Check-in computation and optimalization by simulation and IP in combination

van Dijk, N.M.; van der Sluis, E.

Publication date
2003

CHECK-IN COMPUTATION AND OPTIMIZATION BY SIMULATION AND IP IN COMBINATION

Nico M. van Dijk and Erik van der Sluis

University of Amsterdam, FEE, Roetersstraat 11, 1018 WB Amsterdam, Netherlands

Abstract

This paper presents and investigates a check-in problem for a twofold reason:

i) as a problem of practical and novel scientific interest in itself and
ii) as a problem which requires both a stochastic (simulation) and deterministic (integer programming) approach.

First, simulation is used to determine minimal numbers of desks in order to meet a service level for each separate flight. Next, integer programming formulations are provided to minimize the total number of desks and the total number of desk hours under the realistic constraint that desks for one and the same flight should be adjacent. Both opening intervals with constant and variable capacities will be studied.

A numerical example of real world order shows a triple win in waiting time performance, in number of desks and in number of desk hours (staffing). As simulation and integer programming tools are widely available, this combination of simulation and integer programming can thus be regarded as an illustration of a 'new' practical OR-tool for optimization.

Keywords: Airports; Integer Programming; Queueing; Scheduling; Simulation

1 Introduction

1.1 Practical motivation

Despite the temporary stagnation after September 2001 and the present economic recession, the steady growth in aviation seems to have taken up and further growth is anticipated in the next decade. Also, competition between airports and airlines seems to have grown with a general strive for globalization.

Two important objectives in this respect are customer satisfaction and cost effectiveness. Both objectives are important for check-in processes, with queues on the one hand and limited capacities on the other.

Correspondence: H.J. van der Sluis. Tel: +31-20-5254318. E-mail address: H.J.vanderSluis@uva.nl.

Over the last couple of years there has been a strong development of self-service check-in facilities. Nevertheless, traditional check-in desks will certainly remain present for a number of reasons such as security, logistical baggage aspects and, last but not least, traveler preference for personal treatment and ease of use.

At many airports, however, check-in capacity is a scarce resource and during specific hours the number of check-in counters can appear to be rather restrictive for the total demand. Consequently, long waiting times at check-in desks may be encountered.

This in turn, especially in situations of dedicated check-in as generally preferred by (non-domestic) airlines, may lead to undesirable long queues and excessive waiting times at check-in desks.

For airlines this may result in daily conflicts and contention with other airlines for available check-in resources. For airports, considerations might arise for the necessity of check-in capacity expansions in the near future. +

An efficient if not 'optimal' planning of check-in capacities is therefore required at various levels: at daily and operational level to determine the number of desks and opening and closing hours for each individual flight, at weekly level for flight allocation and reservations, at monthly level for contract negotiations with airlines and finally at yearly level for the total desk capacity required.

Clearly, for any of these levels an optimization of capacity resources is thus required at the most detailed level of its daily utilization, as depending on the primary queueing processes and flight demands.

1.2 Operations Research approaches

This optimization essentially involves two steps:

Step 1: A computation and optimization of the number of desks for an individual flight (or group of flights that share a common check-in), in order to meet a specified service level (in term of waiting times).

Step 2: An optimization (minimization) of the total number of desks and staffing hours, in order to meet these numbers.

Step 1 clearly deals with queueing and thus stochastic aspects. Step 2 in contrast is of a scheduling and thus deterministic nature. A pure stochastic approach for step 1 would at best lead to a feasible planning and would not meet the objective of minimizing the overall desk capacities (staffing hours and number of desks). A pure deterministic approach would ignore the essential stochastic aspects that are intrinsically involved and related to the other objective: customer satisfaction (waiting time perceptions and service norms).

A two-step, or more precisely, a combination of a stochastic and deterministic, approach is thus required, for both of which standard OR-tools are widely available:

- Queueing and simulation tools
- MP (ILP and LP) tools

In practice though, check-in planning is often governed by pragmatic check-in rules based on 'expert rules of thumb' and 'executional simplicity'. In general, at best either one of the two approaches is adopted in combination with manual or spreadsheet calculations. In

other words, the potential of OR-modelling and OR-tools, particularly not a combination of them, seems to remain rather unexploited for check-in planning.

In fact, for each of the two steps the approach that will be followed is of interest by itself. A simulation approach for step 1 as in combination with but also as extension of standard queueing and standard steady-state simulation. An integer programming modelling for step 2, which appears to be a new and *NP*-complete problem.

1.3 Objective

The objective of this paper therefore is threefold:

1. To promote the potential of a combined stochastic and deterministic approach.
2. To illustrate this combination in detail for a specific check-in problem of both practical and novel scientific interest.
3. To illustrate its practical potential.

The paper is thus meant as both a promotional and research paper for:

> *The combination of a classical deterministic and stochastic OR-approach as a new opportunity for OR and state-of-the-art OR-tools, for a real world problem of interest.*

1.4 Outline

The outline is as follows. First, related literature is discussed in section 2. Next, in section 3 on step 1, we will first argue the necessity of simulation, more precisely of terminating simulation, as opposed to queueing (sections 3.1 and 3.2). In section 3.3 we discuss the two optimization steps and the minimization by simulation at flight level. Section 3 will be completed by a brief discussion on the advantages of visualization (section 3.4) and of the combination of queueing and simulation (section 3.5).

Then, in section 4, step 2 for the overall desk allocation will be addressed. An integer programming formulation is provided for both the constant desk allocation (sections 4.2 and 4.3) and variable desk allocation (section 4.4). An instructive and real world based example will be included to illustrate the gains. An evaluation concludes the paper.

2 Literature

So far, the combination of the tools described in section 1.2 seems to remain rather unexploited and restricted. As one noteworthy and recent exception, in an elegant promotional article for a special issue of OR/MS-Today, the bimonthly magazine of Informs, a similar combination and coincidentally for the same area of interest, that is, passenger and baggage flows within airports and check-in planning, has been reported by Atkins et al [2].

Also in this article the application of both simulation and LP is reported. Simulation in order to determine queue times and to meet the service criterion (“The simulation was run until the minimum staffing level to meet the service criterion was determined.”), while LP for determining minimum staff numbers and optimal shift schedules (“Given a list of possible

shifts, we used a linear programming model to determine the optimal shift schedule that satisfied the airport-wide staffing requirements in each time period.").

