Column Generation for Outbound Baggage Handling at Airports

Markus Frey, Rainer Kolisch, Christian Artigues

Column Generation for Outbound Baggage Handling at Airports

Frey Markus, Kolisch Rainer[1] and Christian Artigues [2]

[1]TUM School of Management, Technische Universität München, Munich, Germany,
markus.frey@tum.de, rainer.kolisch@tum.de

[2]LAAS–CNRS/Université de Toulouse, CNRS, Toulouse, France
artigueslaas.fr

Abstract

The planning of outbound baggage handling at international airports is challenging. Outgoing flights have to be assigned and scheduled to handling facilities at which the outgoing baggage is loaded into containers. To avoid disruptions of the system the objective is to minimize workload peaks over the entire system. The resource demand of the jobs, which have to be scheduled, is depending on the arrival process of the baggage. In this paper we present a time-indexed mathematical programming formulation for planning the outbound baggage. We propose an innovative decomposition procedure in combination with a column generation scheme to solve practical problem instances. The decomposition significantly reduces the symmetry effect in the time-indexed formulation and also speeds up the computational time of the corresponding Dantzig-Wolfe formulation. To further improve our column generation algorithm we propose state-of-the-art acceleration techniques for the primal problem and pricing problem. Computational results based on real data from a major European Airport show that the proposed procedure reduces the maximal workloads by more than 60% in comparison to the current assignment procedure used.

keywords: outbound baggage handling; binary programming; column generation; cutting planes

1 Introduction

The number of flight passengers worldwide is rapidly growing at about 5.3% per year (see SITA (2014)), which is challenging for the existing infrastructure at international airports. The increasing number of travelers corresponds to a growing volume of baggage transferred through the terminal. Airports have three baggage streams: Inbound baggage, transfer baggage and outbound baggage. While inbound baggage is brought from incoming flights to the baggage claim hall where passengers pick up their baggage, transfer baggage is brought with baggage-tugs from the flight to infeed-stations, located at the apron, where the baggage is transferred into the Baggage Handling System (BHS). The BHS is a conveyor belt network transporting baggage through the terminal to any destination within particular periods of time. Like the check-in baggage, entering the BHS through check-in counters, the transfer baggage is labeled as outbound baggage once it has entered the BHS. Outbound baggage, comprised of transfer and check in baggage, represents the greatest baggage volume to be handled at hub airports.

In this paper we will focus on the planning of outbound baggage. After receiving outbound baggage from check-in passengers or transfer flights, the BHS transports the outbound baggage to its destinations, a handling facility or a central storage system. If baggage for an outgoing flight arrives at the check-in or at a transfer infeed-station during the flight's baggage handling period, which begins one to three hours before the scheduled departure time and ends 10 to 15 minutes before a flight's departure time, the arriving baggage is directed to the assigned handling facility, where ground handlers sort and load the incoming bags into containers. In the case bags arrive before baggage handling has started, the bags are directed to a central storage system, where they are stored until baggage handling begins. For example, the capacity of storage system at in Terminal 2 of Munich Airport is limited to about $3,000$ bags, which is rather small in comparison to the total amount of approximately $30,000$ bags which arrive per day. Figure 1 illustrates the outbound baggage flow before (see Figure 1 (a)) and after the start of a flight's baggage handling (see Figure 1 (b)). At the end of the baggage handling process, the containers are transported to the airplane to be loaded into the cargo hold.

The handling facility, also denoted carousel, is an ovalshaped conveyor belt on which several flights can be processed at one time (see Figure 3). A slide system connects the BHS conveyor belt with the carousel's conveyor belt. Bags from the BHS which are transferred to the carousel via the slide

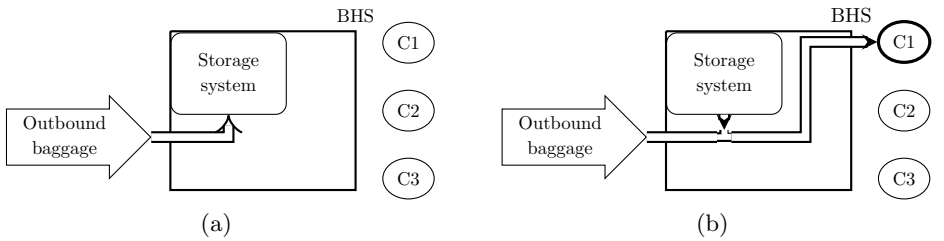


Figure 1: Outbound baggage stream of a flight through the BHS before (a) and after the start of the baggage handling and storage depletion (b). The oval- shaped forms indicate the three available carousels C1 to C3.

system are picked up from the carousel's conveyor belt by workers and loaded into containers. If the arrival rate of bags to the carousel exceeds the loading rate of the workers at the carousel, the number of bags on the carousel is increasing. The number of bags on the carousel defines the carousel's workload. If the workload of a carousel reaches the carousel's capacity, i.e. the maximum number of bags which can be on the carousel, the bags in the BHS' conveyor system are no longer forwarded to the carousel's conveyor belt and remain in the BHS until the workload on the carousel falls below its capacity. From a technical perspective, since infeed stations and the carousels are connected by the same lane system, additional bags in the BHS increase the danger of bottlenecks in the network, which may result in delays and disruptions in the outbound baggage handling process. Delayed baggage might not arrive at the assigned carousel in time and, hence, can not be loaded on the corresponding airplane lowering the service quality of the airport. From the employees' perspective, high workload peaks may lead to an unfair distribution of work among the workers. To avoid both problems, airports seeks to keep the workload on the carousels under given target value during the peak periods of the baggage flow. For example, if we consider Figure 2, showing the amount of baggage arriving at Terminal 2 of Munich Airport during a day, we can identify three peaks, one in the morning, one in the afternoon and one in the evening.

To staff the workers at the carousels adequately, it is important for ground handlers to know the assigned carousels and departing flights' baggage handling periods several hours before the handling of the flights starts. Therefore, this paper focuses on generating a robust plan for the outbound baggage handling for the whole planning day which can be used as a basis for ground handler staffing. The planning of outbound baggage handling

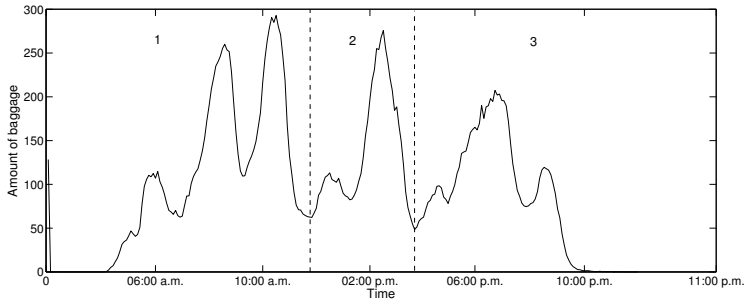


Figure 2: Average arrival curve of baggage during the course of a day

comprises the assignment of flights to the carousels, the scheduling of the start time of flights' handling process and the start time for the storage depletion. Most airports generate a daily plan by using simple allocation heuristics which is then manually re-optimized by a dispatcher. Because the general problem is NP-hard to solve (see §3) and over 350 flights at 22 carousels have to be handled per day on average at an international Airport such as Terminal 2 at Munich Airport, the generated plans result in poor solutions with high workloads on the carousels leading to the problems described above. As the number of available carousels is rather small in comparison to the number of outgoing flights and an expansion of the handling facilities is very costly, the airport seeks to find advanced approaches which reduce the workload peaks subject to the given infrastructure. In this paper, we present a model and a solution procedure to assign outgoing flights to the carousels and schedule the baggage handling, taking into account the BHS' capacities and the arrival profiles of the baggage streams. The objective is to obtain a workload below a given target value across all carousels during the main peak periods of the day.

Literature on planning the outbound baggage handling processes is very scarce. Abdelghany et al. (2006) and Ascó et al. (2013) (see also Ascó et al. (2011)) propose greedy assignment algorithms to assign baggage handling facilities to departing flights assuming a given service periods for flights' baggage handling. However, they neither consider capacities of the airport's infrastructure for baggage handling nor the dynamics of the baggage arrival process in their planning. Furthermore, instead of assuming given handling periods for flights' baggage handling, we determine the schedule for baggage handling, which makes our approach more flexible. A preliminary mixed-integer program for a similar problem statement to ours is presented by Frey

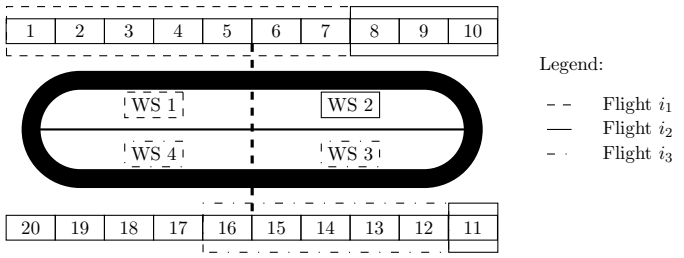


Figure 3: Feasible assignment of three flights at a carousel with 20 parking positions and four working stations

and Kolisch (2010) and Frey et al. (2010) but no efficient method is proposed to solve the problem.

Our main contributions in this paper are as follows:

(1) A time-indexed formulation (TIF) for the outbound baggage handling problem (OBHP) considering the dynamics of the incoming baggage stream and the capacities of airport's infrastructure is presented;

(2) Structural properties and the problem complexity are established;

(3) An efficient solution methodology based on guided column generation is presented which speeds up the convergence time of a standard column generation implementation;

(4) The proposed solution procedure outperforms the airport's manually generated solution by 65.23% in average.

The remainder of this paper is organized as follows. In §2 we provide a formal problem description of the OBHP. The time-indexed mathematical model is presented in §3, where we also discuss structural properties in §3.2 and exhibit dominance criteria to strengthen the solution space in §3.3. The solution procedure is proposed in §4, which decomposes the model into a master problem (MP) and three scheduling subproblems. Techniques to improve the column quality are discussed in §5, and §6 presents computational experiments. Conclusions are drawn in §7.

2 Problem description

The set of all outgoing flights is denoted by $\mathcal{F} = \{1, \ldots, F\}$. As airports plan flights' outbound baggage handling in periods, e.g. 5 minutes, baggage

handling for all flights takes place within a discrete planning horizon $\mathcal{T} = \{t_0, \ldots, t_T\}$ with $t_0 = 0 < t_1 < \ldots < t_T$. All time points are evenly spaced in time, and each time t_k represents the begin of period $[t_k, t_{k+1}[$ for $k = 0, \ldots, T-1$. The capacity of the central storage system is given by K^{s}.

Once baggage enters the BHS, it is transferred to the storage system or to the corresponding carousel slides. To meet a given service quality, the baggage transfer has to take place within a given time window (see Tarău et al. (2011)), e.g. 3 to 7 minutes at Terminal 2 of Munich Airport. As the period length of each t is about the size of the time window for the baggage transfer, it is reasonable to estimate the baggage transfer time by means of the average transfer time. Hence, the amount of baggage $A_i = (A_{i,t})_{t=0,\ldots,T-1}$ arriving during period t at the central storage or any carousel is given by the amount of baggage arriving at the check-in counter or transfer infeed-stations plus the average transfer time of baggage. As the arrival time of passengers at the check-in counter and the arrival time of transfer flights is uncertain, arrival process A_i is a random parameter which has to be estimated properly (see §6.1).

To avoid a high utilization, airport's objective is to obtain an utilization near or below a target utilization $u^{\mathrm{ta}} < 1$ across all carousels during working day's main disjunctive peak periods $\mathcal{T}_1, \ldots, \mathcal{T}_M$ with $M \geq 1$ and $\mathcal{T} = \bigcup_{m=1}^{M} \mathcal{T}_m$ (see Figure 2). Next, we give a detailed description of the assignment process of flights to the carousels and the scheduling of flight baggage handling; the required notation for the assignment and scheduling is summarized in Table 1.

