EURO Journal on Transportation and Logistics 10 (2021) 100041

Contents lists available at ScienceDirect

EURO Journal on Transportation and Logistics

journal homepage: www.sciencedirect.com/journal/euro-journal-on-transportation-and-logistics

The Baggage Belt Assignment Problem

David Pisinger[a,*], Rosario Scatamacchia[b]

[a] Technical University of Denmark, DTU Management Engineering, Akademivej, Building 358, 2800, Kgs. Lyngby, Denmark
[b] Dipartimento di Ingegneria Gestionale e della Produzione, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy

ARTICLE INFO

Keywords:
Combinatorial optimization
Baggage belt assignment problem
Inbound baggage handling
Branch and price
Knapsack problem

ABSTRACT

We consider the problem of assigning flights to baggage belts in the baggage reclaim area of an airport. The problem is originated by a real-life application in Copenhagen airport. The objective is to construct a robust schedule taking passenger and airline preferences into account. We consider a number of business and fairness constraints, avoiding congestion, and ensuring a good passenger flow. Robustness of the solutions is achieved by matching the delivery time with the expected arrival time of passengers, and by adding sufficient buffer time between two flights scheduled on the same belt. We denote this problem as the Baggage Belt Assignment Problem (BBAP). We first derive a general Integer Linear Programming (ILP) formulation for the problem. Then, we propose a Branch-and-Price (B&P) algorithm based on a reformulation of the ILP model tackled by Column Generation. Our approach relies on an effective dynamic programming algorithm for handling the pricing problems. We tested the proposed algorithm on a set of real-life data from Copenhagen airport as well as on a set of instances inspired by the real data. Our B&P scheme outperforms a commercial solver launched on the ILP formulation of the problem and is effective in delivering high quality solutions in limited computational times, making it possible to use the solution approach in daily operations in medium-sized and large airports.

1. Introduction

We consider the problem of assigning flights to a fixed set of baggage belts (carousels) for arriving luggage in an airport. The objective is to construct a robust schedule taking passenger and airline preferences into account. We denote this problem as the Baggage Belt Assignment Problem (BBAP). This work is motivated by a real-life application in Copenhagen airport and may as well find application in daily operations of other airports. We focus on the following aspects in the delivery process of bags to passengers. Depending on the arrival gate of an airplane, the passengers may have a quite long walking distance to the baggage claim area. If the passengers, furthermore, have to go through passport control, it can take up to 1 h before the passengers can pick up their luggage. If the bags are delivered to the baggage belts before the passengers arrive, it can cause serious congestion problems, since the belts are filled to the limit and cannot accept further bags. In the light of this, we will consider solutions where the bags are not delivered before the arrival of the passengers at the baggage claim area. On the other hand, if the passengers arrive much earlier to the baggage claim area than their luggage, then it will lead to dissatisfied passengers, and overcrowded waiting areas. Hence, we assume that every flight has a *preferred time* for starting the delivery of bags at the baggage claim area. This requested time coincides with the arrival of the passengers at the baggage area. If the bags are delivered later than the requested start time, it will be penalized in the objective function. Also, there may be a *maximum start time* of the delivery, to ensure an upper limit on when bags must be delivered to the passengers. For obvious reasons, every flight has a *minimum start time* for the delivery of the bags. The delivery cannot be scheduled before the arrival time to the gate (time on block, TOB) plus the minimum unloading time and driving time. In our study, we reasonably assume that this delivery time is always smaller than the arrival time of passengers at the belts.

For each flight, we have an expected number of bags. This number can either be reported by the airline, or it can be forecasted from historic information for the same route. Based on the expected number of bags, we can calculate the expected *unloading time* of the bags to the baggage belts. This time can be shortened, but it will be strongly penalized in the objective, since it means in practice that two flights will be unloading bags to the same baggage belt at the same time.

Due to an inherent uncertainty in the walking times of the passengers, number of bags to deliver, and productivity of unloading, a robust plan should assign some *buffer time* between the delivery of bags of two

* Corresponding author.
E-mail addresses: dapi@dtu.dk (D. Pisinger), rosario.scatamacchia@polito.it (R. Scatamacchia).

https://doi.org/10.1016/j.ejtl.2021.100041
Received 10 September 2020; Received in revised form 24 April 2021; Accepted 27 April 2021
Available online 11 May 2021

flights to the same belt. The buffer time is rewarded in the objective function in such a way that the first minutes of buffer time get a large reward, while following minutes of buffer time get a smaller and smaller reward. It is assumed that after a given upper limit on the buffer time, the solution does not become more robust by adding additional buffer time.