The underlying motivation and details of either approach, however, are not outlined explicitly and appear to be planned for in a more detailed and technical subsequent paper. Nevertheless, the successive and interactive steps mentioned, as well as the general flavour, seem to be highly related and to convey the same general mission of:

> *A combined stochastic and deterministic approach with corresponding tools.*

There are two main differences with our research. In the first place, in Atkins at al. [2], the total staffing requirements are determined during the day at four security points based on the total passenger flows. In this paper in contrast, we need to determine desk requirement for each individual flight. While a steady-state simulation seems justified for the first situation, the setting of this paper requires terminating simulations as will be argued in section 3. Secondly, based on the results of the simulation they determine an optimal shift allocation by solving an LP-model while we determine an optimal flights assignment to desks by formulating and solving an IP-model. LP-models for shift scheduling give integer solutions so that Atkins et al can solve their model in Microsoft Excel. The IP-formulation of our assignment problem, however, seems to be new and appears to be an *NP*-complete problem which requires dedicated (M)IP software like MPL.

As another noteworthy exception and more generally, in Krug [7] the combination of simulation and various search procedures, such as a greedy or gradient search method, for optimization by simulation has been studied in detail. These procedures have also been implemented in software (under the name of ISSOP). However, this automated search optimization does not include more technical specific OR-techniques such as LP.

A somewhat related approach as applied to check-in counter scheduling is presented in Chun [3]. Here simulation is combined with constraint based reasoning which leads to a repetitive algorithm for optimization under constraint patterns. ("The algorithm treats scheduling as a process of kneading and molding a multi-dimensional *counter profile glob* into free unallocated space until…"). The approach seems of interest but does not seem to guarantee a formal optimization.

Most recently, a check-in desk assignment problem has also been addressed by Yan et al [17]. In this paper only the pure deterministic scheduling problem is studied. The assignment problem itself, however, is different. Their objective is to determine an assignment on a monthly basis such that the total passenger walking distance is minimized combined with a constraint of allowable inconsistency. A flight assignment is considered consistent when the same flight number is assigned to the same block of desks on different days. As the model for determining an assignment for one week becomes to large, they propose a three-step heuristic where single day assignment problems are solved.

In Johnson [5], the method of Multiple Service Costing, adopted from the area of micro-economics, is applied to determine total check-in capacities. However, the method is based on total numbers of travelers and not on specific flights. Furthermore, queueing aspects such as waiting time service levels are not taken into account.

A 'historical' queueing reference for the check-in service process, in contrast, is the paper by Lee and Longton [8]. Based on Pollaczek-Khintchine's formula for M|G|1-queues, a comparison is made of combining specific activities for check-in, somewhat related to comparing common with dedicated check-in.

3 Step 1: Simulation

3.1 Why Simulation?

Clearly, the check-in process encounters a number of variable and stochastic aspects. In the first place, there is the variability of flight patterns over months, weeks and days up an hourly level of how many flights are to be checked-in during which hours. Depending on the time-scale of capacity and check-in planning prior to the actual operations, for example at yearly or monthly levels, these variabilities are to be regarded as subject to uncertainty (stochastics).

However, this paper will purely consider the check-in planning problem at the level for which the flight demands are known. That is, with the flights and check-in times for these flights known, say at a daily level. Nevertheless, a number of aspects remain uncertain as:

the number of actual travelers,
the traveler arrival times and
the check-in times.

At first sight, check-in desks can be regarded as simple queueing systems with designated customers and a simple service process, for which standard queueing results and formula are available. However, the opposite appears to be true for a number of reasons.

- First of all, the arrival process, roughly from 50-300 passengers during 1 up to 4 hours prior to flight departure (strongly depending on the type of flight) in relation to the check-in times in the order of a couple of minutes up to quarters, does not at all justify a steady-state type analysis of any kind.
- Second, the arrival process itself is far from homogeneous over the opening interval (see Figure 1 below as an example). A Poisson assumption thus seems far from realistic.
- Third, also for the service (check-in) process itself, standard exponentiality assumptions are clearly violated. At least approximate M|G|s-results would therefore have to be applied.
- Finally, pooling of different flights and overflow from one desk to another may take place, neither of which can adequately be dealt with by (standard) queueing results (see also section 3.2).

Due to the first two aspects a steady-state computation of a check-in process seems far from justified, as also illustrated in the example below, so that a transient computation would have to be in order. Standard queueing results (such as for M|M|c or M|G|c-queues) strongly rely upon a steady-state justification. For the transient case, however, analytic results are hardly available.

Simulation therefore will necessarily be required. More precisely, as will be argued more detailed in subsection 3.2. below, terminating simulation will have to be employed in order to capture the non-steady (or transient) behaviour of check-in processes. Also the other issues can then be dealt with by simulation at the same time.

Example

For example, for the arrival pattern in Figure 1, as based on realistic data, and average check-in times of 3.2 minutes, by queueing theory (Pollaczek-Khintchine's formula) a mean waiting time of 7 minutes would be predicted. More realistically though, 20 minutes was measured, as also verified by simulation.

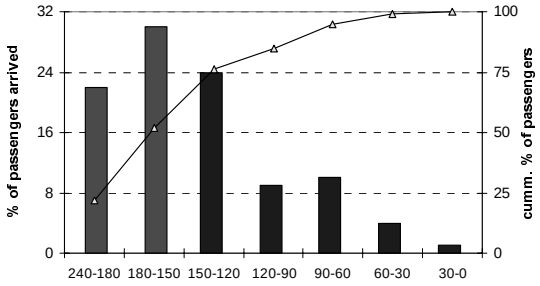

Figure 1: Arrival pattern.

3.2 Terminating Simulation

There can be a number of reasons why a queueing system has a transient or non-steady state behaviour:

1. Most notably, as the arrival process is far from constant (homogeneous) over time (at hourly up to minute level).
2. As the number of arrivals is too small to justify an 'averaging'.
3. As there is a strong initial influence (bias).
4. As the process itself is time-limited.

All of these aspects are in order for check-in processes with strongly fluctuating arrival numbers, with peak periods as well as periods with just a few arrivals, with arrival bursts at opening times and limited desk opening times.

Consequently, terminating simulation will have to be applied. Specific initial situations, such as when desks are opened, limited time intervals during which the desks are open, and varying arrival numbers such as shown in Figure 1 can so be covered.

