HUB RESEARCH PAPER
Economics & Management

Integrated staffing and scheduling for an aircraft line maintenance problem

Jeroen Beliën, Erik Demeulemeester, Brecht Cardoen

HUB RESEARCH PAPER 2010/23
JULI 2010

partners in
Hogeschool-Universiteit Brussel

Integrated staffing and scheduling for an aircraft line maintenance problem

Jeroen Beliën[1] †‡, Erik Demeulemeester‡, Brecht Cardoen§‡

†Hogeschool Universiteit Brussel, Faculty of Economics and Management, Center for Modelling and Simulation

‡Katholieke Universiteit Leuven, Faculty of Business and Economics, Department of Decision Sciences and Information Management

§Vlerick Leuven Gent Management School, Operations and Technology Management Center

Abstract

This paper presents a method for constructing the workforce schedules of an aircraft maintenance company. The method integrates both the staffing and the scheduling decision. We formulate the optimization problem using a mixed integer linear programming approach and solve it heuristically using a branch-and-bound enumeration framework. Each node of the tree represents a combination of teams and team sizes. An extensive computational experiment is added to illustrate the algorithmic performance using a set of test instances that are derived from a real-life setting. We conclude this paper by discussing some results and by addressing interesting future extensions to the model.

Keywords: aircraft maintenance, workforce scheduling, integer programming, branch and bound

[1]Corresponding author: Jeroen Beliën, Hogeschool Universiteit Brussel, Faculty of Economics and Management, Center for Modelling and Simulation, Stormstraat 2, B-1000 Brussels, Belgium, 0032(0)26098271, jeroen.belien@hubrussel.be, erik.demeulemeester@econ.kuleuven.be, brecht.cardoen@vlerick.be

1 Introduction

This paper examines how to minimize the labor costs of an aircraft maintenance company that stem from the provision of its line maintenance programme. A line maintenance programme represents the regular short inspections of aircraft between their arrival and their departure at some specific airport. Two decisions are of particular interest in minimizing labor costs. On the one hand, one has to decide on the appropriate staffing levels that are necessary to meet the demand of customers over time, which implies that we are interested in finding the appropriate number of technician teams, as well as their corresponding team size. On the other hand, one has to determine accurate rostering or scheduling that translates the availability of the teams into an adequate shift structure. Both decisions are examined in an integrated way while considering numerous conditions that stem from imposed collective agreement requirements or coverage constraints.

The literature provides an extensive amount of research on airline staffing and scheduling problems, such as the flight scheduling problem, the fleet assignment problem, the aircraft routing problem, the crew scheduling problem and the crew bidding problem. We refer to Mercier and Soumis (2007) for a clear description of these problem settings. In particular the aircraft routing problem may be of interest in clarifying the scope of this paper, as this problem involves aircraft maintenance. In the aircraft routing problem, one determines the sequence of flight legs to be flown by each individual aircraft so that each leg is covered exactly once while ensuring the required level of aircraft maintenance (Mercier and Soumis, 2007). The type of maintenance in this setting is preventive in nature, which implies that it needs to be performed depending on the accumulated number of flight hours and the number of take-off and landing cycles (Sriram and Haghani, 2003). One distinguishes between type A checks (every 65 flight-hours), type B checks (every 300 to 600 flight-hours), type C checks (once a year) and type D checks (once every 4 years). Besides these types of maintenance, there is also a line maintenance programme. Line maintenance includes regular short inspections of the aircraft between arrival and departure at an airport (Gupta et al., 2003). Yan et al. (2004) refer to this line maintenance as short-term layover maintenance and define it as a combination of preflight checks (prior to take-off), transit checks (between two connected flights serviced by the same aircraft) and daily checks (executed when an aircraft stays overnight). Together with the type of the aircraft, these checks determine the workload to be performed in line maintenance.

In this paper, we investigate how a line maintenance service company should determine its staffing and scheduling decisions in order to minimize the resulting labor costs. When an individual airline also has its private maintenance company, line maintenance decisions can be incorporated into the aircraft routing problem and optimized in an integrated way. This means that the line maintenance workload can be adapted and spread by taking the staffing capacity into account. Dijkstra et al. (1991), for instance, present a decision support system for determining the size and the composition of the workforce at the aircraft maintenance department of the Dutch national airline company at the main airport in the Netherlands. However, airlines may also contract a maintenance company that services multiple airlines. In this way, one could save costs by taking advantage of pooling benefits (see, e.g., Kilpi et al. 2009 for an extensive study on cooperative strategies for the availability service of repairable aircraft components). This implies, however, that the workload of line maintenance is pre-determined by the airline, so that the maintenance company has to adapt its staffing requirements to meet the demand pattern. This latter presumption applies to this research too.

In the literature, decisions on aircraft maintenance are frequently studied as part of the aircraft routing problem (see Papadakos (2009) for a recent example) and integrated with other problems that constitute the airline scheduling problem. As mentioned in the previous paragraph, this paper studies maintenance from the viewpoint of a maintenance company that serves multiple, competing airlines, which implies that decisions on the spread of workload cannot be changed by adapting the routing decisions. Besides this particular setting, this paper focuses on staffing and scheduling decisions concerning the line maintenance programme, which is a maintenance aspect that has rarely been studied up to now. Alfares (1999) presents a case study on an aircraft line maintenance workforce scheduling problem. In particular, he compared the impact of a five-day work week and a seven-day work week in combination with morning and afternoon shifts on the amount of weekend overtime, while meeting the expected increase in workload. The resulting alternatives are formulated as integer programming models. Yang et al. (2003) aim at minimizing the total line maintenance manpower supply, taking flexible management strategies into account. These strategies relate to numerical and temporal flexibility. They allow one to decide on team sizes, the number of shifts needed and their starting times. They furthermore distinguish between full-time shifts and part-time shifts. They formulate their problem using a mixed integer programming approach

and apply the model to a case study. Yan et al. (2004) elaborate on the previous model by adding constraints concerning certification (i.e., the degree of training and functional abilities) of the maintenance personnel. Gupta et al. (2003) formulate a simulation-based optimization approach to minimize the total number of technicians working overtime by adapting the applied shift structure. Particular interest was given to the utilization of the technicians, as well as the amount of workload that had to be passed on to the next shift.

Although some of the above references integrate staffing and scheduling decisions in their problem setting, it should be noted that these decisions are frequently considered as two separate problems (see Mundschenk and Drexl (2007) or Thompson (1997) for examples). In a first phase the staffing problem is solved, which results in workforce sizes. Next, these workforce sizes form the input for the scheduling problem. In practice, however, the staffing and scheduling problem is sometimes solved at the same moment. This entails interesting opportunities, as such a solution to the staffing problem has a serious impact on the scheduling problem. A model that integrates the staffing and the scheduling problem could take advantage of this dependency. Og[breve]uz et al. (2010) show that integrating the scheduling decision with a higher level of decision-making, i.e. order acceptance, can be both effective and computationally tractable. The achievement of an integrated staffing (i.e., determining how many employees should be hired) and scheduling (i.e., determining when these employees should work) system as formulated in this research paper is basically a non-linear problem. We propose an enumeration approach in which a mixed integer linear problem (MILP) is solved for every interesting combination of team sizes and weeks in the roster cycle. In this way, we do not advocate a single solution to the problem, but list a set of qualitative solutions as input for successive bargaining rounds.

