KU LEUVEN
HOGESCHOOL-UNIVERSITEIT BRUSSEL
FACULTEIT ECONOMIE EN BEDRIJFSWETENSCHAPPEN

A rolling horizon algorithm for an airline line maintenance scheduling problem

Jorne Van den Bergh
Philippe De Bruecker
Jeroen Beliën
Liesje De Boeck
Erik Demeulemeester

FEB@HUBRUSSEL RESEARCH PAPER
NOVEMBER 2013
Nr. 2013/11

FACULTEIT ECONOMIE EN BEDRIJFSWETENSCHAPPEN
CAMPUS BRUSSEL (HUBRUSSEL)
Warmoesberg 26
1000 BRUSSEL, België
tel. + 32 2 210 12 11

A rolling horizon algorithm for an airline line maintenance scheduling problem

Jorne Van den Bergh[a,b,1,2], Philippe De Bruecker[a,b], Jeroen Beliën[a,b], Liesje De Boeck[a,b], Erik Demeulemeester[b]

[a]*KU Leuven, Center for Information Management, Modelling and Simulation, Faculty of Economics and Business, campus Brussels (HUBrussel), Warmoesberg 26, 1000 Brussels, Belgium*
[b]*KU Leuven, Research Center for Operations Management, Faculty of Economics and Business*

Abstract

This paper studies the problem of scheduling malleable jobs without preemption while minimising the cost of delays and extra capacity. Jobs can either have the ascending or descending property, which means that while they are processed, the number of allocated resources are either non-decreasing or non-increasing, respectively. We propose a rolling horizon algorithm, and apply it to a line maintenance problem. Our results show that the planning process benefits from creating a robust baseline schedule.

Keywords: rolling horizon, scheduling, aircraft line maintenance, mathematical programming

1. Introduction

Many papers on scheduling problems focus on the midterm scheduling of jobs or personnel. One might think of creating a feasible production schedule at minimum cost, developing personnel rosters while trying to satisfy as many personnel requests as possible, or scheduling jobs while minimising the makespan or the total weighted tardiness. Mostly, the problem instances are solved offline, and the algorithm has complete access to the whole instance input. In reality, however, the input for these problems is stochastic and the solution algorithm learns about the input piece by piece, reacting to the new requests with only a partial knowledge of the input (Pruhs et al. [1]). Examples are treating emergency patients in hospital applications, absenteeism in personnel scheduling, etc. By incorporating a rolling horizon to partially account for stochasticity and a dynamic environment, we better succeed in capturing reality in the scheduling process.

Email addresses: jorne.vandenbergh@kuleuven.be (Jorne Van den Bergh), philippe.debruecker@kuleuven.be (Philippe De Bruecker), jeroen.belien@kuleuven.be (Jeroen Beliën), liesje.deboeck@kuleuven.be (Liesje De Boeck), erik.demeulemeester@kuleuven.be (Erik Demeulemeester)

[1]Corresponding author

[2]TEL +32 2 2101611 FAX +32 16 326624

This research was motivated by a visit to the line maintenance department of an airline maintenance company at Brussels airport. For this department, one operator is in charge of allocating technicians to the flights that need maintenance. By considering the available technicians, the shift sequences that they are following, whether they did or did not have their lunch break yet, etc., this operator tries to link the technicians with the list of flights that need maintenance. This linking is done manually and depending on the experience of the operator and the stochastic behavior of the flight arrivals, the schedule has to be adjusted frequently in the peak periods. However, the stochasticity of the flight arrivals, the constraints imposed on the scheduling of the maintenance jobs, and the creation of the schedule by hand create a far from optimal solution. Moreover, because of the stochasticity of the flight arrivals, the work rosters that are provided may not be suitable to create a feasible schedule for all the flights. Therefore, we present a rolling horizon procedure that can be used for solving various scheduling problems. The basic situation is the following. Each job is characterized by a demand, a release time, a (soft) due time and a (hard) deadline. The release time of a job is stochastic and is updated over the time horizon, taking into account new information. The number of available resources fluctuates over the time horizon and can be increased within certain limits and at a given cost in the rolling horizon procedure. The goal is to produce a feasible schedule, which means that each job is assigned to one or more resources and one or more time slots, while minimising the delay and capacity costs.

In Section 2 we discuss the literature on rolling horizon procedures and we address some papers with a similar problem setting. The nomenclature and mathematical programming formulation are presented in Section 3. This is followed by the description of the rolling horizon algorithm in Section 4. A case study on instances inspired by a scheduling problem in a line maintenance company is addressed in Section 5, followed by the discussion of the results in Section 6. Finally, some conclusions are drawn in Section 7.

2. Literature review

The majority of the literature concerning rolling horizon decision making deals with production planning problems. In this context, the production schedule aims at minimising the cost per period, considering for instance the cost of production, inventory, backlogging or shortage, etc. The demand may either be deterministic or stochastic. Sethi and Sorger [2] develop a theoretical framework for rolling horizon decision making, considering that forecasting the future is a costly activity. To this end, they present a dynamic programming formulation for a general, discrete-time, stochastic dynamic optimization problem in which the decision maker has the possibility to obtain information on the uncertain future at a given cost. The motivation for incorporating this cost lies in the difficulty to retrieve accurate forecasts. The forecast of the future is either expensive, unreliable or both and gets more complicated the more distant the future. As'ad and Demirli [3] develop optimal master production schedules in a steel rolling mill operating in a dynamic environment. To capture this dynamic nature, the models are implemented on a rolling horizon basis. A one product, uncapacitated master production scheduling problem in which decisions are made under rolling planning horizons is studied by Vargas and Metters [4]. Hereby, they can explicitly consider the stochastic nature of the demand. This demand is time-varying and

the effectiveness of their approach is measured by inventory holding, production setup and backorder costs.

In the personnel scheduling literature, rolling horizon procedures are scarce. Gronalt and Hartl [5] minimise the total wage costs for the assembly of mid-volume trucks, given a desired production rate. They develop a solution approach for the assignment of operations to stations, searching for a loading sequence for the products, and for a worker and floater time allocation. Bard and Purnomo [6] use the technique to cope with emergencies, call-outs and normal fluctuations in personnel requirements in hospitals. The problem associated with making the daily adjustments has to take into account the individual preferences of the nurses. Bard and Purnomo formulate the problem as an Integer Program (IP), that they solve with a branch-and-price algorithm. This solution method enables them to solve problem instances with up to 200 nurses within 10 minutes, considering 24 hours at a time. Campbell [7] shows that for single-period scenarios, on-call overtime can slightly reduce costs. Therefore, he discusses the possibility of using multiperiod models within a rolling horizon framework that can benefit from forecast updating, which is the topic of this paper. Other applications can be found in disruption management (e.g., Nielsen et al. [8]) or inventory routing problems (e.g., liquefied natural gas annual delivery program by Rakke et al. [9] or oil transportation by Shen et al. [10]).

In this work we consider the problem of scheduling malleable jobs. These are jobs that may be processed simultaneously by several processors, where the number of processors used influences the processing speed of a job. Two extreme cases exist: in the first case, the job processing time is independent of the number of processors assigned to the job, while in the second case the processing time is exactly inversely proportional to the number of processors assigned to that job (Burke et al. [11]). An overview on the scheduling of multiprocessor tasks (i.e., multiprocessor or parallel tasks include both malleable and non-malleable tasks, which require a specific number of processors for specific units of time) can be found in Drozdowski [12]. Sadykov [13] presents a dominant class of schedules for malleable jobs if the goal is to minimise the total weighted completion time. He introduces the ascending property, which states that the number of processors assigned to a job does not decrease over time while this job is being processed. In order to find an optimal ascending schedule, however, the sequence of job completion times must be determined. Therefore, an alternative to reducing the search space by taking advantage of the class of dominant schedules in solving the problem, can be to enumerate the sequences of jobs without using the ascending property. In our problem, we expand this to the ascending and descending property. This means that while jobs are being processed, the number of allocated resources can only increase or decrease, respectively. We call this the monotonicity property. Note that this monotonicity property implies non-preemptive execution of jobs. Blazewicz et al. [14] deal with the problem of scheduling a set of n independent non-preemptive malleable tasks on an m processors system, starting from a continuous version of the problem. The criterion they assume is schedule length. In Blazewicz et al. [15], they also schedule multiprocessor tasks on parallel processors with limited availability. For some specific cases, polynomial time algorithms are given. This is somewhat related to the problem setting in this paper, except for the preemptive characteristic of the jobs and the minimisation of the makespan.

