Scheduling optimization of a real flexible job shop including side constraints regarding maintenance, fixtures, and night shifts

Karin Thörnblad[a,b,1], Ann-Brith Strömberg[a], Michael Patriksson[a], Torgny Almgren[b]

[a]*Mathematical Sciences, Chalmers University of Technology and University of Gothenburg, SE-421 96 Göteborg, Sweden*
[b]*GKN Aerospace Engine Systems, Dept. of Logistics Development, SE-461 81 Trollhättan, Sweden*

Abstract

We present a generic iterative scheduling procedure for the scheduling of a real flexible job shop, the so-called multitask cell at GKN Aerospace Engine Systems in Sweden. A time-indexed formulation of the problem is presented including side constraints regarding preventive maintenance, fixture availability, and unmanned night shifts. This paper continues the work in Thörnblad et al. [35], with an improvement of the iterative solution procedure and the inclusion of constraints regarding night shifts during which only unmanned processing is allowed. The resulting schedules are compared with schedules constructed using two priority dispatching rules. Computational results show that the gain of including the night shifts constraints is significant and that the methodology developed is able to produce near-optimal schedules for industrial data instances for the coming shift within an acceptable practical time frame.

Keywords: Flexible job shop scheduling problem, Mixed integer linear programming (MILP), Unmanned night shifts, Fixture availability, Preventive maintenance, Priority dispatching rules

Email addresses: `karin.thornblad@gknaerospace.com` (Karin Thörnblad), `anstr@chalmers.se` (Ann-Brith Strömberg), `mipat@chalmers.se` (Michael Patriksson), `torgny.almgren@gknaerospace.com` (Torgny Almgren)

[1]Corresponding author, GKN Aerospace Engine Systems, Dept. of Logistics Development, 9510KT, SE-461 81 Trollhättan, Sweden. Tel. +46 520 29 22 66.

August 19, 2013

1. Introduction

The multitask cell at GKN Aerospace Sweden is a production cell containing ten resources, of which five are multi-purpose machines capable of performing three types of processing tasks: turning, milling, and drilling. The problem of optimally scheduling this production cell is recognized as a *flexible job shop scheduling problem* (FJSP) [33].

In our previous work [35], we propose an iterative solution procedure that is able to produce near-optimal schedules for real data instances of the multitask cell including side constraints regarding a limited fixture availability and required preventive maintenance (PM). The modelling and solution development work presented in ibid. is here continued by an improvement of the iterative solution procedure and the inclusion of constraints regarding night shifts—during which only unmanned processing is allowed—in the scheduling optimization model. The resulting schedules are compared with schedules constructed using two priority dispatching rules, of which one is often utilized in practice, and the other is a built-in scheduling method of the control system of the multitask cell.

2. Literature review

In a *job shop*, each job follows a predetermined sequence of operations [26, Chapter 2]. Each operation must be scheduled for processing in a designated machine during a predefined amount of processing time. A *flexible job shop* is a generalization of a job shop such that each operation may be scheduled in any of the machines in a given subset of all resources [7, Chapter 4].

Since the job shop scheduling problem (JSP) and hence also the FJSP are NP-hard [6], these problems have attracted the interest from researchers ever since the late 1950's when Wagner [36], Bowman [4], and Manne [21] proposed the first mathematical models for similar scheduling problems. Since the computation times required for solving JSPs and FJSPs by exact methods previously have been considered too long for practical purposes, a lot of research has focused on finding approximate solutions to these problems using heuristic methods [5].

Nowadays, due to the development of mathematical optimization theory and practice—and of computer hard- and software—it is possible to find near-optimal schedules using exact methods for real instances. In Thörnblad et al. [35], we solve a real FJSP within an acceptable time frame for the scheduling of the coming work shift in the multitask cell.

The objective that is most often utilized for scheduling problems is the minimization of the makespan [17], i.e., the time between the start of the first operation and the completion of the last operation of the schedule. In Thörnblad et al. [34], we noted a large difference in the performance of the scheduling models depending on which objective was considered. Hence, it is of great importance that the evaluation of scheduling models are made with respect to objective functions that are well suited for the real applications for which the models are intended. In ibid., we argue that the minimization of the makespan is not well suited for a dynamic environment, in which there is a need to repeatedly reschedule the job shop, which is the case in most real applications. Instead, we propose to minimize an objective function being the weighted sum of the completion times and the total weighted tardiness, which—when the tardiness weight for a given job is a non-increasing functions of its due date—works well in a dynamic environment. The rescheduling policy proposed is a *hybrid event-driven policy*, in which the rescheduling is periodically performed with respect to a rolling time horizon, but whenever there is an urgent event (e.g., a machine break-down), a rescheduling is immediately performed. In the survey of dynamic scheduling by Ouelhadj and Petrovic [23], several dynamic scheduling policies are described.

In the operations research literature, there are many mathematical optimization models of JSPs and FJSPs employing variables similar to those deployed by Manne [21] in 1960. In the evaluations performed by Pan [25] and Demir and İşleyen [9] for the JSP and the FJSP, respectively, the Manne models were found to be the best. Both these studies were conducted with the objective of minimizing the makespan and the comparisons were made with several alternative models employing variables similar to those deployed by Wagner [36] and Bowman [4]. The Bowman models are called *time-indexed models* since they are based on a time discretization of the planning horizon. The decision variables used by Bowman [4] equal 1 if the corresponding job is processed by a specific resource during a specific time period, and 0 otherwise. An alternative definition of time-indexed decision variables is employed by Sousa and Wolsey [30]; the variables equal 1 if the corresponding job *starts* in a specific discrete time period, and 0 otherwise.

In Thörnblad et al. [34], the best available Manne model—by Özgüven et al. [24] according to Demir and İşleyen [9]—is compared with our iterative scheduling procedure employing a time-indexed model with the alternative decision variables. When the tardiness objective was employed, our iterative scheduling procedure outperformed the Manne model. The opposite result was obtained in Azem et al. [1], in which a Manne model (therein referred to as a "disjunctive model") outperformed a time-indexed model for a JSP with non-fixed resource availability constraints (i.e., with PM activities included). The latter comparison was, however, made for the objective of minimizing the makespan, and the time-indexed model was solved as is, without an iterative procedure for determining a suitable value of the time horizon.

To the best of our knowledge, there is no published work with a mathematical optimization model of an FJSP extended to take fixture availability, PM, and/or unmanned night shifts into account. The only published work found concerning an FJSP including fixture availability constraints are Rahimifard and Newman [27] and Syberfeldt et al. [32]; both describe simulation-based scheduling approaches. The latter studies the multitask cell at GKN Aerospace and includes both fixture availability constraints and unmanned night shifts.

The constraints describing the necessary scheduling of PM activities are often called *availability constraints* in the literature. Gao et al. [13] categorize availability constraints into two types: fixed and non-fixed. When employing fixed constraints, the PM activities are already scheduled at fixed time slots, during which the resources are unavailable for processing. We consider non-fixed availability constraints, i.e., the starting time of the period of unavailability is flexible within a time window and is determined within the scheduling procedure. Gao et al. [13], Wang and Yu [37], and Rajkumar et al. [28] all consider the FJSP with non-fixed availability constraints and three objectives: makespan, total workload, and critical machine workload. They propose a hybrid genetic algorithm, a filtered beam search algorithm, and a GRASP algorithm, respectively.

Apart from the above-mentioned reference [32], we have found no published work considering the opportunity to schedule unmanned processes during night shifts, although the problem is not unique for the multitask cell at GKN Aerospace: for example, Slomp and Zijm [29] find it desirable to utilize the unmanned night shifts as much as possible. A similar problem arises in the scheduling of personnel having varying skills and working in shifts; see Van den Bergh et al. [2] for a recent survey in this field of research.

In Section 3, the scheduling problem is described in detail and the notation required for presenting the mathematical model is given. Section 4 describes a number of dispatching rules. In Section 5, the time-indexed model for the FJSP, including limited the fixture availability, the required PM activities, and the unmanned night shifts, is presented. The iterative solution procedure is described in Section 6 and in Section 7 the computational results based on real data are presented. Finally, in Section 8, conclusions are drawn.