The remainder of this paper is organized as follows. In Section 2 we introduce the problem setting and describe the objective and the specific set of constraints that constitute the optimization problem. We address particular attention to the coverage constraints, as we allow for a certain flexibility with respect to the exact timing of the maintenance and the number of persons working on a particular aircraft. Hence, the timing of the workload (i.e., the format in which the service is delivered) is an extra decision in the model. Section 3 elaborates on the developed solution heuristic, which combines mathematical programming with an enumerative branch-and-bound framework. Section 4 reports on the algorithmic performance of

the solution approach by means of a set of test instances that are derived from a real-life setting. We conclude this paper in Section 5 by recapitulating some interesting findings and by addressing future extensions to the model.

2 Problem setting

2.1 Cycles, teams and shifts

As mentioned in the introduction, the line maintenance problem of interest consists of a staffing decision and a rostering decision. The staffing decision entails the determination of the team sizes working on the line maintenance programme, while the rostering decision concerns the definition and timing of the shifts. Teams may vary in size, depending on the cycle they are working in. A cycle can be seen as a number of consecutive weeks. The number of weeks in a cycle is equal to the number of teams to whom the cycle applies. The number of weeks in a cycle is also equal to the number of shift sequences that can be fulfilled by the teams working in the cycle. In one particular week, each team of the cycle will execute one specific shift sequence. Figure 1 illustrates the above reasoning. In this figure a cycle is depicted that consists of three weeks, and thus of three teams. Each row in Figure 1 (a) represents a shift sequence. Each column in Figure 1 (a) denotes a day of the week (Monday till Sunday). The transition from Figure 1 (a) to (b) illustrates that the short notation actually captures all the information needed to write out the cycle in full. It should be clear that team 1 will carry out shift sequence 1 in week 1, that team 2 will carry out shift sequence 2 in week 1 and that team 3 will carry out shift sequence 3 in week 1. In other words, the set of shifts that is delivered in each week of the cycle is the same. The team that will carry out a specific shift sequence, however, will differ. Figure 1 (c) is added to illustrate that each shift sequence consists of a specific sequence of shift types. In the example, we differentiate between a morning shift (M), a day shift (D) and a night shift (N). In the problem setting, we will also define an evening shift (E). Note that the exact starting time and finishing time of such a shift type may differ between cycles. Team 1 starts week 1 of the cycle with a morning shift on Monday, followed by a day shift on Tuesday till Sunday. In its second week, team 1 starts with five days off, etc. Consequently, after the cycle time (in this case, 3 weeks) each team has worked all shifts in the roster and restarts working the same sequence of shifts.

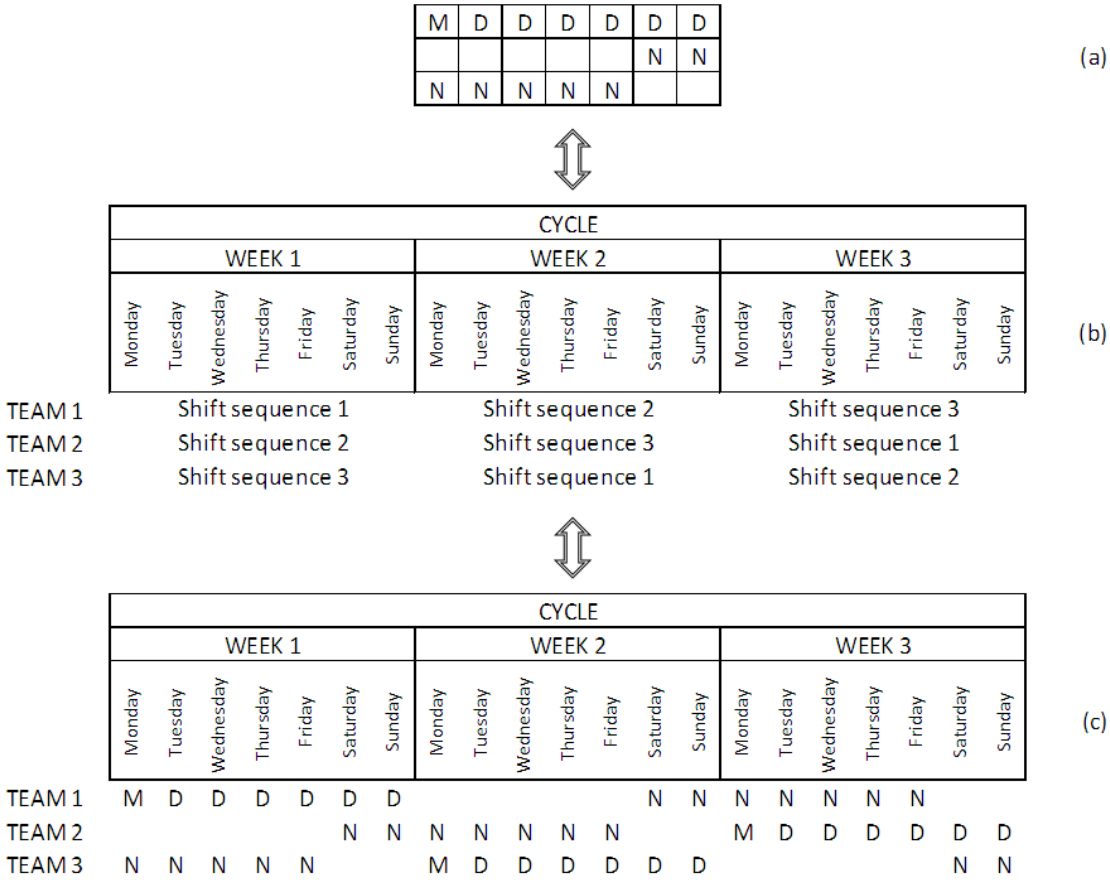

Figure 1: Illustrating the short notation of a cycle and its translation to shifts and teams

To solve the problem of interest to us, we need to find a feasible combination of cycles, with for each cycle a number of teams (= weeks), team sizes and team rosters (shifts and shift hours) that minimizes the total costs of labor. We refer to this combination, which integrates a staffing and rostering decision, as a schedule.