Table 1: Carousel type and flight parameters

Carousel type $r \in \mathcal{R}$	
K_r^{cb}	Conveyor belt capacity
K_r^{pp}	Number of parking positions for containers
K_r^{ws}	Number of working stations
K_r^{ppws}	Number of parking positions in a working station segment
Flight $i \in \mathcal{F}$	
A_i	Baggage arrival vector
P_i	Number of required containers
$\left[W_{i,r}^{\min}, W_{i,r}^{\max}\right]$	Minimal and maximal number of required working stations at carousels type r
$\left[W_i^{\min}, W_i^{\max}\right]$	Minimal and maximal number of required working stations across all carousels
$\left[S_i^{\mathrm{es}}, S_i^{\mathrm{ls}}\right]$	Time window to start flight's baggage handling
S_i^{e}	End time of flight i's baggage handling

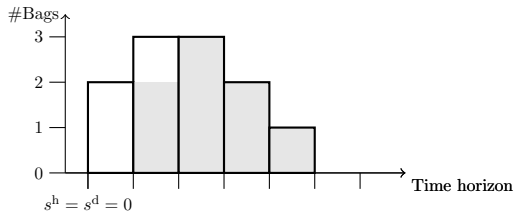
Flight assignment to the carousels In the assignment decision each flight is assigned to a carousel $c \in \mathcal{C} = \{1, \ldots, C\}$ with two long-sides, where each long-side of a carousel is comprised of a limited number of working stations and offers space for a limited number of flight containers on its parking positions lined-up along the long-side of a carousel's conveyor belt. On both sides, a carousel has the same number of working stations as well as the same number of parking positions. For example, the carousel in Figure 3 offers two working stations and 10 parking positions on each side. We distinguish between $R = \{1, \ldots, R\}$ carousel types where the set of carousels of type $r \in \mathcal{R}$ is denoted by $\mathcal{C}_r = \{1, \ldots, C_r\}$ with $\mathcal{C}_{r_1} \cap \mathcal{C}_{r_2} = \emptyset$ for each pair $r_1, r_2 \in \mathcal{R}$ with $r_1 \neq r_2$; index r_c indicates the type of carousel c. The carousel types differ in the number of offered parking positions K_r^{pp}, working stations K_r^{ws} and the number of bags K_r^{cb} which can be on a carousel's conveyor belt in a period (see Table 1). A carousel's parking position can accommodate one container at a time. The number of required containers for flight i, given by P_i, is provided by the airline and considers sorting criteria for baggage such as first, economy or transfer baggage. To meet airline's defined sorting criteria when loading baggage, the airport is forced to place all P_i containers simultaneously at the carousel at a time. Each flight has to be assigned to exactly one carousel, but a carousel can handle more than one flight at a time. At a working station, workers load bags circling on the conveyor belt into containers. To ensure that each bag is placed into the right container, each bag and container is identified with a unique bar-code. With a hand-held scanning device of a working station, a worker scans the bar-code of a bag and a container. If the two codes match, the bag is loaded into the containers. For security reasons, a working station is equipped with exactly one scanning device which can only be used for one flight at a time. Once a working station is assigned to a flight, the working station can not interrupt the handling for the assigned flight.

In addition to the carousels' resource capacities, the assignment has to obey assignment regulations described in the following: To ease the towing of flight containers with a baggage tug to the aircraft, the containers are sequentially ordered. If flight's containers are assigned to both sides of the carousel, then the container rows are lined up opposite of each other and start at the rightmost or leftmost parking position on each side such that the two container rows of the flight remain easily accessible for a baggage-tug (see flight i_2 in Figure 3). Each working station belongs to a segment of the carousel's conveyor belt and has a given number of K_r^{ppws} uniquely adjacent parking positions lined-up along its segment. In Figure 3, for example, each working station has five adjacent parking positions. To reduce bags' lift

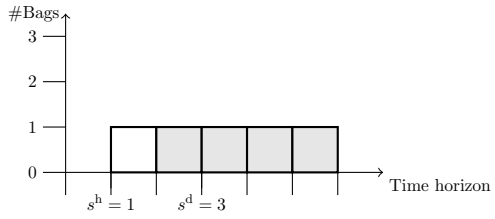
distances for workers from the conveyor belt to the containers, a flight can only be handled at a working station if at least one of the flight's containers is placed on one of the parking positions in its segment. A flight's container-row can overlap with parking positions of another working stations segment without using its working station (see flight i_1 in Figure 3). Moreover, the airport does not allow a worker to load baggage for parking positions of two working station segments, i.e. a worker can serve at most $2 \cdot K_r^{\text{ppws}} - 1$ parking positions. Thus, the minimal number of working stations $W_{i,r}^{\min}$ for flight i requiring P_i containers at a carousel type r is equal to the number of working station segment which could be completely covered by the flight's containers, i.e. $W_{i,r}^{\min} = \max\left\{\left\lfloor \frac{P_i}{K_r^{ppws}} \right\rfloor, 1\right\}$. For flight i_1 in Figure 3, for example, we have $W_{i,r}^{\min} = \left\lfloor \frac{7}{5} \right\rfloor = 1$. Furthermore, given the assignment regularities, the maximal number of working stations for flight i at carousel type r is given by $W_{i,r}^{\max} = \left\lceil \frac{P_i}{K_r^{ppws}} \right\rceil + \mathbb{1}_{>1}(P_i)$; $\mathbb{1}_{\mathcal{B}}(x)$ is equal to 1 if $x \in \mathcal{B}$, and 0 otherwise. For example, given the carousel type of Figure 3, flight i_1's container rows could be placed on parking positions 5 to 11 such that up to $W_{i_1,r}^{\max} = \frac{7}{5} + 1 = 3$ working station segments are reached.

Flights baggage handling schedule The start time of flight i's baggage handling has to be within time window $\left[S_i^{\text{es}}, S_i^{\text{ls}}\right]$. The earliest and latest start time for flight i's baggage handling, S_i^{es} and S_i^{ls}, respectively, are set according to flight i's expected number of bags (see Table 3 (a)). The depletion of flight i's stored bags can only start, if flight i's baggage handling has already been started. As soon as flight i's storage depletion begins, all stored bags of flight i are transferred with a given transfer rate from the central storage to the assigned carousel. To demonstrate the effect of the two scheduling decisions, start time of a flight's baggage handling and start time of the storage depletion, consider the baggage arrival process $A_i = (3, 2, 1, 0, 0, 0)$ for a flight i. Let us assume that the flight is assigned to one working station offering a loading rate of one bag per period. For the central storage system we assume a capacity of at least three bags and a transfer rate of one bag per period to the carousel. If the start time of the baggage handling and the start time of the storage depletion is zero (see Figure 4 (a)), three bags arrive on carousel's conveyor belt in period Δ_0 from which one bag can be immediately loaded. Thus, two bags remain on the conveyor belt until period Δ_1 in which two additional bags arrive leading to a workload of three. The calculation of the workload in the next periods is accordingly. If the start time of the baggage handling and the start time of the storage depletion are postponed by one and three periods,

respectively, the three bags arriving in period Δ_0 are stored in the central storage system resulting in a maximal workload of one in total (see Figure 4 (b)). The baggage handling for flight i ends at time S_i^{e} which is set some



(a)



(b)

Figure 4: The corresponding workload for baggage arrival process $A_i = (3, 2, 1, 0, 0, 0)$, if the start time of the baggage handling s^{h} and the start time storage depletion s^{d} is 0 (see (a)), and the start time of the baggage handling and storage depletion are different with $s^{\mathrm{h}} = 1$ and $s^{\mathrm{d}} = 3$ (see (b)). Gray shaded bars show the number of bags from the previous period

time periods, e.g. 10 minutes, before the scheduled departure time to have time to transport and load flight i's containers into airplane's cargo hold.

In the following, we denote a baggage handling schedule by start time tuples $\tau = \langle s_\tau^{\mathrm{h}}, s_\tau^{\mathrm{d}} \rangle$, where decision s_τ^{h} and s_τ^{d} stand for the start time of the baggage handling at a carousel and start time of storage depletion, respectively. We denote by $\mathcal{S}_i$ the set of flight i's feasible start time tuples. A start time tuple τ is feasible for flight i iff $s_\tau^{\mathrm{h}} \in \left[S_i^{\mathrm{es}}, S_i^{\mathrm{ls}}\right]$, and $S_i^{\mathrm{e}} - \Delta(s_\tau^{\mathrm{h}}) > s_\tau^{\mathrm{d}} \geq s_\tau^{\mathrm{h}}$ where $\Delta(s_\tau^{\mathrm{h}})$ represents an offset for the storage depletion, guaranteeing that all stored baggage up to time s_τ^{h} is transferred from the storage system to the carousel several periods, e.g. 30 minutes, before baggage handling ends. The later part of the paper requires to distinguish between feasible start time tuples for different number of assigned working stations. Therefore, we introduce subset $\mathcal{S}_i(w) \subseteq \mathcal{S}_i$ denoting the subset of feasible start time tuples, if w working stations assigned are assigned to flight i. Start time

tuples $\tau_1, \tau_2 \in \mathcal{S}_i$ are ordered with relation $\preceq$ where $\tau_1 \preceq \tau_2$ iff $s^{\mathrm{h}}_{\tau_1} < s^{\mathrm{h}}_{\tau_2}$ or iff $s^{\mathrm{h}}_{\tau_1} = s^{\mathrm{h}}_{\tau_2}$ and $s^{\mathrm{d}}_{\tau_1} \leq s^{\mathrm{d}}_{\tau_2}$.

3 Mathematical model

To solve the OBHP, we will show in §3.1 that it is sufficient to solve a TIF. The TIF constraints the assignment of flights to the carousels such that the carousels' capacities in terms of number of parking positions and working stations are not violated as well as the scheduling of flights' baggage handling at the carousels subject to the capacity of the central storage. Having a feasible solution for the TIF, the required order of containers at the carousels can be derived by a post-processing procedure in polynomial time as we will show in Theorem 3. For the TIF, we provide a mathematical analysis in §3.2. As the model includes a high number of variables, we present in §3.3 dominance criteria to strengthen the solution space.

3.1 Time indexed formulation

Threshold values $\mathcal{U} = \{u_1, \ldots, u_k\}$ with $0 < u_1 < \ldots < u_K$ measure the excess from target utilization u^{ta} of each carousel during any time interval $1 \leq m \leq M$. A deviation greater than u_k incurs a penalty of $0 < p_1 < \ldots < p_k < \ldots < p_K$ for $k = 1, \ldots, K$.

The number of feasible start time tuples for each flight i in set $\mathcal{S}_i$ is bounded by $\max_{i \in \mathcal{F}} \{|\mathcal{S}_i|\} \leq \max_{i \in \mathcal{F}} \left\{ \sum_{0 \leq z \leq S^{\mathrm{ls}}_i - S^{\mathrm{es}}_i} S^{\mathrm{e}}_i - (S^{\mathrm{es}}_i + z) \right\}$. Given flight i's arrival process A_i, a feasible start time tuple $\tau \in \mathcal{S}_i$ and a given number of working stations w, we can derive the following two parameters for flight i (see Appendix B):

- $\Gamma^{i,w}_{t,\tau}$ – flight i's workload on a carousel during period t;
- $\Phi^{i}_{t,\tau}$ – the amount of flight i's stored baggage in the central storage system during period t.