Hu [16, 17] tries to minimise the total tardiness in the model of an identical parallel-machine version with non-preemptive jobs of the worker assignment scheduling problem. The worker assignment scheduling problem in this model is shown to be NP-complete. Therefore, he developed some heuristics to solve the problem in two phases: job scheduling and worker assignment.

Our decision problem can also be related with another branch in the literature, namely capacity management or more specificly rough-cut capacity planning (RCCP). In RCCP, the aim is to compare the available capacity with the resource profile, retrieved from a tentative scheduling of the jobs. It allows to evaluate different scenarios, such as the usage of non-regular resource capacity such as overtime work and outsourcing, and the determination of due dates and other project milestones. Hans [18] distinguishes between two problems in RCCP: resource driven and time driven RCCP. In resource driven RCCP, all (non-)regular capacity levels are fixed and one tries to minimise the maximum lateness of the projects, preferably using regular capacity. In the case of time driven RCCP, each job in a project is characterized by a deadline, which the company wants to meet while minimising the usage of nonregular capacity. In his thesis, Hans [18] develops some branch-and-price algorithms to solve the RCCP with precedence constraints and compares these methods with approximation techniques such as rounding heuristics and an improvement heuristic. Only time driven RCCPs are tested and as a result, the projects are not allowed to be tardy. In our paper, we aggregate the time driven and the resource driven RCCP in one model and evaluate different scenarios by changing the cost parameters for both nonregular capacity and tardy jobs.

The contribution of our model lies in the combination of scheduling malleable tasks with a soft due time, while respecting the monotonicity property of the tasks. Moreover, we apply this model to a stochastic problem setting, where the timing of the workload is uncertain. Therefore, we combine a proactive and reactive scheduling policy into a rolling horizon framework.

3. Problem setting

In this section, we will present the nomenclature, the problem setting and the mathematical programming formulation.

Nomenclature

Indices and sets

j index of jobs ($j = 1,...,n$)

t index of time periods ($t = 1,...,T$)

Decision variables

δ_j $= 1$ (0) if job j has the ascending (descending) property

x_{jt} $= 1$ if job j is active in period t, 0 otherwise

χ_{jt} $= 1$ if period t is the last active period of job j, 0 otherwise

y_{jt} number of resources performing job j in period t ($y_{jt} \in \mathbb{N}_0$)

o_t number of extra resources in period t ($o_t \in \mathbb{N}_0$)

Coefficients and parameters

d_j	due time of job j		sources assigned to job j when it is active
$\bar{d}_j$	deadline of job j		
r_j	release time of job j	W_j	workload of job j (in resource periods)
L_j	lower bound on the number of resources assigned to job j when it is active	C_j^d	delay cost per period for job j
		C^o	cost per period for extra capacity
U_j	upper bound on the number of re-	μ_t	available capacity in period t

The problem under consideration consists of n malleable jobs, that need to be processed without preemption on a set of resources, of which the capacity is time-varying (μ_t). We assume the resources to be identical. The release time r_j of a job is the first period in which a job can be processed and is stochastic. Each job is characterized by a (soft) due time d_j and a (hard) deadline $\bar{d}_j$. Completion of a job after its due time is allowed, but it will incur a cost C_j^d per period of tardiness. If the ascending (descending) property is assigned to a job, this means that the number of processors assigned to that job does not decrease (increase) over time while this job is being processed. The processing time of a job j is exactly inversely proportional to the number of resources assigned to it (y_{jt}), which is bounded by a lower bound L_j and an upper bound U_j. At a certain limit, the processing time of the jobs can no longer be decreased by assigning more resources. This is the upper bound U_j. Because of the stochasticity of the release times and the need to complete all the jobs, the available capacity μ_t may not be sufficient. To this end, nonregular resource capacity o_t is available at a cost C^o (e.g., outsourcing, extra resources, etc.).

Note that the intensity or the number of resources y_{jt} doing a job j in period t, is required to be integer. By relaxing this requirement, non-preemption is no longer a valid assumption. Consider, for instance, four periods $1, 2, 3$ and 4. The corresponding y_{jt}-values are $0.6, 1.5, 1.8, 2.25$. These values satisfy the monotonicity constraints by following a non-decreasing pattern and, at first sight, also ensure non-preemption. However, from the resources' point-of-view, this pattern could mean that one resource (A) starts executing job j in period 1 when 40% of the period has passed. In the second period, at least 2 resources are needed to cover 1.5 resource periods. Resource A needs to keep on working in the second period (non-preemption constraint) and an additional resource B is called, who starts at the half of the period, then a problem arises in period 3. Resources A and B are present now, but they cannot both work a complete period, since only 1.8 resource periods are needed. Therefore, this execution pattern will lead to preemption in the third period for at least one of the resources, since in the fourth period, again more than 2 resources are necessary. Of course, this behavior could be avoided by a new formulation where the float intensities need to increase (resp. decrease) to at least the next (previous) integer value (i.e., minimally round up or resp. round down the current float value). Considering the previous example, this new formulation would demand for the intensity in period 3 to be 2 or higher (intensities can still be float values). Then, resource B can remain active for the entire period and if the new value is again a float, this represents a third resource C, starting at a given time in the period, leading to an adjustment of the intensity in period 4. This formulation, however, has the drawback that it is no longer possible to aggregate the intensities of all jobs in a given period to check whether the available capacity is not exceeded by the resource consumption, since the corresponding resource usage is not uniformly spread over the period. For these

two reasons, we will keep the integer requirement of the variables y_{jt}.

The integer programming model for minimising the cost of scheduling malleable non-preemptive jobs is as follows:

$$\text{minimise} \qquad \sum_{j=1}^{n} C_j^d \left(\sum_{t=d_j+1}^{\bar{d}_j} (t - d_j)\chi_{jt} \right) + \sum_{t=1}^{T} C^o o_t \tag{1}$$

subject to

$$\sum_{t=r_j}^{\bar{d}_j} y_{jt} \geq W_j \qquad j = 1, \ldots, n \tag{2}$$

$$L_j x_{jt} \leq y_{jt} \leq U_j x_{jt} \qquad j = 1, \ldots, n \quad t = r_j, \ldots, \bar{d}_j \tag{3}$$

$$y_{j,t-1}/W_j \leq y_{jt}/W_j + (1 - \delta_j) + (1 - x_{jt}) \qquad j = 1, \ldots, n \quad t = r_j + 1, \ldots, \bar{d}_j \tag{4}$$

$$y_{jt}/W_j \leq y_{j,t-1}/W_j + \delta_j + (1 - x_{j,t-1}) \qquad j = 1, \ldots, n \quad t = r_j + 1, \ldots, \bar{d}_j \tag{5}$$

$$x_{jt} - x_{j,t+1} \leq \chi_{jt} \qquad j = 1, \ldots, n \quad t = r_j, \ldots, \bar{d}_j \tag{6}$$

$$\sum_{j \in J} y_{jt} \leq \mu_t + o_t \qquad t = 1, \ldots, T \tag{7}$$

$$x_{jt}, \delta_{jt}, \chi_{jt} \in \{0, 1\} \tag{8}$$

$$y_{jt}, o_{jt} \in \mathbb{N}_0 \tag{9}$$

The objective function (1) minimises the total costs, which consist of the delay costs and the cost of extra capacity. Constraint set (2) guarantees that each job is completely maintained. Constraint set (3) makes sure that the number of resources assigned to a job lies within the specified interval $[L_j, U_j]$. Constraint sets (4) and (5) are the monotonicity constraints. They imply that, if δ_j equals 1 (0), the number of resources assigned to a certain job j, can only increase (decrease) in successive periods. Constraint set (6) fixes the χ_{jt}-variables, which indicate the last period in which processing takes place. Finally, constraint set (7) ensures that in every time period, the number of allocated resources does not outnumber the available capacity (i.e., regular and nonregular capacity). Depending on the problem under consideration, the nonregular capacity has multiple sources such as, overtime, outsourcing, inhouse resources from other departments, etc.