As mentioned before, a roster consists of one or more cycles. Usually, a 'night' cycle, containing nothing but night shifts, is included in order to be able to better match the workforce capacity with the workload during the night, which often differs substantially from the workload during the day. Figure 2 shows a night cycle of two weeks. In this cycle the first team starts with four consecutive night shifts, followed by seven days off and three night shifts, after which the cycle restarts. The second team starts with four days off, followed by seven consecutive night shifts and three

days off, after which the cycle restarts. A number of consecutive shifts (e.g., the seven consecutive night shifts in the night cycle) without a day off is further referred to as a 'block'.

N	N	N	N			
				N	N	N

Figure 2: An example of a night cycle containing two weeks (=teams)

In conclusion, the problem we try to solve involves deciding on:

- the number of cycles,
- the number of teams (= weeks) per cycle,
- the team sizes,
- the starting time and finishing time of the shifts,
- the schedule of the shifts (day, week).

2.2 Overview of the constraints

Obviously, only feasible schedules are of interest in minimizing labor costs. A feasible schedule is a roster that satisfies the following restrictions.

First, the roster must satisfy all collective agreement requirements. These constraints are typical for services involving around-the-clock workforce scheduling (see, e.g., Chiaramonte and Chiaramonte 2008 for an example of these constraints in a nurse scheduling case study).

1. The shift starting hours must lie within certain intervals, depending on the type of shift (e.g. a morning shift (M) must start between 5 AM and 6.30 AM).
2. The total duration of a shift must lie between a minimum and a maximum number of hours.
3. The average number of working hours per week per worker must be between a minimum and a maximum, e.g., between 36 hours and 38 hours.

4. Workers must have a minimum number of weekends off, expressed as a fraction of the total number of weekends, e.g. a worker must have 50% of the weekends off.

5. There must be a minimum number of hours between two successive shifts. Starting from realistic feasible starting time intervals for the different shifts (see earlier), this constraint implies that:

    - a night (N) shift cannot be followed by a morning (M), day (D) or evening (E) shift on the next day,
    - an evening (E) shift cannot be followed by a morning (M) or a day (D) shift on the next day.

    We will refer to these sets of constraints as the shift succession constraints.

Second, within a cycle, all shifts of a given shift type (M, D, E or N) must have the same starting and finishing hours. The result is that all workers within a certain cycle have the same hours for a morning (evening, day, night) shift. The reason for this constraint is that it substantially simplifies the daily control of the workforce as the operational scheduler, who assigns specific flights to (teams of) workers, is faced with a limited number of shift times. Indeed, the maximum number of different morning (evening, day, night) shifts equals the number of cycles in the schedule. As a result, it is much easier to compose teams and have a clear view on the number of workers available during each time interval.

Third, the service to clients, signed up in service contracts, must be guaranteed. Therefore, our solution must result in sufficient capacity to cover the workload at any time instance. In line with the literature, we will refer to this third class of constraints as the coverage constraints. We will elaborate on the coverage constraints of our problem setting in Section 2.3, as they substantially differ from traditional coverage constraints.

Fourth, even if no client flights are scheduled, there must be at least one team on standby 24h/24h, seven days per week, to take care of clients with a disrupted flight schedule.

None of the preceding constraints can be violated in a feasible solution and are therefore referred to as hard constraints. Alongside these hard constraints, there are a

number of soft constraints. Soft constraints can be violated, but it is preferable to satisfy them as far as possible. The soft constraints in this application apply to shift transitions in blocks. Recall that a block is defined as a sequence of shifts without a day off. First of all, the length of a block is ideally between 5 and 8. Second, within a block, employees prefer to work only one type of shift, e.g., always a morning shift.

2.3 Coverage constraints

Nearly all workforce scheduling problems contain coverage constraints. Coverage constraints indicate for each time period how many workers are needed to 'cover' the work in that time period: e.g., Monday before noon: 10 workers, Monday afternoon: 15 workers, etc. Coverage constraints of this form are referred to as normal coverage constraints. For two reasons, however, the aircraft maintenance company could not provide data on normal coverage constraints.

First, the workload is not given per time period, but per client (flight). Consequently, the number of workers that need to be present depends heavily on the flight schedules of their clients. Moreover, the exact timing of the maintenance is not fixed. The only requirement is that the maintenance takes place within a certain time interval; i.e., between the flight scheduled time of arrival (STA) and the scheduled time of departure (STD). Hence, the crucial information needed to define the coverage constraints includes: (1) the scheduled departure and arrival times and (2) the workload of each flight.

It is important to notice that this demand for services is repeated on a weekly basis, resulting in a cyclical demand pattern. For the company that inspired us for this research, this weekly demand pattern changes two times per year: at the end of the winter season (emphasis on business flights) and at the end of the summer season (emphasis on holiday flights). It is therefore at these moments that a new schedule needs to be produced, in order to rematch the capacity with the changed demand pattern. This cyclical demand pattern justifies a cyclical supply pattern and therefore a schedule with cycles having fixed team sizes (see Section 3.1).

Second, although the total workload is given for each flight, the company can decide how to divide this workload between different workers. For instance, a workload equal to four hours, can be performed by four workers, each working for one hour,

by two workers, each working for two hours, or by two workers, one working for one hour and the other working for three hours. As a matter of fact, an endless number of combinations is possible.

Since time can be divided fractionally, this requires for the inclusion of continuous variables (see Section 3), which, in contrast with binary and integer decision variables, only have a marginal impact (or even no impact at all) on the difficulty (and, hence, required computation time) of solving the problem.

3 An enumerative MILP approach

It is well known that workforce scheduling problems of realistic dimensions involving all the real-life constraints mentioned in Section 2.2 can hardly ever be solved to optimality within a reasonable computation time (see, e.g., Burke et al. (2004) who conclude this fact after reviewing the literature on nurse rostering). Hence, an exact approach (like MILP) does not seem to be the most promising approach to solving this problem. Metaheuristic approaches have proven to be more robust with respect to finding an acceptable solution in limited computation time. The main drawbacks of metaheuristics, however, are (1) that no guarantee or even information can be given on the quality of the solution and (2) that metaheuristics are often very problem-specific and, therefore, harder to adapt if the problem (slightly) changes. Both drawbacks partly explain our choice for a MILP based approach. Besides the special coverage constraints which can very easily be handled by MILP, (see Section 2.3), the most important reason is that the problem could be considerably simplified, making a MILP approach possible, without harming the quality of the outcome. More specifically, the hard shift transition constraints and the soft constraints mentioned above dramatically impact the problem complexity, while their management information contribution is rather limited. Hence, we will ignore these constraints and assume that, given a solution that satisfies all other constraints, an overall acceptable solution can be found. Figure 3 (a) visualizes the output of the model, after which the cycle of Figure 3 (b) could be constructed, by dividing the shifts over the different weeks.

