A Dynamic Programming Approach for the Aircraft Landing Problem with Aircraft Classes

Alexander Lieder[*◇], Dirk Briskorn[†], Raik Stolletz[*]

[*] University of Mannheim, Business School, Chair of Production Management, Germany

[†] University of Siegen, Department of Operations Research, Germany

[◇] Corresponding author (lieder@uni-mannheim.de)

October 21, 2013

Abstract

The capacity of a runway system represents a bottleneck at many international airports. The current practice at airports is to land approaching aircraft on a first-come, first-served basis. An active rescheduling of aircraft landing times increases runway capacity or reduces delays. The problem of finding an optimal schedule for aircraft landings is referred to as the "aircraft landing problem". The objective is to minimize the total delay of aircraft landings or the respective cost. The necessary separation time between two operations must be met. Due to the complexity of this scheduling problem, recent research has been focused on developing heuristic solution approaches. This article presents a new algorithm that is able to create optimal landing schedules on multiple independent runways. Our numerical experiments show that problems with up to 100 aircraft can be optimally solved within seconds instead of hours that are needed to solve these problems with standard optimization tools.

1 Introduction

The number of passenger flights and cargo flights has been increasing over recent years and is expected to continue to increase. The number of aircraft in use and the number of passengers carried is expected to double within the next two decades (Boeing, 2013). An important limitation in aviation, however, are the runway systems of airports, which limit the number of take-offs and landings per hour. The runway capacity of major European airports is exceeded in periods of high demand, which leads to delays in take-offs and landings. The cost of delays incurred by air traffic flow management (ATFM) (*i.e.,* during take-off, flight, or landing) for all European airports was estimated to be as high as 1.25 billion € (1.61 billion $) in 2011 (Cook and Tanner, 2011). The total ATFM delay cost in North America was estimated to be as high as 4.6 billion $ in 2010 (Ball et al., 2010).

The number of possible landings per hour depends on the types of aircraft involved and on the sequence of operations. Depending on its size and shape, each aircraft causes air turbulence ("wake vortices") that affects the following aircraft. Therefore a minimum separation time between two operations is required. Aircraft are usually divided into a small number of aircraft classes. Table 1 shows a matrix of class-dependent minimum separation times. The values in this matrix are based upon the spacing requirements imposed by the US Federal Aviation Administration (FAA, 2012). Different separation matrices can be found in the related literature, but most of these matrices consist of three to five aircraft classes and have a similar structure (Psaraftis, 1978; Beasley et al., 2001; Soomer, 2008; Harikiopoulo and Neogi, 2011).

		trailing aircraft		
		small	large	heavy
leading aircraft	small	82	69	60
	large	131	69	60
	heavy	196	157	96

Table 1: Separation requirements (in seconds)
Source: Balakrishnan and Chandran (2010)

The aircraft landing problem (ALP) assigns landing times and runways to a given set of aircraft approaching an airport. The planning horizon is very short, as the mean time of an aircraft from the time it arrives within the radar range of an airport (the Terminal Maneuvering Area, TMA) to the targeted landing time is approximately 30 minutes (Balakrishnan and Chandran, 2010). As each aircraft has a preferred landing time, the objective is to minimize the total delay costs for all aircraft landings while respecting the separation requirements. The cost function approximates the actual costs such as fuel, maintenance, exhaust emissions, and passengers missing their connecting flights.

By re-arranging the sequence of runway operations instead of using a priority rule, such as FCFS, a significant reduction of total cost can be achieved. For congested runway systems, this optimization leads to either a reduction of the number of aircraft in holding patterns or to an increase of capacity, *i.e.,* more landings per hour that can be performed. This would lead to a considerable increase in revenue.

Extensive reviews of the literature on the ALP are given by Beasley et al. (2000) and Bennell et al. (2011). Table 2 provides an overview of related articles. The columns in the table show the underlying assumptions of the ALP discussed in the respective articles; most of the articles discuss the ALP with a single runway $(R = 1)$ while others consider multiple (parallel and independent) runways $(R \geq 1)$. The most common

objective is to minimize the total delay costs, but other objectives (such as minimizing the longest delay or minimizing the makespan) are also presented. The delay costs are determined by *cost functions* that are linear or piecewise linear and convex (*i.e.,* the additional cost per period of delay increases). Regarding the target time, we can distinguish two streams of literature; the target times are assumed to be zero or are allowed to be positive. Some of the papers allowing positive target times allow *early landings* to occur, that is, landings before the target time, which are also associated with costs. Most articles assume limited time windows for landings, *i.e.,* there is a *latest landing time* for each aircraft that must not be exceeded by its actual landing time. The last column shows which solution approaches are discussed in the respective articles.

To date, no efficient methods have been proposed in the reviewed literature for the multi-runway ALP that are capable of solving large problem instances. The most common solution approaches are (1) mixed-integer programming (MIP) formulations, which are solved with a standard solver; (2) branch-and-bound (B&B) algorithms; (3) dynamic programming (DP) approaches; and (4) heuristic solution approaches.

MIP formulations: the first mixed-integer formulation for the ALP on a single runway was published by Abela et al. (1993). The extension to multiple runways by Beasley et al. (2000) is the most cited MIP formulation of the ALP to date. Pinol and Beasley (2006) further generalize this formulation to runway-dependent time windows and separation times. Briskorn and Stolletz (2013) proposed a modification of the MIP of Beasley et al. (2000) that explicitly considers aircraft classes.