Proposition 1. *The problem is strongly NP-hard.*

Proof: If we consider the instances for which $L_j = U_j = 1/W_j$ and enough resources are available ($\mu_t = M$ and $o_t = 0$ $(t = 1, \ldots, T)$), the resulting problem of minimizing total tardiness is equivalent to the parallel machine scheduling problem with ready times and without preemption. According to the three-field classification system (Graham et al. [19]), this problem is denoted by P $\mid r_j \mid \sum T_j$ and is proven to be NP-hard in the strong sense (Garey and Johnson [20], Blazewicz [21]). □

Test results based on real-life instances show that IP (1)-(9) is very difficult to solve, even for small instances (e.g., short planning periods with only a limited number of flights

to schedule). Relying only on this formulation in the rolling horizon approach leads to very high computation times. Hereby, one of the main advantages of a rolling horizon approach, i.e., to quickly reschedule a list of jobs based on updated information fades out. Therefore, we first create a baseline schedule, which will be adjusted in the rolling horizon approach, and try to tackle the rescheduling heuristically. If the heuristic fails to create a feasible schedule, IP (1)-(9) will be invoked.

4. Methodology

Our approach starts with an initialization phase, prior to the rolling horizon algorithm. Figure 1 presents the steps in the initialization phase. The first step is to transform the job characteristics (specified in Section 5.1) according to the uncertainty distributions. These distributions can be based on historical information or assumptions on the stochasticity. The newly generated list of jobs will be used as input to create a personnel schedule. The idea is that the personnel schedule(s) generated in this step tend(s) to be more robust than those that are based on the original list of jobs. In the final step of the initialization phase, the original list of jobs and the generated work schedule are used to develop a robust baseline schedule for the rolling horizon algorithm. In the baseline schedule, jobs are scheduled as late as possible, as if it were a worst case scenario. Hereby, we prevent (small) delays from disrupting the schedule, since the test results in Section 3 reveal that rescheduling is computationally expensive. After the initialization phase, the rolling horizon algorithm is executed (Figure 2). This algorithm gradually moves along the time horizon, each time adapting the baseline schedule to the most recent information. This information is used to check whether a job will be available to schedule in the current planning horizon or not. In each step of the algorithm, the available jobs are sorted by earliest release time. If jobs can be scheduled earlier than the execution start time according to the baseline schedule, they are leftshifted heuristically. Hereby, they generate extra capacity in later periods that can be used for jobs, for which the delay harms the baseline schedule. The initialization phase can thus be seen as a proactive approach to cope with uncertainty in the timing of the workload, whereas the rolling horizon algorithm is the reactive approach. Some of the concepts applied here are related to scheduling strategies in robust project scheduling. We refer the reader to Demeulemeester and Herroelen [22] for more information on this topic.

5. Case study

As already mentioned in the introduction, the idea for this research originates from an aircraft line maintenance scheduling problem. In this case study, the workload is determined by a cyclic weekly list of flights that need maintenance. Each flight is characterized by a scheduled time of arrival (STA), a scheduled time of departure (STD) and an estimated workload, which correspond with r_j, d_j and W_j, respectively. The interval $[r_j, d_j]$ is the time window of job j. The arrival time of a flight is stochastic, which is modeled by adjusting the STA to the real time of arrival, i.e., r_j^*. The STD is the latest period to complete the maintenance of a flight in order to avoid delay costs. We allow for maintenance being scheduled beyond this due time to guarantee that all flights can be processed completely. For the same

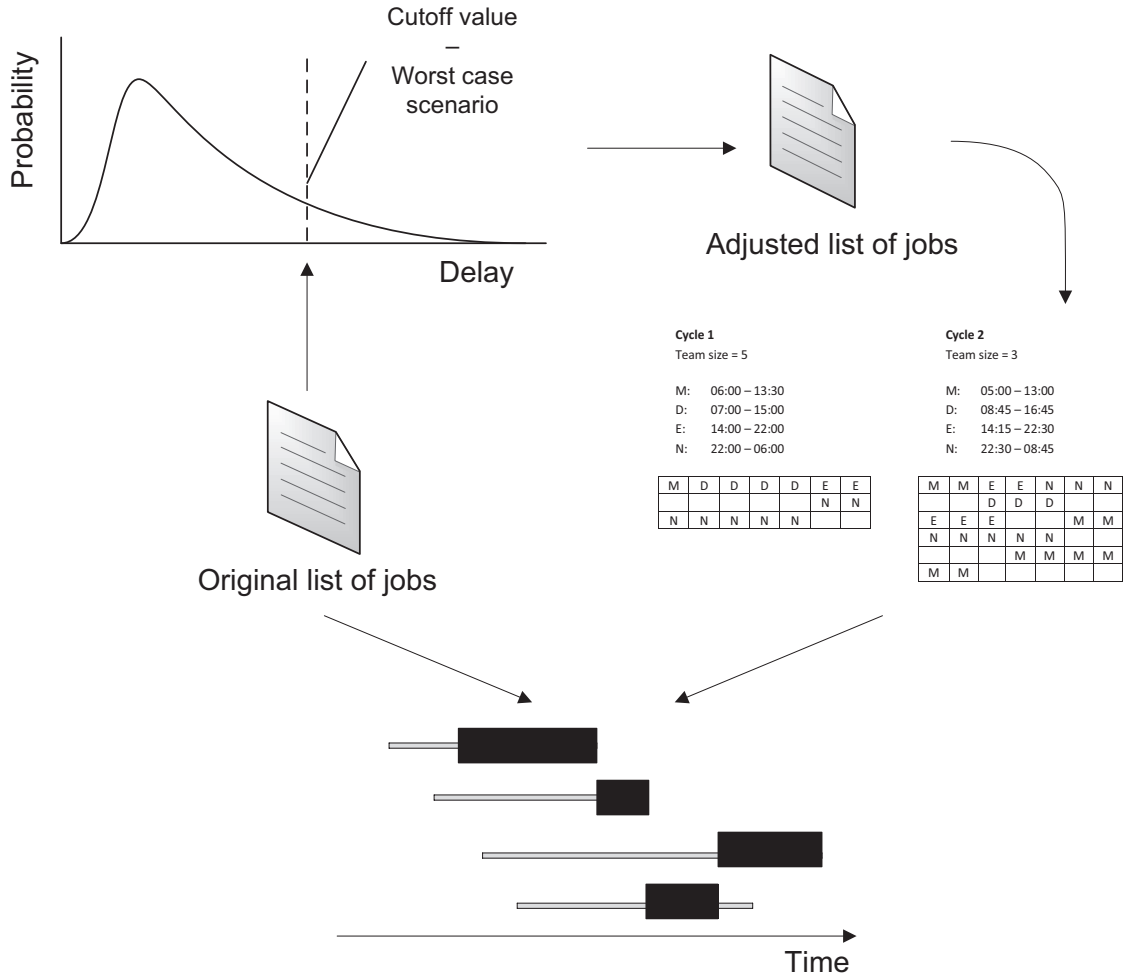

Figure 1: Illustration of the initialization phase: 1) data transformation, 2) creation of the personnel schedule and 3) creation of the baseline schedule.

reason, the deadline $\bar{d}_j$ of a flight j is set high enough (see Section 5.2). The time horizon is discretized into quarters of an hour, i.e., one period equals 15 minutes.