For the studied problem dimensions (which correspond to real-life dimensions), this last step can be done either manually or by using an automated procedure like an IP assignment problem (which is beyond the scope of this paper). Nevertheless, there is

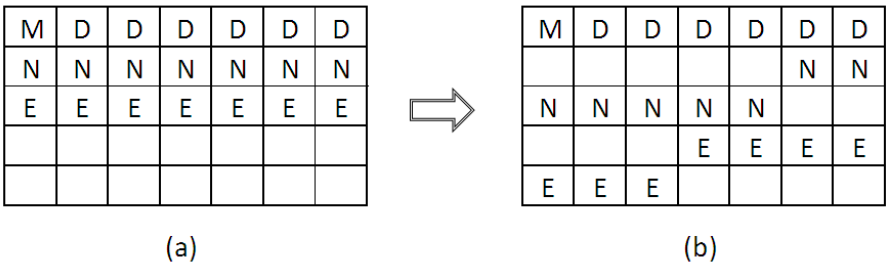

Figure 3: Example of the output of our approach for a cycle containing five weeks (= teams); all shifts are placed in the first three weeks, as only the number and type of shifts on each day are determined

no guarantee that, starting from our solution, a good (or even feasible) overall schedule can be constructed. Therefore, our approach can easily be adapted to generate not one, but several solutions (see Section 3.3.2) and, moreover, special constraints are added in order to increase the probability of finding an overall feasible schedule (see Section 3.2). Before stating the model, however, Section 3.1 explains which parts of the decision process are frozen in order to develop a workable model that could be optimized for a number of interesting combinations of these fixed values. The enumeration approach that is used to explore the promising combinations is described in Section 3.3.

3.1 Fixing the number of cycles, number of weeks per cycle and team sizes

It must be clear now that the more cycles we introduce, the more flexibility we have in building the schedule and therefore the lower the cost of the resulting schedule will be. For instance, the maximum number of different morning (day, evening, night) shifts equals the number of cycles, as at most one shift can be defined for each shift type within a particular cycle. Moreover, several cycles allow for more combinations of worker teams in order to better match the available capacity (workers) to the required demand for service (flights). Having many cycles, however, complicates the task of the operations manager, and therefore four or more cycles are often not considered as a realistic option to work with. Consequently, only schedules of one, two or three cycles should be evaluated in practice. The algorithm presented below starts from a fixed number of cycles and solves the problem for this number. The advantage of this approach lies in the fact that the optimization model simplifies as

the decision on the number of cycles could be removed. In order to evaluate other numbers of cycles, the algorithm should be rerun.

Besides the number of cycles, the number of weeks in each cycle is also part of the decision process. The minimum number of weeks in a cycle, however, equals 2, because a cycle of 1 week implies that the workers either work all weekends (if a shift during the weekend is scheduled) or none at all (if no shift is scheduled during the weekend). A so-called weekend cycle is an exception as workers in this cycle would agree to work each weekend, and so, a cycle of one week is perfectly possible. The maximum number of weeks in a cycle was also bounded, as a large number of weeks (= number of teams) corresponds to a small average team size. To ensure that teams could be composed in which all important skills were present, the maximum number of weeks per cycle was set at 8. Let W^c denote the number of weeks in cycle c. as with the number of cycles C, W^c is considered as being fixed in one optimization model.

Finally, the number of workers in each team is fixed in advance and thus left out of the MILP optimization process. The reason for this is that incorporating a team size variable leads to a non-linear objective function, as the scheduling variable (see below) is multiplied by the team size variable in order to calculate the total costs. Recall that the demand for services follows a weekly pattern (see Section 2.3). Knowing this the teams, working within a cycle, must have the same team size in an optimal solution. For a proof, consider the case in which there is a team with a different size to that of the other teams in the cycle. There must be at least one shift on a particular day in which this team is scheduled while another team is not scheduled. Otherwise, we would have two (or more) identical week schedules and we can decrease the number of weeks (= teams) in the schedule by replacing the identical weeks by one week. Given the cyclical pattern, there must come a moment during which this other team performs this shift, while the first team is having a day off. Since a capacity shortage is not allowed, the team sizes of both teams must suffice to cover the demand. Hence, we can decrease the size of the largest team to the size of the smallest team and obtain a cheaper schedule (having the same team sizes for each team in the cycle).

In conclusion, we fix the number of cycles in advance. Given a particular number of cycles, all interesting combinations of numbers of weeks and a fixed team size for each team in each cycle are explored using an intelligent enumeration approach.

For each interesting combination a MILP is solved. The next section presents this MILP.

3.2 The MILP model

The notation used in this model is as follows:

The indices and sets are:

$i \in I$:	feasible shifts
$t \in T$:	shift types $\{M, D, E, N\}$
$i \in I^t$:	feasible shifts of shift type t
$c \in C$:	cycles in the schedule
$d \in \{1, 2, ..., 7\}$:	days in the week
$p \in P$:	time periods in one week
$f \in F$:	flights to be serviced
$f \in F^p$:	flights that are available to be serviced during time period p
$p \in P^f$:	time periods during which flight f can be serviced

The data parameters are:

$WORKLOAD_f$:	the workload (in man-hours) of flight f
M_c :	the team size in cycle c
W_c :	the number of weeks (= teams) in cycle c
CO_{id} :	the total cost (for one worker) of shift i on day d
DUR_i :	the duration of shift i (in hours)
a_{ip} :	= 1 if period p is included in shift i; = 0 otherwise
b_i :	fraction of duration of shift i available to work = 1 - (duration lunch break / duration of shift i)
$MINAVGHOURS$:	the minimum average number of working hours per week
$MAXAVGHOURS$:	the maximum average number of working hours per week
$WEEKENDFRAC$:	the maximum fraction of working weekends; i.e., weekends during which at least one shift is scheduled

Set I contains all feasible shifts i, i.e., all combinations of feasible shift starting times and feasible shift durations.

A first way of formulating this problem uses the following binary decision variable:

$$x_{idwc} = \begin{cases} 1, & \text{if shift } i \text{ is scheduled during day } d \text{ of week } w \text{ in cycle } c; \\ 0, & \text{otherwise.} \end{cases}$$

Preliminary computational results, however, indicated that the resulting model could not be solved within a reasonable computation time. The problem was twofold. First, for realistic problem sizes, the number of binary decision variables, which equals $|I| * 7 * \sum_{c \in C} W_c$, was very large, leading to (too) many branches in the branch-and-bound tree. Second, the shift succession constraints (see Section 2.2) dramatically complicated the problem, raising the required computation times to an unacceptable level. Fortunately, both problems could be solved by removing the week dimension from the problem. Therefore, consider the following integer decision variable:

x_{idc} = the number of shifts i scheduled during day d in cycle c.

This decision variable, however, does not allow us to formulate the shift transition constraints (see earlier). Since the shift transition constraints could easily be handled afterwards, given the solution(s) provided by our approach, this was not considered a major issue.