B&B algorithms: Abela et al. (1993) present a B&B approach for the single-runway ALP. Ernst et al. (1999) develop a B&B solution procedure for the ALP that outper-

forms standard solvers using the MIP formulation by Beasley et al. (2000) but, nevertheless, results in excessive computation times for all instances, except for small problem instances.

Dynamic programming approaches: Bianco et al. (1999) present a DP approach for a single-machine scheduling problem with sequence-dependent setup times that is equivalent to the single-runway case of the ALP. Dear (1976) and Psaraftis (1978) present dynamic programming formulations with a constrained-position-shifting (CPS) assumption, *i.e.,* they assume that each aircraft can be shifted only by a limited number of positions from the sequence of arrivals at the runway system. CPS approaches are also presented by Dear and Sherif (1991) and, more recently, by Balakrishnan and Chandran (2010).

Heuristic solution approaches: Abela et al. (1993) propose a genetic algorithm (GA) as a heuristic solution approach. Bianco et al. (1999) propose two heuristic approaches (cheapest addition and cheapest insertion) for their DP approach. Fahle et al. (2003) compare different exact and heuristic solution approaches for the ALP on a single runway: MIP, integer programming (IP), constraint programming (CP), hill climbing (HC), and simulated annealing (SA). Pinol and Beasley (2006) develop two population-based heuristic approaches (scatter search and a bionomic algorithm) for the ALP. Soomer (2008) and Soomer and Franx (2008) introduce fairness aspects to the ALP by providing airlines the opportunity to define their own cost functions. The numerical study is performed using a local search heuristic.

Many articles assume a fixed number of aircraft classes, but only a few actually use this property in their solution approaches (Psaraftis, 1978; Harikiopoulo and Neogi,

2011; Bojanowski et al., 2011; Briskorn and Stolletz, 2013). The algorithms presented in the remaining articles assume aircraft-dependent cost functions and separation requirements. However, most of the problems discussed feature class-dependent cost functions and separation requirements.

This article contributes to the current state of research in the scheduling of airport runway operations by providing a new optimization algorithm for the ALP with general assumptions (multiple runways, limited time windows, and positive target times). The numerical study indicates that the algorithm generates schedules significantly faster than standard MIP solvers, such as Cplex.

In Section 2, we provide a formal definition of the ALP as a mixed-integer problem. In Section 3, the new optimization algorithm is described in detail. The numerical study in Section 4 compares the results of the algorithm to optimal results of a MIP solver and provides a sensitivity analysis. Section 5 summarizes the major insights and outlines future research.

2 Problem definition

2.1 Problem description

We consider a set, $A = \{1, \ldots, |A|\}$, of aircraft partitioned into a set, $A_1, \ldots, A_W$, of W classes and a set of R identical and independent runways. Class $w(a)$ is the class that aircraft a is contained in, as determined by its respective cost function and separation requirements.

Source	R	Objective	Cost function	Time windows	Solution approach(es)
Dear (1976)	1	min Σ costs	linear	$E_a = T_a = 0$; $L_a = \infty$	Dynamic Program (with CPS)
Psaraftis (1978)	1-2	min Σ costs / min makespan	linear	$E_a = T_a = 0$; $L_a = \infty$	Polynomial dynamic program (with "FCFS within classes")
Dear and Sherif (1991)	1	min Σ delay / min makespan	linear	$E_a = T_a \geq 0$; $L_a = \infty$	Dynamic Program (with CPS)
Abela et al. (1993)	1	min Σ costs	linear	$E_a \leq T_a \leq L_a$	MIP, exact B&B, heuristic (GA)
Ernst et al. (1999)	≥ 1	min Σ costs	linear	$E_a \leq T_a \leq L_a$	MIP, exact B&B, problem search space (PSS) heuristic
Bianco et al. (1999)	1	min Σ delay	linear	$E_a = T_a \geq 0$; $L_a = \infty$	(exponential) dynamic program, heuristics (addition, insertion)
Beasley et al. (2000)	≥ 1	min Σ costs	piecewise linear, convex	$E_a \leq T_a \leq L_a$	MIP, IP
Fahle et al. (2003)	1	min Σ costs	linear	$E_a \leq T_a \leq L_a$	Overview: MIP, IP, CP, heuristics (HC, SA)
Pinol and Beasley (2006)	≥ 1	min Σ costs	linear	$E_a \leq T_a \leq L_a$	MIP, scatter search, bionomic algorithm
Soomer (2008)	1	minmax delay costs	piecewise linear, convex	$E_a = T_a \geq 0$; $L_a = \infty$	MIP, local search heuristic
Soomer and Franx (2008)	1	minmax delay costs	piecewise linear, convex	$E_a \leq T_a \leq L_a$	MIP, local search heuristic
Balakrishnan and Chandran (2010)	1	min makespan	n/a	arbitrary	Dynamic Program (with CPS)
Briskorn and Stolletz (2013)	≥ 1	min Σ costs	piecewise linear, convex	$E_a \leq T_a \leq L_a$	Polynomial dynamic programs, MIP (with "FCFS within classes")

Table 2: Overview of related articles

Each aircraft, $a \in A$, belongs to exactly one class of aircraft in W denoted by $w(a)$ and has a target landing time, T_a, and a latest possible landing time, L_a. Consider two aircraft, a and a', with $w(a) = w(a')$. As in Briskorn and Stolletz (2013), we assume throughout the paper that there is no pair (a, a') of aircraft with $T_a < T_{a'}$ and $L_a > L_{a'}$.