3. Problem description

The multitask cell is described in detail in Thörnblad et al. [35] and Thörnblad [33]. The notation of sets, parameters, and indices used in this paper is summarized in Table 1. The assumptions regarding the flexible job shop of the multitask cell are given by the following list of items:

1. Job j has n_j operations that must be processed according to a predefined sequence.
2. The operations $i \in \mathcal{N}_j := \{1, \ldots, n_j\}$ of the jobs $j \in \mathcal{J}$ are non-preemptive, i.e., once started, the operations must be completed.
3. The execution of operation i of job j requires a machine selected from a subset $\mathcal{M}_{ij} \subseteq \mathcal{K}$ of the set of available machines (i.e., resources) $k \in \mathcal{K}$.
4. Time is discretized into the set $\mathcal{T}$ of time steps. We will use the terms "time" and "time step" to denote points in time.
5. Resource $k \in \mathcal{K}$ is available for processing at time a_k and onwards in time.
6. The transportation times between the machines inside the multitask cell are neglected.
7. All the machines can be simultaneously maintained.
8. At the time when a maintenance task $j \in \mathcal{J}^{\text{maint}}$ is performed on a machine, no operation can be processed on that machine.
9. Each maintenance task must start within a predefined time window.
10. Each job $j \in \mathcal{S}_f \subset \mathcal{J}$ occupies a fixture of type $f \in \mathcal{F}$ during all of its operations.
11. The number of available fixtures of each type f is limited.
12. During the unmanned (night) shifts, solely unmanned processing is allowed.
13. Some jobs $(j, q) \in \mathcal{Q} \subset \mathcal{J} \times \mathcal{J}$ are subject to precedence constraints with a time lag v_{jq}, i.e., job j has to be completed at least v_{jq} time steps before job q starts.

Table 1: Nomenclature

Indices	*Descriptions*
i/i'	operations
j/q	jobs (tasks)
k/l	machines (resources)
$u/\mu/\nu$	time steps
f	fixture types
n	night shifts
Sets	*Descriptions*
$\mathcal{N}_j$	set of operations of job j, $i \in \mathcal{N}_j = \{1, \ldots, n_j\}$
$\mathcal{J}$	set of jobs, $j \in \mathcal{J}$
$\mathcal{J}^{\text{maint}}$	set of preventive maintenance (PM) tasks, $j \in \mathcal{J}^{\text{maint}}$
$\mathcal{Q}$	set of pairs of jobs with precedence constraints, $(j, q) \in \mathcal{Q}$
$\mathcal{K}$	set of machines, $k \in \mathcal{K}$
$\mathcal{K}^{\text{setup}}$	set of set-up stations, $k \in \mathcal{K}^{\text{setup}}$
$\mathcal{M}_{ij}$	set of machines allowed to process operation i of job j
$\mathcal{T}$	set of time steps, $u \in \mathcal{T}$
$\mathcal{F}$	set of fixture types, $f \in \mathcal{F}$
$\mathcal{S}_f$	set of jobs that use fixture of type f, $j \in \mathcal{S}_f \subset \mathcal{J}$
$\mathcal{P}$	set of night shifts, $n \in \mathcal{P}$
Parameters	*Descriptions*
n_j	total number of operations of job j
p_{ij}	processing time of operation i of job j
r_{ij}	release date (time) of operation i of job j
$\hat{r}^n(\bar{r}^n)$	start (end) time of night shift n
$\hat{r}_{ijn}(\bar{r}_{ijn})$	intermediate deadline (release date) of operation i of job j before (after) night shift n
δ_{ij}	the total remaining processing time of job j at the start of operation i
d_j	due date (time) of job j
a_k	time when resource k becomes available
T	length of the time horizon
ℓ	length of the time steps
λ_{ijk}	1 if operation i of job j can be processed on resource k 0, otherwise
γ_f	total number of fixtures of type f
κ_{jk}	$= \begin{cases} 1, & \text{if maintenance task } j \text{ should be performed in resource } k, \\ 0, & \text{otherwise} \end{cases}$
τ_{jk}	start time of the time window for maintenance task j in resource k
Δ	length of the maintenance time window
$\alpha_j(\beta_j)$	objective weights for the completion time (tardiness) of job j

During a (possibly empty) time interval at the start and/or end of each operation a certain amount of unmanned processing is allowed. Hence, also parts of unmanned jobs may be scheduled during night shifts (Item 12). In fact, for some operations, unmanned processing is allowed throughout the whole operation, a feature which is desired for all operations; should this property hold, the night and day shifts could be equivalently scheduled. However, the first ($i = 1$) and the last ($i = n_j$) operation of every job j, i.e., the mounting into and demounting out of fixtures, are totally manual. Currently, for around half of the operations that are to be performed in one of the five multipurpose machines, an operator must be present during a part of the processing time. Two partly unmanned scheduled operations are illustrated in Figure 1, where the part of the processing times $p_{ij}(p_{i'q})$ of operations $i(i')$ of jobs $j(q)$ during which unmanned processing is allowed correspond to the striped sections of the processing times.

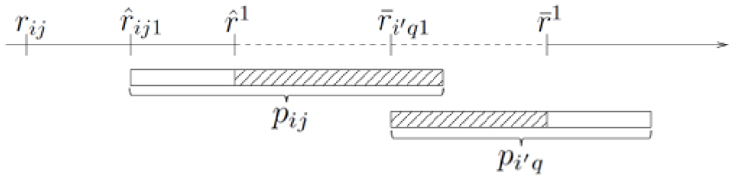


Figure 1: Night shift 1 starts at $\hat{r}^1$ and ends at $\bar{r}^1$. Operation $i(i')$ of job $j(q)$ with processing time p_{ij} ($p_{i'q}$) is scheduled such that its allowed unmanned periods are maximally utilized.

Let us by p_{ij}^{end} denote the amount of unmanned processing time towards the end of operation i of job j. An *intermediate deadline* for night shift n is then defined as

$$\hat{r}_{ijn} := \hat{r}^n - (p_{ij} - p_{ij}^{\text{end}}), \tag{1a}$$

provided that night shift n starts at time $\hat{r}^n$. Similarly, an *intermediate release date* for night shift n is defined as

$$\bar{r}_{ijn} := \bar{r}^n - (p_{ij} - p_{ij}^{\text{start}}), \tag{1b}$$

where p_{ij}^{start} denotes the amount of unmanned processing time at the beginning of operation i of job j, and $\bar{r}^n$ denotes the ending time of night shift n. In Figure 1, operations i and i' are scheduled such that the night shift is maximally utilized, i.e., operation i of job j is scheduled at its intermediate deadline $\hat{r}_{ij1}$, and operation i' of job q at its intermediate release date $\bar{r}_{i'q1}$.

The precedence relations between jobs (Item 13 above) arise due to the fact that the products typically visit the multitask cell multiple times on their way to completion. After job j is completed, the corresponding part is transported to and processed in other workshops of the factory during v_{jq} time steps before returning to the multitask cell for processing of job q, $(j,q) \in \mathcal{Q}$.

The release date r_{1j} of the first operation of job j corresponds to the time of the expected arrival of the corresponding part at the multitask cell. The release date of operation $i = 2,\dots,n_j$, is then calculated as $r_{ij} := r_{i-1,j} + p_{i-1,j}$.

4. Priority dispatching rules

The most well-known and widely used dispatching rule is propably the First–In, First–Out (FIFO) priority rule; it is also currently used as decision support in the detailed production planning of a majority of the workshops at GKN Aerospace. Research on dispatching rules has been active for several decades and over 100 dispatching rules are proposed in the literatur; see Hopp and Spearman [15]. For an extensive summary and discussion on priority dispatching rules, see Blackstone et al. [3] and Haupt [14]. It is still a relevant topic: dispatching rules are widely used in practise (e.g. at GKN Aerospace), researchers still propose new dispatching rules [20], and priority dispatching rules are often included in meta-heuristics [8].