In order to formulate the coverage constraints, we need a second decision variable, which is a normal, continuous variable and reflects the flight's timing of workload decision:

y_{fp} = the number of workers assigned to maintain flight f during time period p.

Note that y_{fp} is not required to be an integer, as the number of workers assigned during a time period might be fractional. E.g. if 2.5 workers are assigned to a particular flight, then two workers spend the whole time period maintaining this flight, while a third worker spends half of the time period on this flight.
Finally, the binary decision variable z_{ic} is required in order to formulate the restriction of only one shift definition of each shift type (M, D, E and N) per cycle:

$$z_{ic} = \begin{cases} 1, & \text{if shift } i \text{ is used in cycle } c; \\ 0, & \text{otherwise.} \end{cases}$$

Now, the model can be stated as follows:

minimize

$$\sum_{c\in C}\sum_{i\in I}\sum_{d=1}^{7} CO_{id}M_c x_{idc} \tag{1}$$

subject to

$$\sum_{i\in I} x_{idc} \leq W_c, \qquad \forall d = 1,...,5 \qquad \forall c \in C \tag{2}$$

$$\sum_{i\in I} x_{idc} \leq WEEKENDFRAC * W_c, \qquad \forall d = 6,7 \qquad \forall c \in C \tag{3}$$

$$\sum_{i\in I}\sum_{d=1}^{7} DUR_i x_{idc} \geq MINAVGHOURS * W_c, \qquad \forall c \in C \tag{4}$$

$$\sum_{i\in I}\sum_{d=1}^{7} DUR_i x_{idc} \leq MAXAVGHOURS * W_c, \qquad \forall c \in C \tag{5}$$

$$x_{idc} \leq W_c z_{ic}, \qquad \forall i \in I \qquad \forall d = 1,...,7 \qquad \forall c \in C \tag{6}$$

$$\sum_{i\in I^t} z_{ic} \leq 1, \qquad \forall t \in T \qquad \forall c \in C \tag{7}$$

$$\sum_{i\in I}\sum_{c\in C} a_{ip} x_{idc} \geq 1, \qquad \forall p \in P \tag{8}$$

$$\sum_{i\in I}\sum_{c\in C} a_{ip} b_i M_c x_{idc} \geq \sum_{f\in F^p} y_{fp}, \qquad \forall p \in P \tag{9}$$

$$\sum_{p\in P^f} y_{fp} = WORKLOAD_f * \frac{|P|}{24*7}, \qquad \forall f \in F \tag{10}$$

$$x_{idc} \in \{0,1,...,W_c\}, \qquad \forall i \in I \qquad \forall d = 1,...,7 \qquad \forall c \in C \tag{11}$$

$$y_{fp} \geq 0, \qquad \forall f \in F \qquad \forall p \in P \tag{12}$$

$$z_{ic} \in \{0,1\}, \qquad \forall i \in I \qquad \forall i \in C \tag{13}$$

The objective function (1) minimizes the total costs. The cost of a shift CO_{id} depends on the duration of shift i, on the shift type of shift i (night shifts are typically the most expensive shifts, day shifts are the cheapest) and on the day of the week (weekend days are typically more expensive than week days).

Constraint set (2) ensures that the number of shifts scheduled on a particular day during the cycle does not exceed the number of weeks (= teams) in the cycle.

Constraint set (3) models the weekend restriction; i.e. the number of shifts on Saturdays (on Sundays) can be at most the weekend fraction multiplied by the number of weeks in the cycle. For instance, if the weekend fraction equals 0.67, i.e. at least one weekend out of three must be a free weekend, and if the cycle consists of six weeks, then there can be at most four shifts on Saturday (on Sunday).

Constraint sets (4) and (5) make sure that the average number of working hours per week per worker lies within a specified interval, for instance between 36 and 38 hours. Note that the constraint applies to the average. Hence, there might be weeks in which the number of working hours is larger than the upper bound or smaller than the lower bound.

Constraint sets (6) and (7) guarantee that all M (D, E, N) shifts have the same hours within one cycle. Note that shift i is defined by a starting and finishing hour. If a particular shift i is scheduled once or several times, constraint (6) makes the corresponding z_{ic} equal to one. Next, constraint (7) ensures that no more than one z_{ic} per shift type can equal one within each cycle.

Constraint sets (8)-(10) model the coverage constraints. Constraint set (8) ensures that, at any time, at least one team is present in order to take care of flights with disrupted schedules. Constraint set (9) makes sure that, at any time, the capacity, i.e. workers present, exceeds the demand, i.e. the planned maintenance. Constraint set (10) guarantees that each flight is maintained. Observe that, in order to calculate the workload in person-periods, we have to multiply the workload (expressed in person-hours) by the number of periods per hour (= the total number of periods per week (= $|P|$) divided by the total hours per week (= 24*7)). Furthermore, note that, instead of modeling the lunch break as a zero capacity time interval during each shift, the decrease in capacity is spread over the entire shift. The reason for this is that the employees are required to have their lunch break during non-busy moments and not all at the same time.

Finally, constraints (11)-(13) define the decision variables.

Preliminary results indicated that the weekends often cause infeasibility in the second step, which involves assigning the shifts, as defined by the solution of the MILP, to the weeks (see the beginning of Section 3). This is due to the weekend constraint, which states that a specific fraction of the weekends must be free of any shift. Con-

sequently, only a limited number of weekends is available for assigning shifts. As a result, all available Saturdays and Sundays must be assigned a shift, which often causes shift transition violations from Saturday to Sunday. Figure 4 gives an example of an infeasible solution. Here, the weekend fraction is 50%, and so half of the weekends must be free of any shifts. This means that the three Saturday shifts (N, N and M) and the three Sunday shifts (M, M and N) must be combined into three weekends, which is impossible without violating the shift transition constraint that an N shift cannot be followed by an M or E shift on the next day.

M	D	D	D	D	N	M
N	N	N	N	N	N	N
E	E	E	E	E	M	E
M	M	D	M	D		

Figure 4: Example of a solution that cannot be turned into a feasible schedule if the weekend fraction equals 50%; it cannot be avoided that an N shift on Saturday is followed by an M or E shift on Sunday

In order to increase the probability of finding a good overall schedule in the second step, a number of extra constraints can be added to the MILP. The following set of constraints tries to avoid infeasibility problems caused by the weekend bottlenecks:

$$\sum_{i \in I^t} x_{i6c} = \sum_{i \in I^t} x_{i7c} \qquad \forall c \in C \qquad \forall t \in T \tag{14}$$

Constraint set (14) ensures that the number of M (D, E, N) shifts on Saturday equals the number of M (D, E, N) shifts on Sunday. As a result, the shift transition constraints can easily be satisfied in the second step, by scheduling the same shift on Saturday and Sunday in each weekend.