5.1 Algorithm

Step 1: Data transformation. Instead of relying on the original list of jobs to create the initial baseline schedule, we opt to transform this list by taking into account uncertainty distributions. As mentioned earlier, we only take into account stochasticity in the release times of the jobs. Assuming that the uncertainty distributions are known upfront, this information can be beneficial for the solution approach. In our case study the delay distributions generated in Van den Bergh et al. [23] are used. The aggregation of these distributions reveals that 80% of the flights arrive early or within at most 3 quarters of an hour after the scheduled time of arrival (STA) and 95% arrives within 6 quarters of an hour after the STA. If this information would be neglected and the personnel schedule is created based on

the original list of (deterministic) jobs without incorporating robustness constraints, flight delays will definitely influence the baseline schedule in real-life operations. Therefore, the time window and/or workload of the initial list of jobs is adjusted in this step. Note that the time window of a job is the time interval between two consecutive flights for a given airplane. Jobs that have a wide time window will not be affected by (most of) the delays. The starting time of the time window of these jobs, that is used as input for building the personnel schedule, will be rightshifted such that it covers a certain percentage of the arrivals delays, in our case 95%. Most of our interest goes to jobs with narrow time windows, for instance less than two hours. If no additional measures are taken, the personnel schedule only provides capacity during the original time window. Large delays can occur that hinder these jobs from (partial or complete) execution within their time window. Therefore, the time window for these jobs, that is used as input for building the personnel schedule, is not shifted, but prolonged. The reason for prolonging the time window is clear: if no capacity is available after the original time window, this could lead to very high tardiness values. On the other hand, the starting time of the (narrow) time window of these jobs cannot be shifted. If so, these jobs could possibly not be executed when the corresponding flights arrive on time (or with a short delay), since the scheduled capacity would be situated at the end of, and beyond the original time window. Therefore, the workload of these jobs is increased to create a personnel schedules that provides more capacity for these jobs. In Algorithm 1 the details of the modification are given. The release time, due date and workload followed by a 'a' represent the adjusted variables of the adjusted list of jobs. The first IF-statement is used when the 'worst-case delay' of 6 periods (referred to as 'delay' in the rest of the paragraph) prevents a job from being scheduled within its original time window (i.e., the time window of that job is smaller than or equal to 6 periods). The second IF-statement indicates that the delay results in an increase in workload, making it impossible to cover the demand with the maximum number of workers. In the third IF-statement the delay implies that on average two extra workers per period are needed to cover the job compared to the original case. In all the previous cases, the due date and workload are adjusted. If these conditions are not fulfilled, one can start the job after the delay without needing more capacity, time or both. This is the last case in Algorithm 1.

Algorithm 1 Modification of jobs

for all jobs $j \in J$ **do**
  old_interval $= d_j - r_j + 1$;
  new_interval $= d_j - (r_j + 6) + 1$;
  if $new\_interval \leq 0$ **then**
    $r_j^a = r_j, d_j^a = d_j + old\_interval, new\_interval = d_j^a - r_j^a + 1, W_j^a = (W_j / old\_interval) * new\_interval$;
  else if $W_j / new\_interval \geq U_j$ **then**
    $r_j^a = r_j, d_j^a = r_j + 6 + \lceil W_j / U_j \rceil, new\_interval = d_j^a - r_j^a + 1, W_j^a = (W_j / old\_interval) * new\_interval$;
  else if $W_j / new\_interval \geq W_j / old\_interval + 2$ **then**
    $r_j^a = r_j, d_j^a = r_j + 6 + \lceil (W_j / (W_j / old\_interval + 1)) \rceil, new\_interval = d_j^a - r_j^a + 1, W_j^a = (W_j / old\_interval) * new\_interval$;
  else
    $r_j^a = r_j + 6, d_j^a = d_j, W_j^a = W_j$;
  end if
end for

Step 2: Creation of the personnel schedule. For the second step, we use the heuristic procedure by Beliën et al. [24] to create personnel schedules. The transformed list of jobs is used as input and the created personnel schedule is thus capable to cope with the majority of the delays, assuming the delay distribution of the first step. The problem of deciding on the number of cycles, team sizes, shift durations and the shift sequence is tackled heuristically by an enumeration algorithm that is based on the team sizes and the number of cycles. Some constraints in the formulation of Beliën et al. [24] can be seen as robustness measures. One of the constraints, for instance, prohibits jobs from execution in the first or last quarter of the time window. Another constraint ensures that a capacity buffer is provided to cope with unexpected workload.

Step 3: Creation of the baseline schedule. In the third step, the original list of jobs, the personnel roster retrieved in step 2 and the IP-model (1)-(9) are used to create the robust baseline schedule. Since this baseline schedule is a tactical schedule, no nonregular capacity can be used and all the jobs need to be finished before their due date d_j. Hence, the o_t-variables are removed from constraint set (7) and the objective (1) is replaced by (10):

$$\text{minimise} \qquad \sum_{j=1}^{n} \left(\sum_{t=r_j}^{d_j} P_{jt} y_{jt} \right) \tag{10}$$

In this new formulation, the objective function is penalized by a penalty P_{jt} for scheduling jobs in the beginning of their time window. Different penalty functions can be used to take into account the delay distributions, i.e., scheduling a job during the first X periods is penalized harder than in the next X periods, etc. This step is very time consuming, but since we assume a cyclic list of jobs and a personnel schedule that is operational for several months, this does not harm the effectiveness of our approach.

Step 4: Rolling horizon algorithm. As mentioned earlier, the previous steps can be seen as the initialization phase or proactive approach to construct a robust schedule that is based on historical information. In the fourth and final step, real-life information will be considered within a rolling horizon framework. The rolling horizon procedure gradually moves along the time horizon. A 'planning procedure' has a certain time window, called 'procedure length', which is defined by a start and an end time. Two consecutive planning procedures (or iterations) are separated by a 'planning interval', which is the time between the start times of planning procedure i and planning procedure $i + 1$. New iterations are performed until a certain end criterion is met, which can be a maximum number of iterations or a certain timing of the time horizon, or a computation time limit. In each step of the rolling horizon framework, the following information is available:

- The availability (i.e., real release time) of the jobs with a scheduled release time between the start and end of the current planning procedure and of the jobs with a scheduled release time after but with a real release time before the end of the current planning procedure.

- The current schedule of the jobs, which is the baseline schedule for the newly released

jobs or the adjusted schedule if these newly released jobs have been rescheduled in the previous planning procedure, but have not yet been finished before the starting time of the current procedure.

- The capacity availability of the resources during the current planning procedure.

Each planning procedure starts with an update of the jobs, based on the newly generated information. Jobs that have a scheduled release time within the time window of the planning procedure could for instance be delayed, resulting in a new release time after the end time of the procedure length. The opposite can also be true: jobs that originally had a release time greater than the end time of the current planning procedure, arrived early and can be planned in the current planning procedure. Of course, in the last scenarios, the actual release time differs positively or negatively from the scheduled release time, but the job still needs to be scheduled in the current planning procedure, since its time window overlaps with the time window of the current planning procedure.

An example of the rolling horizon algorithm is presented in Figure 2. Using the start and end period in Figure 2, jobs 5 to 7 will definitely be addressed in the current planning procedure i. The release time for job 8 lies beyond the end of the planning horizon, hence, it is eliminated from the set. Job 1 will also be excluded, as its deadline lies before the start time of the current planning horizon, and therefore, has already been completed. Jobs 2, 3 and 4 have a release time which is smaller than the current start period. Since their time windows overlap with the current planning procedure, one has to check whether they have been completely processed in the previous planning procedure. Since the workload for job 2 and 4 was only partly covered in the previous planning procedure, it has to be processed from the start period until it is finished because of the non-preemptive constraint. Not only the non-preemption is important, also the value of δ_j has to be passed from the previous planning procedure to the current one, ensuring the ascending or descending property of job j. Job 4 was already assigned the ascending property. In the current planning horizon, δ_4 remains 1 and the minimum number of resources to be assigned to job 4 equals y_{4t}, where t is the period just before the start period of the current planning horizon. Full flexibility remains in the current planning procedure with respect to the monotonicity property for job 2. However, the value of y_{2t} in the last period of the previous planning period needs to be saved. Because of the non-preemption constraint, this value has an influence on δ_2. No workload has been covered for job 3 in the previous planning procedure, so full flexibility remains with respect to the monotonicity and the start time of processing.
Considering the next start period in Figure 2, jobs 2, 4 and 5 definitely join job 1 in the set of finished jobs, while jobs 3, 6 and 7 can have the partly or completely processed status or might not be started yet. Job 8 will be added to the set of jobs that need to be scheduled.