Dispatching rules can be classified in various ways; a distinction can, e.g., be made between *static* and *dynamic* rules [26]. Dynamic rules are time dependent, i.e., the priorities change over time, whereas static rules (as, e.g., the FIFO priority rule) are not.

The built-in scheduling algorithm in the control system of the multitask cell is based on a critical ratio (CR) [18] defined as

$$\mathrm{CR}_j(t) := \begin{cases} \min\limits_{i \in \mathcal{N}_j} \left\{ \dfrac{1 + (d_j - t)|\mathcal{M}_{ij}|}{1 + \sum_{i'=i}^{n_j} p_{i'j}} \right\}, & d_j > t, \\ \min\limits_{i \in \mathcal{N}_j} \left\{ \dfrac{1}{1 + (t - d_j)|\mathcal{M}_{ij}| \left(1 + \sum_{i'=i}^{n_j} p_{i'j}\right)} \right\}, & d_j \le t, \end{cases} \qquad (2)$$

for all $j \in \mathcal{J}$, where the summation in the formula results in the total remaining processing time. The job is assigned the minimum critical ratio among its operations. At each time point t, the job j possessing the lowest value of $\mathrm{CR}_j(t)$ is given the highest priority. When a job is late, i.e., when $d_j \leq t$, the job is given a higher priority, since then $\mathrm{CR}_j(t)$ is decreased. If a machine is available at time t, the operation corresponding to the job with the highest job priority is scheduled in this machine, provided that all precedence constraints and other side constraints are fulfilled at time t.

In this article, we will compare the schedules found by our iterative scheduling procedure with the schedules constructed by the static FIFO rule and the dynamic CR rule that is supplied by the built-in control system of the production cell.

5. The time-indexed formulation

Time-indexed models are based on a time discretization of the planning horizon, which is divided into a set $\mathcal{T} := \{0, \dots, T-1\}$ of intervals, each of length ℓ hours. The number T of time intervals must be large enough such that the time interval $[0, T\ell]$ can contain an optimal schedule. In our time-indexed model, the decision variables are defined as

$$x_{ijku} = \begin{cases} 1, & \text{if operation } i \text{ of job } j \text{ starts at time } u \text{ in resource } k, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where, $k \in \mathcal{K},\ i \in \mathcal{N}_j,\ j \in \mathcal{J}$, and $u \in \mathcal{T}$. Similar variable definitions have been employed by Dyer and Wolsey [10] and Kedad-Sidhoum et al. [19] for single and parallel machine scheduling problems, respectively. Let us for the moment denote the objective function by $z(\cdot)$, without being more precise. The base model for a general flexible job shop is then formulated as that to minimize $z(\cdot)$ subject to

$$\sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{N}_j} \sum_{\mu=(u-p_{ij}+1)_+}^{u} x_{ijk\mu} \leq 1, \qquad k \in \mathcal{K},\ u \in \mathcal{T}, \tag{4a}$$

$$\sum_{k \in \mathcal{M}_{ij}} \sum_{\mu=0}^{u} x_{ijk\mu} - \sum_{l \in \mathcal{M}_{i+1,j}} \sum_{\nu=0}^{u+p_{ij}} x_{i+1,jl\nu} \geq 0, \qquad u \in \{0, \dots, T-p_{ij}\},\ i \in \mathcal{N}_j \setminus \{n_j\},\ j \in \mathcal{J}, \tag{4b}$$

$$\sum_{k \in \mathcal{M}_{ij}} \sum_{u \in \mathcal{T}} x_{ijku} = 1, \qquad i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{4c}$$

$$\sum_{k \in \mathcal{K} \setminus \mathcal{M}_{ij}} \sum_{u \in \mathcal{T}} x_{ijku} = 0, \qquad i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{4d}$$

$$x_{ijku} = 0, \qquad u \in \{0, \ldots, \max\{r_{ij}, a_k\} - 1\},\ k \in \mathcal{M}_{ij},\ i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{4e}$$

$$x_{ijku} = 0, \qquad u \in \{T - \delta_{ij} + 1, \ldots, T-1\},\ k \in \mathcal{M}_{ij},\ i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{4f}$$

$$x_{ijku} \in \{0,1\}, \qquad i \in \mathcal{N}_j,\ j \in \mathcal{J},\ k \in \mathcal{K},\ u \in \mathcal{T}, \tag{4g}$$

where $(b)_+ := \max\{b, 0\}$ for any $b \in \mathbb{R}$. In the model (4), all parameters are assumed to be integer-valued; the sometimes necessary approximation of the real data is discussed in Section 6. The constraints (4a) ensure that at most one operation at a time is scheduled in each resource. The constraints (4b) define the precedence relations between the operations of a job. The constraints (4c) ensure that each operation is scheduled to be processed exactly once in an allowed resource. The constraints (4d) set all variables corresponding to an operation to zero for the set of resources in which the operation is not allowed to be processed; these redundant constraints are included since we discovered in Thörnblad et al. [34] that the computations ran faster (w.r.t. clocktime) when these constraints were included. Finally, the constraints (4e) ensure that no operation is scheduled before its release date or before the corresponding resource is available, and the constraints (4f) make sure that no operation is scheduled too late such that the succeeding operations can not be completed before the end of the planning horizon. The constraints (4g) impose the binary constraints on the variables.

In Thörnblad et al. [35], a time-indexed model was formulated for the problem of scheduling the multitask cell using a parameter λ_{ijk} which equals 1 if operation i of job j is allowed to be processed in resource k, and 0 otherwise. This is an alternative way, where the set $\mathcal{M}_{ij}$ is not needed to be defined. In that model, the set $\mathcal{M}_{ij}$ is replaced by the set $\mathcal{K}$ in the constraints (4b)–(4f), and the constraints

$$\sum_{u \in \mathcal{T}} x_{ijku} \ \leq\ \lambda_{ijk}, \quad i \in \mathcal{N}_j,\ j \in \mathcal{J},\ k \in \mathcal{K},$$

are added to the model. Before the preprocessing is done in the optimization solver, this model employing the parameter λ_{ijk} contains more constraints than the model (4) employing the sets $\mathcal{M}_{ij}$. However, after the preprocessing with AMPL-CPLEX12 [12, 16], these two models contain the same number of variables and constraints. The inclusion of the constraints (4d) however enables a higher degree of parallelization of the computations; as a result, the computation times required to solve the model (4) employing the sets $\mathcal{M}_{ij}$ are shorter (w.r.t. clocktime) than those required to solve a model employing the parameter λ_{ijk}.

5.1. Constraints unique for the multitask cell

The precedence constraints with a time lag between two jobs that are to be performed on the same physical component, are formulated similarly to the precedence constraints (4b) between the operations of the same job:

$$\sum_{k \in \mathcal{M}_{n_j j}} \sum_{\mu=0}^{u} x_{n_j j k \mu} - \sum_{k \in \mathcal{M}_{1q}} \sum_{\nu=0}^{u+v_{jq}} x_{1qk\nu} \geq 0, \quad u = 0, \ldots, T - v_{jq}, \ (j,q) \in \mathcal{Q}. \tag{5}$$

In the multitask cell, the three set-up stations, denoted by the set $\mathcal{K}^{\text{setup}}$ and in which the physical parts are mounted into and demounted out of fixtures, are identical. All such operations can thus be performed in any of the set-up stations. In order to eliminate the corresponding mathematical symmetry, these resources are treated as one, with a capacity of three. The constraints (4a), for $k \in \mathcal{K}^{\text{setup}}$, are hence reformulated as

$$\sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{N}_j} \sum_{\mu=(u-p_{ij}+1)_+}^{u} x_{ijk\mu} \leq 3, \quad k \in \mathcal{K}^{\text{setup}}, \ u \in \mathcal{T}, \tag{6a}$$

and the range for the index k in the constraints (4a) is altered to $\mathcal{K} \setminus \mathcal{K}^{\text{setup}}$, according to

$$\sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{N}_j} \sum_{\mu=(u-p_{ij}+1)_+}^{u} x_{ijk\mu} \leq 1, \quad k \in \mathcal{K} \setminus \mathcal{K}^{\text{setup}}, \ u \in \mathcal{T}. \tag{6b}$$