Some flights may be *incompatible* with a given belt. For instance, large flights cannot be assigned to small belts or flights arriving from non-Schengen countries must be assigned to specific belts. For every flight we may have *preferences* to the chosen baggage belts. Airline companies may prefer to use baggage belts close to their customer service, so passengers easily can solve questions related to damaged/ missing bags. Furthermore, if the baggage claim area has several entrances, it will be preferred to use baggage belts close to the entrance where the passengers will come from.

Finally, we may assume that the schedule should be *fair* in the sense that if a flight j has a requested time for delivery smaller than the time of another flight j', and both flights are scheduled on the same belt, then flight j must be scheduled before flight j'. If two flights are scheduled to different belts, then no constraint is imposed on the ordering, since waiting passengers are not standing at the same belt, and hence do not block for each other.

The problem is *dynamic* in nature, since arrival times of flights may change due to weather, delays, or technical issues. Therefore, the problem is solved using a *rolling horizon* with segments of about 2 h. Once a plane is on block (i.e. after TOB), the assignment cannot be changed, but all other assignments can be continuously amended to account for new information about forecasted arrival times, and expected number of bags. The assignment problem is typically solved several times per hour, hence fast solution times below, say, 5 min are needed.

For an excellent overview of baggage handling at airports we refer to the comprehensive theses by Barth (2013b) and Frey (2015). Different variants of the Baggage Belt Assignment Problem have been studied in the literature. Barth (2013a) and Barth and Böckmann (2012) study the assignment of incoming flights to baggage belts. They model the problem as an extended assignment problem and present a detailed verification study as well as some sensitivity analysis. Delonge (2012) presents a model for local and global load balancing of baggage belts. Frey, Kiermaier and Kolisch (Frey et al., 2017) consider inbound baggage handling at airports, and present a hybrid heuristic combining a greedy randomized adaptive search procedure (GRASP) with a guided fast local search (GLS) and path-relinking. Barth and Pisinger (2021) formulate a variant of the Baggage Belt Assignment Problem as an extended assignment problem, and solve it as a MIP problem. The objective is to balance customer satisfaction with operational needs. Computational results are reported for real-life instances from Frankfurt Airport. Pisinger and Rude (2020) report results for a variant of the Baggage Belt Assignment Problem that minimizes overlap of aircraft-to-belt assignments, and passenger cross-flow. A simulation analysis presented in Borille and Correia, 2013a, 2013b investigates how different factors impact the service level of arriving passengers.

However, none of the above papers try to match arrival times of bags to the arrival times of passengers in order to avoid congestion and ensure a good passenger flow. Furthermore, nearly all algorithms solve the problem heuristically, and hence optimality cannot be guaranteed.

For the problem we consider in this study, we first present a general Integer Linear Programming (ILP) formulation. Next, we propose a reformulation of the ILP model of the problem and propose a Branch-and-Price (B&P) algorithm using Column Generation (CG) to derive bounds in each node. Our approach relies on an effective dynamic programming algorithm for handling the pricing problems. We tested our solution methods on a set of real-life data from Copenhagen airport as well as on a set of synthetic instances inspired by the real data. Our B&P scheme appears to strongly outperform the commercial solver CPLEX 12.9 launched on the ILP model, and we show that the solution approach is fast enough to be used in a dynamic environment of an airport, improving the daily operational processes.

The main contribution of this paper is threefold: We present a new formulation of the Baggage Belt Assignment Problem that aims at matching the arrival time of bags with the expected arrival time of passengers. We develop an *exact* algorithm for solving the problem based on Branch-and-Price. Finally, we show that the pricing problem can be solved efficiently using dynamic programming.

The paper is organized as follows. In Section 2, we formally introduce the BBAP and a related ILP formulation. We outline the proposed Branch-and-Price scheme in Section 3 and discuss the computational experiments in Section 4. Section 5 draws some conclusions.

2. Notation and formal problem definition

In BBAP, let there be given a set $N = \{1, \dots, n\}$ of flights, a set $M = \{1, \dots, m\}$ of baggage belts and a set of time steps $T = \{0, \dots, (t_{\max} - 1)\}$, where $t_{\max}$ is the time horizon. All flights have to be scheduled within the time horizon, i.e., for each flight, the start time of the delivery of the bags plus the corresponding duration on a belt must not exceed $t_{\max}$. Each belt $i \in M$ has associated a set of compatible flights N_i and a productivity π_i (e.g., bags per minute) for the unloading operations of the bags. For every flight $j \in N$, let b_j denote the number of bags and let t^j_{req} denote the requested delivery start time corresponding to the arrival of passengers at the baggage area. Thus, for each flight j we have a set of possible start times $T_j = \{t^j_{req}, (t^j_{req} + 1), \dots, (t_{max} - 1)\}$. For every belt $i \in M$ and flight $j \in N_i$ we have a set of possible unloading durations W_{ij}. Set W_{ij} includes a nominal duration $w'_{ij} = b_j / \pi_i$ and other durations that may reflect a minimum allowed unloading time, and a maximum time including buffer time. Since the productivity of each belt may differ, the unloading time may be faster at some belts than others. Therefore, the set of possible durations W_{ij} depends on the belt i. Let p_{ijtw} be the profit that can be obtained by assigning flight $j \in N_i$ to belt i with start time t and duration w. The profit is a sum of scores reflecting:

- Capacity: A score is assigned depending on whether the capacity of belt i is sufficient for flight j.
- Preferences: A score is obtained if belt j is located close to the entry gate, or close to the help desk of the handling agent.
- Start time: The closer the scheduled belt time is to the expected arrival of passengers, the higher score.
- Duration: A score is given if sufficient time w is scheduled for unloading.

Buffer time, and hence implicitly robustness, is modeled by increasing the profits for longer durations w. If a flight has a nominal duration w', then we increase the profit of all durations $w > w'$. Similarly, we penalize durations $w < w'$ to reflect that this may mean that baggage from two different flights will be unloaded at the same time to the same belt. We also penalize situations where the delivery of the bags of a flight j starts after the requested start time t^j_{req}. In these cases, lower profits reflect the presence of disappointed passengers and overcrowded waiting areas.

Finally, to ensure a fair schedule, we have a set of precedences to consider, namely we must ensure that each belt processes flights in increasing order of the requested start times t^j_{req} $(j = 1, \dots, n)$. From now on, we assume this ordering of the flights. In order to derive an ILP formulation for the BBAP, we introduce binary variables x_{ijtw} equal to one if and only if flight j is assigned to belt i with start time t and duration w. Clearly, the relevant variables x_{ijtw} are the ones with $t + w \le t_{max}$ and $j \in N_i$. The problem can be formulated as the following ILP model, denoted as $BBAP_{ILP}$.

$BBAP_{ILP}$:

$$\max \sum_{i\in M}\sum_{j\in N_i}\sum_{t\in T_j}\sum_{\substack{w\in W_{ij}:\\ t+w\le t_{max}}} p_{ijtw}x_{ijtw} \tag{1}$$

$$\text{s.t.} \sum_{i\in M}\sum_{t\in T_j}\sum_{\substack{w\in W_{ij}:\\ t+w\le t_{max}}} x_{ijtw} = 1, \qquad j\in N \tag{2}$$

$$\sum_{t\in T_j}\sum_{\substack{w\in W_{ij}:\\ t+w\le t_{max}}} (t+w)\cdot x_{ijtw} \le \sum_{t\in T_{j'}}\sum_{\substack{w\in W_{ij}:\\ t+w\le t_{max}}} t\cdot x_{ij'tw}$$

$$+t_{max}\left(2-\sum_{t\in T_j}\sum_{\substack{w\in W_{ij}:\\ t+w\le t_{max}}} x_{ijtw} - \sum_{t\in T_{j'}}\sum_{\substack{w\in W_{ij'}:\\ t+w\le t_{max}}} x_{ij'tw}\right),$$

$$i\in M, j,j'\in N_i\ (j<j') \tag{3}$$

$$x_{ijtw}\in\{0,1\}, i\in M, j\in N_i, t\in T_j, w\in W_{ij} : t+w\le t_{max} \tag{4}$$

The objective (1) maximizes the profits of the assignments. The first constraints (2) ensure that every flight j is assigned to exactly one belt with a certain duration and start time. Constraints (3) ensure that if two flights j, j' are both assigned to belt i (i.e. $\sum_{t\in T_j}\sum_{w\in W_{ij}} x_{ijtw} = \sum_{t\in T_{j'}}\sum_{w\in W_{ij'}} x_{ij'tw} = 1$), they will not overlap on the belt and will fulfill the precedence constraints. Otherwise, the constraints are inactive. Finally, constraints (4) define the domain of the decision variables.

Using the fact that $|W_{ij}| \le |T|$ we can estimate the size of the model. The formulation contains $O(|M||N|^2)$ constraints, and $O(|M||N||T|^2)$ variables.

It is easy to see that the BBAP is strongly NP-hard by a reduction from the bin-packing problem. In the bin-packing problem, we are given a set of items to be packed in bins. Each item has a weight and all bins have the same capacity. The goal is to find the minimum number of bins that permits the packing of all items. Consider the decision version of the problem that asks whether all items can be packed by using at most K bins. Given an instance of the bin-packing problem, we can construct an instance of the BBAP by associating items with flights and bins with K identical belts. Each belt can process each flight with one possible duration that reflects the weight of the corresponding item. The reduction is completed by setting $t_{\max}$ equal to the capacity of the bins, identical and time invariant profits for all the flights and request start times equal to zero. Solving the BBAP instance and checking if it admits a feasible solution allows us to decide the corresponding instance of the bin-packing problem.