By sufficiently large numbers of independent replications with the same initial conditions and by sampling from experimental arrival and service distributions, sufficiently sharp confidence intervals can be obtained.

As a special but direct additional feature (advantage) that terminating simulation provides, also the actual capacity, that is the number of desks during the opening period does not need to be constant. For example, it is thus allowed to adjust these numbers beforehand, say by the hour, as based upon the arrival pattern or even dynamically as based upon the actual number of passengers waiting. This feature can be exploited for check-in allocation in order to reduce the number of desks and staffing hours.

Instructive example 1

As an instructive, but nevertheless realistic, example consider the arrival curve from Figure 1 as obtained by realistic data for an international overseas flight from Schiphol airport. The standard planning would allocate a fixed number of 4 desks during 3 consecutive hours prior to the flight. By simulation (as well as measurements), this led to mean waiting times of approximately 20 minutes. More seriously though, during the first opening hour, during which over 50% of all passengers have arrived, excessive waiting times in the order of 40 minutes were measured, while in the second and third hour waiting completely disappeared.

With the service level (norm) that 90% of all passengers reach their desk within 10 minutes, and by running a number of opening and closing scenarios, by simulation it was shown that it suffices to have 6 desks open during the first hour and only 1 for the remaining two.

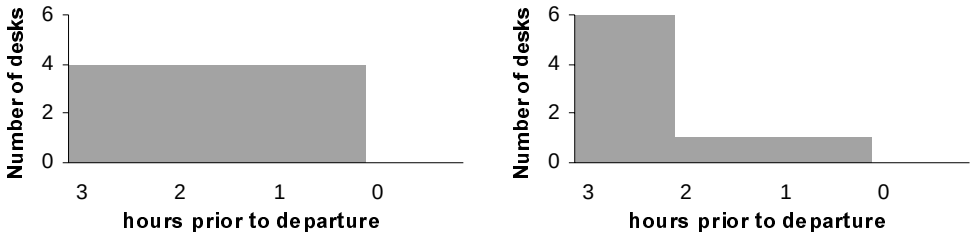

Figure 2: Constant versus variable desk capacity computation.

A *triple* win was so obtained. Not only, in the number of desk (staffing) hours required (from 12 to 8) but also in mean waiting time (from 20 to 12 minutes) as well as in having excluded excessive waiting times during the first opening hour!

3.3 Minimization

As argued in the preceding sections, simulation is thus required for the check-in problem. So far, however, it just provides a tool for computing performances, like in terms of waiting times and queue lengths, when desk capacities (either constant or time depending) are known during the opening intervals.

So far however, there has been no optimization at all. Here a distinction can be made in two optimization steps that are required in order to minimize desk capacities.

Step 1: A minimization of the desk capacity required for each individual flight as based upon simulation, given that some service level is met, such as of 90% travelers with a waiting time less than 10 minutes.

Step 2: A minimization of the desk capacities and staffing hours for all flights, given that the desk requirements for each flight are known.

For the first step (**step 1**), there are no sophisticated methods other than structured search procedures such as described in Krug [7]. In other words, it is essentially based on common sense and experimentation. Clearly, when desk capacity is constant the minimal number is easily found in just a few scenarios. When the desk capacities may vary during the opening interval, it can be more complicated. Even when changes are only allowed at fixed times,

which we will refer to as *'fixed planning',* say by the hour, more options can be found that meet the service level and that have the same minimal number of desk hours. Especially in the latter case, visualization and animation (see section 3.4) can hereby be most supportive.

For example the solutions could be

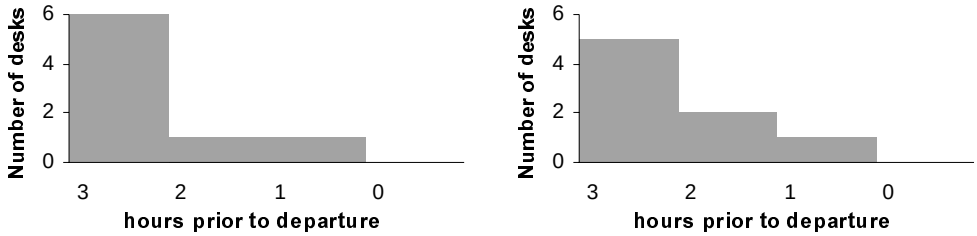

Figure 3: Two alternative fixed plannings with variable capacity.

In that case the option with the 'best overall' service level is chosen. (However, for the overall optimization purpose of step 2, this 'choice' might even be reconsidered for further improvement, as will briefly be argued in section 4.4.) But also options with dynamic capacity changes can be thought of, that is, with changes as based upon the actual queue lengths (called dynamic planning), that might even reduce the number of desk hours further. In this case, the minimization is far more complex and essentially boils down to 'trial and error'. However, as an implementation it is also far more complicated and less appealing. In this paper from hereon (with the exception of section 5) only 'fixed planning' will be considered in which we distinguish between

i) constant capacities (section 4.2)
ii) variable capacities by the hour (sections 4.3 en 4.4)

The second step (**step 2**) is the major and global optimization step for the overall minimization of the total desk and staffing capacities. This step is of a scheduling nature and thus purely deterministic. It will extensively be dealt with separately in the next section.

Remark (planning)

So far, the word planning has been used as a commonly used word up to the term 'fixed planning' as defined above. In what follows we deliberately will not use it in a more technical context, as it could be confusing in 'practical environments' with various interpretations.

3.4 Animation

Though visualization and animation is generally looked upon by OR-specialists, as for manager purposes and non-functional profiling, there are in fact also functional reasons on more serious grounds for which these features can be useful; as they have been for check-in studies at the Dutch airport Schiphol.

- First and foremost, to observe and monitor the *transient* and *non-linear* aspects. As mentioned in section 3.2, this transient behaviour is most natural and essential for check-in. By animation the strong effects of rapidly growing or slowly diminishing queues, of arrival bursts and of space limitations is observed rather easily without requiring consuming simulations for sufficiently reliable computational results.

- Secondly, to see the effects on *dynamic level* of changes (opening or closing desks, overflow, speeding up, etc.). Particularly, in order to 'optimise' the capacity as outline in section 3.3, by visualization a first and quick impression of capacity changes is obtained easily to support the search for 'optimizing' directions.

Figure 4: Animation of a check-in bay.