Let $L_w^{\max} = \max\{L_a - T_a \mid w(a) = w\}$. For each class w, a non-decreasing and convex cost function, $c_w(d) : [0, L_w^{\max}] \to \mathbb{R}$, reflects the additional cost depending on the deviation, d, of the actual landing time from the target time of an aircraft of class w. Finally, for each pair (w, w') of classes, a minimum separation time, $s_{w,w'}$, is given.

A solution to the ALP is a schedule $S \subset A \times \{1, \ldots, R\} \times \mathbb{R}$ with exactly one $(a, r, t) \in S$ for each $a \in A$. A triple $(a, r, t) \in S$ represents a being scheduled on r at time t. For $(a, r, t) \in S$ and $(a', r, t') \in S$ with $t' > t$ such that no $(a'', r, t'') \in S$ with $t < t'' < t'$ exists we say that a' immediately follows a .

A solution S is called feasible if

- for each $(a, r, t) \in S$ we have $T_a \leq t \leq L_a$, that is, each aircraft lands in its landing window, and
- for each pair (a, a') of aircraft such that a' immediately follows a, we have $(a, r, t) \in S$ and $(a', r, t') \in S$ with $t' - t \geq s_{w(a),w(a')}$, that is, the minimum separation time is satisfied.

The latter condition describes successive separation, see Beasley et al. (2000).

The problem, then, is to find a feasible solution S which minimizes

$$\sum_{(a,r,t)\in S} c_{w(a)}(t - T_a) \tag{1}$$

among all feasible solutions. We refer to this problem as the *ALP* in the following sections.

2.2 MIP Model

In this section, we present a MIP formulation of the ALP on multiple runways and with successive separation. It assumes aircraft classes with common separation requirements and cost functions. The solution of the MIP with Cplex serves as a benchmark in our numerical study.

The sets, parameters, and variables of the model are shown in Table 3. We use binary variables, $\gamma_{a,a'}$, to indicate if aircraft a' immediately follows aircraft a. Additional binary variables, f_{ar} and l_{ar}, indicate the first and last aircraft on each runway. Variable C_a represents the assigned landing time for each aircraft.

The MIP with successive separation requirements can be stated as follows. Note that the objective function is not necessarily linear, but it can replaced by any (piecewise) linear objective function that is convex and non-decreasing.

Sets:

A	set of aircraft a to be scheduled

Parameters:

W	number of different aircraft classes w
R	number of (identical) runways r
$w(a)$	class of aircraft a: $w(a) \in W \; \forall a \in A$
T_a	target landing time of aircraft $a \in A$
L_a	latest possible landing time of aircraft $a \in A$
$S_{w(a),w(a')}$	minimum separation time between aircraft a and a'
M	a sufficiently large number

Decision Variables:

C_a	assigned landing time for aircraft $a \in A$	
$\gamma_{aa'}$	$= \begin{cases} 1 & \text{if } a \text{ lands } \textit{immediately} \text{ before } a' \text{ on the same runway} \\ 0 & \text{otherwise} \end{cases}$	$\forall \; (a, a' \in A \times A \vert a \neq a')$
f_{ar}	$= \begin{cases} 1 & \text{if } a \text{ lands } \textit{first} \text{ on runway } r \\ 0 & \text{otherwise} \end{cases}$	$\forall \; (a, \in A, \; r = 1...R)$
l_{ar}	$= \begin{cases} 1 & \text{if } a \text{ lands } \textit{last} \text{ on runway } r \\ 0 & \text{otherwise} \end{cases}$	$\forall \; (a, \in A, \; r = 1...R)$

Table 3: Sets, parameters, and variables of the MIP model

$$\text{Minimize } F = \sum_{a \in A} c_{w(a)}(C_a - T_a) \tag{2}$$

subject to the constraints

$$T_a \leq C_a \leq L_a \qquad \forall\, a \in A \tag{3}$$

$$C_a + S_{w(a)w(a')} \leq C_{a'} + M(1 - \gamma_{aa'}) \qquad \forall\, a, a' \in A;\ a \neq a' \tag{4}$$

$$C_a \leq C_{a'} \qquad \forall\, a, a' \in A;\ T_a < T_{a'};\ w(a) = w(a') \tag{5}$$

$$\sum_{a' \in A} \gamma_{a'a} + \sum_{r=1}^{R} f_{ar} = 1 \qquad \forall\, a \in A \tag{6}$$

$$\sum_{a' \in A} \gamma_{aa'} + \sum_{r=1}^{R} l_{ar} = 1 \qquad \forall\, a \in A \tag{7}$$

$$\sum_{a \in A} f_{ar} \leq 1 \qquad \forall\, r = 1, \ldots, R \tag{8}$$

$$\sum_{a \in A} l_{ar} \leq 1 \qquad \forall\, r = 1, \ldots, R \tag{9}$$

$$C_a \geq 0 \qquad \forall\, a \in A \tag{10}$$

$$\gamma_{aa'},\ f_{ar},\ l_{ar} \in \{0;1\} \qquad \forall\, a,\ a' \in A\ \ \forall\, r = 1, \ldots, R \tag{11}$$

The objective function (Equation 2) sums up the total delay costs of all aircraft landings incurred by the delay of the respective aircraft's scheduled landing time, C_a, from its target time, T_a. Equation (3) ensures that each landing is scheduled within the respective time window, $[T_a, L_a]$. The separation requirement for all pairs of subsequent aircraft that land on the same runway is ensured by Equation (4): if $\gamma_{aa'} = 1$, *i.e.,* a

lands immediately before a' on the same runway, the respective landing times, C_a and $C_{a'}$, must be separated by at least $S_{w(a)w(a')}$. Otherwise, if $\gamma_{aa'} = 0$, the equation is valid for a large enough M, *e.g.*, $M \geq L_a + S_{w(a)w(a')} - T_{a'}\ \forall a, a' \in A$.