In the rolling horizon algorithm (Algorithm 2), a greedy heuristic is used to schedule the newly released jobs. This greedy heuristic tries to schedule a job as close as possible to the release time (i.e., a leftshifted schedule). Per job, it enumerates all the possibilities for non-decreasing and non-increasing monotonocity, considering the remaining capacity in these periods. Of course, this approach does not guarantee the creation of an optimal schedule. Multiple jobs can interfere with each other because of overlapping time windows and limited

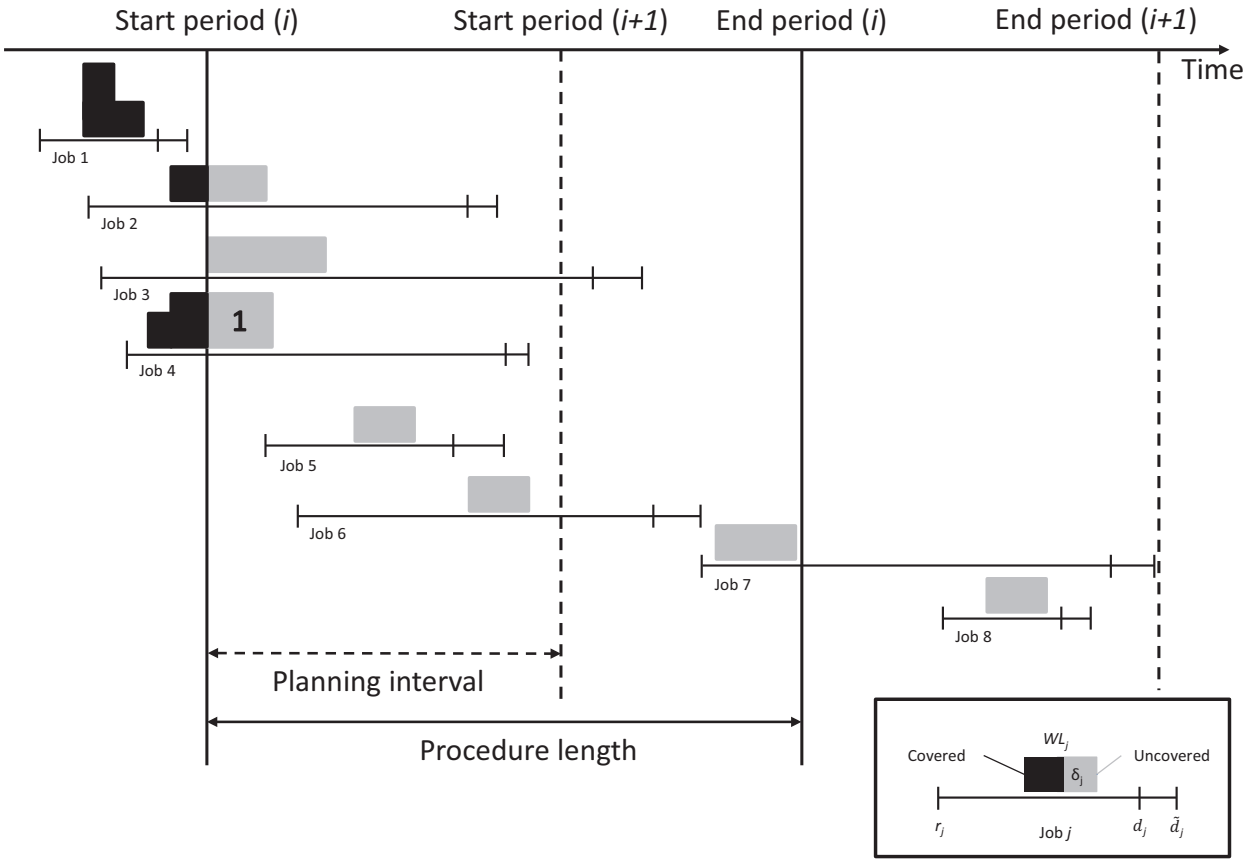

Figure 2: Illustration of the rolling horizon procedure: each job is defined by its workload (divided into covered and uncovered workload), release time, due time, deadline and delta. The value of delta is only shown if the job already possesses the ascending or descending property.

capacity. However, the approach is quite effective to transform the robust base schedule into a leftshifted schedule. The jobs are sorted by earliest (real) release time and by this approach we try to generate as much available capacity as possible in later periods. Of course, the greedy approach does not (always) give the optimal solution. Consider, for instance, two jobs, with $r_1 < r_2$ and $d_1 > d_2$. The greedy heuristic will schedule job 1 before job 2, whereas this could lead to capacity problems. The main reason behind this approach is that this capacity can be used by delayed jobs. In the worst case, when a job was not delayed (or arrived early), it cannot be leftshifted by the heuristic. Thus, the baseline schedule for this job becomes the actual schedule. Delayed jobs, however, have to be rescheduled if the actual release time is later than the start of the execution time according to the baseline schedule. In this case, a part or even the total workload of that job needs to be (re)scheduled. The greedy heuristic, however, might fail to provide a feasible solution since it does not allow nonregular capacity. In that case we rely on the IP-formulation (1)-(9). All jobs that are available in the current procedure (i.e., all jobs of which the available time window partly overlaps that of the current planning procedure) are then rescheduled in order to obtain a

feasible schedule while minimizing the use of nonregular capacity.

Algorithm 2 Rolling horizon algorithm

```
INITIALIZATION;
Set S = { };
while (end criterion met = FALSE) do
    for all jobs j ∈ J do
        update r*_j ← r_j+delay;
        update d*_j and d̄*_j;
        update schedule and available capacity;
        add job to or remove it from active set S;
    end for
    sort all jobs of S by earliest r*_j;
    MAIN STEP
    still_feasible ← TRUE;
    j ← 1;
    while (j ≤ |S| AND (still_feasible)) do
        try greedy heuristic for job j;
        if not all workload covered then
            solve IP (1)-(9);
            still_feasible ← FALSE;
        end if
        update schedule and available capacity;
    end while
    save solution;
    check end criterion;
    renew start_period, end_period, next_start_period;
end while
```

In Algorithm 2, the complete procedure is given. The variables r_j^*, d_j^* and $\bar{d}_j^*$ represent, respectively, the real release time, the real due date and the real deadline of job j (instead of the scheduled times).

Algorithm 3 shows how the due dates of the jobs are adjusted according to the simulated delays and shows similarities to Algorithm 1. If the job cannot be completed within its new interval or it needs at least one extra worker per period (on average) to complete the job before the due date, then the due date, and correspondingly the time window, of that job will be adjusted (first IF-statement). This due date is also extended when the number of workers needed to execute the job before the due date exceeds the maximum number of workers allowed to process that job. If these two conditions are not fulfilled, the original due date remains.

