|G| = 10$, and $\sigma = 6$. Then, the aforementioned reactions are emulated by: (i) stepwise increasing the number of gates ($|G| = 10, \ldots, 30$), (ii) stepwise increasing any departure time d_o by one time unit, and (iii) stepwise reducing transshipment times $t_{g,o}$ by one time unit per gate g and outbound truck o until $t_{g,o} = 0$. Figure 4 displays the average relative improvement (avg inc in %) in relation to the initial TSFD solution value (gained by FA-BEST), when increasing the door number and departure times and decreasing transshipment times by Δ units.

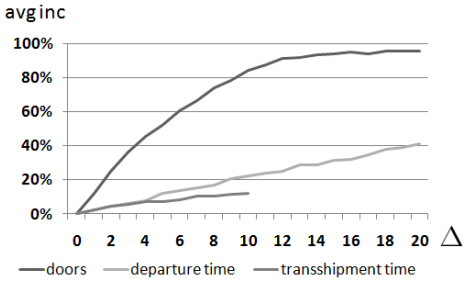

Figure 4: Improvement of initial solution value by applying reactions to an increasing extent

Clearly, general implications are hard to gain considering only a stylized test. To determine the right reaction to threatening delays in a specific cross dock setting, alternative costs for reactions (additional doors, postponed departure or decreased transshipment time) need to be traded off for the gains of reduced delays. However, the results of Figure 4 indicate, that gains promised by additional gates are highest. Thus, when planning a cross dock over-dimensioning of the terminal with regard to its gate number should be seriously considered. This, especially holds true for light-weight terminal buildings, because here comparatively low additional construction cost might be rapidly earned when opening additional gates will avoid penalty cost for delayed shipments. Already a few standby gates allow an over-proportional reduction of delayed shipments while avoiding extra cost for later on extending an existing terminal. On the other hand, Figure 4 reveals, that postponed departures and reduced transshipment times are of the same value until minimum transshipment times (zero in our instances) are reached. Therefore, additional manpower is only sensible up to a certain extent.

7 Conclusion and Outlook

In this paper we investigate a scheduling problem of inbound trucks at a cross docking terminal (TSFD), where outbound trucks operate on a given departure schedule, so that each inbound truck faces several due dates and varying time lags for transshipping goods inside the terminal. In addition to some complexity results, we develop an efficient heuristic procedure, which shows a satisfying average gap of less then 5% compared to optimal results obtained with an off-the-shelf solver and is able to solve even instances with 100 inbound trucks and 30 gates in fractions of a second. Furthermore, we present a real-world case of a provider of express postal services, where the application of TSFD and parameter estimation is discussed.

Future research should concentrate on additional solution procedures. On the one hand, meta heuristics could be developed to further improve solutions quality (for the price of additional solution time). On the other hand, exact solution procedures are required for testing heuristic

procedures in a more challenging test bed. A second branch of research should focus on generalizations of TSFD. For example, as discussed in Sections 3.1 and 4.2, release dates complicate the application of our approaches. In real world applications differing from the one outlined in Section 6 trucks may arrive over time and, therefore, release dates have to be taken into account.

References

[1] N. Boysen. Truck scheduling at zero-inventory crossdocking terminals. *Computers & Operations Research*, 37:32–41, 2010.

[2] N. Boysen and M. Fliedner. Cross dock scheduling: Classification, literature review and research agenda. *Omega*, 38:413–422, 2010.

[3] N. Boysen, M. Fliedner, and A. Scholl. Scheduling inbound and outbound trucks at cross docking terminals. *OR Spectrum*, 32:135–161, 2010.

[4] D. Briskorn, B.-C. Choi, K. Lee, J. Leung, and M. Pinedo. Complexity of single machine scheduling subject to nonnegative inventory constraints. *European Journal of Operational Research*, 207(2):605–619, 2010.

[5] P. Brucker. *Scheduling Algorithms*. Springer, Berlin, 2004.

[6] F. Chen and C.-Y. Lee. Minimizing the makespan in a two-machine cross-docking flow shop problem. *European Journal of Operational Research*, 193:59–72, 2009.

[7] F. Chen and K. Song. Minimizing makespan in two-stage hybrid cross docking scheduling problem. *Computers & Operations Research*, 36:2066–2073, 2009.

[8] Z. Chen and W. Powell. Solving parallel machine scheduling problems by column generation. *Informs Journal on Computing*, 11:78–94, 1999.

[9] G. Forger. Ups starts world's premiere cross-docking operation. *Modern Material Handling*, Nov:36–38, 1995.

[10] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimisation and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:236–287, 1979.

[11] K. Gue. The effect of trailer scheduling on the layout of freight terminals. *Transportation Science*, 33:419–428, 1999.

[12] M. Held and R. Karp. A dynamic programming approach to sequencing problems. *SIAM Journal*, 10:196–210, 1962.

[13] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Berlin, 1 edition, 2004.

[14] E. Lawler. A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, 1:331–342, 1977.

[15] B. T. Lowerre. *The HARPY Speech Recognition System*. PhD thesis, Carnegie-Mellon University, USA, 1976.

[16] R. M'Hallah and R. Bulfin. Minimizing the weighted number of tardy jobs on parallel processors. *European Journal of Operational Research*, 160(2):471–484, 2005.

[17] Z. Miao, A. Lim, and H. Ma. Truck dock assignment with operational time constraint within crossdocks. *European Journal of Operational Research*, 192(1):105–115, 2009.

[18] P. Ow and T. Morton. Filtered beam search in scheduling. *International Journal of Production Research*, 26:35–62, 1988.

[19] I. Sabuncuoglu, Y. Gocgun, and E. Erel. Backtracking and exchange of information: Methods to enhance a beam search algorithm for assembly line scheduling. *European Journal of Operational Research*, 186:915–930, 2008.

[20] G. Stalk, P. Evans, and L. Shulman. Competing on capabilities: The new role of corporate strategy. *Harvard Business Review*, 70(2):57–69, 1992.

[21] J. Van den Akker, J. Hoogeveen, and S. Van de Velde. Parallel machine scheduling by column generation. *Operations Research*, 47:862–872, 1999.

[22] C. Witt. Crossdocking: Concepts demand choice. *Material Handling Engineering*, 53(7): 44–49, 1998.

[23] W. Yu and P. J. Egbelu. Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *European Journal of Operational Research*, 184:377–396, 2008.