The key decision variable for the presented TIF is the four indexed binary variable $x_{i,c,w,\tau}$, which is equal to 1 if flight i is processed at carousel $c \in \mathcal{C}$ using w working stations and employs start time tuple τ. Binary auxiliary variable $y_{c,k,,m}$ is equal to 1, if the utilization of carousel c exceeds target utilization u^{ta} by u_k in time interval m. The TIF can now be stated as follows

$$\text{minimize} \quad \sum_{c \in \mathcal{C}} \sum_{1 \leq k \leq K} \sum_{1 \leq m \leq M} p_k \cdot y_{c,k,m} \tag{1}$$

subject to

$$\sum_{c\in\mathcal{C}} \sum_{W_{i,r}^{\min}\le w\le W_{i,r}^{\max}} \sum_{\tau\in\mathcal{S}_i(w)} x_{i,c,w,\tau} = 1 \quad \forall\ i\in\mathcal{F} \tag{2}$$

$$\sum_{i\in\mathcal{F}:S_i^{\mathrm{es}}\le t<S_i^{\mathrm{e}}} \sum_{W_{i,r}^{\min}\le w\le W_{i,r}^{\max}} \sum_{\tau\in\mathcal{S}_i(w):s_\tau^{\mathrm{h}}\le t} w\cdot x_{i,c,w,\tau} \le K_r^{\mathrm{ws}} \quad \forall\ r\in\mathcal{R}, c\in\mathcal{C}_r, t\in\mathcal{T} \tag{3}$$

$$\sum_{i\in\mathcal{F}:S_i^{\mathrm{es}}\le t<S_i^{\mathrm{e}}} \sum_{W_{i,r}^{\min}\le w\le W_{i,r}^{\max}} \sum_{\tau\in\mathcal{S}_i(w):s_\tau^{\mathrm{h}}\le t} P_i\cdot x_{i,c,w,\tau} \le K_r^{\mathrm{pp}} \quad \forall\ r\in\mathcal{R}, c\in\mathcal{C}_r, t\in\mathcal{T} \tag{4}$$

$$\sum_{i\in\mathcal{F}:S_i^{\mathrm{es}}\le t<S_i^{\mathrm{e}}} \sum_{c\in\mathcal{C}} \sum_{W_{i,r}^{\min}\le w\le W_{i,r}^{\max}} \sum_{\tau\in\mathcal{S}_i(w)} \Phi_{t,\tau}^{i}\cdot x_{i,c,w,\tau} \le K^{\mathrm{s}} \quad \forall\ t\in\mathcal{T} \tag{5}$$

$$\frac{1}{K_r^{\mathrm{cb}}}\cdot \sum_{i\in\mathcal{F}:S_i^{\mathrm{es}}\le t<S_i^{\mathrm{e}}} \sum_{W_{i,r}^{\min}\le w\le W_{i,r}^{\max}} \sum_{\tau\in\mathcal{S}_i(w)} \Gamma_{t,\tau}^{i,w}\cdot x_{i,c,w,\tau} - \sum_{1\le k\le K} u_k\cdot y_{c,k,m} \le u^{\mathrm{ta}}$$
$$\forall\ r\in\mathcal{R}, c\in\mathcal{C}_r, 1\le m\le M, t\in\mathcal{T}_m \tag{6}$$

$$\sum_{1\le k\le K} y_{c,k,m} \le 1 \quad \forall\ c\in\mathcal{C}, 1\le m\le M \tag{7}$$

$$x_{i,\tau,c,w}\in\{0,1\} \quad \forall\ i\in\mathcal{F}, c\in\mathcal{C}, \tau\in\mathcal{S}_i, W_{i,r_c}^{\min}\le w\le W_{i,r_c}^{\max} \tag{8}$$

$$y_{c,k,m}\in\{0,1\} \quad \forall\ c\in\mathcal{C}, 1\le k\le K, 1\le m\le M \tag{9}$$

Objective function (1) minimizes the sum of all penalty values for the violation of threshold value in the M time intervals. Constraints (2) ensure that for each flight $i\in\mathcal{F}$ a feasible start time tuple of set $\mathcal{S}_i$ is selected and each flight i is assigned to one carousel and to $W_{i,r}^{\min}\le w\le W_{i,r}^{\max}$ working stations. Capacity constraints (3) and (4) impose the resource restrictions given by the available number of working stations and parking positions at a carousel, respectively. The capacity of the central storage system is bounded by constraints (5). Constraints (6) set the violation of the threshold value during time interval m. The threshold value for the capacity violation in each time interval $1\le m\le M$ is selected in constraints (7).

A solution of the TIF does not consider the assignment rules described in §2. However, given a feasible solution for problem formulation (1)-(9) it is always possible to map it into a solution respecting the assignment rules in a post-processing step.

Theorem 1. *Given a TIF optimal solution it is always possible obtain an optimal OBHP solution in polynomial time.*

The proof of Theorem 1 (see Appendix A) is constructive and contains a polynomial time assignment algorithm to obtain a feasible solution for the OBHP.

3.2 Model analysis

The TIF contains at most $(3 \cdot F \cdot \max_{i \in \mathcal{F}} \{|\mathcal{S}_i|\} + K \cdot M) \cdot C$ variables and $T \cdot (C \cdot (2 + M) + 1) + C \cdot M + F$ constraints. Consider a real-world scenario with 350 flights, a flights' baggage handling time window of 120 minutes and 22 carousels for a planning horizon of 236 periods, where we plan from 03:00 a.m. to 10:40 p.m. in five minutes intervals. If the number of threshold values and time intervals is equal to three, the TIF leads to at most $554,598$ variables and $34,246$ constraints.

As part of the TIF, the question arises whether a feasible solution exists, i.e. whether F flights (items) can be assigned and scheduled to C carousels (bins). Hence, the decision problem can be reduced from the Bin-Packing Problem, which is known to be NP-complete (see Garey and Johnson (1979)) yielding the TIF's and, hence, OBHP's complexity.

Theorem 2. *The OBHP is NP-complete.*

However, the NP-completeness is not only established because of the assignment problem, but also the scheduling problem is NP-hard to solve. Assume a feasible assignment of flights to carousels is given and the scheduling decision to start flights' baggage handling still has to be done such that a minimized utilization is obtained. Let us consider one carousel only and assume that each flight $i \in \mathcal{F}$ (items) stores s_i bags (item size) in the central storage at time t_0. At the carousel with F working stations we assume a loading rate of B bags per period with $B \geq s_i$ and the stored bags can be sent within the time intervals t_1 to t_T (bins) to the carousel. Then the answer to the question weather it is possible to obtain a workload of 0 on the carousel would also provide a answer to the decision problem whether it is possible to store the items in the T bins.

Theorem 3. *Given a feasible assignment of flights to the carousels, the scheduling problem to obtain a minimized workload is NP-complete.*

3.3 Preprocessing

We strengthen the solution space by tightening the bound for the storage capacity in constraints (5) and by reducing the number of decision variables $x_{i,\tau,c,w}$. In the central storage system, all bags of a flight i arriving up to

period $S_i^{\text{es}} - 1$ have to be stored independently of selected start time tuple $\tau \in \mathcal{S}$. Taking all flights into account, which can start their baggage handling only after time t, storage capacity K^{s} in constraints (5) can be strengthened by $K_t^{\text{s}} = K^{\text{s}} - \sum_{i \in \mathcal{F}: t < S_i^{\text{es}}} \sum_{t \in \mathcal{T}: t < S_i^{\text{es}}} A_{i,t}$ for all $t > 0$. For $t = 0$ we set $K_0^{\text{s}} = K^{\text{s}} - \sum_{i \in \mathcal{F}} A_i^{\text{early}}$ where A_i^{early} stands for the amount of early baggage which has arrived before time 0. The number of variables $x_{i,\tau,c,w}$ heavily depends on the cardinality of sets $\mathcal{S}_i$. To reduce the number of feasible start time tuples, and thus, the number of variables, we strengthen the time windows for the baggage handling and we exhibit dominance criteria for two different start time tuples of the same flight i.

Earliest start time Assume for flight $i \in \mathcal{F}$ that no bag has arrived up to the earliest baggage handling starting time S_i^{es}. Thus, if baggage handling is delayed by one period to $S_i^{\text{es}} + 1$, there will be no consumption of the central storage capacity by flight i. At the same time, the time span in which flight i is assigned to the carousel is decreased. As the objective function does not increase when starting flight i at $S_i^{\text{es}} + 1$ instead of S_i^{es}, the earliest start time can be set to $S_i^{\text{es}} = S_i^{\text{es}} + 1$. The update is repeated as long as no bags have arrived to the current earliest start time for baggage handling.

Latest start time Assume that flight i is scheduled with the latest possible start time tuple $\tau_i^{\text{ls}} = \langle S_i^{\text{ls}}, S_i^{\text{ls}} \rangle$ while all other flights which can be handled in parallel with flight i start the baggage handling and the storage depletion at the earliest possible time. If the capacity of the storage system is violated at any time $t \in \left[S_i^{\text{ls}}, S_i^{\text{e}}\right[$, the latest start time of a flight i does not lead to a feasible solution. Therefore, flight i's latest start time can be left-shifted by one period. Let $\mathcal{F}_i^{\text{parF}}$ be the the subset of flights which can be handled in parallel with flight i. If inequality $\left(\sum_{j \in \mathcal{F}_i^{\text{parF}}} \Phi^j_{t,\tau_j^{\min}}\right) + \Phi^i_{t,\tau_i^{\text{ls}}} \leq K_t^{\text{s}}$ with $\tau_j^{\min} = \min\{\tau \in \mathcal{S}_j \mid \tau \preceq \tau' \text{ for all } \tau' \in \mathcal{S}_j\}$ is violated for at least one $t \in [S_i^{\text{es}}, S_i^{\text{e}}]$, the latest start time is updated to $S_i^{\text{ls}} = S_i^{\text{ls}} - 1$. The update is repeated as long as the capacity of the central storage system is violated when start time tuple τ_i^{ls} is selected.

Arbitrary start time tuple Consider two start time tuples τ_1 and τ_2 with the same start time for the baggage handling and in which the start time of the storage depletion in τ_1 is earlier then in τ_2. Assuming that the workload imposed by start time tuple τ_1 is less than or equal to the workload imposed by start time tuple τ_2 in any period, i.e. $\Gamma^{i,\bar{w}}_{t,\tau_1} \leq \Gamma^{i,\bar{w}}_{t,\tau_2}$ for

one $W^{\min} \leq \bar{w} \leq W^{\max}$. While both start time tuples occupy the assigned carousel for the same period of time, start time tuple τ_1 uses the storage for a shorter period than start time tuple τ_2 and leads to a lower workload on the carousel. Start time tuple τ_1 is therefore preferable to, or dominates, start time tuple τ_2.

Proposition 1. *Let $\tau_1, \tau_2 \in \mathcal{S}_i$ be two feasible start time tuples for flight $i \in \mathcal{F}$ with $s^h_{\tau_1} = s^h_{\tau_2}$ and $s^d_{\tau_1} < s^d_{\tau_2}$. If there is a $W_i^{min} \leq \bar{w} \leq W_i^{max}$ such that $\Gamma^{i,\bar{w}}_{t,\tau_1} \leq \Gamma^{i,\bar{w}}_{t,\tau_2}$ for all $t \in \mathcal{T}$, then τ_2 can be removed from set $S_i(w)$ for $\bar{w} \leq w \leq W_i^{max}$ without excluding the optimal solution.*

Proposition 1 is not valid for two start time tuples $\tau_1, \tau_2 \in \mathcal{S}_i$ with $s^{\mathrm{h}}_{\tau_1} < s^{\mathrm{h}}_{\tau_2}$, as a postponed start for baggage handling may violate the central storage's capacity. However, if start time tuple τ_2 never leads to a violation of the storage capacity, and if the workload imposed by τ_2 is less then or equal to the workload imposed by τ_2 in any period, then tuple τ_2 dominates tuple τ_1 as the flight is assigned to the carousel for a shorter time period with no drawback in terms of the objective function and storage capacity.