3.3 A Branch-and-Bound enumeration scheme

Solving MILP problem (1)-(14) to optimality for realistic problem dimensions requires a lot of computation time, even with a state-of-the-art MILP Solver like CPLEX© 10. Recall that we have to solve this MILP for every combination of team sizes and number of weeks in each cycle. In order to be able to generate solutions in

acceptable computation times, our procedure will only spend time on the promising combinations of team sizes and number of weeks in each cycle using a branch-and-bound enumeration framework. Each node in the branch-and-bound tree represents a specific combination of team sizes and number of weeks in each cycle. Moreover, the MILPs in each node are not solved to optimality, due to a limit on the computation time. As a result, we obtain the heuristic branch-and-bound framework depicted in Algorithm 2. This algorithm is preceded by Algorithm 1. As a result, a good upper bound is available at the start of the search.

Algorithm 1 Initial UB search

```
{Search for good initial UB}
UB ← M;
sort all combinations;
while (UB = M) do
    take next combination;
    infeasibility_proven ← FALSE;
    while (infeasibility_proven = FALSE) AND (UB = M) do
        allowed_MILP_comp_time ← UB_time_limit;
        solve corresponding MILP;
        if (feasible solution found within time limit) then
            UB ← MILP_objective_value;
        else if (infeasibility not proven by MILP) then
            allowed_MILP_comp_time ← allowed_MILP_comp_time + UB_time_limit ;
        else
            infeasibility_proven ← TRUE;
        end if
    end while
end while
```

3.3.1 Initial upper bound

Algorithm 1 contains the pseudo code of the search for an initial upper bound. In this pseudo code, M is the so-called 'big M', that represents a very large number. The algorithm starts with sorting all combinations in order to find the 'most promising' combination to start with. The 'most promising' combination is the combination for which the occupancy level is closest to (and less than or equal to) 85%. The occupancy level is defined as the total number of paid work hours divided by the total number of workload hours, required by all flights. Note that 100% occupancy can never be achieved because this would mean that the planned capacity always exactly matches the demand for workload, which is simply impossible. The number of paid work hours is calculated by multiplying, for each cycle, the number of teams (= weeks) by the team size and by the number of working hours per week and adding

Algorithm 2 Heuristic branch-and-bound

```
{Heuristic branch-and-bound loop}
for all combinations do
    LB_of_this_combination ← -M;
end for
while (end criterion met = FALSE) do
    for all combinations do
        if (LB_of_this_combination < UB) then
            MILP_UB ← UB;
            solve corresponding MILP;
            if (feasible solution found within allowed_MILP_comp_time) then
                save solution;
                UB ← MILP_objective_value;
            end if
            if (MILP not solved to optimality within allowed_MILP_comp_time) then
                if (MILP_LB > LB_of_this_combination) then
                    LB_of_this_combination ← MILP_LB;
                end if
            else
                LB_of_this_combination ← M;
            end if
        end if
    end for
    {Detecting the overall LB}
    LB ←M;
    for all combinations do
        if (LB_of_this_combination < LB) then
            LB ← LB_of_this_combination;
        end if
    end for
    check end criterion;
end while
```

up over all cycles. The combinations are thus sorted in increasing order of difference from the preset 'ideal' occupancy level. Preliminary computational results indicated that an occupancy level of 85% could be achieved for realistic problem instances.

If the first ranked combination results in a feasible solution within a limited time limit, the objective value of this solution is our first upper bound. If, within the given time limit, no feasible solution is found, the time limit is extended with the same amount of time. This is done until either a feasible solution is found or infeasibility is proven. If infeasibility is proven, the next ranked combination having an occupancy smaller than or equal to 85% is considered until a feasible solution is detected.

3.3.2 Heuristic branch-and-bound search

Upon finding an initial upper bound, the algorithm starts with the branch-and-bound search (see Algorithm 2) which involves a loop through all combinations of team sizes and weeks in each cycle. The combinations are also explored in increasing order of difference from the ideal occupancy level, starting at the first unexplored combination after an upper bound has been detected. For each combination, a MILP is solved. In order to avoid spending MILP time on combinations which cannot result in a better solution, each MILP is provided with the current upper bound. Consequently, whenever the MILP lower bound (which continuously increases during the MILP search) becomes larger or equal to this UB cut-off value, the MILP calculation stops and the algorithm moves on to the next combination. Note also that because of this UB cut-off value, every time a MILP results in a feasible solution, this solution will automatically be the best solution found so far, and for this reason has to be saved. If we are interested in finding several near-optimal solutions, we only have to increase the UB cut-off value by a certain percentage, e.g. 5%. Since solving an individual MILP itself can be very time-consuming, a time limit has been set on each MILP. Note that the lower bound is saved only for those combinations that are not solved to optimality within the available computation time. In order to save computation time, the MILP solution status of these nodes is saved and loaded when the node is revisited later on in the search process. In all other cases (infeasible combination, optimal solution found, MILP stopped because of bound comparisons), the lower bound of the combination is set to M, excluding the node from further search. After all the nodes have been explored a first time, the overall lower bound is set to the minimum of all the nodes' lower bounds, and the end criterion is checked. The end criterion can be a computation limit or an optimality gap, e.g. 5%, defined as the relative difference between the best found solution (UB) and the overall LB, i.e. $(UB - LB)/UB$.

4 Computational results

4.1 Test set

The algorithm was tested using a set of 40 instances which have been generated based on real-life data from a major airline maintenance company active at Brussels airport. This company maintains about 300 flights weekly. The workload of these flights ranges from 30 minutes to 10 hours and the company was faced with peak arrivals around 7 AM and 5 PM. In our testing we also assumed a cycle time equal

to one week divided into 672 (= 4*24*7) periods of 15 minutes. The test set was generated using a random generator in which the demand for services differs with respect to three factors each having two factor settings. These factors are:

- The number of flights:
  1. 100 flights
  2. 300 flights
- The distribution of the flight workloads:
  1. The workloads are drawn from a uniform distribution between 0 and 10 hours.
  2. The workloads are drawn from an exponential distribution with average 3.5 hours.
- The flight arrivals:
  1. Peak arrivals: in this setting, 50% of the flight arrivals (STAs of the maintenance time windows) take place within either the morning (between 6 AM and 9 AM) or the afternoon peak (between 4 PM and 7 PM). The arrivals of the other flights are drawn from a uniform, random distribution.
  2. Uniform arrivals: all flight arrivals are drawn from a uniform, random distribution.

For each of the eight (=2*2*2) combinations, five instances have been generated, resulting in a set of 40 instances. These instances are named "Test_a_b_c_d", with a, b and c $\in \{1, 2\}$ the factor settings for the three factors in the same order as discussed above, and d $\in \{1, 2, 3, 4, 5\}$ the number of the instance. This test set is available for download on www.

In this test setting we opted to vary the demand side, while workforce regulations have been kept constant and in line with reality:

- The average number of work hours per week must be between 36 and 38 hours.
- For each employee, at least half of the weekends must be completely off.