- Last but not least, to compare different scenarios, such as common versus dedicated check-in, on a *sample path* basis rather than on just average numbers by visualizing the scenarios synchronously in one and the same screen. It can so be checked whether one scenario is indeed stochastically superior to another. Here, rather counterintuitive results might occur for which stochastic monotonicity results might be required to obtain more insight (e.g. Massey [9], Van Dijk and Van der Wal [15], Van Dijk and Taylor [14]).

For example, for a case study at Schiphol airport, the synchronization approach showed that a specific common check-in procedure was **not** superior to the dedicated check-in, as would intuitively have been expected. (The reason for it being that the common check-in led to significantly more variability in check-in times.)

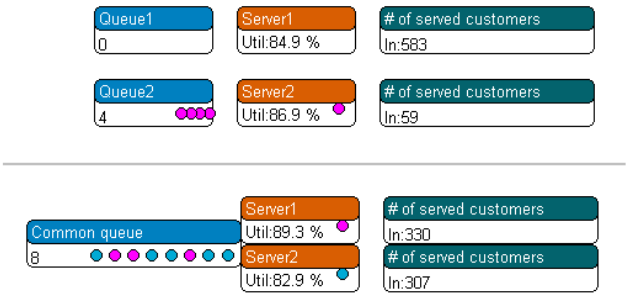

Figure 5: Synchronized comparison of two scenario's.

3.5 Queueing and Simulation in combination

As argued in section 3.1 and 3.2, standard queueing results are too restricted for a sufficient realistic modeling and computation of the queueing features at check-in desks. This is not to say that queueing theoretic results would not still be useful, if not necessary, and practical (see Van Dijk [12], Huisman et al [4]), even in situations such as for check-in analysis when executing simulation.

In the first place, queueing results can provide a first order of magnitude and an indication of results that one may expect. For example, for a single desk and a workload of 75% the most basic queueing rule (or formula $W = 1/(\mu-\lambda)$ with μ the service capacity and λ the arrival rate) would already predict a waiting time of roughly triple the check-in time, despite 25% 'idleness'. This estimate will at least provide a rough lower estimate, as relying upon the assumption of constant workload.

Secondly, queueing theory will provide general insights that can be helpful in order to convince managers. Most notably, it will enable one to suggest possible scenarios of interest for further investigation by simulation. For example, the question may come up whether or not check-in queues should be pooled together. Counterintuitively though, as based on queueing results (such as Pollaczek-Khintchines formula) it turns out that pooling is not always beneficial (e.g. see Rothkopf and Rech [11] or Van Dijk [12]). In fact, a large if not infinite number of pooling and overflow scenarios can be thought of. By queueing insights and depending on the specific situation, these can be limited to just one or a few scenarios of real interest by which the waiting times may be reduced. This in turn will have to be tested by simulation. As mentioned already in section 3.4, in a case-study for Schiphol airport, it was so tested that a dedicated check-in for a specific set of flights should be preferred over a common check-in option.

Finally, 'standard' queueing results can still be most supportive for technical verification of a simulation program itself or for a 'real world' validation. For example, parameters can be given special values (e.g. zero) and assumptions can be made such that the simulation model reduces to an analytical solvable model.

Queueing	**Simulation**
Advantages	Disadvantages
Insights	Numbers
100% exact	Confidence?
Generic components	Too much complexity
Few detailed data necessary	Detailed data required
Disadvantages	Advantages
Too restricted for real life modeling	Allows real-life complexity
Strongly simplified uncertainty-assumptions (exponentiality)	Allows real-life uncertainties
Steady-state analysis	Also allows transient situations

Figure 6: The combined advantages of queueing and simulation.

A combined approach of queueing and simulation is thus suggested. More generally, the major advantages and disadvantages of both approaches are listed in Figure 6 so that by the combination of both approaches also the advantages of both are combined.

In fact, queueing theory seems more applicable for check-in analysis when common check-in is applied. As several flights may check in at the same set of counters, the collective arrival pattern will generally show less fluctuation then the arrival pattern of individual flights. This in turn can lead to a *pseudo* steady state situation in which results from queueing theory become more realistic (also see Van Dijk et al [16]).

4 Step 2: Integer Programming Formulations

4.1 Introduction

Once the required numbers of desks, or rather desk hours, for each individual flight (or group of flights in the case of partial common check-in) are determined by simulation as outlined in section 3 (step 1), the following questions become of practical interest:

1. What is the minimum total number of desks required over all periods?
2. Can we always find a solution with no more desks than the maximum number of desks required during any hour?
3. In the case of variable desk hours, can desk-hours, as saved for one flight, be exploited for another concurrent flight?

In other words, we wish to minimize the total desk requirements, both in terms of desk hours that are to be open and of physical desks. Roughly speaking we thus need to find the best possible, if not optimal allocation. However, also a practical constraint has to be taken into account: desks for one and the same flight should be adjacent. To this end, two Integer Programming formulations will be provided:

i) one that can deal with constant desk requirements and
ii) one that can deal with variable desk requirements.

The IP-models will be referred to as the *constant desk allocation* and the *variable desk allocation* model.

In combination with the types of two results from the simulation step, that is, with constant or variable desk requirements, we will distinguish three practical situations each of which falls within the category of fixed planning (see section 3.3):

1. A *traditional* allocation with for each flight a *constant* number of desks and *constant* number of desk hours during its entire opening interval. In this case, the IP-model for constant desk allocation is used (section 4.2).
2. A *pragmatic* allocation with for each flight a *constant* number of reserved desks, but a (by the hour) *variable* number of desk hours during its opening interval. However, desks that are closed are not used by other concurrent flights. This approach is thus concerned with the total number of desk hours, but not the number of desks! Also here, the IP-model for constant desk allocation is used (section 4.3).
3. The *variable allocation* with both a *variable* number of desks and a *variable* number of desk hours during its opening interval. This offers the opportunity to exploit desks that are closed by one flight to be used by other flights. In this case the IP-model for *variable desk allocation* is applied (section 4.4).

Complication

Even the first situation is not a standard and easy to solve problem due to the adjacency constraints. This desk allocation problem needs to be seen as an extension of the *Fixed Interval Scheduling* (FIS) or Reservation Problem, which is well-known in the literature, (e.g. see Pinedo and Chao [10]).