5.2 Parameters

Random instances. To get an insight into the effects of the different parameters, a test set of 20 instances in total is created with a random generator, based on real-life data from the company. These instances differ in two factor settings, listed in Table 1. For each combination, 5 instances are created. They are called 'a_b_c', where the first two parameters represent the factor settings and the last one $c \in \{1, 2, 3, 4, 5\}$, the instance number. The distribution of the workload depends on the duration of the time window and the maximum

Algorithm 3 Incorporation of delays

for all jobs $j \in J$ **do**
  old_interval $= d_j - r_j + 1$;
  new_interval $= d_j - r_j^* + 1$;
  if $new\_interval \leq 0$ **or** $W_j/new\_interval \geq W_j/old\_interval + 1$ **then**
    $d_j^* = r_j^* + old\_interval - 1$;
  else if $W_j/new\_interval \geq U_j$ **then**
    $d_j^* = r_j^* + \lceil W_j/(U_j - 1) \rceil - 1$;
  else
    $d_j^* = d_j$;
  end if
end for

number of workers allowed to be allocated to a flight. The parameters of the uniform distribution are 0 and the minimum of, on the one hand, a workload of 10 hours and, on the other hand, the maximum workload that can be covered by 4 workers in the time window of the flight. The generated value (r) of the triangular distribution (0.125; 0.375; 1) represents the average number of workers needed per period to maintain the flight before its STD ($W_j = r * (STD - STA + 1)$). This value r is not limited by a maximum workload (as in the instances with uniform distribution), but by a maximum average value of workers/period of 1. In the instances with peak arrivals, there are more arrivals during the morning and evening hours, whereas in the instances with spread arrivals, the arrivals are uniformly distributed over the day.

Table 1: Factor settings

	1	2
a) Distribution of workloads (hours)	Uniform	Triangular
b) Flight arrivals	Peak arrivals	Spread arrivals

Overtime and extra capacity. As stated in Section 3, the nonregular capacity can have multiple sources. In our case study, extra capacity can be retrieved by allowing workers to do overtime or by hiring workers either from another maintenance division within the company or from a partner company.
The cost of overtime depends on the shift type to which overtime was added and on the day of the week. In our instances, 4 different shifts are used: morning (M), day (D), evening (E) and night (N). The shift premium for a morning, evening or night shift is respectively, 7%, 16.67% or 20%. This premium is added to the regular cost of one hour of overtime. We used a 50% premium for the overtime cost per period compared to a regular work hour for a day shift (15 EUR), resulting in 22.5 EUR/hour of overtime after a day shift, and, e.g., (15 * 1.07) * 1.5 EUR/hour of overtime after a morning shift, etc.

To take into account the overtime constraints, the following nomenclature is used:

i	index of shift in the planning procedure (i=1,...,I)
C_i^o	cost of overtime in shift i
S_i	start time of shift i
E_i	end time of shift i
N_i	number of times shift i is scheduled
T_i	team size of shift i
o_{it}	number of workers doing overtime of shift i in period t ($o_{it} \in \mathbb{N}_0$) Note that overtime is allowed to start at most two hours before a shift and to end at most two hours after a shift. $o_{it} = 0$ if $t \leq S_i$ - 9 or if $t \geq E_i + 9$
γ_t	number of extra workers needed in period t
C^γ	cost of extra worker per period

The shifts set I is composed of those shifts that are active in the planning horizon. Model (1)-(9) needs to be adjusted by adding (11)-(14).

$$o_{it} \leq T_i N_i \qquad i = 1, \ldots, I \quad t = 1, \ldots, T \tag{11}$$

$$\sum_{t=1}^{T} o_{it} \leq 8 T_i N_i \qquad i = 1, \ldots, I \tag{12}$$

$$o_{i,t-1} \leq o_{it} \qquad i = 1, \ldots, I \quad t = S_i - 1, \ldots, S_i - 7 \tag{13}$$

$$o_{i,t+1} \leq o_{it} \qquad i = 1, \ldots, I \quad t = E_i + 1, \ldots, E_i + 7 \tag{14}$$

In each period and for every shift, the number of workers doing overtime, cannot exceed the team size of that shift, multiplied by the number of times that shift is scheduled (11). Additional constraints state that the total number of periods of overtime per team member needs to be smaller than two hours (i.e., eight quarters of an hour) for each shift (12). Constraint sets (13-14) ensure that the number of workers doing overtime fades when the time difference to the start or end of a shift increases. This constraint is used to guarantee continuity in the work patterns, hereby avoiding idle time between two successive working periods on the level of an individual worker.

During a preliminary test, it became clear that the overtime variables cannot guarantee a feasible solution. Sometimes overtime cannot be used (e.g., if there are no preceeding or succeeding shifts to which the overtime can be added). However, our rolling horizon procedure requires a feasible solution for every horizon in order to allow the procedure to continue. Hence, we introduce the variable γ_t, which represents the number of workers added to period t. A high cost C^γ for every man-period of this extra capacity is added to the objective function. Since the overtime variables are now stated for every shift and period, the capacity constraint set (7) needs to be changed into (16) and the objective function (1) becomes (15):

$$\text{minimise} \sum_{j=1}^{n} C_j^d \left(\sum_{t=d_j+1}^{\bar{d}_j} (t - d_j)\chi_{jt} \right) + \sum_{t=1}^{T} ((\sum_{i=1}^{I} C_i^o o_{it}) + C^\gamma \gamma_t) \tag{15}$$

$$\sum_{j=1}^{J} y_{jt} \leq \mu_t + \sum_{i=1}^{I} o_{it} + \gamma_t \qquad t = 1, \ldots, T \tag{16}$$

Delay cost and distributions. In model (1)-(9) tardy flights are penalized by a tardiness cost C_j^d per period of delay per flight, considering the due dates. The deadline of a job has been set 8 periods higher then the due date in order to ensure that a flight can be finished on time and to prevent huge tardiness values. Many studies have been conducted to estimate the delay cost to airlines. In the NEXTOR-report (Ball et al. [25]), for instance, a variety of cost components caused by flight delays are analyzed, including cost to airlines, cost to passengers, cost of lost demand, as well as the indirect impact of delay on the US economy. In other approaches the overall cost of delay for both airlines and passengers is computed (e.g., Zou and Hansen [26]). They calculate the total cost, without breaking it down into crew, maintenance, fuel and other costs. In this study, data provided by Eurocontrol (the European Organisation for the Safety of Air Navigation) will be used . In 2004, Cook et al. [27] developed a rigorous methodology and collected data for estimating the components of airline delay costs for various segments of a scheduled flight by order of Eurocontrol (a more recent version of this study can be found in Cook and Tanner [28]). We use the values of the three scenarios recommended by Eurocontrol for the overall delay cost per minute. The recommended values are presented as a range from a low value to a high value with a base value as the medium term. For the ground delays with network effect (i.e., taking into account the effect of consequential delay), these values are represented in Table 2.

Table 2: Eurocontrol [29] recommended values for the delay cost of ground delay considering network effects (in €/minute)

	Base	Low	High
Fuel costs	0.1	0.1	0.2
Maintenance costs	0.5	0.5	0.7
Crew costs	8.8	-	18.7
Ground and passenger handling	-	-	2.7
Airport charges	0.5	0.4	2.1
Passenger compensation	27.8	15.3	45.9
Direct cost to an airline	37.8	16.3	70.3
Passenger opportunity cost	43.5	7.0	43.8
Overall cost	81.3	23.3	114.1

Procedure length and interval. The considered lengths for the planning procedures are 8, 12, 16 and 20 periods of 15 minutes. With the small planning horizons, we can assume that perfect information exists about the flight arrivals. Two consecutive planning procedures are separated by an interval of 4, 6 or 8 periods. This interval is set low enough to be able to update the schedule regularly based on the new information. The total horizon length (i.e.,

the time difference between the start time of the first and the last planning procedure) is set to 100 weeks, resulting in a different number of iterations according to the interval length. In Table 3 we give an overview of the different parameter combinations.

Table 3: Parameter combinations

Procedure length (15-min periods)	Procedure interval (15-min periods)	Delay cost	Workload distribution	Arrival distribution
8	4	Base	Uniform	Peak
12	6	Low	Triangular	Spread
16	8	High		
20				

6. Results and discussion

Before analyzing the influence of the different parameters on the decisions made in the rolling horizon procedure, note that since we are examining a rolling horizon procedure, decisions made in one planning procedure affect those in the following one(s). Therefore, some results can seem counter-intuitive. The solution method was programmed using C++ and CPLEX 12.5 and tested using a Duo Core Processor of 2.4 GHz.