3. A Branch-and-price approach

We propose a B&P approach to effectively tackle the BBAP. The approach is based on a reformulation of model $BBAP_{ILP}$ in which Column Generation is used to solve the LP relaxation. The ingredients of the algorithm and related CG framework are described in the following sections. For an excellent introduction to Column Generation see, e.g., (Desaulniers et al., 2005).

3.1. ILP formulation with fewer constraints, but an exponential number of variables

We derive an alternative ILP formulation with an exponential number of variables. Assume that for each belt i we have a set $\mathscr{S}_i$ of all feasible schedules of the flights. For every schedule $s\in\mathscr{S}_i$, let q_{si} denote the profit of the schedule, and let the binary parameter a_{sij} be one if schedule s for belt i covers flight j. We introduce binary variables x_{si} equal to one if schedule s is used for belt i. We obtain the following formulation, denoted as $BBAP_{exp}$.

$BBAP_{exp}$:

$$\max \sum_{i\in M}\sum_{s\in\mathscr{S}_i} q_{si}x_{si} \tag{5}$$

$$\text{s.t.} \sum_{i\in M}\sum_{s\in\mathscr{S}_i} a_{sij}x_{si} = 1 \qquad j\in N \tag{6}$$

$$\sum_{s\in\mathscr{S}_i} x_{si} \le 1 \qquad i\in M \tag{7}$$

$$x_{si}\in\{0,1\} \qquad i\in M, s\in\mathscr{S}_i \tag{8}$$

The objective (5) maximizes the profits of the selected schedules. Constraints (6) ensure that every flight j is assigned to exactly one belt. The next constraints (7) ensure that at most one schedule will be selected for each belt i. Finally, constraints (8) define the binary domain of the variables.

3.2. Master problem

Since model $BBAP_{exp}$ may contain an exponential number of schedules in sets $\mathscr{S}_i$, it may be intractable to solve. Therefore, we replace the integrality constraints (8) with non-negativity constraints on variables x_{si} and solve the LP-relaxed problem for a subset $\mathscr{R}_i \subseteq \mathscr{S}_i$ of the schedules for each belt i. We obtain the following Restricted Master Problem (RMP) from model $BBAP_{exp}$.

RMP :

$$\max \sum_{i\in M}\sum_{s\in\mathscr{R}_i} q_{si}x_{si} \tag{9}$$

$$\text{s.t.} \sum_{i\in M}\sum_{s\in\mathscr{R}_i} a_{sij}x_{si} = 1 \qquad j\in N \tag{10}$$

$$\sum_{s\in\mathscr{R}_i} x_{si} \le 1 \qquad i\in M \tag{11}$$

$$x_{si}\ge 0 \qquad i\in M, s\in\mathscr{R}_i \tag{12}$$

In our algorithmic implementations, we initialize the RMP with a dummy schedule for each belt in order to guarantee the feasibility of the RMP in any node of the B&P tree. Each dummy schedule processes all flights and has a profit equal to $-\infty$ (i.e., an arbitrarily negative value).

3.3. Pricing problem

Let y_j ($j\in N$) denote the dual variables corresponding to constraints (10), and let u_i ($i\in M$) denote the dual variables corresponding to constraints (11). For each belt $i\in M$, we consider the following Pricing Problem PP_i with binary variables x'_{jtw} representing the assignment of flight j to belt i with duration w and start time t.

PP_i :

$$\max \sum_{j\in N_i}\sum_{t\in T_j}\sum_{\substack{w\in W_{ij}:\\ t+w\le t_{max}}} (p_{ijtw}-y_j)x'_{jtw} - u_i \tag{13}$$

$$\text{s.t.} \sum_{t\in T_j} \sum_{\substack{w\in W_{ij}:\\ t+w\le t_{max}}} x'_{jtw} \le 1, \qquad j \in N_i \tag{14}$$

$$\sum_{t\in T_j} \sum_{\substack{w\in W_{ij}:\\ t+w\le t_{max}}} (t+w)\cdot x'_{jtw} \le \sum_{t\in T_{j'}} \sum_{\substack{w\in W_{ij'}:\\ t+w\le t_{max}}} t\cdot x'_{j'tw}$$

$$+t_{max}\left(2 - \sum_{t\in T_j} \sum_{\substack{w\in W_{ij}:\\ t+w\le t_{max}}} x'_{jtw} - \sum_{t\in T_{j'}} \sum_{\substack{w\in W_{ij'}:\\ t+w\le t_{max}}} x'_{j'tw}\right), \qquad j,j' \in N_i\ (j<j') \tag{15}$$

$$x'_{jtw} \in \{0,1\}, \qquad j\in N_i, t\in T_j, w\in W_{ij} : t+w \le t_{max} \tag{16}$$