Proposition 2. *Let $\tau_1, \tau_2 \in \mathcal{S}_i$ be two feasible start time tuples for flight $i \in \mathcal{F}$ with $\tau_1 \prec \tau_2$. Assume there is a $W_i^{min} \leq \bar{w} \leq W_i^{max}$ such that $\Gamma^{i,\bar{w}}_{t,\tau_1} \geq \Gamma^{i,\bar{w}}_{t,\tau_2}$ for all $t \in \mathcal{T}$, and inequality $\left(\sum_{j \in \mathcal{F}_i^{parF}} \Phi^j_{t,\tau_j^{max}}\right) + \Phi^i_{t,\tau_2} \leq K^s_t$, with $\tau_j^{max} = \max\{\tau \in \mathcal{S}_j \mid \tau \succeq \tau' \text{ for all } \tau' \in \mathcal{S}_j(\bar{w})\}$, is satisfied for any $t \in \left[S^{es}_i, s^d_{\tau_2} + p^d_i(s^h_{\tau_2}) - 1\right]$ where $p^d_i(s^h_\tau) \geq 0$ is the duration to send all stored bags to the assigned carousel if the baggage handling start at s^h_τ. Then tuple τ_1 can be removed from sets $\mathcal{S}_i(w)$ for all $\bar{w} \leq w \leq W_i^{max}$ without excluding the optimal solution.*

In our computational study in §6, Proposition 1 and 2 lead to a column reduction of 35.16% and 11.25%, respectively (see Table 7 and 4).

4 Solution methodology

Solving the TIF with a branch-and-cut algorithm as it is implemented in off the shelf MIP-Solvers such as CPLEX results in a bad convergence behavior with high computational times. Reasons are the OBHP's complexity, the large number of variables and the large number of equivalent assignment patterns of flights to the carousels due to the symmetry effect, i.e. during the branch-and-cut procedure an given assignment of flights to the carousels and flights' baggage handling schedule are investigated more than one time. Indeed, we could not solve any real-world instance using CPLEX.

To overcome these problems, we provide a Dantzig-Wolfe formulation of the TIF leading to a reduction of symmetry. The resulting MP requires the definition of a duty $d \in \mathcal{D}_r$ representing a feasible assignment of flights to a carousel of type $r \in \mathcal{R}$ and schedule for flights' baggage handling period. A duty d contains the information

- $\Theta^{\mathrm{a}}_d = \left(\Theta^{\mathrm{a}}_{d,i}\right)_{i \in \mathcal{F}}$ – $\Theta^{\mathrm{a}}_{d,i}$ is equal to 1, if flight i is contained in duty d;
- $\Theta^{\mathrm{s}}_d = \left(\Theta^{\mathrm{s}}_{d,t}\right)_{t \in \mathcal{T}}$ – $\Theta^{\mathrm{s}}_{d,t}$ gives the number of bags in the central storage of flights assigned to duty d during time period t;
- H_d – penalty value for the threshold violations summed up over all time intervals.

Then, binary variable z_d is 1, if duty $d \in \mathcal{D} = \bigcup_{r \in \mathcal{R}} \mathcal{D}'_r$ is selected, and 0 otherwise. The MP, which is equivalent to the TIF, can now be stated as

$$\text{minimize} \quad \sum_{r \in \mathcal{R}} \sum_{d \in \mathcal{D}_r} H_d \cdot z_d \tag{10}$$

subject to

$$\sum_{r \in \mathcal{R}} \sum_{d \in \mathcal{D}_r} \Theta^{\mathrm{a}}_{d,i} \cdot z_d = 1 \quad \forall\, i \in \mathcal{F} \tag{11}$$

$$\sum_{r \in \mathcal{R}} \sum_{d \in \mathcal{D}_r} \Theta^{\mathrm{s}}_{d,t} \cdot z_d \leq K^{\mathrm{s}}_t \quad \forall\, t \in \mathcal{T} \tag{12}$$

$$\sum_{d \in \mathcal{D}_r} z_d \leq |\mathcal{C}_r| \quad \forall\, r \in \mathcal{R} \tag{13}$$

$$z_d \in \{0, 1\} \quad \forall\, d \in \mathcal{D}_r \tag{14}$$

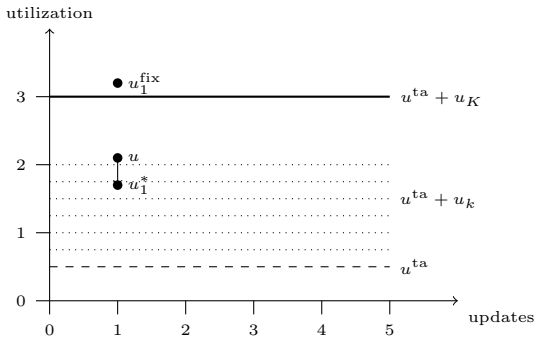
Constraints (11) assign each flight i to one duty. The storage capacity is bounded in constraints (12). Constraints (13) restrict the number of available carousels of each type r. Objective function (10) minimizes the sum of penalties for the utilization across all carousels.

The number of feasible duties for each carousel type is exponential in size, why we restrict the duties of each type r considered in the MP formulation to a subset $\mathcal{D}'_r \subseteq \mathcal{D}_r$, leading to the restricted MP (RMP). As $\mathcal{D}'$ may not contain duties resulting in an optimal or even feasible solution, we generate new columns by means of column generation (see Desaulniers et al. (2005) or Lübbecke and Desrosiers (2005) among others). Hence, the linear relaxation of RMP (L-RMP) yields the dual variables of constraints (11)

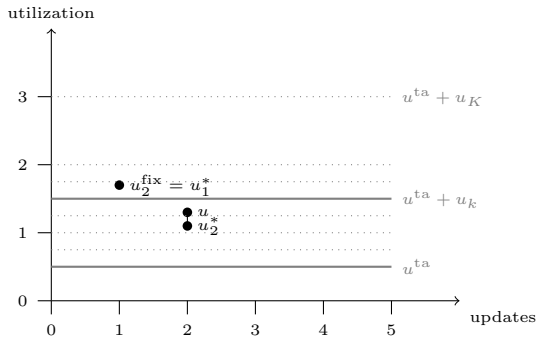
to (13) as query to find new duties having negative reduced cost in one of the R pricing problems (PP) (see §4.1). If one negative reduced cost column for type r is found, we add this column RMP.

To obtain a low penalty value for the threshold violations, the column generation will only assign few flights to a duty which, however, may be at odds with the requirement to find a feasible solution. As a consequence, a standard column generation implementation shows slow convergence times (see §6.2, Table 5). To "simplify" the answer in finding a feasible solution in shorter time, we relax the NP-hard problem of finding a minimized utilization. Given any feasible solution for MP, let u_l^{fix} be the current maximal utilization on the carousels across all time intervals where index $l \geq 0$ represents an iteration counter. To relax the minimization problem of carousels' utilization, the considered threshold values for RMP are restricted to subset $\mathcal{U}_m = \{u_{k-n}, \ldots, u_k\} \subseteq \mathcal{U}$ for each time interval m with $k-1 \geq n \geq 0$ and each $u_k \in \mathcal{U}_m$ satisfy $u^{\text{ta}} + u_k \leq u^{\text{fix}}$. For example, let us consider Figure 5 in which we minimize carousels' utilization for one time interval, i.e. $M = 1$. In iteration 1, RMP is restricted to subset $\mathcal{U}_1 = \{u_K\}$ and the maximal utilization of the carousels is bounded by u_l^{fix} (see Figure 5 (a)). Once we have found a feasible solution, we improve the solution by re-scheduling flights' baggage handling while keeping the assignment of flights' to carousels fixed to the assignment given by RMP's solution; the effect of re-scheduling can be seen in Figure 4. In Figure 5 (a) we reach utilization u_1^* by re-scheduling flights' baggage handling. Given the the new solution, we set the upper bound for the maximal allowed utilization $u_{l+1}^{\text{fix}} \leq u_l^{\text{fix}}$ for iteration $l+1$ to the maximal utilization across all time intervals reached in the solution of iteration l. To further improve the solution, the threshold value subset $\mathcal{U}_m$ is updated for each time interval $1 \leq m \leq M$ such that $\mathcal{U}_m \neq \{\emptyset\}$ and $u^{\text{ta}} + u_k \leq u_{l+1}^{\text{fix}}$ for all $u_k \in \mathcal{U}_m$. RMP with the updated set $\mathcal{U}_m$ and upper bound u_{l+1}^{fix} is run again to further improve the solution (see Figure 5 (b)). In this manner, we bound RMP from above, relax its feasibility problem and iteratively improve the objective function such that column generation is guided to the target value u^{ta} (see see Figure 5 (c)).

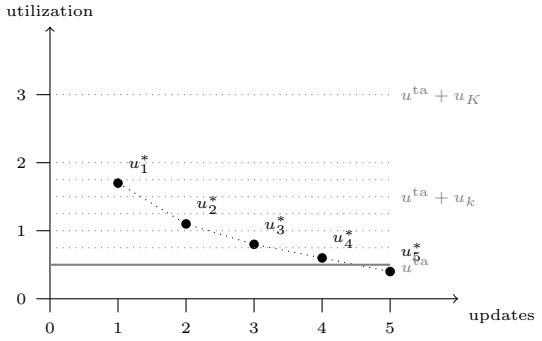
In the guided column generation (GCG) three types of subproblems are considered: RMP's PP generating new columns (see §4.1), two scheduling problems improving the solution found by RMP (see §4.2), and updating threshold value subset $\mathcal{U}'_m$ for the carousels' utilization in each time interval (see §4.3).



(a) RMP restricted to subset $\mathcal{U}_1 = \{u_K\}$



(b) Update upper bound to $u_2^{\text{fix}} = u_1^*$ and restrict RMP to subset $\mathcal{U}_2 = \{u_k\}$



(c) Course of the GCG until target value is reached

Figure 5: Example for GCG for $M = 1$; thick lines represent the considered threshold values in RMP

4.1 Pricing problem

We restrict the dual space of dual variable corresponding to constraints (11) to the negative side of the dual polyhedron by replacing the "=" constraint by the"$\geq$" constraint. Thus, each flight is assigned to at least one duty. As we minimize the utilization, flights assigned to more than one duty can be removed by which the objective function does not increase (see Ben Amor et al. (2006)).

Let $\lambda = (\lambda_i)_{i \in \mathcal{F}} \in \mathbb{R}_+^F$, $\mu = (\mu_t)_{t \in \mathcal{T}} \in \mathbb{R}_-^T$ and $\nu = (\nu_r)_{r \in \mathcal{R}} \in \mathbb{R}_-^R$ be the dual variables of constraints (11) to (13), respectively. In the PP, a duty d is represented by vector $x_{d,r} = (x_{d,r,i,w,\tau})_{i \in \mathcal{F}, \tau \in \mathcal{S}_i(w), W_{i,r}^{\min} \leq w \leq W_{i,r}^{\max}}$, where $x_{d,r,i,w,\tau}$ is 1, if flight i belongs to duty d with w working stations

assigned to flight i at a carousel of type r and start time tuple τ is selected, and 0 otherwise. Auxiliary binary vector $y_{d,r} = (y_{d,r,k,m})_{1\leq k\leq K, 1\leq m\leq M}$ is 1 at entry $y_{d,r,k,m}$, if duty d, corresponding to a carousel of type r, causes a violation of u_k in time interval m, and 0 otherwise. For time interval m, let $\mathcal{K}_m$ denote the indexes belonging to the subset of threshold values $\mathcal{U}_m$. Moreover, to speed up the computing time of the PP, we bound the maximal allowed carousel utilization in iteration l by u_l^{fix}. The corresponding subset of possible duties is given by $\mathcal{D}(u_l^{\text{fix}})$. Then, a duty of carousel type $1 \leq r \leq R$ with negative reduced cost is a negative solution of one of the restricted PP (RPP)

$$rc_r(x_{d,r}, y_{d,r}) = \min \left\{ \sum_{1\leq m\leq M} \sum_{k\in\mathcal{K}'_m} p_k \cdot y_{d,r,k,m} - \left(\sum_{i\in\mathcal{F}} \sum_{W_{i,r}^{\min}\leq w\leq W_{i,r}^{\max}} \sum_{\tau\in\mathcal{S}_i(w)} x_{d,r,i,w,\tau} \cdot \rho_{r,i,w,\tau} + \nu_r \right) \mid d \in \mathcal{D}_r(u_l^{\text{fix}}) \right\} \tag{15}$$

with $\rho_{r,i,w,\tau} = \left(\lambda_i + \sum_{t\in\mathcal{T}} \left(\mu_t \cdot \Phi_{t,\tau}^i\right)\right)$. In the RPP we have to assign a given number of items (flights) with a given size (number of containers and working station) to a given number of capacitated bins (carousels). Thus, the RPP is a generalization of the Multi-dimensional Knapsack Problem (MDKP) which is known to be NP-complete (see Garey and Johnson (1979)). As we have to solve the RPP in each iteration R-times, we present a LP-heuristic to solve RPP in §5.4.