Without the adjacency constraints, the minimal number of desks could be found easily by a straightforward *Earliest Release Date first (ERD)* rule as follows: First, order all requests in increasing order of release date and assign request 1 to resource 1. Sequentially, assign requests to the first available resource that already has been utilized. When all resources are busy, assign the next request to a so far unexploited resource. And so on. The number of resources used will then be the minimal number of resources required.

The adjacency constraint, however, makes the problem considerably harder to solve, i.e. desks assigned to a certain flight should be located next to each other. For instance, an ERD rule like the one for FIS problems no longer guarantees an optimal solution for the constant allocation problem. This will be illustrated by the following instructive example.

Instructive example 2

Consider the situation of 5 flights and desk requirements as in Figure 7. The total desk requirement in each period is shown in Figure 7. The busiest periods are easily identified, i.e. the periods with the maximum number of desks required. This number will be denoted by N_{max}. In this example $N_{max} = 4$. Clearly, N_{max} is always a lower bound for the optimal number D^*. A first question now arises whether it is possible to find a desk allocation that uses no more than N_{max} desks, that is, $D^* = N_{max}$?

flight	1	2	3	4	5
starting period	1	3	4	5	7
ending periode	3	5	6	7	9
# desks required	3	1	2	1	3

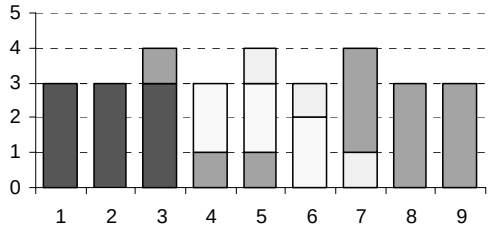

Figure 7: Desk requirements for five flights and total desk requirements for 9 periods.

Starting with an ERD rule would lead to a situation where the first four flights are assigned to (adjacent) desks, while for the fifth flight a sufficient number of desks is available, except that these desks are not adjacent (see Figure 8). Without further modifications this rule will thus lead to either an infeasible schedule or a schedule that requires more than N_{max} desks.

Infeasible schedule

d\t	1	2	3	4	5	6	7	8	9
1	1	1	1	3	3	3	5	5	5
2	1	1	1	3	3	3	5	5	5
3	1	1	1		4	4	4		
4			2	2	2		5	5	5

Feasible schedule

d\t	1	2	3	4	5	6	7	8	9
1	1	1	1		4	4	4		
2	1	1	1	3	3	3	5	5	5
3	1	1	1	3	3	3	5	5	5
4			2	2	2		5	5	5

Figure 8: An infeasible and a feasible (also optimal) schedule.

If the flights are assigned to the first available desk in the sequence (1, 2, **4**, **3**, 5) then a feasible schedule is found (as shown in Figure 8). But, there is no logic for this sequence. In fact, even for this constant allocation problem it can be shown that there is no simple dispatching rule that guarantees an optimal solution.

It will be shown, however, that for both the constant and variable allocation case, integer programming formulations can be provided which guarantee an optimal solution, that is, which compute $\boldsymbol{D}^*$ and an optimal desk allocation with $\boldsymbol{D}^*$ desks.

In order to illustrate these formulations as well as to give an indication of the results that can be obtained, below we provide a numerical example that will be used throughout sections 4.2, 4.3 and 4.4. By this example also more insight will be given in whether $\boldsymbol{D}^* = \boldsymbol{N}_{max}$.

Numerical example

This example is shown by the flight schedule in Figure 10 and is based upon a flight realization at the Dutch airport Schiphol. It represents a flight schedule for the majority of a particular day for one of the bays. At Schiphol airport there are 3 departure halls each with a number of bays.

Except for the main carrier, for which flights can be checked in by common check-in at a number of bays in the central hall, for all other airlines one and the same flight cannot be checked in at more than one bay. As a consequence, as far as desk allocation for other airlines is concerned by dedicated, or partially shared common check-in, the allocation can be restricted to a single bay with a maximum of at most 24 desks, with 12 desks on either side.

With this example, we will illustrate the transformation of a constant schedule into a variable schedule and show that improvements can be achieved in service level, in number of desk hours and in number of desks. First, by terminating simulation as described in section 3.2, we determined the desk requirements for 10 flights based on the expected number of passengers and the following assumptions for all flights:

- Check-in intervals of 3 periods;
- Check-in times which are Gamma (2,1)-distributed;
- Triangle-shaped arrival patterns as shown in Figure 9;
- The service level (norm) that 90% of all passengers will have to wait at most 10 minutes.

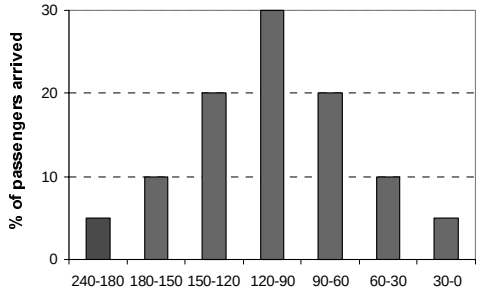

Figure 9: Arrival pattern in minutes prior to the flight departure.

The results from the terminating simulation are summarized in Figure 10 for both constant and variable capacities. For the variable case, note that the number of desks required in the first period can be larger than in the last period, although the arrival pattern is symmetric, as there are already some passengers waiting (5% in our example).

Flight		1	2	3	4	5	6	7	8	9	10
Number of passengers		150	210	240	180	270	150	210	300	180	270
Starting period		1	2	3	3	4	5	6	7	7	8
Constant		3	4	4	3	5	3	4	5	3	5
Variable	1st period	3	3	3	3	4	3	3	4	3	4
	2nd period	3	5	5	4	5	3	5	6	4	5
	3rd period	1	1	2	1	2	1	1	2	1	2

Figure 10: Minimal required number of desks in constant and variable case.

4.2 Traditional allocation

Let us assume that desks are numbered from 1 to D. Due to the adjacency restriction combined with the fixed interval of periods, the assignment of a flight to desks can be described by a single variable d_f which denotes the desk with the largest index assigned to flight f (see Figure 11).