- A morning shift must start between 5 AM and 6:30 AM. Only each 30-minute interval is considered to be a feasible starting time. Hence, for the morning shifts we have 5 AM, 5:30 AM, 6 AM and 6:30 AM as possible starting times. A day shift must start between 7 AM and 9 AM. An evening shift must start between 12:15 PM and 2:45 PM. A night shift must start between 9 PM and 10:30 PM. The total duration of a shift must either be 8, 8.5, 9 or 9.5 hours (including a half hour break). Combining all possible starting times with all possible durations for each shift type, we have in total 57 feasible shifts: 12 morning shifts, 15 day shifts, 18 evening shifts and 12 night shifts. These 57 shifts have a specific cost depending on the shift type, duration and starting time. The cost of a day shift during the week is set at 15 Euro per hour (gross cost). There is a shift premium for morning shifts (= 7% extra compared to the cost of a day shift), evening shifts (9%) and night shifts (20%). There is an extra weekend premium (on top of the shift premium): 16.67% for Saturday shifts and 95% for Sunday shifts. Finally, there is an extra premium for each night shift of 45 Euro (fixed), regardless of the shift length.

Finally, without loss of generalization, we only considered solutions having two workforce cycles, as this number allows for sufficient flexibility in developing the rosters. Preliminary computational experiences indicated that rosters having only one cycle are too inflexible to allow for a decrease in costs, while the marginal gain for rosters having three or more cycles is too small to warrant the dramatically increased computation time required. The reason why this test design only involves different demand settings is that the demand profile changes from season to season (and from company to company), while the supply side (workforce regulations) tends to be more constant over time.

4.2 Results

Table 1 lists the results of the computational experiment. The column 'Nodes' reports the number of nodes visited, i.e. the number of MILPs started within the given computation time. The column 'Improvements' indicates the number of times an improvement was found, while the column 'Cut-offs' indicates the number of times a node was pruned due to bound comparisons. Neither number includes the cut-offs and improvements found within the MILPs. Each MILP was provided with a computation time limit of 30 seconds, which is extended by 30 seconds every time the node is revisited. The end criterion (see 'check end criterion' in Algorithm 2) for each instance was set to 1 hour of computation time, provided that each node

has been evaluated at least once. This last condition explains why some computation times (e.g., instance Test_2_1_1_4) are far above 1h. The reason why all the combinations must be evaluated at least once is to be able to calculate and report an overall lower bound.

The results give rise to a number of conclusions and interesting insights. First, although none of the instances could be solved to optimality, the algorithm consequently succeeds in finding quality solutions. The gaps range from 0.66% to 6.51% with an average of 3.56%. These results indicate that an integrated approach to staffing and scheduling is possible from a computational point of view. Second, if we consider the gap as a measure of the problem difficulty, we can draw three conclusions corresponding to the three test design factors. Each time, the significance is tested using a T-test with equal variances assumed (confirmed by an F-test).

1. The number of flights has a significant impact (p-value: 0.025) on the problem difficulty: the average gap for problems involving 100 flights (factor setting 1) is 3.08% versus 4.04% for problems involving 300 flights (factor setting 2).

2. The distribution of the workload has no impact (p-value: 0.43) on the problem difficulty: the average gap for problems involving uniformly distributed workloads (factor setting 1) is 3.61% versus 3.51% for problems involving exponentially distributed workloads (factor setting 2).

3. The distribution of flight arrivals has a significant impact (p-value: 0.017) on the problem difficulty: the average gap of problems involving peak arrivals (factor setting 1) is 3.04% versus 4.08% for problems involving uniformly distributed arrivals (factor setting 2).

Third, concerning the impact of the factor settings on the cost (column 'Sol. value'), we observed that both the number of flights and the workload distributions have a significant impact (very small p-values). This is not a surprising result, as the total workload for 100 flights is much smaller than for 300 flights and the uniformly distributed workload has an average workload of 5 hours, while the exponentially distributed workload has an average workload of only 3.5 hours. One could ask why we didn't use an average of 5 hours for the exponentially distributed workload. The reason is that in that case there would be a fair chance that flights have a workload higher than 25 hours, which is not realistic.

Table 1: Computational results

	Sol. value (UB)	LB	Gap (%)	Nodes	Improvements	Cut-offs	Time (s)
Test_1_1_1_1	16,025.40	15,343.52	4.25	2,208	1	2,194	4,025.38
Test_1_1_1_2	20,604.74	20,277.78	1.59	2,017	3	2,006	3,638.89
Test_1_1_1_3	16,286.82	15,614.46	4.13	2,202	6	2,191	3,730.97
Test_1_1_1_4	16,427.25	16,000.96	2.60	2,162	3	2,148	3,771.89
Test_1_1_1_5	17,967.34	17,699.03	1.49	2,163	4	2,147	3,677.05
Test_1_1_2_1	13,304.74	12,922.32	2.87	2,238	2	2,221	3,768.17
Test_1_1_2_2	13,468.37	12,593.03	6.50	2,248	4	2,235	3,835.59
Test_1_1_2_3	20,965.44	20,365.86	2.86	2,022	3	2,010	3,911.25
Test_1_1_2_4	17,625.14	17,084.46	3.07	2,131	2	2,119	3,820.31
Test_1_1_2_5	16,747.22	16,194.63	3.30	2,170	2	2,157	3,849.60
Test_1_2_1_1	13,660.94	13,557.60	0.76	2,196	2	2,185	3,657.46
Test_1_2_1_2	16,337.67	16,229.98	0.66	2,126	5	2,116	3,842.51
Test_1_2_1_3	9,871.54	9,646.04	2.28	2,263	6	2,249	3,720.04
Test_1_2_1_4	11,631.43	11,272.72	3.08	2,257	5	2,243	3,717.18
Test_1_2_1_5	16,822.78	16,350.01	2.81	2,138	5	2,120	4,670.44
Test_1_2_2_1	9,922.02	9,294.42	6.33	2,260	4	2,246	3,748.76
Test_1_2_2_2	12,241.35	11,983.61	2.11	2,231	4	2,218	3,768.12
Test_1_2_2_3	12,495.54	12,088.22	3.26	2,229	3	2,217	3,853.41
Test_1_2_2_4	9,786.03	9,391.21	4.03	2,278	2	2,265	3,829.22
Test_1_2_2_5	9,939.31	9,573.15	3.68	2,278	2	2,264	3,788.43
Test_2_1_1_1	35,368.39	34,592.15	2.19	1,551	13	1,407	26,293.19
Test_2_1_1_2	34,615.57	33,456.17	3.35	1,551	24	1,311	38,601.75
Test_2_1_1_3	37,568.48	36,749.23	2.18	1,551	19	1,409	23,978.84
Test_2_1_1_4	35,181.45	33,755.59	4.05	1,511	22	1,300	35,780.37
Test_2_1_1_5	35,478.65	33,803.15	4.72	1,551	12	1,385	29,506.14
Test_2_1_2_1	34,408.12	32,554.69	5.39	1,610	10	1,494	20,368.88
Test_2_1_2_2	47,102.59	45,566.79	3.26	1,247	3	1,233	3,883.53
Test_2_1_2_3	34,357.41	32,122.12	6.51	1,551	14	1,350	34,027.81
Test_2_1_2_4	33,657.49	32,502.67	3.43	1,610	15	1,496	20,453.16
Test_2_1_2_5	34,283.47	32,772.07	4.41	1,551	11	1,394	27,854.49
Test_2_2_1_1	20,407.92	19,817.43	2.89	2,015	17	1,919	16,792.45
Test_2_2_1_2	19,924.38	19,149.55	3.89	2,059	12	1,984	14,586.63
Test_2_2_1_3	25,455.80	24,336.81	4.40	1,952	6	1,914	9,353.25
Test_2_2_1_4	21,724.74	20,811.46	4.20	2,059	2	2,017	9,355.23
Test_2_2_1_5	22,757.02	21,539.42	5.35	2,015	4	1,960	11,271.97
Test_2_2_2_1	18,080.64	16,992.41	6.02	2,106	5	2,050	10,285.25
Test_2_2_2_2	20,851.12	19,642.60	5.80	2,015	7	1,929	16,170.98
Test_2_2_2_3	25,238.39	24,934.57	1.20	1,958	2	1,940	4,685.85
Test_2_2_2_4	22,791.79	21,550.92	5.44	2,015	3	1,990	5,444.42
Test_2_2_2_5	24,462.98	23,959.07	2.06	1,964	3	1,941	5,263.76