5.2. Side constraints regarding fixtures, maintenance, and night shifts

The number of fixtures of type f that are simultaneously in use is limited to γ_f. Since the part is mounted into the fixture during all the operations of a job, the limited fixture availability is formulated as

$$\sum_{j\in\mathcal{S}_f}\sum_{k\in\mathcal{K}}\left(\sum_{\mu=0}^{u} x_{1jk\mu} - \sum_{\nu=0}^{(u-p_{n_j j})_+} x_{n_j jk\nu}\right) \leq \gamma_f, \quad f\in\mathcal{F},\ u\in\mathcal{T}. \tag{7}$$

A PM task occupies the corresponding machine during the maintenance operation. Hence, it can be characterized as a job consisting of one operation. Therefore, for each $j \in \mathcal{J}^{\text{maint}}$, $n_j := 1$ and the time duration of maintenance task j equals p_{1j}. Each maintenance task must start in a predefined time window of length Δ:

$$\sum_{\mu=\max\{\tau_{jk},a_k\}}^{\min\{\tau_{jk}+\Delta-1,T-p_{1j}\}} x_{1jk\mu} \geq \kappa_{jk}, \qquad j\in\mathcal{J}^{\text{maint}},\ k\in\mathcal{K}\setminus\mathcal{K}^{\text{setup}}, \tag{8a}$$

where x_{1jku} are the decision variables corresponding to the maintenance task $j \in \mathcal{J}^{\text{maint}}$. The constraints (8a) allow more than the required κ_{jk} maintenance tasks of type j to be scheduled on resource k, but this option will be suppressed by the objective function; see Section 5.3. No PM tasks need to be performed in the set-up stations. Since no operation can be processed in a resource while maintained, the capacity constraints (6b) are altered as

$$\sum_{j\in\mathcal{J}\cup\mathcal{J}^{\text{maint}}}\sum_{i\in\mathcal{N}_j}\sum_{\mu=(u-p_{ij}+1)_+}^{u} x_{ijk\mu} \leq 1, \qquad k\in\mathcal{K}\setminus\mathcal{K}^{\text{setup}},\ u\in\mathcal{T}. \tag{8b}$$

For some alternative formulations of scheduling PM tasks, see [35].

The constraints required to take the unmanned night shift $n \in \mathcal{P}$, into account are given by

$$x_{ijku} = 0, \quad k\in\mathcal{M}_{ij},\ u\in\{\hat{r}_{ijn}+1,\dots,\bar{r}_{ijn}-1\},\ i\in\mathcal{N}_j,\ j\in\mathcal{J},\ n\in\mathcal{P}, \tag{9a}$$

where the parameters $\hat{r}_{ijn}$ and $\bar{r}_{ijn}$ are defined in (1a) and (1b), respectively. The constraints (9a) ensure that operation i of job j is not scheduled to start during the interval between its operation-specific intermediate deadline, $\hat{r}_{ijn}$, and release date, $\bar{r}_{ijn}$. Note that this interval may occur during daytime, depending on the allowed time of unmanned processing at the start and/or end of the corresponding operation. Each PM task requires an operator present in the multitask cell throughout its duration. Therefore, the constraints ensuring that no maintenance task is scheduled during an unmanned night shift are formulated as

$$x_{1jku} = 0, \quad k \in \mathcal{K}, u \in \{\hat{r}^n - p_{1j} + 1, \dots, \bar{r}^n - 1\},\ j \in \mathcal{J}^{\text{maint}},\ n \in \mathcal{P}, \tag{9b}$$

where $\hat{r}^n$ and $\bar{r}^n$ denote the starting and the ending time of night shift n, respectively.

5.3. The objective function

As stated in Thörnblad et al. [34] the minimization of the makespan is not suitable as an objective if the flexible job shop is operating in a dynamic environment. In fact, some jobs then risk never being processed since nothing prevents a certain job from being scheduled close to the end of each of a set of subsequent schedules—if the rescheduling is performed at a time point before the last job of the previous schedule has started. If the objective function instead includes the tardiness $T_j \geq 0$ of job j, multiplied by a weight $\beta_j \geq 0$, where the value of β_j increases with and increased delay of the job, this risk is reduced. Another reason why the minimization of the makespan is not suitable is that at the start of the scheduling time interval, typically not all jobs are yet released, but they are expected to arrive at the workshop at given release dates. For an instance with at least one job possessing a very late release date—such that all the other jobs can be finished before the earliest possible completion time of this job—the minimization of the makespan would yield an abundance of optimal solutions, some of which would result in very poor throughput times for the remaining jobs.

We propose an objective function mainly focusing on minimizing the total weighted tardiness. Since instances with no tardy jobs may very well occur in real industrial cases, we also include a sum of weighted completion times $C_j \geq 0$ in the objective, as

$$\text{minimize} \sum_{j \in \mathcal{J}} (\alpha_j C_j + \beta_j T_j), \tag{10}$$

where $T_j := (C_j - d_j)_+$, and $\alpha_j > 0$, $j \in \mathcal{J}$, are objective weights for the completion times of the jobs. We define the tardiness objective weights $\beta_j := B(1 - d_j/|d_D|)_+,$, as in Thörnblad et al. [35], where D denotes the job having the largest absolute due date that is not an outlier, and the scalar parameter $B \gg \alpha_j$, since tardiness is meant to be the main objective.

Since the completion time C_j can be expressed as

$$C_j := \sum_{k \in \mathcal{M}_{n_j j}} \sum_{u \in \mathcal{T}} (u + p_{n_j j}) x_{n_j jku}, \tag{11}$$

the objective function for our base model is formulated as to minimize

$$z'(x) := \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{M}_{n_j j}} \sum_{u\in\mathcal{T}} \Big(\big[\alpha_j(u+p_{n_j j}) + \beta_j(u+p_{n_j j}-d_j)_+\big] x_{n_j jku} - \varepsilon u x_{1jku} \Big), \quad (12)$$

where the term $\varepsilon u x_{1jku}$ is included in order to schedule the first operation of each job as late as possible; the relation $0 < \varepsilon \ll \alpha_j$ then must hold for all $j \in \mathcal{J}$, since the completion times of the jobs should not be affected by this term. This inclusion facilitates rescheduling; see Thörnblad et al. [35].

If the scheduling problem includes PM tasks, yet another term needs to be included in the objective function, according to

$$z(x) := z'(x) + \varepsilon^{\text{maint}} \sum_{j\in\mathcal{J}^{\text{maint}}} \sum_{k\in\mathcal{K}\setminus\mathcal{K}^{\text{setup}}} \sum_{u\in\mathcal{T}} x_{1jku}. \quad (13)$$

The problem of scheduling the multitask cell including side constraints regarding maintenance, fixtures, and night shifts can now be formulated as that to minimize $z(x)$, defined in (13), subject to the constraints (4b)–(4g), and (5)–(9). We have computationally tested our time-indexed model of the FJSP with different sets of side constraints, as indicated in Table 2.

Table 2: Definitions and notation for the models implemented and tested. The capital letters refer to the time-indexed model (TI), fixture availability (F), maintenance activities (M), and unmanned night shifts (N).

Notation	*Description*	*Model*
TI-FMN	FJSP including all side constraints	minimize (13) subject to (4b)–(4g), (5)–(9)
TI-FM	FJSP including fixture and PM constraints	minimize (13) subject to (4b)–(4g), (5)–(8)

6. The iterative solution procedure

As stated by Stefansson et al. [31], the time-indexed formulations are flexible and capable of accounting for many scheduling features, such as all the side constraints proposed in Section 5.2. There are, however, two major drawbacks in utilizing a discrete time representation, namely

i. the approximation of time, and
ii. the large number of binary variables and constraints required.

In order to decrease the gap between the optimal objective value of the approximate solution obtained from the time-indexed model and the real optimal objective value, we have developed a *squeezing procedure* [34]. This procedure retains the sequence of operations scheduled in each of the machines, while the starting times of the operations are reset such that each operation starts as early as possible (i.e., without violating any constraints).