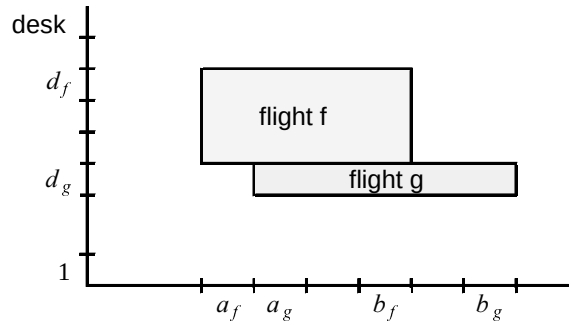

Figure 11: Desk assignment for flights f and g.

For the constant case, the problem of minimizing the total number of desks, given the desk requirements and the adjacency constraint, can then be formulated as:

$$\begin{array}{lll} \min & D & \\ s.t. & n_f \le d_f \le D & \text{for all } f \\ & d_f + n_g \le d_g \text{ or } d_g + n_f \le d_f & \text{for all } f, g \text{ with } I_f \cap I_g \neq \varnothing \end{array} \qquad (4.1)$$

Here

D: Total number of desks required (indexed from 1 to D);
I_f: Check-in time interval of flight f (with $f = 1, \ldots, F$);
n_f: Number of desks required for the check-in process of flight f;
d_f: Largest desk number assigned to flight f.

The first set of constraints ensures that the desks assigned to a flight fall within the interval of 1 to D. The second set ensures that two flights are not assigned to the same desk at the same time. Stated differently, the second set avoids overlap of rectangles defined by d_f and d_g (see Figure 11). By using a binary variable o_{fg} for each disjunctive constraint, (4.1) can be rewritten in the following Mixed Integer Programming.

$$
\begin{array}{lll}
\min & D & \\
s.t. & n_f \le d_f \le D & \text{for all } f \\
 & \left.\begin{array}{l} d_f + n_g \le d_g + M\, o_{fg} \\ d_g + n_f \le d_f + M(1 - o_{fg}) \end{array}\right\} & \text{for all } f, g \text{ with } I_f \cap I_g \neq \varnothing \\
\text{with} & & \\
 & o_{fg} = \begin{cases} 1 & \text{when flight } f \text{ is assigned to lower desk than flight } g \\ 0 & \text{otherwise and} \end{cases} & \\
 & M & \text{an arbitrarily large number.}
\end{array}
\qquad (4.2)
$$

During first experiments we observed that for all problems (randomly generated) a solution was found with $\boldsymbol{D}^* = \boldsymbol{N}_{max}$. This suggested that the constant allocation problem is no more than fitting all flights within $\boldsymbol{N}_{max}$ desks and that an efficient algorithm might exist that finds a solution with $\boldsymbol{D}^* = \boldsymbol{N}_{max}$.

Further analysis of the problem, however, showed that we can construct (counter) examples for which $\boldsymbol{D}^* > \boldsymbol{N}_{max\ x}$. In fact, in a subsequent technical paper it will be shown that the constant allocation problem is *NP*-complete (see Al-Ibrahim et al [1]). In the same paper, it will also be argued that constant allocation problems, or more generally *Adjacent Resource Scheduling* problems, do not belong to any standard class of scheduling problems.

Numerical example (continued)

The required numbers of desks are already given in Figure 10. The busiest period is period 8 in which flights 7 to 10 are checked-in and in total 17 desks are needed ($\boldsymbol{N}_{max} = 17$). Using the IP-model (4.1) an optimal solution was found with $\boldsymbol{D}^* = \boldsymbol{N}_{max} = 17$. An optimal solution is given in Figure 12 and one can easily find alternative solutions. As the desks assigned to a flight are fully staffed during the entire check-in interval, the total number of desk hours is 117.

d \ t	1	2	3	4	5	6	7	8	9	10
1		2	2	2	6	6	6	10	10	10
2		2	2	2	6	6	6	10	10	10
3		2	2	2	6	6	6	10	10	10
4		2	2	2				10	10	10
5	1	1	1	5	5	5		10	10	10
6	1	1	1	5	5	5	9	9	9	
7	1	1	1	5	5	5	9	9	9	
8				5	5	5	9	9	9	
9				5	5	5	8	8	8	
10			3	3	3		8	8	8	
11			3	3	3		8	8	8	
12			3	3	3		8	8	8	
13			3	3	3		8	8	8	
14			4	4	4	7	7	7		
15			4	4	4	7	7	7		
16			4	4	4	7	7	7		
17						7	7	7		

Figure 12: Constant desk requirement and constant desk allocation.

4.3 Pragmatic allocation

As shown by the instructive example 1 in section 3.2, a fixed planning with variable desk capacities can lead to a reduction in both excessive waiting times and in required desk hours. On the other hand, the number of desks required for the period facing the peak of passenger arrivals will generally be larger than the number required for the constant desk capacity case.

Airports that have sufficient check-in desks could then proceed in the following pragmatic manner: a constant desk allocation is obtained by solving model (4.1) with n_f based on the peak demand for flight f. In other words, the airport ensures that airline companies have enough desks available during the entire check-in interval and that they are free to use as many desks as needed (but no more than n_f). During the check-in time interval, desks can be opened (staffed) and closed (say by the hour). In comparison to the traditional allocation, this will lead to a saving in desk hours, but more desks and larger check-in areas will be needed. This will be illustrated below for our numerical example.

Numerical example (continued)

Continuing our example, we first determine the variable requirements such that the total number of desk-hours is minimized while maintaining a service level of at least 90% of all passengers having to wait time at most 10 minutes. The results, as based on simulation, are summarized Figure 10.

Based on the peak requirement for each flight (during the second period of the check-in interval) we determined a constant desk allocation. Again, the busiest period is period 8 with $\boldsymbol{N}_{max} = 20$, although only 15 desks are staffed. Again, $\boldsymbol{D}^* = \boldsymbol{N}_{max}$. An optimal solution with $\boldsymbol{D}^* = \boldsymbol{N}_{max} = 20$ is shown in Figure 13. Now, the total number of desk-hours (staffing hours) is reduced to 92 (a reduction of over 20%).

d \ t	1	2	3	4	5	6	7	8	9	10
1		2	2	2	6	6	6	10	10	10
2		2	2		6	6		10	10	10
3		2	2		6	6		10	10	
4			2					10	10	
5			2						10	
6	1	1	1	5	5	5	9	9	9	
7	1	1		5	5	5	9	9		
8	1	1		5	5		9	9		
9				5	5			9		
10					5		8	8	8	
11			3	3	3		8	8	8	
12			3	3	3		8	8		
13			3	3			8	8		
14				3				8		
15				3				8		
16			4	4	4	7	7	7		
17			4	4		7	7			
18			4	4		7	7			
19				4			7			
20							7			

Figure 13: Variable desk requirement but constant desk allocation.