A key property of aircraft classes is that an FCFS sequence within each class can be assumed (Briskorn and Stolletz, 2013). Equation (5) implements this property by forcing all pairs of aircraft in the same aircraft class to land in the same order as their target times. Equations (6) and (7) ensure that each aircraft, a, has exactly one predecessor (unless it is the first aircraft landing on its runway) and exactly one successor (unless it is the last aircraft). Each runway has at most one aircraft landing first and one aircraft landing last, as stated in Equations (8) and (9).

2.3 Additional position shift constraints

Many articles on the single-runway ALP consider a constrained-position-shifting (CPS) restriction. We generalize this restriction for the multiple-runway setting:

A position shift between aircraft a and a', $T_a < T_{a'}$, in a feasible solution, S, occurs if $(a, r, t) \in S$ and $(a', r, t') \in S$ with $t' < t$, *i.e.,* aircraft a' is assigned to land earlier than a on the same runway. We consider only position shifts among pairs of aircraft on the same runway because we assume independent runways. Given a maximum number *mps* of allowed position shifts, we say that a feasible solution is CPS-feasible if each aircraft, $a \in A$, is involved in no more than *mps* position shifts. The *ALP-CPS* problem then is to find a CPS-feasible solution S that minimizes Equation (1) among all CPS-feasible solutions.

3 Dynamic programming approach

3.1 States and transitions

We propose a dynamic programming algorithm based on the framework proposed by Briskorn and Stolletz (2013). Briskorn and Stolletz (2013) developed a DP approach to prove that the ALP can be solved polynomially in $|A|$ but exponentially in W and R. Their approach was not implemented, as they argue that the size of the state space is too large for a straightforward implementation.

We define each state of the dynamic program as a tuple $(k_1, ..., k_W, rop)$, where

- k_w, $w = 1, \ldots, W$, is the number of aircraft of class w that have been scheduled, and
- *rop* is a *runway occupation profile* (ROP). It is defined as a vector $((O_1, w_1), ..., (O_R, w_R))$ that contains the time O_r and aircraft class w_r of the latest landing on each runway r.

A runway with no operations scheduled is denoted in a ROP as $(-1, -1)$. Note that the state tells us which aircraft have already been scheduled to land due to the FCFS assumption within in each class, and the earliest possible time of the next landing for each class on each runway.

The initial state of the program is $(0^W, (-1, -1)^R)$, *i.e.,* no landings have been scheduled yet. A feasible state transition

$$(k_1, ..., k_W, (O_1, w_1), ..., (O_R, w_R)) \Rightarrow (k'_1, ..., k'_W, (O'_1, w'_1), ..., (O'_R, w'_R)) \tag{12}$$

corresponds to the scheduling of the (k_w+1)th aircraft in class w, namely a, on runway r, that is, we have

- $k'_{w'} = k_{w'}$ for each $w' \neq w$,
- $k'_w = k_w + 1 \leq |A_w|$,
- $(O'_{r'}, w'_{r'}) = (O_{r'}, w_{r'})$ for each $r' \neq r$, and
- $(O'_r, w'_r) = (\max\{T_a, O_r + S_{w_r,w}\}, w)$ with $O'_r \leq L_a$.

This transition is associated with a cost, $c_{w(a)}(O'_r - T_a)$. The cost of a state s can be defined via a **Bellman recursion** as

$$Z(s) = \min_{s' \in \Pi(s,a,t)} \left(Z(s') + c_{w(a)}(t - T_a) \right) \tag{13}$$

where $\Pi(s,a,t)$ is the set of states for which a feasible transition to s by landing a at time t exists, and $Z(0^W, (-1,-1)^R) = 0$.

We consider the set Π of terminal states with $s = (|A_1|, \ldots, |A_W|, rop)$ for each $s \in \Pi$. The ALP then can be solved by finding a state

$$s^* = \arg\min_{s \in \Pi} \{Z(s)\}. \tag{14}$$

The actual schedule can be derived by tracking the sequence of transitions that transform the initial state into s^*, inducing $Z(s^*)$.

This DP approach corresponds to the approach by Briskorn and Stolletz (2013). Both the states and the transitions are defined in a similar fashion. However, because we do

not consider *early landings*, the number of transitions and the state space of the DP are significantly smaller. In Briskorn and Stolletz (2013), for each transition there is a multitude of possible landing times for the next scheduled landing. Without early landings, it is optimal to schedule the next landing as early as possible.

To consider CPS, the dynamic program described above can be modified as follows: For each state transition, *i.e.,* for each landing of an aircraft a that is added to a partial schedule, we count the number of aircraft a' scheduled earlier on the same runway $(C_{a'} < C_a)$ with a later target $(T_{a'} > T_a)$. If this number of position shifts exceeds *mps*, the resulting state is not CPS-feasible and is therefore removed from further consideration. Note that when CPS is considered we still keep the FCFS sequence of aircraft of the same class.