The number of binary variables and constraints depend on the choice of the time horizon T and the length ℓ of the time steps. Our *iterative procedure* is designed to keep the number of variables and constraints at minimum by iteratively decreasing the length ℓ of the time steps and utilizing the solution from the previous iteration to feed the next iteration with a feasible solution and an appropriate value of T. The iterative procedure [34, Algoritm 1] is generic in the sense that it can be used to solve any time-indexed model.

For iteration s of the iterative procedure, let ℓ^s and T^s denote the length and number of time steps in the planning horizon, respectively, and let the parameter values be scaled and rounded up or truncated, respectively, according to

$$p_{ij} := \left\lceil \frac{\widetilde{p_{ij}}}{\ell^s} \right\rceil ;\ r_{ij} := \left\lceil \frac{\widetilde{r_{ij}}}{\ell^s} \right\rceil ;\ \hat{r}_{ijn} := \left\lfloor \frac{\widetilde{\hat{r}_{ijn}}}{\ell^s} \right\rfloor ;\ \bar{r}_{ijn} := \left\lceil \frac{\widetilde{\bar{r}_{ijn}}}{\ell^s} \right\rceil ; \tag{14a}$$

$$\hat{r}^n := \left\lfloor \frac{\widetilde{\hat{r}^n}}{\ell^s} \right\rfloor ;\ \bar{r}^n := \left\lceil \frac{\widetilde{\bar{r}^n}}{\ell^s} \right\rceil ;\ d_j := \left\lfloor \frac{\widetilde{d_j}}{\ell^s} \right\rfloor ;\ i \in \mathcal{N}_j,\ j \in \mathcal{J},\ n \in \mathcal{P} ; \tag{14b}$$

$$v_{jq} := \left\lceil \frac{\widetilde{v_{jq}}}{\ell^s} \right\rceil ,\ (j,q) \in \mathcal{Q} ;\quad a_k := \left\lceil \frac{\widetilde{a_k}}{\ell^s} \right\rceil ,\ k \in \mathcal{K} ; \tag{14c}$$

$$\tau_{jk} := \left\lceil \frac{\widetilde{\tau_{jk}}}{\ell^s} \right\rceil ,\ j \in \mathcal{J}^{\text{maint}},\quad k \in \mathcal{K} ;\quad \Delta := \left\lfloor \frac{\widetilde{\Delta}}{\ell^s} \right\rfloor , \tag{14d}$$

where the superscript "$\sim$" denotes the respective original (non-discretized) parameter values.

The process of the iterative procedure is defined in Figure 2. In step 1, the input data, denoted `data`^{1}, is generated employing the assignments in (14) for $s = 1$. In the computational tests, we used $\ell^1 := \lceil \widetilde{p_{\max}}/2 \rceil$, where $\widetilde{p_{\max}} := \max_{i \in \mathcal{N}_j, j \in \mathcal{J}} \{\widetilde{p_{ij}}\}$. The value of ℓ^1 is chosen sufficiently large such

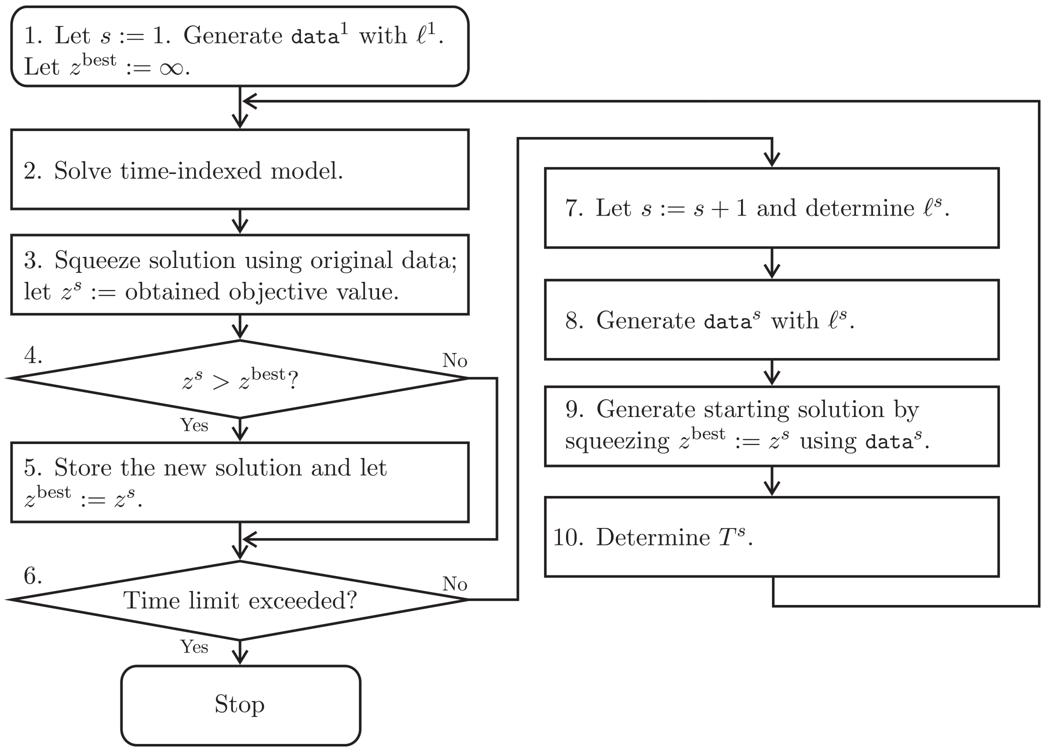

1. Let $s := 1$. Generate $\texttt{data}^1$ with ℓ^1. Let $z^{\text{best}} := \infty$.
2. Solve time-indexed model.
3. Squeeze solution using original data; let $z^s :=$ obtained objective value.
4.
$z^s > z^{\text{best}}$?
No
Yes
5. Store the new solution and let $z^{\text{best}} := z^s$.
6.
Time limit exceeded?
No
Yes
Stop
7. Let $s := s + 1$ and determine ℓ^s.
8. Generate $\texttt{data}^s$ with ℓ^s.
9. Generate starting solution by squeezing $z^{\text{best}} := z^s$ using $\texttt{data}^s$.
10. Determine T^s.

Figure 2: The process of the iterative procedure.

that most of the processing times p_{ij} are valued 1. Hence, most instances can be solved by the time-indexed model in a few seconds, even for large values of T. Consequently, we let $T^1 := \sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{N}_j} p_{ij}$, which theoretically can be too short (if for example the time lags v_{jq} are very long), but in most cases this time horizon is more than sufficient and hence considerably longer than the makespan of an optimal schedule. If this value of T^1 is too small, the problem would be infeasible in the first iteration.

In step 2 in Figure 2, for $s = 1$, the model TI-FM is solved. The reason for solving the problem without considering unmanned night shifts in the first iteration is that it may be impossible to find feasible solutions when the data, generated according to (14) using the large value of ℓ^1, is very coarse. In our previous work [33, p. 35] the value of T^1 was determined by a problem-specific heuristic; the first iteration of the iterative procedure presented here replaces this heuristic.

In step 3, the solution obtained after solving the time-indexed model in step 2 is squeezed and the objective value z^s, obtained through the squeezing with respect to the original data, is stored. In step 4, we check if the value of z^s is the best found so far. If yes, the new solution is stored, and $z^{\text{best}} := z^s$ in step 5. In step 6, we check if the time limit is exceeded. If yes, the procedure is terminated. If no, $s := s + 1$ in step 7 and the value of ℓ^s is determined (the values used in the computational tests are listed in Table 4). In step 8, $\texttt{data}^s$ is generated according to (14). In step 9, the best solution found so far is squeezed with respect to $\texttt{data}^s$. The resulting solution is stored and used as a starting solution for the time-indexed model in iteration s (where $s > 1$) in step 2. Finally in step 10, the value of T^s is determined according to

$$T^{s+1} := \max \left\{ \left\lceil \frac{\widetilde{C_{\max}} + \widetilde{p_{\max}}}{\ell^s} \right\rceil, C_{\max} \right\}, \tag{15}$$

where $C_{\max}$ is the makespan of the starting solution, and $\widetilde{C_{\max}}$ denotes the makespan of the solution found in iteration $s - 1$ and squeezed with respect to the original data. The reason for not letting T^s equal $C_{\max}$ is that the optimal solution in iteration s may have a larger makespan than $C_{\max}$; see Thörnblad [33, pp. 34–36] for a discussion. This is more likely to happen when the difference $\ell^{s-1} - \ell^s$ is small, since then the likelihood that $C_{\max}$ is close to the makespan of the optimal solution in iteration s is large. During our computational testing, we used a time limit as termination criterion, since the main focus of this work is to investigate whether or not this procedure can be of practical use. In Thörnblad et al. [34, 35], we used a certain level of accuracy as the termination criterion, i.e., the computations were terminated when a desired length of the time step was reached.