4.4 Variable allocation

In order to fully exploit the saving in desk-hours, desks that are not used should be made available for other flights. To this end, we need to extend the IP-model (4.1) and specify the required number of desks for any flight in the check-in interval I_f. As desks are opened and closed, the last desk (desk with the largest index) assigned to flight f may change during the check-in interval I_f. We adjust the notation as follows:

n_{ft} Number of desks required for the check-in process of flight f in period t ($t \in I_f$);
d_{ft} Largest desk number assigned to flight f in period t ($t \in I_f$).

For this variable case the problem of minimizing the total number of desks needed for all flights, can now be formulated as:

$$
\begin{array}{lll}
\min & D & \\
s.t. & n_{ft} \le d_{ft} \le D & \text{for all } f \text{ and } t = a_f \\
 & d_{ft} + n_{ft} \le d_{gt} \text{ or } d_{gt} + n_{gt} \le d_{ft} & \text{for all } f, g \text{ and } t \in I_f \cap I_g \\
 & d_{ft} - d_{ft-1} \le \max\{0,\, n_{ft} - n_{ft-1}\} & \text{for all } f \text{ and } t \in (a_f, b_f] \\
 & d_{ft-1} - d_{ft} \le \max\{0,\, n_{ft-1} - n_{ft}\} & \text{for all } f \text{ and } t \in (a_f, b_f]
\end{array}
\qquad (4.3)
$$

The first two sets of constraints are similar to those in model (4.1). The other two sets are added to avoid that during the check-in process of a flight desks are opened and closed at the same time. When there is an increase in the demand for desks ($n_{ft} > n_{ft-1}$) then exactly n_{ft}-n_{ft-1} desks are opened (no closures) and (c) and (d) reduce to $0 \le d_{ft} - d_{ft-1} \le n_{ft} - n_{ft-1}$. Similarly, when there is a decrease in the demand for desks ($n_{ft} < n_{ft-1}$) then exactly $n_{ft-1} - n_{ft}$ desks are closed and (c) and (d) reduce to $0 \le d_{ft-1} - d_{ft} \le n_{ft-1} - n_{ft}$.

Numerical example (continued)

By solving model (4.3) for the example, we so obtained a variable schedule that requires only $\boldsymbol{D}^* = \boldsymbol{N}_{max} = 15$ desks. An optimal schedule is shown in Figure 14.

d \ t	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***
1			2	5	5	5	7	10	10	10
2			2	5	5	5	7	10	10	10
3		2	2	5	5	7	7	10	10	
4		2	2	5	5	7	7	10	10	
5		2	2	2	5	7	7	7	10	
6	1	1	1	4				9		
7	1	1	4	4			9	9		
8	1	1	4	4			9	9		
9			4	4	4		9	9	9	
10								8		
11				3	6	6	6	8		
12				3	6	6	8	8		
13			3	3	6	6	8	8		
14			3	3	3		8	8	8	
15			3	3	3		8	8	8	

Figure 14: Variable capacities and variable desk allocation.

The three situations from sections 4.2, 4.3 and 4.4 are summarized in Figure 15.

Situation	Desk requirements	Allocation	Desk hours	D^*
Situation 1	Constant	Constant	117	17
Situation 2	Variable	Constant	92	20
Situation 3	Variable	Variable	92	15

Figure 15: Summary.

4.5 Computational aspects and extensions

Computation

The IP-models can be solved by standard IP-software. We used the Modeling System MPL with the LP-solver CPLEX 6.6 and a 400 MHz Pentium II computer.

For the purpose of this paper we presented only a straightforward IP-formulation for the constant desk allocation model. In a subsequent and more technical paper we present alternative and more complex formulations in order to improve the computational times. For the constant desk allocation problem a pure zero-one integer program gives the best running times. For example, average computation times of less than 3 minutes were obtained for examples with 96 flights and 48 periods.

The introduction of variable desk requirements, as in section 4.3, leads to larger problems, since for each flight several variables (n_{ft}) are needed instead of only one (n_f). As reported in Al-Ibrahim et al [1], these problems require more computation time. For example, average times of more than 30 minutes were obtained for examples with 48 flights and 24 periods.

Airport reality

Clearly, for real world airport application the complexity and order sizes will be substancially larger than as dealt with herein with hundreds of flights per day and flight patterns that may vary by the week if not daily. Nevertheless, for a number of reasons the results do appear to be of interest, also for such practical realities

First of all, at many airports check-in problems can be decomposed in smaller problems due to natural separations such as in domestic and international flights, in separate check-in areas (bays) and in airline consortia. For example, at the Dutch airport Schiphol a dedicated flight can only be checked-in at one and the same bay. Each bay contains up to 24 desks with 12 on either side and a flight is mostly checked-in at only one side.

Secondly, even though flight patterns might vary, by running and optimizing a number of representative patterns, some general allocation structures and rules might be developed that perform reasonably well with some form of 'practical minimum' in staffing and desks.

Thirdly, also other constraints can be thought of, such as on lay-out structures and staffing restrictions. For example, the layout of desks in bays, each with 24 desks, requires additional constraints for the overall optimization. The following constraints ensure that all desks assigned to a flight are in one and the same bay.

$$n_f + 24(b_f - 1) \le d_f \le 24b_f$$
$$24b_f \le D$$

Multiple step approach

In this paper only a two-step approach is highlighted in order to promote the potential of the combination of simulation and mathematical programming with each step as essential ingredient. An obvious extension, as already mentioned in section 3.3, is a third step when there are more desk patterns with the same number of total desk hours as in Figure 3. The approach from section 4.2 or 4.3 can then be applied to each combination of patterns for all flights and be compared for the best result. To restrict the number of combinations additional constraints could be inserted, for example by setting upper and lower bounds for n_{ft} with n_{ft} as by:

$$l_{ft} \leq n_{ft} \leq u_{ft} \qquad \text{for all } f \text{ and } t$$

In fact, in the setting of section 4.3 it can even be possible that a desk pattern which is not minimal in number of desk hours can lead to an allocation with a minimal number of desks. And, as to be mentioned in section 3.3, further improvements can be sought by allowing to change the number of desks more frequently than just by the hour (fixed). Or even at any time based on the actual queue lengths and some prescribed rule (dynamic). An extensive simulation search would then be required.