Table 4: Average cost and time for random instance parameters

Arrival distribution	Workload distribution					
	Uniform		Triangular		Total	
	Cost (EUR)	Time (s)	Cost (EUR)	Time (s)	Cost (EUR)	Time (s)
Peak	22,476	411	17,800	1,813	20,138	1,112
Spread	17,960	371	5,041	1,010	11,501	691
Total	20,218	391	11,421	1,412	15,819	901

Table 5: Comparison exact and heuristic approach for cost, time, extra capacity (E), overtime (O), and tardiness (T)

Approach	Cost (EUR)	Time (s)	E	O	T
Exact	17,480	5,161	12.7	42.3	4.6
Heuristic	15,587	862	12.2	28.1	3.5

We first focus on the computation times (Table 4-7). The results show that the instances with peak arrivals are nearly twice as hard to solve as those with the spread arrivals (1,112s vs 691s). The main differences in computation time, however, result from the workload distribution. The instances with triangularly distributed workload take more than 3.6 times the computation time of the ones with uniformly distributed workload (1,412s vs 391s). The

Table 6: Average cost and time for procedure length and replication interval parameters

Procedure length	Replication interval							
	4		6		8		Total	
	Cost (EUR)	Time (s)	Cost (EUR)	Time (s)	Cost (EUR)	Time (s)	Cost (EUR)	Time (s)
8	16,260	965	16,712	910	38,024	90	23,665	655
12	15,493	1,042	15,213	946	15,322	991	15,342	993
16	12,979	870	13,990	1,035	13,787	1,032	13,586	979
20	9,856	1,025	10,703	871	11,492	1,037	10,684	978
Total	13,647	976	14,154	941	19,656	787	15,819	901

Table 7: Average cost and time for delay cost and extra capacity cost parameters

Cost extra capacity	Delay cost							
	Low		Base		High		Total	
	Cost (EUR)	Time (s)	Cost (EUR)	Time (s)	Cost (EUR)	Time (s)	Cost (EUR)	Time (s)
200	5,210	1,252	3,858	817	3,911	784	4,326	951
500	8,763	935	10,895	815	10,859	905	10,172	885
1,000	11,626	928	21,423	953	24,498	826	19,182	902
2,000	16,238	864	31,971	834	40,579	901	29,596	866
Total	10,459	995	17,037	855	19,962	854	15,819	901

parameters on replication interval or procedure length do not really influence the computation time, except for the case where both parameters take value 8 (i.e., there is no overlap between consecutive planning procedures). In this case, the computation time is only 10% of the average computation time for the other parameter combinations, but the costs increase significantly. In this worst case, one cannot leftshift the jobs as effective as in the other scenarios, which decreases the flexibility of the approach, leading towards costly solutions. For the cost parameters, we see that the situation with base cost for delays and the lowest cost of non-regular capacity (200 EUR/man-period) takes about 30% more computation time than the other combinations. This is mainly due to the trade off between these two options. In the Low cost case, delaying a flight for one period costs about 350 EUR. For this amount, you can almost hire two extra workers for one period in the scenario under consideration. This offers many more trade off effects as in the scenarios with a higher cost for non-regular capacity. The switch towards the heuristic approach (5) results in a gain of factor six in terms of computation time (5,160s vs 862s). These results are based on only a subset of the instances (220 combinations), since it takes too much time to do the computations for all the instances. Since we stop the IP in the exact approach after 3,600s and continue with the best obtained solution, the exact approach is in a sense also a heuristic approach. Without drawing conclusions, we see that the solution quality of the heuristic approach is better than that of the exact approach.

The total costs (measured for all the flights over the entire planning horizon, averaged per parameter combination) depend greatly on the instance (Table 4-7). Instances with peak arrivals and those with uniformly distributed workload are both 75% more costly than instances with spread arrivals and triangularly distributed workload, respectively. This results

in a factor of 4.45 for the instances with both uniformly distributed workload and peak arrivals compared to the instances with triangularly distributed workload and spread arrivals. During the peak hours (in the case with peak arrivals), a high utilization rate is observed. As these jobs have typically narrow time windows, a delay implies expensive solutions such as non-regular capacity or delays, whereas for the spread arrivals rescheduling could be beneficial without incurring extra costs. This should motivate the airlines to spread the flights as much as possible. Moreover, maintenance companies should take this behavior into account while negotating the service contracts with the airlines. Shorter replication intervals and, especially, longer procedure lengths result in lower aggregated costs. However, short replication intervals are expensive in computation time. On the other hand, we might not be able to predict the (uncertain) behavior of the release times of jobs for longer procedure lengths.

Table 8: Average use of extra capacity (E), overtime (O) and tardiness (T) for random instance parameters. E and O are expressed in worker-periods. T is expressed in periods.

Arrival distribution	Workload distribution								
	Uniform			Triangular			Total		
	E	O	T	E	O	T	E	O	T
Peak	15.2	30.1	8.0	11.3	34.5	4.7	13.3	32.3	6.4
Spread	13.0	49.6	7.5	4.0	11.6	1.5	8.5	30.6	4.5
Total	14.1	39.8	7.7	7.6	23.1	3.1	10.9	31.5	5.4

Table 9: Average use of extra capacity (E), overtime (O) and tardiness (T) for procedure length and replication interval parameters. E and O are expressed in worker-periods. T is expressed in periods.

Procedure length	Replication interval											
	4			6			8			Total		
	E	O	T	E	O	T	E	O	T	E	O	T
8	10.4	22.3	5.8	10.4	18.8	6.2	32.4	24.4	6.3	17.7	21.8	6.1
12	10.3	27.2	5.9	9.8	24.5	5.7	10.0	24.4	5.8	10.0	25.4	5.8
16	9.0	34.9	5.1	9.3	32.5	5.7	8.9	30.7	5.5	9.0	32.7	5.4
20	6.4	50.6	3.9	6.9	42.8	4.4	7.1	44.4	4.9	6.8	45.9	4.4
Total	9.0	33.8	5.2	9.1	29.6	5.5	14.6	31.0	5.6	10.9	31.5	5.4

Another evaluation criterion is the tardiness (Table 8-10). This value represents the aggregated tardiness for all the flights over the entire planning horizon (averaged per parameter combination). In instances with peak arrivals, tardiness is 42.5% higher compared to instances with spread arrivals. This is partly due to the increased frequency of tardy flights (33%), but is also due to the increased length of the tardiness. The instances with uniformly distributed workload even face an increase of 150% regarding tardiness and an increase of 227% of departure delays (i.e., departures between the due date and the hard deadline) compared to the instances with triangularly distributed workload. Again, the increased flexibility of longer procedure lengths results in lower tardiness values. Whereas the

Table 10: Average use of extra capacity (E), overtime (O) and tardiness (T) for cost parameters. E and O are expressed in worker-periods. T is expressed in periods.

Extra capacity	Delay cost											
	Low			Base			High			Total		
	E	O	T	E	O	T	E	O	T	E	O	T
200	11.6	30.2	3.3	18.3	28.5	0.0	18.3	28.3	0.0	16.1	29.0	1.1
500	5.7	31.5	9.0	17.6	30.1	0.4	17.2	29.6	0.3	13.5	30.4	3.2
1000	4.6	33.8	11.6	11.1	31.3	3.5	11.0	30.6	3.3	8.9	31.9	6.2
2000	3.6	40.0	15.3	5.5	32.1	9.5	6.1	31.4	8.8	5.1	34.5	11.2
Total	6.4	33.9	9.8	13.1	30.5	3.3	13.1	30.0	3.1	10.9	31.5	5.4

difference in terms of average tardiness between the base cost scenario (81.3 EUR/minute of delay) and the high cost scenario (114.1 EUR/minute of delay) is only 7%, the difference between the low (23.3%) and the high cost scenario is more than 215%.