The squeezing procedure resets all starting times of the operations such that each operation starts as early as possible and without violating any constraints. Even though the constraints (9), regarding the unmanned night shifts, are not considered when the model TI-FM is solved in step 2 of the first iteration, the squeezing procedure is in most cases able to find a feasible schedule for the model TI-FMN that can be used as a starting solution in the next iteration; otherwise the values from the starting solution are retained by the optimization solver, and used for finding the first feasible solution.

In Thörnblad et al. [34], the general FJSP without side constraints was investigated; the coding of the squeezing procedure was then fairly simple. In this article the side constraints (5)–(9) are added—making the coding (in

MATLAB [22]) of the squeezing procedure much more complicated. From these observations we conclude that using the iterative procedure without the squeezing, as done in Thörnblad et al. [35], can be an option worth to consider. To investigate the benefits of the squeezing procedure, we tested the iterative procedure with and without the squeezing. This investigation is discussed in the next section.

7. Computational results

Through the manufacturing site's Enterprise Resource Planning (ERP) system, real data were collected from the multitask cell during 2012 and 2013. Nine out of totally twelve real scenarios, from each of which a set of test instances was created, are identical to the scenarios employed in Thörnblad et al. [35], except for the data describing the unmanned night shifts. At the time, some personnel was working during the night, and in Thörnblad et al. [35] we assumed that the work force was sufficient for the schedules produced. Since the beginning of 2013, the night shifts are, however, unmanned. In this paper, we use the scenarios from 2012 as if they were to be performed using the work force of today, i.e., with no personnel during the night shifts.

In the multitask cell, there are about 30 different types of jobs consisting of 3–7 operations to be performed on eight products. The computations were carried out using AMPL-CPLEX12 [12, 16] on a computer with two 2.66GHz Intel Xeon X5650 processors, each with six cores (24 threads), and a total memory of 48 Gbytes of RAM.

In Section 7.1 the test setting is defined, and in Section 7.2 we report on the results from our computational tests. In the latter section, the schedules obtained by our iterative procedure are also compared with the schedules constructed by the FIFO and the CR priority rule, respectively.

7.1. The test setting

The objective weights (see Section 5.3) used in the computations are equivalent to those used in Thörnblad et al. [35], i.e., $\alpha_j = 1$, $B = 10$ (implying that $\beta_j \in [0, 20]$, $j \in \mathcal{J}$, except for very much delayed outliers, for which $\beta_j > 20$ may hold), $\varepsilon := 0.001$, and $\varepsilon^{\text{maint}} := 0.0001$.

As mentioned in Section 2, we propose a hybrid event-driven policy for the rescheduling. This means that rescheduling is performed periodically with a rolling time horizon, but whenever an urgent event, such as a machine break-down, occurs, a rescheduling is immediately performed. Therefore, the scheduling procedure needs to be carried out within an acceptable time frame, and produce near-optimal schedules for the period chosen. For the case of the multitask cell, the manager of the cell has agreed that a reasonable clock time to spend on the computing of a near-optimal schedule for the coming shift is around 15 minutes. From the twelve real scenarios, instances for the coming shift including all jobs and PM tasks with release dates $\widetilde{r_{1j}} \leq 8$h were created; the sizes of these instances vary between 20 and 33 jobs.

As discussed in Thörnblad et al. [35], the makespan of an optimal schedule for a specific instance may cover a period that is substantially longer than the coming shift. In order to guarantee a schedule of good quality for the subsequent shift, a rescheduling must, however, be performed at the start of this shift; the new instance considered in this rescheduling includes also the jobs that are expected to arrive at the cell during this subsequent shift. Hence, the details of the later part of the schedule is not of practical interest, since the corresponding jobs are likely to be moved when rescheduled at a later point in time. Therefore, we consider the first two night shifts only in each instance, i.e., $\mathcal{P} = \{1, 2\}$.

In Table 3, the variants of the scheduling procedures tested are defined. The lengths, ℓ^s, of the time intervals chosen for each scheduling procedure

Table 3: Test setting. For all tests denoted TI-FMN_x, the model TI-FMN was employed by the iterative procedure in all iterations except for the first where TI-FM was employed. The time limits refer to clock times.

Scheduling procedure	Time limit (s)	Implementation of the iterative procedure
TI-FMN_2h	7200	including squeezing
TI-FMN_15min	900	including squeezing
TI-FMN_no_squeeze_2h	7200	without squeezing
TI-FMN_no_squeeze_15min	900	without squeezing
TI-FM_2h	7200	including squeezing

listed are indicated in Table 4. TI-FMN_15min was computed with all three variants of ℓ^s, i.e., such that $\ell^s := \ell^{s-1}/\zeta$, for all $s \geq 5$ and the scalar parameter $\zeta \in \{1.8, 2, 2.2\}$. The value 1.8 was chosen since good results were achieved using this scalar in Thörnblad et al. [34]. However, for the case of TI-FMN_no_squeeze, only $\zeta = 2$ was chosen since the iterative procedure without the squeezing procedure works best if the quotient ℓ^{s-1}/ℓ^s is integer-valued. Otherwise, if the starting solution is computed through the rounding

Table 4: The values of ℓ^s and the `mipgap` limit, denoted g^s, employed in the computational tests. The value of ζ is used as a suffix for the variants of the scheduling procedures listed in Table 3.

s	g^s	ℓ^s ($\zeta = 1.8$)	ℓ^s ($\zeta = 2.0$)	ℓ^s ($\zeta = 2.2$)
	(%)	(h)	(h)	(h)
1	5	12	12	12
2	2	3	4	4
3	1	1.66667	2	1.81818
4	0.5	1	1	0.82645
$r \geq 5$	$g^{r-1}/2$	ℓ^{r-1}/ζ	ℓ^{r-1}/ζ	ℓ^{r-1}/ζ

of the values of the time step, it often turns out to be infeasible in the next iteration; this will increase the computation time required. For TI-FMN_2h, $\zeta \in \{1.8, 2\}$ was employed, and TI-FM_2h was computed using $\zeta = 1.8$.

The termination criterion used for the iterative procedure is the time limit. The termination criterion employed when solving the time-indexed model in each iteration was either the total time remaining until the time limit or that `mipgap` $\leq g^s$.[2] Since the approximation of the data is less coarse for smaller values of ℓ^s, the value of g^s is also decreased in each iteration; see Table 4.

7.2. Test results

In order to demonstrate the effect of the squeezing procedure we solved the instances for the coming shift using the setting TI-FMN_15min_2.0. As this turned out to yield slightly better result than the setting TI-FMN_15min_1.8 for most of the instances, we also tested the setting TI-FMN_15min_2.2. These results are reported in Table 5 along with those for the setting TI-FMN_no_squeeze_15min, for the twelve instances of the coming shift; see also the graph in Figure 3. Since the results in the last iteration are computed for different values of ℓ^s—and some of the solutions are also squeezed—it is not possible to use the value of `mipgap` in the comparison. Therefore, the numbers reported in Table 5 are relative to the best feasible objective value found within two hours, denoted z^{best2h}, of all the variants of scheduling

[2]The `mipgap` is defined as the relative difference between the best lower bound LB and the best objective value z found. The definition used by CPLEX version 12 is `mipgap` $:= \frac{|z-\text{LB}|}{10^{-10}+|z|} \cdot 100\%$.