3.2 State-space reduction using a dominance criterion

This section develops a reduction of the state space that is actually searched by employing a dominance criterion and by removing symmetry. We traverse the state space in order by considering states with the smallest number of aircraft being landed yet first. After all states with q aircraft scheduled have been evaluated, we then proceed to those states having $q+1$ aircraft scheduled to land. However, before proceeding to states with $q+1$ aircraft, we sort the R entries of the ROP for each state with q aircraft according to the non-decreasing class of the last aircraft that landed and use the last landing time as a tie-breaker. Note that this does not change the basis for scheduling further aircraft because we assume the runways are identical and independent. This

sorting step will lead to states with symmetric ROPs to be identified in the check for dominated states described as follows.

Using the ROP of a state $(k_1, ..., k_W, rop)$ and the separation requirements, we define $p_{wr} = O_r + S_{w_r,w}$ as the earliest possible landing time for the $(k_w + 1)$th aircraft of class w on runway r for all runways $r = 1, ..., R$ and all aircraft classes $w = 1, ..., W$.

We say that ROP *rop* dominates ROP *rop*$'$ ($rop \succ rop'$) if each runway, r, is available for the next aircraft of each class, w, earlier or at the same time, *i.e.,* if $p_{wr} \leq p'_{wr}$ holds for each $w = 1, ..., W$ and $r = 1, ..., R$. Then we say that a state, s, dominates another state, s', ($s \succ s'$) if

- at least the same number of aircraft k_w of each class w have already been scheduled ($k_w(s) \geq k_w(s')$ for each $w \in W$),
- the cost of s does not exceed the cost of s' ($Z(s) \leq Z(s')$), and
- the ROP of state s dominates the ROP of state s' ($rop(s) \succ rop(s')$).

A dominated state s' can be removed from further consideration, that is, we do not consider any transition that starts in s'.

Note that we should (at least implicitly) check for dominance between each of the pairs of states that is reached. To reduce the computational burden, we carefully traverse the state space by evaluating the states in a non-decreasing order of aircraft that are scheduled. It is then easy to observe that only the dominance needs to be checked between states having the same number of aircraft scheduled for each class.

4 Numerical study

In Section 4.1, we demonstrate the efficiency of the new algorithm by comparing its computation time with that of a standard MIP solver using the formulation presented in Section 2.2. We use two standard data sets from the scientific literature (Bianco et al., 1999) and 10 randomly generated, realistic data sets.

Next, in Section 4.2 we analyze the sensitivity of the algorithm's performance to problem size (40, 50, 60, 70, 80, 90, and 100 aircraft) and to the average inter-arrival times of 30, 35, 40, 50, and 60 seconds. We solve 10 randomly generated data sets for different combinations of the aforementioned parameters on $R = 2,\ 3$, and 4 runways. In Section 4.3, we show the impact of the state-space reduction presented in Section 3.2 on the computation times. Section 4.4 shows how constrained position shifting (CPS) affects the objective values and computation times.

We assume a linear cost function of 1 monetary unit per second of delay for all problem instances in this study. For all randomly generated problem instances, we use the separation matrix in Table 1. All calculations are performed on an Intel Core i5 computer (2.5GHz, 8GB RAM). The MIP formulations are solved with Cplex 12.2; our new algorithm is implemented with Java JDK 1.6. The computation time limit is set to 60 minutes for each problem instance.

4.1 Performance analysis

Bianco et al. (1999) provide two realistic sets of aircraft with target times and separation requirements. These sets are also used in numerical studies, *e.g.,* by Ernst et al.

(1999) and by Briskorn and Stolletz (2013). Set 1 consists of $|A| = 30$ aircraft divided into $W = 4$ classes, and set 2 consists of $|A| = 44$ aircraft divided into $W = 2$ classes. By scheduling each of these sets on $R = 1,\ 2$, and 3 runways, we obtain a total of 6 problem instances. Runway operations can be delayed indefinitely, *i.e.,* $L_a = \infty$ is assumed.

Problem instance	$\|A\|$	W	R	Objective value (Z)	New algorithm CPU time (seconds)	MIP w/ Cplex CPU time (seconds)	MIP w/ Cplex Gap to lower bound	MIP w/ Cplex Gap to opt. objective
1.1	30	4	1	3721	<1	>3600	76%	0%
1.2	30	4	2	219	<1	2	0%	0%
1.3	30	4	3	0	<1	1	0%	0%
2.1	44	2	1	20391	<1	>3600	95%	14%
2.2	44	2	2	660	<1	23	0%	0%
2.3	44	2	3	63	<1	3	0%	0%

Table 4: Computation times of 6 standard problem instances

Table 4 compares the computation times of our algorithm and of the MIP formulation from Section 2.2. The "gap to lower bound" shows the difference between the best feasible solution found by Cplex after one hour and the respective lower bound. The "gap to opt. objective" shows the difference between the solution of our algorithm and the best feasible solution found by Cplex. Our new algorithm optimally solves all 6 problem instances to in less than one second. For $R = 1$, Cplex cannot find proven optimal solutions within one hour.