Overtime (O) represents the aggregated use of overtime in worker-periods for all the flights over the entire planning horizon (averaged per parameter combination). The results (Table 8-10) indicate that the instances with uniformly distributed workload need 75% more overtime than those with triangularly distributed workload, whereas the difference between the instances with peak and with spread arrivals is quite neglectible (1%). However, especially the instances with spread arrivals and uniformly distributed workload are the main consumers of overtime, i.e., more than twice the amount of the average of the other instances. Furthermore, it is particularly the length of the planning horizon that creates possibilities for overtime. Considering the base procedure length of 8 periods, a procedure length of 12, 16 or 20 periods results in 4.24%, 29.9% or 81.64% more overtime usage, respectively. This is mainly because of the restriction to allow overtime before the start of a shift only if that shift starts within the following two hours. This restriction is used to enable the personnel to commute to their work. Of course, when a longer procedure length is considered, the possibility of a shift ending in that planning procedure also increases.

The last cost component addresses the non-regular capacity (E), which represents the aggregated use of extra capacity in worker-periods for all the flights over the entire planning horizon (averaged per parameter combination). In (Table 8-10), we see similar results as for the other two components regarding the set of instances. Again, peak arrivals (55%) and uniformly distributed workload (85%) are more costly than spread arrivals and triangularly distributed workload. Compared to the case of 2,000 EUR/man-period of extra capacity, the case of 1,000, 500 and 200 EUR/man-period, consumes 75%, 166% and 217% more extra capacity. Logically, as the cost of non-regular capacity decreases, usage goes up. The longer the procedure length, the less extra capacity is needed (-32% for a length of 20 periods compared to 12 periods). Looking at the interaction effects between the usage of non-regular capacity and the delay costs, we see that for the low delay cost scenario, the usage of non-regular capacity drops with more than 50% compared to the base and high delay cost scenario. Again, the case where both the replication interval and the replication length take value 8 is very costly. In this scenario, flexibility is limited, resulting in numerous costly measures

to guarantee the processing of the jobs.

7. Conclusion and future research

This paper describes a rolling horizon procedurefor scheduling malleable tasks with an ascending or descending property without preemption. The goal is to minimise the cost of overtime and delays. The approach consists of a proactive and reactive policy to ensure robustness regarding the stochasticity in the release times of the jobs. The results indicate that huge time savings can be achieved by considering a base case scenario, which can be adjusted heuristically or with an exact approach. Furthermore, there exists a clear trade-off between non-regular capacity and delays. The results also indicate that a longer planning procedure length can result in lower costs. However, one has to consider the (lower) quality of the retrieved information and the higher costs of getting this information. Also, shorter replication intervals result in lower delay costs, but increase the computation time.

This research creates some interesting possibilities for further research. First, other heuristics can be developed to further improve the speed and quality of the approach. Second, the problem formulation can be improved. Some possible extensions that deserve our attention are adding valid inequalities and decomposing the model. Finally, other sources of uncertainty can be incorporated.

Appendices

Bibliography

[1] K. Pruhs, J. Sgall, E. Torng, Online scheduling, in: J. Leung (Ed.), Handbook of Scheduling: Algorithms, Models, and Performance Analysis, CRC Press, 2004, pp. 15–1 – 15–41.

[2] S. Sethi, G. Sorger, A theory of rolling horizon decision making, Annals of Operations Research 29 (1991) 387–415.

[3] R. As'ad, K. Demirli, Production scheduling in steel rolling mills with demand substitution: Rolling horizon implementation and approximations, International Journal of Production Economics 126 (2010) 361–369.

[4] V. Vargas, R. Metters, A master production scheduling procedure for stochastic demand and rolling planning horizons, International Journal of Production Economics 132 (2011) 296–302.

[5] M. Gronalt, R. F. Hartl, Workforce planning and allocation for mid-volume truck manufacturing: A case study, International Journal of Production Research 41 (2003) 449–463.

[6] J. F. Bard, H. W. Purnomo, Hospital-wide reactive scheduling of nurses with preference considerations, IIE Transactions 37 (2005) 589–608.

[7] G. M. Campbell, On-call overtime for service workforce scheduling when demand is uncertain, Decision Sciences 43 (2012) 817–850.

[8] L. K. Nielsen, L. Kroon, G. Maróti, A rolling horizon approach for disruption management of railway rolling stock, European Journal of Operational Research 220 (2012) 496–509.

[9] J. G. Rakke, M. Stalhane, C. R. Moe, M. Christiansen, H. Andersson, K. Fagerholt, I. Norstad, A rolling horizon heuristic for creating a liquefied natural gas annual delivery program, Transportation Research Part C: Emerging Technologies 19 (2011) 896–911.

[10] Q. Shen, F. Chu, H. Chen, A Lagrangian relaxation approach for a multi-mode inventory routing problem with transshipment in crude oil transportation, Computers & Chemical Engineering 35 (2011) 2113–2123.

[11] E. K. Burke, M. Dror, J. B. Orlin, Scheduling malleable tasks with interdependent processing rates: Comments and observations, Discrete Applied Mathematics 156 (2008) 620–626.

[12] M. Drozdowski, Scheduling multiprocessor tasks: An overview, European Journal of Operational Research 94 (1996) 215–230.

[13] R. Sadykov, A dominant class of schedules for malleable jobs in the problem to minimize the total weighted completion time, Computers & Operations Research 39 (2012) 1265–1270.

[14] J. Blazewicz, M. Machowiak, G. Mounié, D. Trystram, Approximation algorithms for scheduling independent malleable tasks, volume 2150 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2001, pp. 191–197.

[15] J. Blazewicz, P. Dell'Olmo, M. Drozdowski, P. Maczka, Scheduling multiprocessor tasks on parallel processors with limited availability, European Journal of Operational Research 149 (2003) 377–389.

[16] P.-C. Hu, Minimising total tardiness for the worker assignment scheduling problem in identical parallel-machine models, The International Journal of Advanced Manufacturing Technology 23 (2004) 383–388.

[17] P.-C. Hu, Further study of minimizing total tardiness for the worker assignment scheduling problem in the identical parallel-machine models, The International Journal of Advanced Manufacturing Technology 29 (2006) 165–169.

[18] E. W. Hans, Resource Loading by Branch-and-Price Techniques, Ph.D. thesis, University of Twente, Enschede, 2001.

[19] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey, volume Volume 5, Elsevier, pp. 287–326.

[20] R. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, New York, 1979.

[21] J. Blazewicz, Scheduling preemptible tasks on parallel processors with information loss, Tsi-Technique Et Science Informatiques 3 (1984) 415–420.

[22] E. Demeulemeester, W. Herroelen, Robust Project Scheduling, Foundations and trends in technology, information and operations management, Now Publishers, 2011.

[23] J. Van den Bergh, P. De Bruecker, J. Beliën, L. De Boeck, E. Demeulemeester, A three-stage approach for aircraft line maintenance personnel rostering using MIP, discrete event simulation and DEA, Expert Systems with Applications (2012) In press.

[24] J. Beliën, E. Demeulemeester, P. De Bruecker, J. Van den Bergh, B. Cardoen, Integrated staffing and scheduling for an aircraft line maintenance problem, Computers & Operations Research (2012) In press.

[25] M. Ball, C. Barnhart, M. Dresner, M. Hansen, K. Neels, A. Odoni, E. Peterson, L. Sherry, A. Trani, B. Zou, Total delay impact study: a comprehensive assessment of the costs and impacts of flight delay in the United States, Technical Report, NEXTOR, 2010.

[26] B. Zou, M. Hansen, Impact of operational performance on air carrier cost structure: Evidence from us airlines, Transportation Research Part E: Logistics and Transportation Review 48 (2012) 1032–1048.

[27] A. Cook, G. Tanner, S. Anderson, Evaluating the true cost to airlines of one minute of airborne or ground delay, Technical Report, University of Westminster, London for Eurocontrol, Brussels, 2004.

[28] A. Cook, G. Tanner, European airline delay cost reference values, Technical Report, University of Westminster, London for Eurocontrol, Brussels, 2011.

[29] Eurocontrol, Standard inputs for Eurocontrol cost benefit analysis, Technical Report, Eurocontrol, 2011.