Table 5: Computational results for the settings TI-FMN_x, with x ∈ {15min_1.8, 15min_2.0, 15min_2.2, no_squeeze_15min}. The relative differences, `diff`, to z^{best2h} according to (16) are listed for each of the twelve coming shift scenarios. The notation "hybrid" refers to the objective value resulting from squeezing the solution from the setting TI-FMN_no_squeeze_15min. The bold numbers indicate the best result for each scenario.

		TI-FMN_15min			TI-FMN_no_squeeze_15min	
Scenario	$\lvert\mathcal{J}\rvert$	$\zeta=1.8$ (%)	$\zeta=2.0$ (%)	$\zeta=2.2$ (%)	hybrid (%)	$\zeta=2.0$ (%)
1	25	0.32	**0.00**	4.08	**0.00**	4.49
2	21	**0.00**	0.05	0.05	0.05	0.88
3	26	1.51	**0**	0.90	1.26	3.04
4	26	2.19	**0.01**	2.26	**0.01**	1.31
5	30	1.97	**1.95**	2.97	2.04	6.44
6	25	0.74	0	2.31	**-0.09**	3.06
7	33	1.23	**0.47**	0.79	**0.47**	3.69
8	27	**0.38**	0.52	0.39	0.50	3.07
9	30	2.56	2.56	4.57	**1.95**	8.26
10	24	7.19	1.27	9.87	**0.64**	9.29
11	27	11.59	**10.56**	11.23	10.63	25.24
12	20	8.61	1.56	8.03	**0.12**	0.65
Average	26.2	3.19	1.58	3.95	**1.46**	5.78

procedures tested employing this time limit. In this comparison of the test settings of two hours duration, nine out of the twelve best results were found when employing the setting TI-FMN_2h_2.0. For a few of the instances, CPLEX ran out of memory before reaching the time limit of two hours, whence the last solution obtained was used. The relative differences listed in Table 5 are calculated as

$$\texttt{diff} := \frac{z^{\hat{s}} - z^{\text{best2h}}}{z^{\text{best2h}}} \cdot 100\%, \tag{16}$$

where $\hat{s}$ denotes the iteration for which the time limit was reached.

We conclude from the results reported in Table 5 that the $\zeta = 2.0$ setting performs slightly better than $\zeta = 1.8$ and $\zeta = 2.2$. During the work with the computational tests for [34], $\zeta = 1.8$ worked best for the benchmark test instances employed. However, the best choice of ℓ^s for each iteration is dependent on the instance data, and the multitask cell data differs a lot from the benchmark data used in ibid. One of the reasons why $\zeta = 2$ works

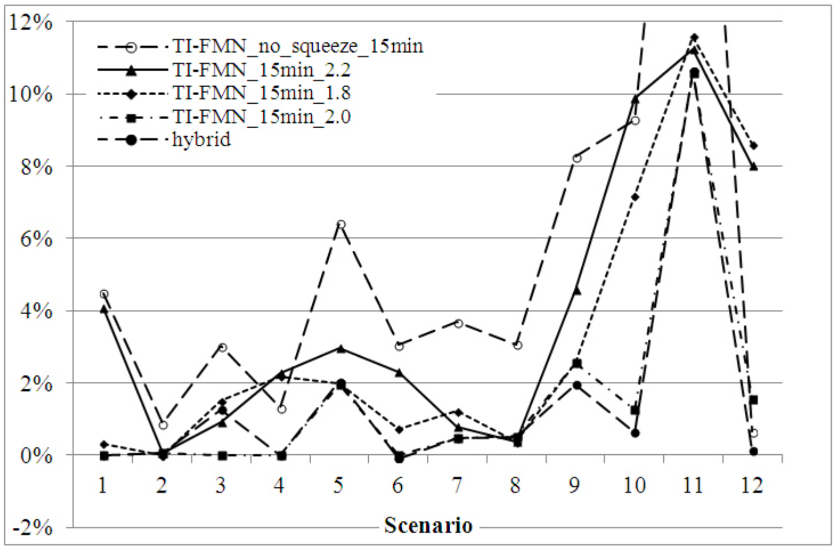


Figure 3: The results reported in Table 5 plotted for all coming shift instances. The vertical axis is truncated at 12%, such that the `diff`-value of 25.24% (for scenario #11) is missing for the setting TI-FMN_no_squeeze_15min.

well is revealed when studying the average of $\ell^{\hat{s}}$ over the twelve scenarios, i.e., the average of the time steps employed at the time limit; see Table 6. Since the lowest average values are found for $\zeta = 2$, the solutions obtained

Table 6: The average length $\ell^{\hat{s}}$ of twelve coming shift instances, where $\hat{s}$ denotes the iteration for which the time limit was reached. An empty space indicates that the corresponding setting was not computed. The best results in each category are written in bold.

Setting	$\zeta = 1.8$	$\zeta = 2.0$	$\zeta = 2.2$
TI-FMN_15min	0.79h	**0.49h**	0.65h
TI-FMN_no_squeeze_15min		**0.59h**	
TI-FMN_2h	0.52h	**0.34h**	
TI-FMN_no_squeeze_2h		**0.34h**	

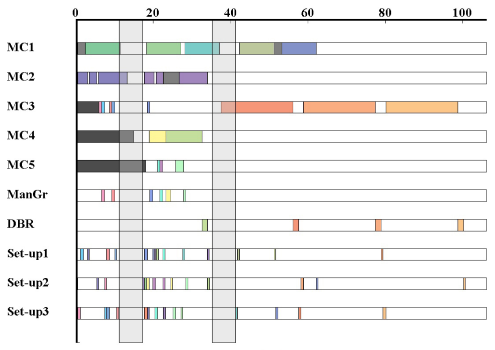
with this setting are closer to the real optimal solution; this is due to the approximation of the data being less coarse. For $\zeta = 2$, an unfavourable data pattern might, however, be reproduced in each iteration. For example, with $\ell^1 = 4$h, $\ell^2 = 2$h, and $\ell^3 = 1$h a processing time of 4.1h, will be rounded up—with big errors—to 2, 3, and 5 time steps, respectively, while 4.0h will be rounded up—with zero errors—to 1, 2, and 4 time steps, respectively.

Table 5 also reveals the effect of the squeezing procedure: The column entitled "hybrid" contains the results obtained by squeezing the solution obtained in the last iteration of TI-FMN_no_squeeze_15min. This hybrid setting yields equally good results as does the setting TI-FMN_15min_2.0, while the value of `diff` is 4% higher on average when no squeezing procedure is employed. In our implementation of the setting TI-FMN_15min, new data is generated in each iteration, but for the setting TI-FMN_no_squeeze_15min, no files need to be generated during the computations other than the result file. The latter is of course more convenient, and therefore, for instances resembling the multitask cell data instances, we recommend the hybrid setting of TI-FMN_no_squeeze_15min. While preparing the results presented in Thörnblad et al. [34], the iterative solution procedure with squeezing in each iteration was, however, observed to be a clear winner. One of the reasons is that the gain of the squeezing procedure is greater for large values of ℓ^s. For example, for the twelve instances reported in Table 5, the average relative difference between the objective values of the squeezed and the non-squeezed solutions obtained in the second iteration, with ℓ^2=4h, is 45%. The benchmark data instances generated by Fattahi et al. [11] and tested in Thörnblad et al. [34] contain operations with very long processing times, which in turn require long planning horizons and therefore also large values of ℓ^s for the first iterations.