For the second comparison, we generate 10 data sets using the following parameters. Each data set consists of $|A| = 50$ aircraft divided into 3 classes, with the separation matrix shown in Table 1. By analyzing the inbound traffic of 9 major American airports, Willemain et al. (2004) show that a Poisson arrival process (*i.e.,* exponentially distributed interarrival times) is a realistic approximation of the inbound traffic of an

airport. We use one set of exponentially distributed interarrival times in 10 different randomized orders, with the first target time, T_a, of 0. Therefore, the last target time, $\max_a T_a$, is the same for all 10 resulting profiles. Each target time is randomly assigned to an aircraft class out of a set of exactly 20% small aircraft, 40% large aircraft, and 40% heavy aircraft, which is a realistic configuration (Balakrishnan and Chandran, 2010). We choose an average interarrival time of 40 seconds and a 30-minute time window for each aircraft ($L_a = T_a + 1800$). The resulting distributions of target times and aircraft classes for all 10 profiles are shown in Figure 1.

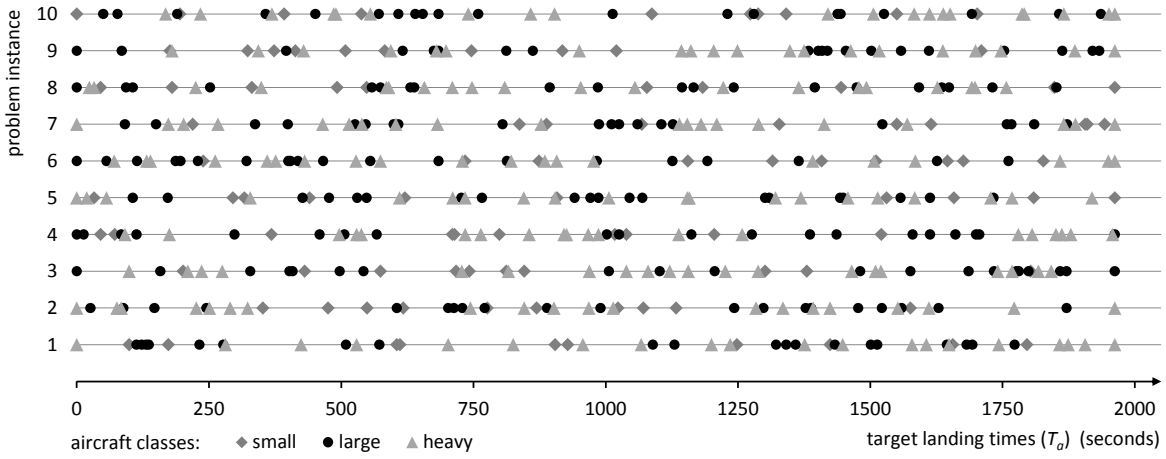

Figure 1: Distribution of target landing times and aircraft classes

Table 5 shows the optimal solutions to our randomly generated problem instances. Each profile is scheduled on $R = 2,\ 3$, and 4 runways. Due to the limited time windows, none of the problem instances has a feasible solution for $R = 1$.

Our new algorithm outperforms the MIP solver in all instances. All instances are solved in less than one minute, while Cplex exceeds the one hour time limit in most cases with $R = 2$ and 3. In particular, for $R = 2$ the gap to the optimal solution is quite

Problem instance	R	New algorithm Objective value (Z)	CPU time	MIP w/ Cplex Solution after 1h	CPU time	Gap to lower bound	Gap to opt. objective
1	2	5410	3	7267	>3600	100%	26%
	3	1027	3	1037	>3600	72%	1%
	4	233	5	233	28	0%	0%
2	2	7277	8	10628	>3600	100%	32%
	3	871	2	876	>3600	72%	1%
	4	109	4	109	13	0%	0%
3	2	5220	2	11445	>3600	100%	54%
	3	966	3	1008	>3600	63%	4%
	4	328	12	328	44	0%	0%
4	2	6002	2	7242	>3600	100%	17%
	3	970	3	970	>3600	70%	0%
	4	222	7	222	19	0%	0%
5	2	5256	3	7960	>3600	100%	34%
	3	454	1	454	2652	0%	0%
	4	42	3	42	9	0%	0%
6	2	8646	3	12115	>3600	100%	29%
	3	928	2	928	>3600	71%	0%
	4	245	3	245	16	0%	0%
7	2	6096	2	8358	>3600	100%	27%
	3	765	2	765	>3600	57%	0%
	4	147	5	147	11	0%	0%
8	2	5859	3	8478	>3600	100%	31%
	3	532	2	532	2875	0%	0%
	4	109	3	109	10	0%	0%
9	2	6491	3	8914	>3600	100%	27%
	3	884	4	889	>3600	54%	1%
	4	219	12	219	24	0%	0%
10	2	5063	2	6115	>3600	100%	17%
	3	590	2	590	>3600	38%	0%
	4	139	5	139	10	0%	0%

Table 5: Comparison of the computation times for the 10 problem instances

large. When $R = 4$, the runway utilization is unrealistically low and the two have a comparable performance.