The objective (13) maximizes the reduced cost of the schedule of flights to be determined. Notice that u_i is a fixed contribution in the objective function and can be ignored in the computation of an optimal solution to PP_i. Constraints (14) state that each flight $j \in N_i$ could be assigned to belt i. Constraints (15), in combination with constraints (14), ensure that two flights $j, j' \in N_i$ do not overlap if they are processed on belt i (as constraints (3)). Also, these constraints ensure the precedence constraints.

It is easy to see that PP_i contains the 0–1 Knapsack Problem (see monographs (Kellerer et al., 2004), (Martello and Toth, 1990)) as special case, using the same arguments as for BBAP. The input size of PP_i is $\Theta(nw_{\max}t_{\max})$ if the parameters are given explicitly, where $w_{\max}$ denotes the maximum size of an interval W_{ij}. If instead the parameters can be calculated ad-hoc when needed, the order of magnitude of the number of parameters is $\Theta(\log_2 w_{\max}t_{\max})$ for each flight, thus reducing the input size to only $\Theta(n\log_2 w_{\max} + n\log_2 t_{\max})$.

At each node of the Branch-and-Price tree, we solve the *RMP* and obtain the corresponding values of dual variables y_j, u_i. Then, we solve to optimality the pricing problems PP_i for every belt $i \in M$. If a schedule for belt i with positive reduced cost is found, the schedule is added to set $\mathscr{R}_i$. Then, we solve the *RMP* again and the CG process is repeated until no schedules with positive reduced costs can be determined. An optimal solution of the *RMP* provides an Upper Bound (*UB*) on the BBAP. Also, we keep track of integer solutions of the *RMP* along the CG iterations to possibly update the current Lower Bound (*LB*), namely the best feasible solution computed so far.

Clearly, the pricing problems should be solved effectively to speed up the convergence of the iterative process. Here, we notice that each pricing problem PP_i can be seen as a variant of the Knapsack Problem where the position of the packed items impacts the profits and there are precedence constraints among the items in addition to the standard capacity constraint. We denote this problem as the *Position Dependent Knapsack Problem* (PDKP). A crucial observation is that, in each problem PP_i, we have an implicit capacity constraint because two flights cannot overlap on a belt and the flights must be scheduled within the specified time horizon. Since we assume that flights (items) in N_i are sorted by increasing requested start times, we design a recursion that implicitly satisfies the non-overlap requirement and iteratively processes flights according to the specified order to satisfy the precedence constraints. This means that we only have to choose flights and their duration, considering the flights in N_i, up to (capacity) $t_{\max}$. This leads to an effective dynamic programming algorithm to solve the PP_i related to a belt i.

In the Dynamic Program (DP), let $f(t,j)$ be the maximum profit we can obtain by considering only the first j flights in N_i and a time horizon of t. For each $j = 1,\dots,|N_i|$ we indicate by $[j]$ the associated flight in set N. As initial conditions, we set $f(0,j) = 0$ for $j = 1,\dots,|N_i|$ and $f(t,0) = 0$ for $t = 0,\dots,t_{\max}$. We also assume $f(t,j) = -\infty$ if $t < 0$ or $j < 0$. We then have the following recursion within two nested for-loops:

$\forall j = 1,\dots,|N_i|,\ \forall t = 1,\dots,t_{\max}$:

$$f(t,j) = \max\begin{Bmatrix} f(t,j-1), \\ f(t-1,j), \\ \max\limits_{w\in W_{i[j]},\ t^{[j]}_{req}\le t-w} \left\{f(t-w,j-1) + \left(p_{i[j](t-w)w} - y_{[j]}\right)\right\} \end{Bmatrix} \tag{17}$$

The term $f(t,j-1)$ in the recursion represents the case where flight j is not selected. The inner maximization expression considers the selection of flight j with different durations on belt i and finish time t. Clearly, the relevant cases here are only the ones with a start time $(t-w)$ larger than (or equal to) the requested start time $t^{[j]}_{req}$. The term $f(t-1,j)$ indicates a possible update from the subproblem with the first j flights and the immediately preceding time $(t-1)$. Notice that this term is needed to guarantee the correctness of the recursion for any arbitrary distribution of the profits, as a-priori we do not know if the schedule of flight j should finish in time t in an optimal solution of subproblem $f(t,j)$.