What might be surprising, however, is that there is no significant difference between the peak arrivals and the uniformly distributed arrivals (p-value: 0.42). This could be explained by the presence of more evening and night flights in the uniformly distributed arrivals, which requires more expensive evening and night shifts that compensate for the higher cost of scheduling to peak capacity in the peak arrivals setting.

5 Conclusion and future research

This paper has described a heuristic branch-and-bound methodology for solving a workforce planning problem encountered at a major aircraft maintenance company. The model makes two important contributions. First, whereas the existing models presented in the literature either concentrate on the scheduling problem (and consider the staff levels to be given) or on the staffing problem (and ignore the scheduling problem, considering it to be a lower-level operational problem), our model deals simultaneously with both problem dimensions. Second, the model presented in this paper makes it possible to decide on the timing of the maintenance for each flights, in which this decision is bounded by given time windows. The approach was tested on a test set including 40 instances which were generated based on real-life data. The computational results indicate that this integrated approach succeeds in finding quality solutions within an acceptable computation time. Furthermore, the difficulty of the problem increases with a larger number of flights and a larger spread of flight arrivals. A third and maybe surprising result is that the occurrence of peak periods of flights has no impact on the total cost. The reason for this is probably that peak periods occur during the day (morning peak and afternoon peak) and hence can be covered by relatively cheap shifts.

This study introduces a lot of possibilities for future research. First, it would be very interesting to make our model more robust. We recognize three major sources of uncertainty: (1) uncertainty in supply (e.g., employee illness), (2) uncertainty in demand (e.g., unexpected aircraft problems leading to higher workloads) and (3) uncertainty in the timing of workloads (e.g., flight delays). Is it possible to incorporate this uncertainty into the model by adding 'robust constraints' or developing a stochastic programming model, or is a simulation approach the only feasible alternative? Second, the scheduling decision could be further detailed in the current model. For instance, our model decides on the number and type of shifts on each day for each cycle, but we would like to extend it by means of an automated proce-

dure in order to assign these shifts to the specific weeks within each cycle. Third, the model should be extended in order to allow for part-time workers and different skills. Fourth, the current model does not ensure that the maintenance of a particular flight is scheduled in a consecutive way. In other words, the solution might assign the first part of a flight's workload at the beginning of its time window and the remaining part towards the end of its time window (with a break in between). Obviously, this is not feasible in practice. Finally, this type of problem setting, in which the timing of the workload represents an extra decision dimension, is not specific to the aircraft maintenance industry. For instance, in nurse scheduling and project scheduling, the timing of the workload is also a controllable decision. The workload for nurses depends on the operating room and appointment scheduling, while the timing of the workload in projects depends on the scheduling of different activities. It would be very interesting to identify more workforce applications that might benefit from the approach described in this paper and try to develop a general solution framework.

Acknowledgements

The authors are extremely grateful to Willy Buysse and Thierry Van Damme (both Sabena Technics) for introducing them into this fascinating problem and providing them with input data, real-life constraints and useful feedback on the generated rosters.

References

Alfares H (1999) Aircraft maintenance workforce scheduling: a case study. Journal of Quality in Maintenance Engineering 5(2):78–88

Burke E, De Causmaecker P, Vanden Berghe G, Van Landeghem H (2004) The state of the art of nurse rostering. Journal of Scheduling 7(6):441–499

Chiaramonte MV, Chiaramonte LM (2008) An agent-based nurse rostering system under minimal staffing conditions. International Journal of Production Economics 114(2):697 – 713

Dijkstra MC, Kroon LG, van Nunen JAEE, Salomon M (1991) A DSS for capacity planning of aircraft maintenance personnel. International Journal of Production Economics 23(1-3):69 – 78

Gupta P, Bazargan M, McGrath R (2003) Simulation model for aircraft line maintenance planning. In: Proceedings of the Annual Reliability and Maintainability Symposium

Kilpi J, Töyli J, Vepsäläinen A (2009) Cooperative strategies for the availability service of repairable aircraft components. International Journal of Production Economics 117(2):360 – 370

Mercier A, Soumis F (2007) An integrated aircraft routing, crew scheduling and flight retiming model. Computers & Operations Research 34:2251–2265

Mundschenk M, Drexl A (2007) Workforce planning in the printing industry. International Journal of Production Research 45(20):4849–4872

Oğuz C, Sibel SF, Bilgintürk YZ (2010) Order acceptance and scheduling decisions in make-to-order systems. International Journal of Production Economics 125(1):200–211

Papadakos N (2009) Integrated airline scheduling. Computers & Operations Research 36:176–195

Sriram C, Haghani A (2003) An optimization model for aircraft maintenance scheduling and re-assignment. Transportation Research Part A 37:29–48

Thompson G (1997) Labor staffing and scheduling models for controlling service levels. Naval Research Logistics 44(8):719–740

Yan S, Yang TH, Chen HH (2004) Airline short-term maintenance manpower supply planning. Transportation Research Part A 38:615–642

Yang TH, Yan S, Chen HH (2003) An airline maintenance manpower planning model with flexible strategies. Journal of Air Transport Management 9:233–239