4.2 Sensitivity analysis

We tested how the **number of aircraft** affects the computation times for problem instances with $|A| = 40,\ 50,\ 60,\ 70,\ 80,\ 90$, and 100 aircraft. The other parameters remain unchanged (40 sec. average interarrival times, 30 min. time windows). Table 6 shows the average and maximum computation times for 10 problem instances of each size. For $|A| = 60$ aircraft or less, the computation took less than one minute in all of the test cases. On average, the larger problem instances took longer to compute, but even for $|A| = 100$ aircraft, the one hour time limit was never reached.

problem size $\|A\|$	$R = 2$		$R = 3$		$R = 4$	
	avg.	max.	avg.	max.	avg.	max.
40	1	1	1	2	2	4
50	4	10	3	4	7	15
60	14	43	12	37	15	37
70	28	80	12	28	22	45
80	85	335	25	73	51	252
90	140	611	29	58	54	91
100	392	1420	41	178	92	524

Table 6: Computation times for the different problem sizes (in seconds)

To observe how the **load on the runway system** affected the computation times, we used problem instances with $i = \{30,\ 35,\ 40,\ 50,\ 60\}$ seconds for problems with $|A| = 50$ aircraft and a 30 minute time window. Table 7 shows the computation times for 10 problem instances for each configuration. For short interarrival times ($i = 30$) and $R = 4$ runways, the time limit was exceeded in one problem instance. If this particular

instance is excluded, the average computation time for $i = 30$ and $R = 4$ is only 19 seconds. All of the problem instances with an average interarrival time of 40 seconds or longer were solved in under one minute.

avg. inter-arrival time	$R = 2$		$R = 3$		$R = 4$	
	avg.	max.	avg.	max.	avg.	max.
30	29	66	117	517	753	>3600
35	23	65	7	23	8	14
40	4	10	3	4	7	15
50	1	1	2	2	4	6
60	1	1	1	1	2	3

Table 7: Computation times for the different interarrival times (in seconds)

4.3 Impact of state space reduction

The following tests show that the algorithm presented in Section 3.1 performs well only with the state-space reduction presented in Section 3.2. We generated a set of small problem instances, with $|A| = 10,\ 15$, and 20 aircraft, with the same parameters as our problem instances from Section 4.1. We solved these instances both with and without state space reduction. Table 8 shows the average computation times (in seconds) and the average number of states created for 10 problem instances in each setting. The table entries ">3600" or "n/a" indicate that the algorithm did not return a solution after one hour.

Without state-space reduction, the dynamic solution approach is not solvable. The number of states created is up to 400 times larger than with state-space reduction. The state space contains a vast number of symmetric and sub-optimal states that make even small problem instances intractable.

$\|A\|$	$\|R\|$	computation time (seconds) *without* state space reduction	computation time (seconds) *with* state space reduction	number of states created *without* state space reduction	number of states created *with* state space reduction
10	2	0,1	0,0	365.060	938
10	3	2,3	0,1	384.820	1.643
10	4	32,8	0,1	436.789	2.665
15	2	2,4	0,1	1.646.941	4.124
15	3	276,9	0,1	1.952.312	5.620
15	4	>3600	0,2	n/a	8.882
20	2	23,1	0,1	911.586	9.648
20	3	>3600	0,1	n/a	10.690
20	4	>3600	0,2	n/a	15.534

Table 8: Impact of the state-space reduction on computation time and state space

4.4 Impact of constrained position shifting

Table 9 shows the objective values and computation times for the 10 problem instances in Table 5, assuming CPS with $mps = 1$, 3 and 5. The objective for the unconstrained case is clearly a lower bound for the constrained case. The computation times with CPS are faster in most cases because on the one hand, the respective state space is smaller, while on the other hand, the algorithm has to perform additional checks and comparisons. For $R = 3$ and 4, the constrained problem with $mps = 1$ returns the same objective values as the unconstrained problem, indicating that the optimal, unconstrained schedules do not include position shifts larger than one.

5 Conclusions and further research

This article presents a new dynamic programming algorithm that is capable of efficiently solving the ALP with different aircraft classes on multiple runways. Based

Problem instance	R	unconstrained Objective value (Z)	unconstrained CPU time	CPS ($mps = 1$) Objective value (Z)	CPS ($mps = 1$) CPU time	CPS ($mps = 3$) Objective value (Z)	CPS ($mps = 3$) CPU time	CPS ($mps = 5$) Objective value (Z)	CPS ($mps = 5$) CPU time
1	2	5410	3	5758	2	5596	2	5524	3
	3	1027	3	1027	2	1027	2	1027	3
	4	233	5	233	2	233	4	233	5
2	2	7277	8	7842	1	7417	6	7277	8
	3	871	2	871	1	871	2	871	2
	4	109	4	109	1	109	3	109	4
3	2	5220	2	5502	1	5220	1	5220	1
	3	966	3	966	1	966	1	966	2
	4	328	12	328	4	328	7	328	10
4	2	6002	2	6506	0	6002	1	6002	1
	3	970	3	970	1	970	1	970	2
	4	222	7	222	4	222	5	222	7
5	2	5256	3	5835	1	5456	1	5456	2
	3	454	1	454	0	454	1	454	1
	4	42	3	42	1	42	2	42	3
6	2	8646	3	9734	1	9149	5	8928	5
	3	928	2	928	1	928	3	928	3
	4	245	3	245	1	245	3	245	3
7	2	6096	2	6396	1	6096	2	6096	3
	3	765	2	765	0	765	1	765	2
	4	147	5	147	1	147	4	147	5
8	2	5859	3	6214	1	5946	2	5924	3
	3	532	2	532	0	532	1	532	1
	4	109	3	109	1	109	2	109	3
9	2	6491	3	6757	1	6520	1	6491	2
	3	884	4	884	0	884	2	884	7
	4	219	12	219	1	219	5	219	10
10	2	5063	2	5292	0	5259	1	5063	2
	3	590	2	590	0	590	1	590	2
	4	139	5	139	1	139	4	139	5

Table 9: Comparison of CPS and unconstrained approach

on a DP formulation by Briskorn and Stolletz (2013), we develop a dominance criterion that eliminates states from the state space while maintaining optimality. The numerical study reveals that the algorithm quickly and optimally solves large problem instances and outperforms the standard MIP solver, Cplex. The study also shows that the new dominance criterion significantly improves the performance of the dynamic programming approach. Furthermore, we show that our approach can be generalized to incorporate constrained position shifting.