The time complexity of the recursion for any pricing problem is $O(nw_{\max}t_{\max})$. The space complexity of the recursion is $O(nt_{\max})$. Notice that we have $w_{\max} \ll t_{\max}$ in realistic BBAP instances (see Section 4). The time complexity of PP$_i$ is linear in the input size $\Theta(nw_{\max}t_{\max})$ if the parameters are given explicitly, but pseudo-polynomial if the parameters can be calculated ad-hoc.

The optimal solution value is given by $f(t_{\max},|N_i|)$. Correspondingly, the maximum reduced cost in problem PP$_i$ is equal to $f(t_{\max},|N_i|) - u_i$. Without going into details, we point out that an optimal schedule of a given pricing problem, i.e. a new column for the *RMP*, can be recovered from the DP entries by implementing an appropriate backtracking procedure and using auxiliary data structures (without increasing the space and time complexities).

3.4. Branching strategy

In our B&P approach we adopt a branching strategy where at most m children are created from a father node by assigning a flight to each compatible belt. We branch on the flight that appears more times in the fractional columns of an optimal solution of the *RMP* in the father node. Clearly, if no fractional columns exist, the father node can be pruned.

Else, each child node inherits all the columns of the father node that are not forbidden by the branching decisions. Correspondingly, the *RMP* is solved by taking into account the previous branching operations in the pricing problems. More precisely, when we solve the pricing problem for a given belt, we artificially increase all profits of the flights that must be assigned to the belt, namely we add a sufficiently large value to the profits of the flights to ensure their selection by the dynamic program depicted in Section 3.3. Notice that if the time horizon is not large enough to process all flights assigned to the belt, we can prune the node by infeasibility. Likewise, in the pricing problem of a belt we do not consider the flights that must be assigned to other belts. Finally, we adopt a Best First Search strategy in the exploration of the B&P tree, namely we first select the node with the largest *UB*.

4. Computational experiments

We tested the B&P algorithm on a set of real-life data from Copenhagen airport as well as on a set of randomly generated instances inspired by the real data. We compared the proposed approach with the commercial solver CPLEX 12.9 launched on the ILP formulation of the problem. We first provide a description of the features of the instances and then discuss the performance of our approach.

4.1. Instances and experimental settings

We first consider real data from Copenhagen airport on five representative days. In order to simulate the rolling horizon, for each day, we considered time slots of 2 h (00:00–02:00, 02:00–04:00,…, 22:00–24:00). The corresponding instances have $t_{\max} = 120$ minutes. The number of flights per time slot was 22 on average with up to 44 in the peak hours. Copenhagen airport normally has 7 belts but at the considered days 2 belts were closed due to infrastructure changes, leaving only 5 belts open. All flights could be assigned to 5 claim belts (i.e., $N_i = N$ for each belt i). The belts had the same productivity π_i (unloading speed) of 10 bags per minute. Besides, the unloading speed of the last belt could be doubled for flights with 100 bags or more, since it has two unloading stations. We derived the requested delivery time t^j_{req} for each flight j by considering the arrival time of the flight and the time needed for passengers to get to the baggage claim area. For each belt i and flight j, we considered a nominal duration (in minutes) equal to the ratio, rounded up to the nearest integer value, between the number of bags and the productivity of the belt, i.e., $w'_{ij} = \lceil b_j / \pi_i \rceil$. According to the value of w'_{ij}, we defined a set of durations W_{ij} with five duration values (including w'_{ij}) and at most two positive duration values smaller than w'_{ij} (so there are at least two durations larger than w'_{ij} to represent buffer times). Two consecutive durations have a difference of 2 min. Profits p_{ijtw} were generated taking into account both start time t and duration w. Due to lack of information, we did not consider in the profits possible preferences of air companies for specific belts. More precisely, each profit p_{ijtw} is set to the rounded value (to the nearest integer) of the following weighted sum

$$\alpha f(w) + (1-\alpha) g(t),$$

with $0 < \alpha < 1, f(w) = \beta_1 \frac{e^{(w-w'_{ij})}}{1+e^{(w-w'_{ij})}}, g(t) = \beta_2 \frac{t_{max}-t}{t_{max}-t^j_{req}}$. The sigmoid function $f(w)$ is used to model decreasing durations and increasing buffer times. Function $g(t)$ contributes to decreasing profits with the increase of start time t: Starting from a value of β_2 for $t = t^j_{req}$, the profits are linearly decreased towards 0 as the value of t gets closer to t_{max}. In our tests, we consider parameters $\beta_1 = 500, \beta_2 = 500, \alpha = 0.5$ and $\alpha = 0.8$ to induce more robust solutions by increasing the contributions of buffer times in the objective function. We considered a total number of 78 instances related to the data from Copenhagen airport.