4.2 Flight scheduling problem

Whenever RMP finds a feasible solution, we solve the flight scheduling problem (FSP) to gain further improvements. Given RMP's assignment of flights to the carousels, the FSP re-schedules flights' baggage handling periods and re-assigns the working stations to flights. The FSP has the same number of constraints as the OBHP, and due to Theorem 3, the FSP is NP-complete. However, as the assignment of flights to the carousels is provided in the FSP, the number of variables is reduced to $(3 \cdot F \cdot \max_{i\in\mathcal{F}} \{|\mathcal{S}_i|\} + K' \cdot M)$ with $K' \leq K$. Moreover, as only a subset of threshold values are considered in the objective function, we can apply CPLEX to solve FSP. If FSP leads toa new upper bound $u_{l+1}^{\text{fix}} < u_l^{\text{fix}}$ across all time intervals, we run the descent scheduling problem.

4.3 Descent scheduling problem

Once a new improved solution was found in iteration l leading to a new upper bound $u_{l+1}^{\text{fix}} < u_l^{\text{fix}}$, the set of threshold values $\mathcal{U}_m$ have to be updated. First, all threshold values leading to a violation of upper bound u_{l+1}^{fix} are removed, i.e. all $u_k \in \mathcal{U}_m$ with $u^{\text{ta}} + u_k > u^{\text{fix}}$. To add new threshold values for RMP, we consider the threshold values leading to the next improvement of the objective value. The most obvious but rather brute-force threshold value which could be added is u_k with $u^{\text{ta}} + u_k < u_{l+1}^{\text{fix}}$ and $u^{\text{ta}} + u_{k+1} \geq u_{l+1}^{\text{fix}}$. This threshold value, however, may not be appropriate, due to the irregular arrival of flights' baggage. For example, consider three flights where each flight causes a workload of four bags in the same period, independently of the chosen start time tuple. Given two carousels with a capacity of 12 bags each and the threshold value set $\mathcal{U} = \{0.2, 0.4\}$. Assume that in the current solution all three flights are assigned to the same carousel such that an utilization of 100% is reached, i.e. if we consider one time interval, we obtain $u_l^{\text{fix}} = 1$. To obtain an improvement, we could add threshold value 0.4 to $\mathcal{U}_1$. However, by swapping one flight to the second carousel the actual utilization that we can reach is $\frac{2}{3}$. We should therefore rather add threshold value 0.2 to $\mathcal{U}_1$. To obtain the next threshold values, we apply the DSP.

Let $\mathcal{T}_m(u_{l+1}^{\text{fix}})$ be the points in time interval m in which utilization u_{l+1}^{fix} is reached. For example, let us assume a maximal utilization of $u_{l+1}^{\text{fix}} = 3$. Then, given the utilization shown in Figure 4 (b) we have $\mathcal{T}_m(3) = \{1, 2\}$. In the DSP, decision variable $x_{i,w,\tau}$ is 1, if flight i is planned with w working stations and start time tuple τ. For each period in time interval $1 \leq m \leq M$, $t \in \mathcal{T}_m(u_{l+1}^{\text{fix}})$ and carousel type $r \in \mathcal{R}$, we solve the DSP (DSP(r, t))

$$\text{minimize} \quad z_{\text{DSP}}(r,t) = v \tag{16}$$

$$\text{subject to } (2) - (5) \quad \text{restricted to } r, t$$

$$\sum_{i \in \mathcal{F}: S_i^{\text{es}} \leq t < S_i^{\text{e}}} \sum_{W_{i,r}^{\min} \leq w \leq W_{i,r}^{\max}} \sum_{\tau \in \mathcal{S}_i(w)} \Gamma_{t,s}^{i,w} \cdot x_{i,w,\tau} + v = \left\lfloor K_r^{\text{b}} \cdot u^{\text{fix}} \right\rfloor \tag{17}$$

$$1 \leq v \leq \left\lfloor K_r^{\text{b}} \cdot u^{\text{fix}} \right\rfloor \tag{18}$$

$$x_{i,w,\tau} \in \{0,1\} \quad \forall\, i \in \mathcal{F} : S_i^{\text{es}} \leq t < S_i^{\text{e}}, W_{i,r}^{\min} \leq w \leq W_{i,r}^{\max}, \tau \in \mathcal{S}_i(w). \tag{19}$$

Constraints (2) to (5) ensure the resource capacities of the carousels and central storage system for the considered carousel type r and time period t. Constraints (17) determine the descent direction value v leading to the next threshold value which can be reached. Constraints (18) and (19) set the

domains of the variables. As in DSP only a subset of flights are considered we can solve DSP efficiently with CPLEX.

For each time interval m we distinguish three cases when updating threshold value sets $\mathcal{U}_m$: Either, there is at least one carousel type r such that DSP(r,t) is feasible for all $t \in \mathcal{T}_m(u_{l+1}^{\text{fix}})$, or DSP is not feasible for m and any carousel type $r \in \mathcal{R}$, or there exists no feasible solution for DSP in all time intervals. In the first case, let u_k be the largest threshold value with $u^{\text{ta}} + u_k \leq u_{l+1}^{\text{fix}}$. For the next iteration, we include threshold value u_k up to threshold value u_{k-n} with $u_{k-n} > \max_{r \in \mathcal{R}, t \in \mathcal{T}_m(u_{l+1}^{\text{fix}})} \left\{ \lfloor \frac{z_{\text{DSP}}(r,t)}{K_r^{\text{cb}}} \rfloor \right\}$ in $\mathcal{U}_m$. In the second case, we set $\mathcal{U}_m = \left\{ u_{l+1}^{\text{fix}} \right\}$ for all upcoming iterations, i.e. in time interval m no improvement can be obtained anymore. In the last case, the objective value cannot be further minimized and the optimal solution of the OBHP is found.

5 Implementation details

In this section we present details of our column generation implementation. In §5.1, a sequential allocation heuristic is presented to produce initial start columns. The heuristic is based on the procedure currently used in practice. To increase the chance of finding a feasible solution for RMP, the variety of duties from two consecutive column generation iterations is increased in §5.2 by means of the Chebyshev center cutting-plane method (CCCPM) of Lee and Park (2011). To search for a feasible solution in RMP, we apply a primal set-covering heuristic presented in §5.3. In §5.4 a greedy based MIP-heuristic for the RPP is presented speeding up each iteration of column generation. As GCG may not lead to an immediate improvement, we embed the GCG in a branch-and-price framework described in §5.5.

5.1 Initial columns

To obtain initial columns for RMP we apply a greedy heuristic in which all flights are increasingly ordered according to the latest start time of baggage handling and the scheduled departure time serves as tie breaker. When flight $i \in \mathcal{F}$ is next in the greedy order, the start time for the baggage handling as well as the storage depletion is set in the middle of the flight's baggage handling time window and the minimal number of working stations is assigned. Let $\bar{\tau}_i$ denote the selected start time tuple for flights $i \in \mathcal{F}_c^{\text{as}}$ already assigned to a carousel c and let $\mathcal{C}^{\text{pos}}$ be the subset of feasible carousel in terms of parking positions and working stations for flight i. Then, flight i

is assigned to the carousel leading to the best leveling for the amount of baggage during flight i's baggage handling which is carousel

$$c^* = \arg\min_{c \in \mathcal{C}^{\text{pos}}} \left\{ \sum_{t \in \mathcal{T}} \left(\sum_{j \in \mathcal{F}_c^{\text{as}} : s^h_{\bar{r}_j} \leq t < S^{\text{e}}_i} \frac{A_{j,t} + \mathbb{1}_{\left[s^{\text{h}}_{\bar{r}_i}, S^{\text{e}}_i\right[}(t) \cdot A_{i,t}}{K^{\text{cb}}_{r_c}} \right)^2 \right\}; \quad (20)$$

The sum of the inner most term on the right hand side of equation (20) yields the utilization at carousel c over planning horizon $\mathcal{T}$. As high utilization values have to be avoided, the utilization values are penalized by the quadratic function at any time t (see e.g. Rieck et al. (2012)). If it is not possible to assign a flight i due to a resource conflicts at a carousel, the starting time of the baggage handling and the storage depletion is postponed by 1 period and (20) is evaluated again. If the resource conflict at the carousels cannot be resolved for a flight or the capacity of the central storage system is violated by the flight, we assign the flights to a dummy carousel with infinite capacity. The dummy duty is inserted into the RMP to guarantee that a solution exists; the use of this duty is penalized in the objective function. In a post-processing step we assign not used working stations sequentially to those flights causing the highest workload on the carousels.

5.2 Chebyshev center cutting-plane method

Reduced costs (15) allow a flight $i \in \mathcal{F}$ only to be assigned to a duty if inequality $\lambda_i \geq \sum_{t \in \mathcal{T}} \mu_t \cdot \Phi^i_{t,\tau}$ holds. However, because of dual degeneracy this inequality is often valid only for a small subset of flights which leads to duties with a small flight density. To enforce this inequality for a larger subset of flights we "artificially" increase dual values λ_i by Lee and Park (2011)'s cutting-plane method which sets the dual variables to the gravity point of L-RMP's dual polyhedron, the so called Chebyshev center.

Let $\|x\|_2$ be the 2-norm of a vector x, b be the vector of the right hand side of constraints (11) – (13) and Z^{LB} a lower bound for L-RMP's objective function value. To get the gravity point in the dual polyhedron, consider the Chebyshev centered restricted dual problem

$$\text{maximize} \quad z \quad (21)$$

subject to

$$\sum_{i \in \mathcal{F}} \Theta^{\text{a}}_{d,i} \cdot \lambda_i - \sum_{t \in \mathcal{T}} \Theta^{\text{s}}_{d,t} \cdot \mu_t - \nu_r + \|(\Theta^{\text{a}}_d, \Theta^{\text{s}}_d, 1)\|_2 \cdot z \leq H_d \quad \forall\, d \in \mathcal{D}'_r(u^{\text{fix}}_l), r \in \mathcal{R} \quad (22)$$

$$-\lambda_i + \alpha_i^{\mathrm{a}} \cdot z \leq 0 \text{ and } -\mu_t + \alpha_t^{\mathrm{s}} \cdot z \leq 0 \quad \forall\, i \in \mathcal{F} \tag{23}$$

$$-\sum_{i \in \mathcal{F}} \lambda_i + \sum_{t \in \mathcal{T}} K_t^{\mathrm{s}} \cdot \mu_t + \sum_{r \in \mathcal{R}} C_r \cdot \nu_r + \alpha^{\mathrm{obj}} \cdot \|b\|_2 \cdot z \leq -Z^{\mathrm{LB}} \tag{24}$$

$$\lambda, \mu, \nu, z \geq 0. \tag{25}$$

To obtain an increased assignment variety of flights, we extend Lee and Park (2011)'s method by constraints (23), which set the distance for a dual point from the non-negativity hyperplanes of λ and μ to at least $\alpha^{\mathrm{a}} \cdot z$ and $\alpha^{\mathrm{s}} \cdot z$, respectively. As the constraints do not influence objective function (21), proximity parameters α^{a} and α^{s} can be set ≥ 0 without cutting off the optimal solution. Lee and Park (2011) use the proximity value $\alpha^{\mathrm{obj}} > 0$ (see Theorem 3 of Lee and Park (2011)) to move the point either closer to Z^{LB} by lowering its value or to move the point closer to the current polyhedral describing hyperplanes by increasing α^{obj}. To increase the assignment variety, we choose α_i^{a} randomly within interval $(0, 1]$ for each flight $i \in \mathcal{F}$ after each column generation iteration, while α_t^{s} is set to 0 for all $t \in \mathcal{T}$, which motivates the dual variables to satisfy inequality $\lambda_i \geq \sum_{t \in \mathcal{T}} \mu_t \cdot \Phi_{t,\tau}^i$. Proximity parameter α^{obj} is increased by an increment after each column generation iteration, as proposed by Lee and Park (2011).