In future research, we will extend the solution approach to require less restrictive assumptions than are discussed in the scientific literature regarding the ALP: Some authors assume both take-offs and landings, allow early landings ($C_a < T_a$) with penalty costs, or assume heterogeneous and interdependent runways. In addition, our approach may serve as a framework for heuristic solution methods such as beam search, to make even larger problems tractable.

Acknowledgements

This research was partially supported by the Erich-Becker-Stiftung, a foundation of the Fraport AG, and by the Julius Paul Stiegler Memorial Foundation.

References

Abela, J., D. Abramson, M. Krishnamoorthy, A. De Silva, and G. Mills (1993). Computing optimal schedules for landing aircraft. In *Proceedings of the 12th National Conference of the Australian Society for Operations Research, Adelaide*, pp. 71–90.

Balakrishnan, H. and B. Chandran (2010). Algorithms for scheduling runway operations under constrained position shifting. *Operations Research 58*(6), 1650–1665.

Ball, M., C. Barnhart, M. Dresner, M. Hansen, K. Neels, A. Odoni, E. Peterson, L. Sherry, A. Trani, and B. Zou (2010). Total delay impact study: A comprehensive assessment of the costs and benefits of flight delay in the united states. Technical report, National Center of Excellence for Aviation Operations Research (NEXTOR).

Beasley, J., M. Krishnamoorthy, Y. Sharaiha, and D. Abramson (2000). Scheduling aircraft landings-the static case. *Transportation Science 34*(2), 180–197.

Beasley, J., J. Sonander, P. Havelock, et al. (2001). Scheduling aircraft landings at london heathrow using a population heuristic. *Journal of the Operational Research Society 52*(5), 483–493.

Bennell, J., M. Mesgarpour, and C. Potts (2011). Airport runway scheduling. *4OR: A Quarterly Journal of Operations Research 9*(2), 115–138.

Bianco, L., P. Dell'Olmo, and S. Giordani (1999). Minimizing total completion time subject to release dates and sequence-dependent processing times. *Annals of Operations Research 86*, 393–415.

Boeing (2013). Current market outlook. *http://www.boeing.com/commercial/cmo/*.

Bojanowski, L., D. Harikiopoulo, and N. Neogi (2011). Multi-runway aircraft sequencing at congested airports. In *American Control Conference (ACC), 2011*, pp. 2752–2758.

Briskorn, D. and R. Stolletz (2013). Aircraft landing problems with aircraft classes. *Journal of Scheduling*, doi: 10.1007/s10951–013–0337–x.

Cook, A. and G. Tanner (2011). European airline delay cost reference values. Technical report, EUROCONTROL Performance Review Unit.

Dear, R. (1976). The dynamic scheduling of aircraft in the near terminal area. Technical report, Cambridge, Mass.: Flight Transportation Laboratory, Massachusetts Institute of Technology.

Dear, R. and Y. Sherif (1991). An algorithm for computer assisted sequencing and scheduling of terminal area operations. *Transportation Research Part A: General 25*(2–3), 129–139.

Ernst, A., M. Krishnamoorthy, and R. Storer (1999). Heuristic and exact algorithms for scheduling aircraft landings. *Networks 34*(3), 229–241.

FAA (2012). American federal aviation administration safety alert for operators (safo) #12007. *http://www.faa.gov/other_visit/aviation_industry/airline_operators/airline_safety/safo/*.

Fahle, T., R. Feldmann, S. Götz, S. Grothklags, and B. Monien (2003). The aircraft sequencing problem. In R. Klein, H.-W. Six, and L. Wegner (Eds.), *Computer Science in Perspective*, Lecture Notes in Computer Science 2598, pp. 152–166. Springer Berlin Heidelberg.

Harikiopoulo, D. and N. Neogi (2011). Polynomial-time feasibility condition for multiclass aircraft sequencing on a single-runway airport. *IEEE Transactions on Intelligent Transportation Systems 12*(1), 2 –14.

Pinol, H. and J. Beasley (2006). Scatter search and bionomic algorithms for the aircraft landing problem. *European Journal of Operational Research 171*(2), 439 – 462.

Psaraftis, H. (1978). A dynamic programming approach to the aircraft sequencing problem. Technical report, Cambridge, Mass.: Massachusetts Institute of Technology, Flight Transportation Laboratory.

Soomer, M. (2008). Runway operations scheduling using airline preferences. (PhD Diss., VU University, Amsterdam).

Soomer, M. and G. Franx (2008). Scheduling aircraft landings using airlines' preferences. *European Journal of Operational Research 190*(1), 277–291.

Willemain, T., H. Fan, and H. Ma (2004). Statistical analysis of intervals between projected airport arrivals. Technical report, Pennsylvania State University, Department of Decision Sciences and Engineering Systems.