To get a larger test-bed, we also generated further instances inspired by the real-life data from Copenhagen airport. We considered instances with different number of flights/belts, i.e., $n = 30/m = 5$ and $n = 50/m = 10$ and with $t_{max} = 120$ minutes. We focused on instances where each belt can schedule each flight as these instances are expected to be more difficult to solve. For each belt i, we generated its productivity π_i (baggage per minute) uniformly random in $[10, 20]$. We generated for each flight j a number of bags b_j uniformly random in $[50, 300]$ and a requested start time t^j_{req} uniformly random in $\left[0, \frac{t_{max}}{2}\right]$ and in $\left[0, \frac{3t_{max}}{4}\right]$ to evaluate different congestion scenarios of the baggage claim area. Nominal durations w'_{ij}, sets W_{ij} and profits p_{ijtw} were generated as described above. For each category (identified by the values of n, m, α and the way of generating times t^j_{req}), we generated 10 instances for a total number of 80 instances.

All computational tests were performed on an Intel i5 CPU at 3.0 GHz with 16 GB of RAM. We implemented our B&P scheme in C++ programming language. The Restricted Master Problem in the B&P was solved by CPLEX 12.9 with a relative gap set to 0 and the barrier (interior point) algorithm. We benchmarked the performances of our B&P algorithm on the considered instances against the performances of CPLEX 12.9 launched on model $BBAP_{ILP}$. In the performance comparison the parameters of the solver were set to their default values. We considered a time limit of 300 s for both CPLEX 12.9 and the B&P algorithm. The choice of the time limit was related to the use of the proposed approach in daily operations, which usually require the computation of schedules in short running times.

4.2. Results

We report the computational results for the instances from Copenhagen airport in Table 1. In the considered operational days, there were time slots, such as the late night slots, with very few flights. Since the corresponding instances were trivial to solve, we present only the results for the instances with at least 20 flights landed at the airport in the associated time slot. In Table 1, we report, for each day, the minimum and maximum number of flights n in a time slot, the number of belts m, the value of α considered for profit generation. Each day has 8 instances but the third day for which 7 instances were considered. The table reports the performance of CPLEX 12.9 launched on model $BBAP_{ILP}$ in terms of average computational time (column *time*), average percentage gap $\left(\frac{UB'}{LB'} - 1\right) \cdot 100$ between the best upper bound obtained (UB') and the best solution computed (LB') within the time limit (column *gap (%)*), the number of instances solved to optimality within the time limit (column *# opt*). For the B&P algorithm, the table also reports the average number of nodes explored in the search tree (column *# nodes*). The average values consider also the instances where the solution methods reach the time limit.

Our B&P algorithm managed to solve to optimality 74 out of 78 instances with limited computational times. Notice that the percentage gaps were very small in the instances where the time limit was reached. The number of explored nodes along the branch operations was also reasonably limited. The proposed approach outperformed the solver CPLEX 12.9 that solved to optimality 48 instances. The solver had smaller computational times only in the instances of day 1.

The same trend on the performance emerged in the results reported in Table 2 for the randomly generated instances. The table has the same entries as those of Table 1 except for the first column. In this column, we report the range of values of the requested times t^j_{req}.

The generated instances turned out to be more challenging to solve. This could reasonably be due to more narrow distributions of the delivery requested times and to the presence of belts with different productivity. Our B&P algorithm solved to optimality 56 out of 80 instances. Still, the percentage gaps were significantly small and the approach strongly outperformed CPLEX 12.9. The solver was capable of solving to optimality 3 instances only. We also remark that our approach always provided better feasible solutions and upper bounds than CPLEX 12.9 in all instances where the time limit was reached.

Finally, we notice that the proposed B&P algorithm on average, spent about 90% of the overall computational time in each tested instance for solving the restricted master problems. This highlights the effectiveness of the dynamic programming algorithm in solving the pricing problems. Besides, the number of columns generated in the root node of the search tree was large enough (about 1068 on average) to allow a quick computation of feasible solutions either in the root node or during the exploration of the subsequent nodes.

5. Conclusions

We have presented a B&P scheme for an optimal assignment of flights to baggage belts in the baggage reclaim area. The assignment ensures that for each belt, only one flight is serviced at each time. The approach takes care of a number of business and fairness constraints, avoiding congestion, and ensuring a good passenger flow. Robustness of the solutions is achieved by matching the delivery time with the expected arrival time of passengers, and by adding buffer time between two flights on the same belt. Computational experiments, based on real

Table 1
Results for real-life BBAP instances from Copenhagen airport.