As another possibility, a number of $\boldsymbol{D}^*$ desks, as obtained in the first two steps, might not be available. Also in this case, an iterative procedure comes in mind. For example, to relex the service level requirement by gradually decreasing the number of allocated desks hours for certain flights. In other words, research in the direction of iterative steps is still open.

Other method

As already mentioned in section 4.2, during our experiments we observed that for all examples (randomly generated) $\boldsymbol{D}^* = \boldsymbol{N}_{max}$. In other words, the main purpose of the IP-models is to find a solution that occupies not more than $\boldsymbol{N}_{\max}$ desks. In that respect, other techniques like Constraint Programming, as executed in Chun [3], might also be suitable for this problem. Nevertheless, (counter)examples, can be found as given in Al-Ibrahim et el [1], with $\boldsymbol{D}^*$ strictly larger than $\boldsymbol{N}_{\max}$. In that case, a search procedure for $\boldsymbol{D}^*$ seems necessarily required, which is directly included by IP. Also here, further research will be of interest.

5 Evaluation

In this paper a new problem of check-in planning has been studied as of practical interest as well as to promote the potential if not necessity of a combination of a stochastic and deterministic OR-approach for real world applications. For the check-in problem this combination concerns a two-step approach:

Step 1: Simulation in order to capture the underlying queueing processes.
Step 2: Mathematical (integer) programming in order to minimize the staffing and desk capacities.

Each of these two steps has been argued as of interest in itself. The first step as necessarily required in order to deal with transient behaviour, such as intrinsically involved with a check-in queueing process. The second step as a general approach for scheduling and optimal capacity allocation under all sorts of realistic constraints, such as desk adjacencies for check-in.

Also pragmatic aspects of both steps are addressed. For the simulation step of how simulation is also to be used to ‘optimise’ capacity under given service requirements. For the integer

programming step of how it can also be used under different 'practical' situations (traditional, pragmatic, variable).

The combination was illustrated and shown to be fruitful for a numerical check-in example of realistic order. A *triple* win was achieved in excessive waiting times, in desk (staffing) hours and in number of desks.

A third and iterative step are briefly mentioned for further improvement and possible future research. As software tools for simulation and IP are widely available this combination can thus be regarded as a new opportunity of classical OR techniques.

The same combination also seems of interest for a variety of other optimal planning and scheduling problems such as for check-in planning to explore and optimize the possibilities of more flexible check-in periods prior to the flights, of common versus dedicated check-in procedures and of overflow of (e.g. economy) passengers.

More generally, it also seems useful for optimization problems in other application areas such as:

- Call centers
- Manufacturing
- Transportation
- Health care science
- Administrative logistics

Nevertheless, for practical purposes a number of challenges for this combined approach still remain as also illustrated by the check-in problem. In the first place, a structured optimization procedure for simulation, such as in order to develop dynamic rules for when to open and close desks. Secondly, heuristic practical rules for optimal integer programming formulations, which are generally *NP*-hard. And last but not least, the actual implementation and transformation of these OR-tools into application-oriented and user-friendly practical tools.

Acknowledgement

The authors are grateful to the anonymous referees who made constructive comments, which improved the presentation of the paper. The approval of the Dutch airport Schiphol to include realistic aspects and data related to them is appreciated.

References

[1] A. Al-Ibrahim, C. Duin, E. van der Sluis, Adjacent Resource Scheduling, Technical Paper, AE-Report 2/2003, University of Amsterdam, 2003 (Downloadable from website http://www.fee.uva.nl/ke/sluis/).

[2] D. Atkins, M.A. Begen, B. Luczny, A. Parkinson, M.L. Puterman, Right on Queue, OR/MS Today, April 2003, Vol 30, no 2.

[3] H.N. Chun, Scheduling as a Multi-dimensional Placement Problem, Engineering Applications of Artificial Intelligence 9 (1996) 261-273.

[4] T. Huisman, R.J. Boucherie, N.M. van Dijk, A solvable queueing network model for railway networks and its validation and application for the Netherlands, European Journal of Operational Research 142 (2002) 30-51.

[5] A.P. Johnson, Multi-Service Costing, Journal of the Operational Society 29 (1978) 551-558.

[6] P.E. Joustra, N.M. van Dijk, Simulation of Check-in at Airports, in: Proceedings of Winter Simulation Conference 2001, pp 1023-1028.

[7] W. Krug, Modelling, Simulation and Optimisation for Manufacturing, Organisational and Logistical Processes. Erlangen, Deutschland, Gruner Druck GmbH., 2002.

[8] A.M. Lee, P.A. Longton, Queueing Processes Associated with Airline Passenger Check-In, Operational Research Quarterly 10 (1959) 56-71.

[9] W.A. Massey, Stochastic Ordering for Markov Processes on Partially Ordered Spaces, Mathematics of Operations Research 12 (1987) 350-367.

[10] M. Pinedo, X. Chao, Operations Scheduling with Applications in Manufacturing and Services, McGraw-Hill, 1999.

[11] M.H. Rothkopf, P. Rech, Perspectives on Queues: Combining queues is not always beneficial, Operations Research 35 (1987) 906-909.

[12] N.M. Van Dijk, Why queueing never vanishes, European Journal of Operations Research 99 (1997) 463-476.

[13] N.M. Van Dijk, Hybrid Combination of Queueing and Simulation, in: Proceedings of Winter Simulation Conference 2000, pp 147-150.

[14] N.M. van Dijk, P.G. Taylor, On Strong Stochastic Comparison for Steady State Measures of Markov Chains with a Performability Application, Operations Research 46 (1998) 665-674.

[15] N.M. van Dijk, J. van der Wal, Monotonicity of Throughput in Non-Markovian Networks, Queueing Systems 4 (1989) 1-26.

[16] N.M. van Dijk, L. van Wassenhoven, E Yücesan, An experimental analysis of steady-state convergence, Implications for flexible manufacturing system models. Simulation Practice and Theory 4 (1996) 1-29.

[17] S. Yan, C-H Tang, M. Chen, A model and a solution algorithm for airport common use check-in counter assignments, Transportation Research Part A 38 (2004) 101–125.