The column generation principle is applied to the primal problem of the Chebyshev centered restricted dual problem is denoted as the Chebyshev centered restricted MP (CCRMP) yielding the a feasible lower bound for MP (see Lee and Park (2011)). When applying column generation, duties with negative reduced cost are added to the linear relaxed CCRMP (L-CCRMP). If no such duty exists and $z > 0$, lower bound Z^{LB} is updated with $Z^{\mathrm{LB}} = \sum_{i \in \mathcal{F}} \lambda_i + \sum_{t \in \mathcal{T}} K_t^{\mathrm{s}} \cdot \mu_t + \sum_{r \in \mathcal{R}} C_r \cdot \nu_r$. Otherwise, when $z = 0$, value Z^{LB} yields the lower bound and column generation is stopped.

5.3 Primal set-covering heuristic

To accelerate the search for a feasible solution in RMP, we make use of the framework for a primal heuristic proposed by Joncour et al. (2010). In our primal heuristic, a set-covering heuristic, presented in Appendix C selects those duties containing the most number of flights not already covered. The corresponding z_d-variables in the L-CCRMP are set to 1 in order to set up a primal search tree, i.e. the nodes of a branch represent these fixed duties. To explore the neighborhood of a partial solution, backtracking as a diversification mechanism is applied in which a tabu list avoids choosing the same duties selected in previous branches. The tabu list at a node contains all children of the node as well as the union of the tabu lists of the ancestor

nodes; the tabu list is empty at the root node. A node that is not the first child node of its parent node is explored only if the size of its tabu list is smaller or equal to $maxDiscrepency = 6$ and its tree depth is smaller or equal to $maxDeepth = \lfloor \frac{C}{2} \rfloor$.

Given fixed variables z_d, column generation is applied to seek for a feasible solution by generating appropriate columns for the partial solution. The residual L-CCRMP is re-optimized as long as the L-CCRMP becomes either feasible or the RPP does not return a column with negative reduced costs. In the latter case, the next branch is examined. If there is no branch left, we proceed with column generation for the next $nextIterations = 2 \cdot C$ iterations until a new search tree is built. Because feasibility of RMP can be achieved after the addition of a new duty, the set-covering heuristic is applied after each column generation iteration.

5.4 Pricing problem heuristic

In the RPP we have to add containers and working stations (items) of flights to a carousel with limited resources (knapsack) over the planning horizon. Hence, the RPP is at least as hard to solve as a MDKP which is known to be NP-hard (see Garey and Johnson (1979)). In particular, L-CCRMP produces a great number of ρ-values ≥ 0 increasing RPP's computational intractability. As there is no need to solve RPP exactly until the last iteration of column generation, an approximated solution is used which finds good solutions in an acceptable time. We implement a heuristic based on linear programming and adapted the adaptive fixing procedure of Bertsimas and Demir (2002) for the MDKP. The heuristic iteratively rounds variables of the linear programming relaxation of the PP. In our pre-experiments, the procedure yields much better solutions for RPP than any other greedy based algorithms for the MDKP (see Khan et al. (2002) and Akbar et al. (2006)). When the PP heuristic does not return a duty with negative reduced cost, we solve RPP by means of CPLEX to proof either the optimality of L-CCRMP or to generate a new duty.

5.5 Branch-and-price framework

The column generation terminates as soon as each RPP does not provide a column with negative reduced cost meaning that the LP bound of L-CCRMP and, thus, RMP is reached. If no feasible solution is found at the end of column generation, the solution space is partitioned with the multi-pattern branching of Elhedhli et al. (2011) who extend the two-pattern

branching rule of Ryan and Foster (1981) (see also Vanderbeck and Wolsey (1996), Barnhart et al. (1998)). As search strategy we apply depth-first branching. During the branch-and-price algorithm we can prune an examined node if RMP's LP-bound is greater or equal to current upper bound u_l^{fix}, RMP is not feasible or RMP returns a feasible (integer) solution.

6 Computational study

In this section we report empirical results of the proposed model and solution methodology. For computations we employ an Intel Dual Core 2.8 GHz workstation with 2 GB of RAM memory running on a Windows 7 platform. The mathematical model and algorithms are implemented in JAVA language. LP and MIP problems are solved by CPLEX 12.4.

In §6.1 we provide a description of the underlying data. An experimental study in §6.2 compares the performance of the GCG, a standard column generation (SCG) procedure for RMP in which all threshold values are considered in the PP (see §4.1) and the TIF. The performance of the GCG with all implementation details of §5 for real-world scenarios is shown in §6.3.

6.1 Data base

All test instances are derived from 2010 data of Terminal 2 of Munich Airport. The planning horizon is set from 03:00 am to 10:40 pm, i.e. $T = 236$ for periods of 5 minutes length. As shown in Figure 2 the planning horizon is divided into 3 time intervals, in which the target value for the utilization is set to $u^{\text{ta}} = 0.5$. The threshold values for the deviations from the target value are $\mathcal{U}_m = \{u_1, \ldots, u_{13}\} = \{0.1, 0.2, \ldots, 0.9, 1, 2, 10, 100\}$ in each time interval $1 \leq m \leq 3$. To minimize the utilization and to reach the target value of 0.5 in each of the 3 time intervals, we penalize a deviation of u_k with $p_k = k^k$ for $k \in \{1, \ldots, 13\}$. All presented results are rounded after the second decimal place. In the following, we provide a description of the baggage arrival process, of airport's infrastructure and the considered outgoing flights.

Arrival process The OBHP's solution quality depends to a great extend on the baggage arrival process A_i for each flight $i \in \mathcal{F}$ (see Appendix B). The arrival process consists of the dynamic and stochastic baggage arrival process at the check-in and of transfer flights. According to Stolletz (2011) (see Robertson et al. (2002)) the arrival process of passengers and, hence,

Table 2: Average RMSE and the average $a(i)$-values of 1,000 flights

	50%			70%			80%			90%		
$\mathbb{E}(A)$	< 50	$[50, 100]$	> 100	< 50	$[50, 100]$	> 100	< 50	$[50, 100]$	> 100	< 50	$[50, 100]$	> 100
RMSE	0.58	1.03	1.72	0.60	1.03	1.72	0.68	1.11	1.86	0.91	1.42	2.37
$\bar{a}$	0.10	0.19	0.34	0.07	0.15	0.27	0.06	0.12	0.21	0.04	0.08	0.14

the arrival process of check-in baggage, shows predictable a arrival pattern depending on a flight's destination and scheduled time of departure. Using our data from Terminal 2, a study revealed that the same holds true for the arrival time of incoming flight following a gamma-distribution. Thus, to obtain an estimation for flight i's baggage arrival process A_i for a planning day, we accumulate flight i's historical amount of baggage arriving in 5 minute intervals before the scheduled departure time. A flight i is thereby uniquely identified by its flight number, flight destination, departure day and departure time. To avoid seasonal dependencies (e.g. vacations, holidays, festivals, etc.), data collection is restricted to a maximal 3 months before the planning day. Having the accumulated amount of baggage in each time interval, we derive the 50% (5), 70% (7), 80% (8) and 90% (9) quantile for the estimation of A_i. The quantiles represent the different robustness degrees of the obtained solution. The higher the quantile the lower the chance that the amount of arrived baggage and, thus, the calculated utilization at the carousels are underestimated. On a planning day, we calculate the average underestimation of the historical arrival process A_i^{h} from the estimated arrival process A_i with $a(i) = \frac{1}{T} \cdot \sum_{t \in \mathcal{T}} \left(A_{i,t}^{\mathrm{h}} - A_{i,t} \right)^+$. Table 2 reports the average root mean squared error (RMSE) and the average underestimation ($\bar{a}$) of the historical arrival process of $1,000$ flights according to the expected amount of baggage $\mathbb{E}(A)$. The results show that the higher the quantile the lower the underestimation of the amount of baggage.

Infrastructure The layout for the handling facilities is based on the one of Terminal 2 of Munich Airport with 22 carousels and 3 carousel types (see Table 3 (a)). The capacity of the central storage is bounded by $3,500$ bags, while the storage depletion rate is 19 bags per 5 minute. Each working station has a working rate of 8 bags per 5 minute.

Outgoing flights For each flight the time window for the start of the baggage handling is derived from the minimal and maximal duration for

Table 3: Carousels (a) and flights' handling periods in minutes (min) (b)

(a) Carousels

	Capacity		
Type r	K_r^{pp}	K_r^{ws}	K_r^{ca}
1	8	4	20
2	12	4	25
3	24	6	40

(b) Handling periods

	Handling period (min)	
$\mathbb{E}(A)$	Min	Max
< 50	30	60
$[50, 100]$	60	120
> 100	120	180

the baggage handling periods which depends on flight's expected amount of baggage (see Table 3 (b)). The scheduled time of departure of the flight as well as the number of containers required for each flight is taken from the historical data.

6.2 Experimental study

All column generation implementations are initialized with greedy heuristic of §5.1, the primal set-covering heuristic of §5.3 searches for feasible solutions and the PP is solved as described in §5.4. We denote the test instances with I-$n(q)$ where n stands for the instance number and $q \in \{5, 7, 8, 9\}$ represents the used quantile to estimate flights' baggage arrival process. For the generation of test instances I-1(q) to I-8(q), we systematically vary the number of flights and the number of handling facilities. The flights are randomly selected from a pool of 413 different flights for each instance number. For each instance, Table 4 shows in row "$C(r)$" the number of carousels and the corresponding type (r) as defined in Table 3 (a). As indicator for the hardness of an instance, row "LB" shows the LP-bound for the minimal number of required carousels for each instance. The last row "Pro. 1/2" gives the average percentage reduction of the start time tuple sets for Proposition 1 and 2, respectively.