				CPLEX 12.9			B&P			
day	n	m	α	time	gap (%)	# opt	time	gap (%)	# opt	# nodes
1	[20,35]	5	0.5	5.83	0.00	8/8	12.35	0.00	8/8	381.00
			0.8	6.24	0.00	8/8	11.30	0.00	8/8	361.00
2	[20,43]	5	0.5	46.32	0.02	7/8	16.63	0.00	8/8	401.00
			0.8	49.79	0.04	7/8	13.97	0.00	8/8	320.38
3	[25,41]	5	0.5	173.48	1.66	3/7	50.65	0.02	6/7	547.14
			0.8	176.18	1.51	3/7	48.70	0.00	6/7	396.71
4	[23,41]	5	0.5	226.08	1.25	2/8	3.55	0.00	8/8	67.25
			0.8	226.08	1.10	2/8	55.88	0.03	7/8	1018.50
5	[22,44]	5	0.5	152.24	0.78	4/8	7.18	0.00	8/8	84.75
			0.8	152.32	0.74	4/8	40.30	0.00	7/8	272.88

Table 2
Results for randomly generated BBAP instances.

				CPLEX 12.9			B&P			
$t_{req} \in$	n	m	α	time	gap (%)	# opt	time	gap (%)	# opt	# nodes
$\left[0, \frac{1}{2} t_{max}\right]$	30	5	0.5	300.00	10.74	0/10	100.74	0.00	9/10	1347.00
			0.8	300.00	4.70	0/10	120.91	0.01	7/10	1531.90
$\left[0, \frac{3}{4} t_{max}\right]$	30	5	0.5	300.00	4.16	0/10	116.71	0.03	7/10	1456.00
			0.8	300.00	1.79	0/10	77.82	0.00	8/10	996.80
$\left[0, \frac{1}{2} t_{max}\right]$	50	10	0.5	300.00	7.62	0/10	187.85	0.01	5/10	1463.30
			0.8	300.00	4.77	0/10	230.31	0.00	4/10	1901.00
$\left[0, \frac{3}{4} t_{max}\right]$	50	10	0.5	285.72	1.34	1/10	100.86	0.00	7/10	1106.50
			0.8	257.42	0.67	2/10	63.98	0.00	9/10	666.60

data from Copenhagen airport and on randomly generated instances, show that the proposed algorithm is effective in delivering high quality solutions in limited computational times, making it possible to use the solution approach in daily operations in medium-sized and large airports. In future research, it would be interesting to extend the proposed algorithm to similar real-life applications and to further investigate the Position Dependent Knapsack Problem from both a theoretical and practical point of view.

Acknowledgements

The authors would like to thank Copenhagen Airport for having provided data and insightful comments for this project.

References

Barth, T., 2013a. Optimal Assignment of Incoming Flights to Baggage Carousels at Airports. DTU Management Engineering. Technical Report Report 5.
Barth, T., 2013b. *Optimization Of Baggage Handling at Airports*. PhD Thesis. Department of Management Engineering, Technical University of Denmark (DTU).
Barth, T., Böckmann, F., 2012. Baggage carousel assignment at airports: Model and case study. In: International Conference for Airport Operations Management in Munich, pp. 27–30.
Barth, T.C., Pisinger, D., 2021. Baggage carousel assignment at airports: Model and case study. SN Operations Research Forum 2.
Borille, G.M.R., Correia, A.R., 2013a. Determining factors in airport baggage claim level of service. Int. J. Aviat. Manag. 2, 66–79.
Borille, G.M.R., Correia, A.R., 2013b. A method for evaluating the level of service arrival components at airports. Air Transport Management 27, 5–10.
Delonge, F., 2012. Balancing load distribution on baggage belts at airports. In: Operations Research Proceedings, International Annual Conference of the German Operations Research Society in Hannover. Springer, pp. 499–505.
Desaulniers, G., Desrosiers, J., Solomon, M.M., 2005. Column Generation. Springer US.
Frey, M., 2015. Models and Methods for Optimizing Baggage Handling at Airports. PhD thesis. TUM-Bibliothek.
Frey, M., Kiermaier, F., Kolisch, R., 2017. Optimizing inbound baggage handling at airports. Transport. Sci. 51, 1210–1225.
Kellerer, H., Pferschy, U., Pisinger, D., 2004. Knapsack Problems. Springer.
Martello, S., Toth, P., 1990. Knapsack Problems: Algorithms and Computer Implementations. Wiley.
Pisinger, D., Rude, S. í H., 2020. Advanced algorithms for improved baggage handling. J. Airpt. Manag. 14.