Table 4: Experimental instance information with the reduction of the start time tuples due to Propositions 1 and 2 in %

Instance	I-1(q)	I-2(q)	I-3(q)	I-4(q)	I-5(q)	I-6(q)	I-7(q)	I-8(q)
F	25	50	100	150	200	250	300	413
$C(r)$	10(2)	10(2)	10(2)	7(1)/10(2)	7(1)/10(2)	7(1)/10(2)	7(1)/10(2)	7(1)/10(2)
LB	1	1.89	3.83	4.67	6.33	8.33	12.72	14.17
Pro. 1/2	40.28/14.63	39.88/12.03	42.49/12	42.64/12.24	42.33/11.77	41.16/12.23	42.37/11.97	44.17/12.40

Table 5 shows the results for comparison between the GCG, the SCG and the TIF for the instances I-1(q) to I-4(q). In both column generation implementations, we make no use of the CCCPM and as soon as the LP-bound is reached the procedures terminate. Columns "Cols" and "Time (s)" report the total number of generated columns and the computing time (in seconds) until the LP-bound is reached. From the best solution found we report in column "u^*" the highest utilization across all time intervals. Some instances solved by means of SCG did not reach the LP-bound within 1 hour time. For those instances, indicated by"‡", the computation time until the best utilization obtained in column "u^*" is reported in column "Time (s)". The LP-bound of the column generation and the TIF is reported in columns "LP-CG" and "LP", respectively. Column "Upd" shows the number of updates of the threshold value sets. The results reveal that the

Table 5: Comparison between the GCG, SCG and the TIF

	GCG				SCG				TIF		
Inst	u^*	Time (s)	Cols	Upd	u^*	Time (s)	Cols	LP-CG	u^*	Time (s)	LP
I-1(5)	0.18	2.53	28	2	0.36	148.37	1,878	$4.8 \cdot 10^{-3}$	0.18	12.42	$4.8 \cdot 10^{-3}$
I-1(7)	0.18	3.51	50	2	0.26	113.89	1,814	$4.2 \cdot 10^{-3}$	0.18	12.80	$4.2 \cdot 10^{-3}$
I-1(8)	0.24	4.07	28	2	0.3	106.62	1,886	$8.2 \cdot 10^{-3}$	0.24	4.98	$8.2 \cdot 10^{-3}$
I-1(9)‡	0.31	3.34	50	3	0.74	8.86	258	$6.2 \cdot 10^{-3}$	0.31	0.1	$2.2 \cdot 10^{-3}$
I-2(5)	0.22	3.94	142	3	0.5	507.58	3,387	$8.8 \cdot 10^{-3}$	0.22	66.26	$6.8 \cdot 10^{-3}$
I-2(7)	0.22	4.42	105	2	0.72	405.37	3,314	$9.3 \cdot 10^{-3}$	0.22	55.64	$6.2 \cdot 10^{-3}$
I-2(8)‡	0.26	5.18	106	2	1.18	26.86	101	$7 \cdot 10^{-3}$	0.26	310.24	$7 \cdot 10^{-3}$
I-2(9)	0.26	6.56	102	2	1.9	320.19	2,306	0.03	0.26	280.69	0.01
I-3(5)	0.22	35.18	236	3	1.24	2,982.78	5919	$9.8 \cdot 10^{-3}$	0.22	560.02	$6.8 \cdot 10^{-3}$
I-3(7)‡	0.2	15.66	201	2	0.34	292.92	330	$8.2 \cdot 10^{-3}$	0.2	724.19	$5 \cdot 10^{-3}$
I-3(8)‡	0.26	19.57	371	4	1.66	139.34	266	0.01	0.26	644.36	$9.6 \cdot 10^{-3}$
I-3(9)‡	0.29	32.68	529	6	2.14	168.74	258	0.01	0.29	655.37	$7.6 \cdot 10^{-3}$
I-4(5)‡	0.22	48.60	402	3	0.62	352.30	432	$8.5 \cdot 10^{-3}$	0.22	1434.42	$5.4 \cdot 10^{-3}$
I-4(7)‡	0.24	30.98	656	5	1.02	161.87	332	0.03	0.24	1,463.66	0.01
I-4(8)‡	0.24	44.88	476	5	0.78	570.27	683	$8.2 \cdot 10^{-3}$	0.24	1,523.49	$8.2 \cdot 10^{-3}$
I-4(9)	0.28	57.87	878	7	2.32	2,794.70	5,274	0.06	0.28	1,553.45	0.03

GCG requires not only less columns than the column generation for RMP to reach the LP-solution but also generates better columns in terms of finding a feasible solution with the primal set-covering heuristic of §5.3. All found solutions for GCG are optimal as the target value 0.5 is reached. In contrast, the SCG does not even reach the LP-bound in half of the instances. The LP-bound obtained by MP and TIF is rather loose in comparison to the optimal solution. However, MP's LP bound clearly dominates TIF's LP-bound backing the LP-theory (see Nemhauser and Wolsey (1999)).

In Table 6, we compare the branch-and-price framework described in §5.5 with GCG and the CCCPM (BB-GCG) and without the CCCPM (BB-GCG$^+$). As instances I-1(q) to I-4(q) already could be solved to optimality without the CCCPM, we use the more difficult instances I-5(q) to I-10(q) for which a SCG did not converge at all. Instances not solved to optimality with BB-GCG$^+$ and BB-GCG are indicated with "†" and "‡", respectively. In column "Nodes" the number of visited nodes is reported, while column "∅Cols" shows the average number of generated columns in each node.

Table 6: Comparison between BB-GCG and BB-GCG$^+$

	BB-GCG$^+$				BB-GCG			
Inst	u^*	Time (s)	∅Cols	Nodes	u^*	Time (s)	∅Cols	Nodes
I-5(5)	0.22	96.11	824	0	0.22	156.62	392	0
I-5(7)	0.24	98.19	1,105	0	0.24	83.9	331	0
I-5(8)	0.26	82.93	906	0	0.26	295.33	604	0
I-5(9)	0.28	112.18	843	0	0.28	186.72	686	0
I-6(5)	0.22	135.77	1,214	0	0.22	141.06	471	0
I-6(7)	0.24	145.87	942	0	0.24	169.65	442	0
I-6(8)	0.26	173.85	1,267	0	0.26	307.52	836	0
I-6(9)	0.58	172.46	878	0	0.58	169.44	406	0
I-7(5)	0.22	224.99	710	0	0.22	1642.21	972	0
I-7(7)	0.24	262.96	629	0	0.22	479.23	677	0
I-7(8)	0.26	278.44	560	0	0.26	607.55	697	0
I-7(9)†‡	0.58	297.56	499	104	0.58	417.54	210.2	91
I-8(5)	0.22	449.27	919	0	0.22	360.68	733	0
I-8(7)	0.24	349.64	668	0	0.24	227.73	1184	0
I-8(8)	0.24	1,245.85	1,014	0	0.24	446.48	905	0
I-8(9))†‡	0.96	436.71	613	103	0.7	280	376	90
I-9(5)	0.22	732.78	877	100	0.22	586.08	978	20
I-9(7)	0.24	3,179.72	1,049	83	0.24	614.80	933	64
I-9(8)†	0.56	3,571.46	988	95	0.26	2,772.77	1747	87
I-9(9)†‡	0.58	1,753.24	734	100	0.58	1,417.95	1053	131

Using CCCPM reduces the number of examined nodes due to the increased assignment variety during the first run of column generation which increases also the chance of finding a feasible solution with the primal set-covering heuristic of §5.3. However, due to the higher number of dual variables which have to be considered in the RPP, in some instances the time per column generation in BB-GCP is higher than in BB-GCG$^+$.

6.3 Real-world study

For the real-world instances we randomly selected 7 arbitrary planning days from our historical data and the original layout configuration of Terminal 2 in Munich Airport is applied (see Table 7). The instances are solved by means BB-GCG containing all implementation details of §5. The results are shown

Table 7: Real-world instance with the reduction of the start time tuples due to Proposition 1 and 2 in %

Instance	I-10(q)	I-11(q)	I-12(q)	I-13(q)	I-14(q)	I-15(q)	I-16(q)
F	436	389	413	386	440	331	353
$C(r)$				7(1)/14(2)/1(3)			
LB	19.25	18.5	19	18	18.5	17.5	18
Pro. 1/2	28.2/9.12	37.1/12.18	36.28/12.22	34.48/12.41	38.06/12.18	33.31/10.9	24.64/0.48

in Table 8. As most of the papers propose greedy algorithms for planning outbound baggage (see Abdelghany et al. (2006), Ascó et al. (2013)), we use greedy heuristic of §5.1, denoted by "Heu", as benchmark. If the greedy heuristic does not find a feasible solution (results labeled with "+"), we relax the carousels' parking position capacities and assign and schedule a not planned flight to the carousel having the smallest difference between the minimum number of the flight's required working stations and available working stations. Once all flights are assigned, we run the pre-processing step as described in §5.1. The average improvement of the utilization in % when calling FSP is presented in column "∅ Impr". Column "∅ Time (s)" shows L-CCRMP's average computing time until the next feasible solutions is found. Column "∅ Time (s)" for FSP gives the average computing time over all calls. The algorithm is aborted after a running time of 8 hours. For all test instances labeled with "†", optimality could not be proved. In the last three columns we report the results of an event-based simulation. Given the historical time-stamps of the baggage at the infeed station as parameters, we simulate the transfer of each bag through the BHS to the destinations as well as the loading process of workers at the working stations in an event-based simulation. In the simulation we compare the solution of the heuristic and the BB-GCG with the airport's manually generated solution by the dispatcher "Disp". If dispatcher's solutions is not feasible in terms of resource capacities, the result is labeled with a "+". Column "Heu", "BB-GCG" and "Disp" show the average maximal utilization of the three procedures obtained in 100 simulation runs during the course of the day. To guarantee a clear comparison between the procedures in the simulation,

we allow an utilization greater than one on the carousels' conveyor belts. BB-GCG solves 5 of 21 instances to optimality and always outperforms the

Table 8: Computational results for real-world instances

				L-CCRMP		FSP		Simulation		
Inst	Heu	Upd	u^*	∅Time(s)	∅Cols	∅Impr	∅Time (s)	Heu	BB-GCG	Disp
I-10(5)	1.36	1	0.8	15.51	75	68.25	6.87	3.23	2.21	2.29
I-10(7)	2.72	7	0.96	54.75	291.71	15.77	3.26	3.45	1.14	
I-10(8)†	3.08	12	1.24	747.77	693.66	8.81	4.60	2.93	0.90	
I-10(9)	6.96	12	1.90	359.84	509.33	11.00	4	2.21	0.87	
I-11(5)†	2.8	7	1.05	626.77	523.29	9.75	3.57	4.97	2.44	2.96
I-11(7)†	3.08	7	1.05	2,775.96	443.14	19.00	2.95	4.47	1.23	
I-11(8)†	3.09	7	1.12	4,435.82	1731.2	22.57	3.12	3.87	0.89	
I-11(9)†	8.24	10	2.18	1,498.51	834.6	9.65	4.31	3.98	0.99	
I-12(5)†	2.22	4	1.1	336.30	980.5	26.51	2.06	4.13	2.42	3.85
I-12(7)†	3.12	7	1.32	1,382.55	1145	14.82	2.47	4.60	1.22	
I-12(8)†	4.16	7	1.32	1,177.96	1360.71	14.78	2.98	4.78	0.53	
I-12(9)†	7.52	12	2.25	1,847.16	1043.92	9.34	4.84	4.32	0.84	
I-13(5)†	4.1^+	3	1.15	253.01	926.67	28	2.89	3.23	2.12	2.71^+
I-13(7)†	4.3	4	1.2	78.52	230.75	23.13	2.18	3.12	1.01	
I-13(8)†	4.95^+	3	1.33	182.06	357.67	41.16	3.11	3.02	1.28	
I-13(9)†	7.35	7	1.85	4,803.03	1309.57	14	4.18	2.89	1.07	
I-14(5)	1.72	5	0.68	96.68	411.2	21.14	1.86	2.12	1.34	1.62
I-14(7)	3.08^+	4	0.88	308.33	871.75	24.13	2.14	2.28	1.13	
I-14(8)†	2.72	5	0.96	923.53	1,334	22.92	3.67	2.55	0.64	
I-14(9)†	8.08	8	0.96	312.79	426.38	19.04	5.81	2.10	0.87	
I-15(5)†	3.16^+	3	1.15	2,937.81	1,409.66	32.57	2.12	2.12	1.87	1.91
I-15(7)†	3.44	4	1.2	5,435.51	1,232.75	25.36	2.33	1.98	1.15	
I-15(8)†	3.96	6	1.3	4,882.72	1,696.71	14.82	2.91	1.80	0.78	
I-15(9)†	8.32	6	2.45	406.98	393.5	13.56	4.53	1.76	0.99	
I-16(5)†	3^+	4	1.25	632.31	1,081.67	20.42	1.92	3.02	1.67	2.01^+
I-16(7)†	3.44	6	1.92	754.45	1,065.5	14.69	2.32	3.14	1.10	
I-16(8)†	3.96	7	1.92	4,338.34	2,220.71	15.31	3.56	2.98	0.83	
I-16(9)†	8.10^+	4	3.48	6,903.30	2,450.8	17.16	3.55	2.78	1.14	

greedy heuristic and the dispatcher. Comparing the best solution found for BB-GCG with the dispatcher's solution, we observe a reduction of 65.23% in average. We can also see, that solving the FSP (see §4.2) lead to an average improvement of 20,63% of the solution found during column generation. We identified differences in the assignment and schedule pattern of flights to carousels between the different quantiles of the same instance. Thus, due to the fluctuations in the arrival process of baggage, in 5 of 7 days the 80%-quantile leads to the most robust solution in the simulation with a maximal utilization near one. In the instances I-10(q) and I-13(q) the 70% and 90%

quantile perform best.

7 Conclusion

We presented a TIF for assigning flights to handling facilities and scheduling their baggage handling period. The problem is of high practical relevance. We analyzed the problem complexity and showed dominance criterion to strengthen the solution space. To receive a utilization below or close to a target value during the course of the day, we presented a GCG framework. In our computational study we showed that the presented column generation framework leads to a significant reduction of computer times in comparison to the TIF or a SCG framework. Different scenarios for the arrival process of baggage was tested. In 70% of the test instances, the most robust solution was obtained when the 80% quantile for the baggage arrival process was used.

A Proof of Theorem 1

For the proof it is sufficient to consider flights assigned to the same carousel. We have to show that, given a solution by the TIF, it is possible to place all containers of assigned flights to the carousel such that all assigned working stations for the flights are reached and there is no overlapping of container rows or working stations with other flight handled simultaneously. For the assignment rule, we order the flights by increasing baggage handling ending times. A container row of a flight is assigned clockwise from $1 to K_r^{\text{pp}}$ (see Figure 3) beginning with the parking position immediately following the container row of the previously assigned flight such that all working stations are reached. Assume, flights i_1 to i_k are handled simultaneously where flight i_1 to i_k are ordered according to their baggage handling ending times. Let us assume that flight i_1's to i_{k-1}'s container rows are assigned with the above assignment rule and assume, w.l.o.g., that flight i_k requires the last available working station. If flight i_k's container row has to be split on both sides of the carousel such that it overlaps with flight i_1's container row by $p \geq 1$ parking positions, we right-shift the container row of flight i_1 for p parking positions in clockwise order until the overlapping conflict is solved. If it is not possible to right-shift flight i_1's container row as it is blocked by flight i_2's container row, see e.g. Figure 3, we sequentially right-shift the container rows of the succeeding flights i_2 to i_{k-2}; a right-shift of flight i_{k-1}'s container rows is not required as it would directly lead to a reduction

of available parking positions for flight i_k. If it is not possible to right-shift the container rows of flight i_1 to i_{k-2} such that p parking positions for flight i_k become available in the working station segment of flight i_1, then there are either not enough parking positions available for flight i_k, or a right-shift of flight i_1's container row would contradict the requirement that flight i_1's container row reaches the assigned working stations. In the first case, we violate the assumption that it is possible to handle the flights i_1 to i_k simultaneously due to a violation of the parking position capacity. In the latter case, flight i_k would require at least $p \geq K_r^{\text{ppws}}$ additional parking positions, and hence at least one additional working station, which violates the working station capacity and, thus, also contradicts the assumption that the flights can be handled simultaneously.

B Preprocessing for the TIF

Let $\rho > 0$ be the number of bags transferred from the storage system to any carousel. If start time tuple $\tau = \langle s_\tau^{\text{h}}, s_\tau^{\text{d}} \rangle \in \mathcal{S}_i$ is selected for flight $i \in \mathcal{F}$, the number bags stored up to period t is given by the expected amount of baggage arrived up to time t minus the amount of baggage which have already been sent to the carousel, i.e.

$$\Phi_{t,\tau}^i = \left(\sum_{t < s_\tau^{\text{h}}} \mathbb{E}(A)_{i,t} - \mathbb{1}_{\left\{ s_\tau^{\text{d}} \leq t \right\}} \cdot \left(t - s_\tau^{\text{d}} + 1 \right) \cdot \rho \right)^+ .$$

Let $\alpha > 0$ be the number of bags which can be loaded at working station per period t. Then, if w working stations are assigned to flight i, the number of bags on carousel c in period $t \in \mathcal{T} \setminus \{0\}$ is derived by the recursive formula

$$\Gamma_{t,\tau}^{i,w} = \left(\Gamma_{\tau,t-1}^{i,w} + \mathbb{1}_{\left\{ s_\tau^{\text{h}} \leq t < S_i^{\text{e}} \right\}} \cdot A_{i,t} + \mathbb{1}_{\left\{ s_\tau^{\text{d}} \leq t < S_i^{\text{e}} \right\}} \cdot \min \left\{ \rho, \Theta_{\tau,t} \right\} - \alpha \cdot w \right)^+$$

with $\Gamma_{\tau,0}^{i,w} = 0$. $\Gamma_{\tau,t-1}^{i,w}$ yields the number of bags on the carousel's conveyor belt from the previous period. The indicator function indicates weather the baggage handling has started and thus the amount of baggage $A_{i,t}$ arriving in period t is directed to the carousel. The last term yields the amount of baggage arriving at the assigned carousel from the storage system minus the bags loaded by the assigned working stations.

C Set-covering heuristic

To build the search tree for the primal heuristic, we use the greedy-based set-covering heuristic consisting of the following three steps:

`selection`($\mathcal{D}^{\text{pos}}$): $\mathcal{D}^{\text{pos}}$ denotes the set of duties which can be selected. The procedure first chooses those duties corresponding to an integer solution, i.e. $z_d = 1$. Then, these duties are selected containing at least one priority flight, i.e. a flight which could not be assigned during the last run of the set-covering heuristic. Last, we select duties containing the maximal number of flights not assigned so far in the partial solution (see Johnson (1974) and Korte and Vygen (2012)).

`storageCapacityCheck`($\mathcal{D}^{\text{sel}}$): After a duty d is added to partial solution $\mathcal{D}^{\text{sel}}$, the heuristic checks violations of the central storage capacity. Let $\mathcal{F}_t^{\text{parT}}$ be the subset of flights handled during period $t \in \mathcal{T}$. If the storage capacity during some period t is violated, flight

$$i_d^* = \arg \max_{i \in \mathcal{F}_t^{\text{parT}}} \left\{ \frac{\Phi^i_{t,\bar{\tau}_i} - \Phi^i_{t,\tau_i}}{P_i \cdot \left(s^{\text{h}}_{\bar{\tau}_i} - s^{\text{h}}_{\tau_i}\right)} \mid \{\tau_i, w_i, P_i\} \text{ is feasible for duty } d \right\}$$

is left-shifted to start time tuple $\tau_i \preceq \bar{\tau}_i$; the number of working stations assigned to flight i is equal to the maximal possible number. We continue left-shifting flights as long as the storage conflict is not solved or there is no flight which can be left-shifted. In the latter case, we delete the flight from a duty with the greatest number of stored bags relative to the consumed parking positions capacity, i.e. we remove flight

$$i^* = \arg \max_{i \in \mathcal{F}_t^{\text{parT}}} \left\{ \frac{\Phi^i_{t,\bar{\tau}_i}}{P_i \cdot \left(S_i^E - s^{\text{h}}_{\bar{\tau}_i}\right)} \right\}.$$

`add`($\mathcal{D}^{\text{sel}}$,$\mathcal{F}^{\text{nA}}$): Flights $\mathcal{F}^{\text{nA}}$, not added at the end of the heuristic, are assigned to 1 of the selected duties $\mathcal{D}^{\text{sel}}$ in decreasing order of the number of required parking positions. Flights not added are collected in a priority set.

References

Abdelghany, A., K. Abdelghany, R. Narasimhan. 2006. Scheduling baggage-handling facilities in congested airports. *Journal of Air Transport Management* **12**(1) 76 – 81.

Akbar, M.M., M.S. Rahman, M. Kaykobad, E.G. Manning, G.C. Shoja. 2006. Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. *Computers & Operations Research* **33**(5) 1259 – 1273.

Ascó, A., J.A.D. Atkin, E.K. Burke. 2011. The airport baggage sorting station allocation problem. J. Fowler, G. Kendall, B. McCollum, eds., *In proceedings of the 5th Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2011), 9-11 August 2011, Phoenix, Arizona, USA*. 419–444. URL `http://www.schedulingconference.org/previous/publications/displaypub.php?key=2011-419-444-P&filename=mista.bib`. Paper.

Ascó, A., J.A.D. Atkin, E.K. Burke. 2013. An analysis of constructive algorithms for the airport baggage sorting station assignment problem. *Journal of Scheduling* 1–19URL `http://dx.doi.org/10.1007/s10951-013-0361-x`.

Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W. Savelsbergh, P.H. Vance. 1998. Branch–and–price: Column generation for solving huge integer programs. *Operations Research* **46**(3) 316 – 329.

Ben Amor, H., J. Desrosiers, J.M. Valério de Carvalho. 2006. Dual-optimal inequalities for stabilized column generation. *Operations Research* **54**(3) 454 – 463.

Bertsimas, D., R. Demir. 2002. An approximation dynamic programming approach to multidimensional knapsack problems. *Management Science* **48**(4) 550 – 565.

Desaulniers, G., J. Desrosiers, M.M. Solomon, eds. 2005. *Column Generation*. 1st ed. Springer.

Elhedhli, S., L. Li, M. Gzara, J. Naoum-Sawaya. 2011. A branch-and-price algorithm for the bin-packing problem with conflicts. *Journal on Computing* **23**(3) 404–415.

Frey, M., C. Artigues, R. Kolisch, P. Lopez. 2010. Scheduling and planning the outbound baggage process at international airports. *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM 2010), Macao, China*.

Frey, M., R. Kolisch. 2010. Scheduling of outbound baggage at airports. *12th international Conference on Project Management and Scheduling*.

Garey, M., D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York.

Johnson, D.S. 1974. Approximation algorithms for combinatorial problems. *Journal of Computer and System Science* **9**(3) 156 – 278.

Joncour, C., S. Michel, R. Sadykov, D. Sverdlov, F. Vanderbeck. 2010. Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics* **36**(0) 695–702.

Khan, S., K.F. Li, E.G. Manning, M.M. Akbar. 2002. Solving the knapsack problem for adaptive multimedia system. *Studia Informatica Universalis* **2** 161 – 182.

Korte, B., J. Vygen. 2012. *Combinatorial Optimization*, vol. 5. Springer.

Lee, C., S. Park. 2011. Chebyshev center based column generation. *Discrete Applied Mathematics* **159**(18) 2251 – 2265.

Lübbecke, M. E., J. Desrosiers. 2005. Selected topics in column generation. *Operations Research* **53**(6) 1007 – 1023.

Nemhauser, G.L., L.A. Wolsey. 1999. *Integer and Combinatorial Optimization*. Wiley-Interscience.

Rieck, J., J. Zimmermann, T. Gather. 2012. Mixed-integer linear programming for resource leveleing problems. *European Journal of Operational Research* **221**(1) 27–37.

Robertson, C.V., S. Shrader, D.R. Pendergraft, L.M. Johnson, K.S. Silbert. 2002. The role of modelling demand in process re-engineering. E. Ycesan, C.-H. Chen, J.L. Snowdon, J.M. Charnes, eds., *Proceedings of the 2002 Winter Simulation Conference*.

Ryan, D.M., B.A. Foster. 1981. An integer programming approach scheduling. A. Wren, ed., *Computer and Scheduling of Public Transport Urban Pasenger Vehicle and Crew Scheduling*. North-Holland, Amsterdam, 269 – 280.

SITA. 2014. Baggage report 2014. `http://www.sita.aero/surveys-reports/industry-surveys-reports/baggage-report-2014`.

Stolletz, R. 2011. Analysis of passenger queues at airport terminals. *Research in Transportation Business and Management* **1**(1) 144–149.

Tarău, A.N., B. De Schutter, J. Hellendoorn. 2011. Predictive route control for automated baggage handling systems using mixed-integer linear programming. *Transportation Research Part C* **19**(3) 424–439.

Vanderbeck, F., L.A. Wolsey. 1996. An exact algorithm for ip column generation. *Operations Research Letters* **19**(4) 151 – 159.