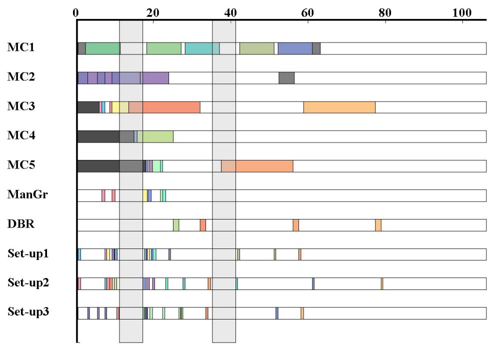
7.2.1. The gain of the night shift constraints

The iterative scheduling procedure was run with the TI-FM_2h_1.8 setting, i.e., without the constraints for unmanned night shifts. In these tests, the solution obtained in each iteration was squeezed—disregarding the night shifts. After the last iteration, which was terminated at the pre-specified time limit, the solution was squeezed taking the night shift constraints (9) into consideration such that a feasible schedule was obtained. Hence, this last solution is eventually not squeezed—but expanded (in the time dimension). The resulting schedule was then compared with the best schedule obtained within two hours of computing by calculating the relative differences `diff` according to (16). The average relative difference over the twelve coming shift instances was 9.90% (ranging between 1.87% and 27.63%). Hence, for the case of the multitask cell, the gain—in terms of the objective value—of including the night shift constraints in the scheduling procedure is around 10%. This figure is of course dependent on the instance data. For operations in the multitask cell, the share of allowed unmanned processing ranges from 0% to 100%. If 100% unmanned processing is allowed for all operations, the night shift constraints are obviously not needed.

In Figure 4, the gain of including the night shift constraints is illustrated for instance #12 comprising 20 jobs. In (a)—without the night shift constraints—the third last operation in machine MC3 cannot be scheduled before the second night shift due to the required manned processing in the later part of this operation. The resulting makespan is 100.5 hours. In the best solution, illustrated in (b), this operation is scheduled to start during the first night shift such that it is completed before the start of the second night shift. The resulting makespan is 79.3 hours. The dark grey bars in the beginning of the planning period represent the parameter a_k, i.e., that the corresponding resources are occupied in the beginning of the planning period.



(a)



(b)

Figure 4: The gain of the night shift constraints illustrated for instance #12. The night shifts are marked by vertical light grey, transparent bars.
(a) The feasible schedule obtained from the solution of the TI-FM_2h_1.8 setting.
(b) The best schedule obtained [the squeezed solution of TI-FMN_no_squeeze_2h (hybrid) in this case].

7.2.2. Comparison with the FIFO and the CR priority dispatching rules

The construction of schedules by the priority dispatching rules was coded such that each possible scheduling occasion was considered from time zero to the planning horizon. At the scheduling occasion, which may be when an operation is completed, the operation currently possessing the highest priority and which is available for scheduling, regarding, for example, precedence and fixture constraints, is scheduled in this resource. As the FIFO rule is static, the priorities remain unchanged throughout the scheduling—in contrary to the CR rule [defined in (2)].

The value of the objective function was computed from the schedules constructed by the dispatching rules and the relative difference to z^{best2h} was computed according to (16). The relative differences between the objective values from the FIFO (CR) schedules and the best solution found (z^{best2h}) were on average 37.4% (41.1%). Remarkably, the CR rule performs worse than the FIFO rule; hence the use of the built-in scheduling method of the control system of the multitask cell is not recommended.

Figure 5 shows four schedules for instance #8 constructed by FIFO (a), CR (b), TI-FMN_15min_2.0 (c), and the best solution found [the squeezed solution of TI-FMN_no_squeeze_2h (hybrid)] (d), with makespans of 91h, 107.1h, 76.1h, and 73.3h, respectively. For this instance the solution with the shortest makespan was also the best solution found (having the lowest objective value), but this is not a general property. For instance #8, the iterative procedure terminated at the pre-specified time limit at iteration $s = 6$ and with $\ell^6 = 0.25$h for both settings TI-FMN_15min_2.0 and TI-FMN_no_squeeze_2h. Note that in the squeezed schedule produced using the setting TI-FMN_no_squeeze_2h [Figure 5(d)] there is no idle time in the machine MC2 during any of the night shifts, while none of the other scheduling procedures manages to completely fill the night shifts in MC2. The relative difference, according to (16), between the TI-FMN_15min_2.0, the FIFO, and the CR schedule and the best solution found, was 0.52%, 58%, and 65%, respectively. The idle times between operations in the four schedules in Figure 5 are due to either the precedence constraints (4b) and (5), the limited fixture availability (7), the night shift constraints (9), or the jobs' release dates (4e).

Figure 6 shows the relative difference, according to (16), between the objective value obtained by each of the scheduling procedures tested and z^{best2h}. The schedules corresponding to the three rightmost columns disregard the night shift constraints (9). It is clear that our proposed scheduling procedures—considering the night shift constraints—are able to find significantly better schedules within an acceptable time frame (15 minutes). If the proposed hybrid event-driven policy discussed in Section 7.1 is applied,—with a rescheduling at the start of each shift—this difference will be even larger,

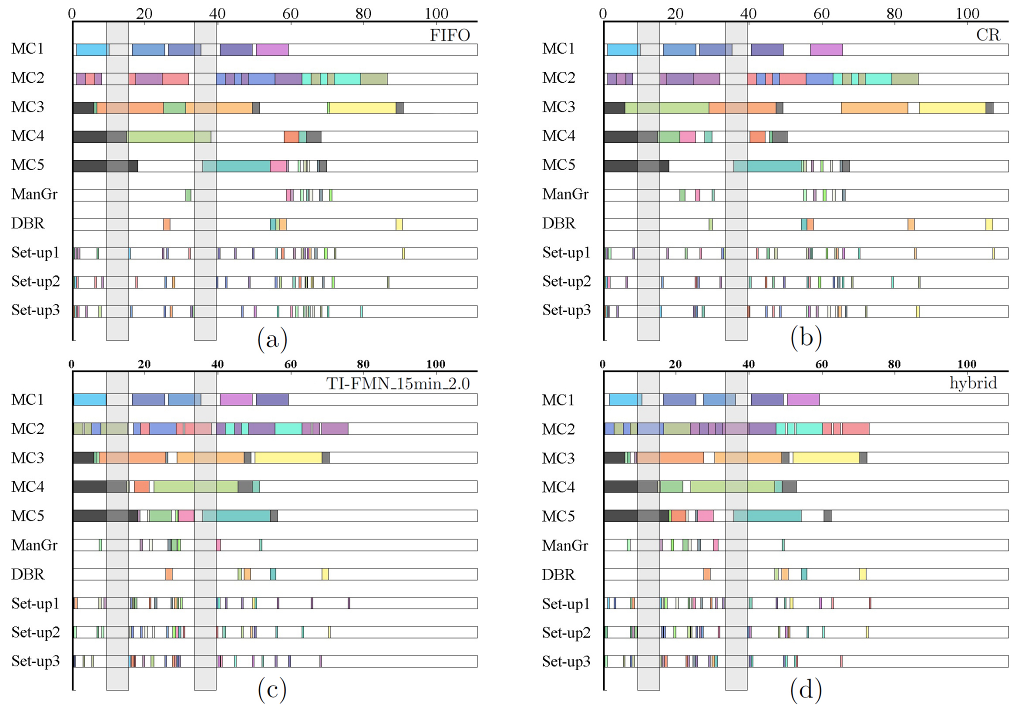


Figure 5: The schedules of instance #8 constructed by (a) FIFO, (b) CR, (c) TI-FMN_15min_2.0, and (d) the best solution found [the squeezed solution of TI-FMN_no_squeeze_2h (hybrid)]. The night shifts are marked by vertical light grey, transparent bars. The grey jobs close to the end of the schedules represent PM tasks.

since then the machines will be available for processing (via the parameter a_k) at times such that the coming unmanned shift can be even better utilized (by the jobs that were scheduled but not processed during the previous shift).

In contrary to the computations performed in Thörnblad et al. [35], where the iterative procedure was terminated when a pre-specified value of ℓ^s was reached, here we let the iterative procedure run until a pre-specified time limit is reached. In previous work employing this procedure [34, 35], we have not considered any night shift constraints; such constraints may have a big impact on a job's completion time, depending on whether the required manual processing of an operation can be scheduled before or after a night shift. In [34] we noted that a near-optimal solution was often found during the first iterations of the iterative procedure. Due to the added night shift constraints, this has not been the case during the work with the result presented

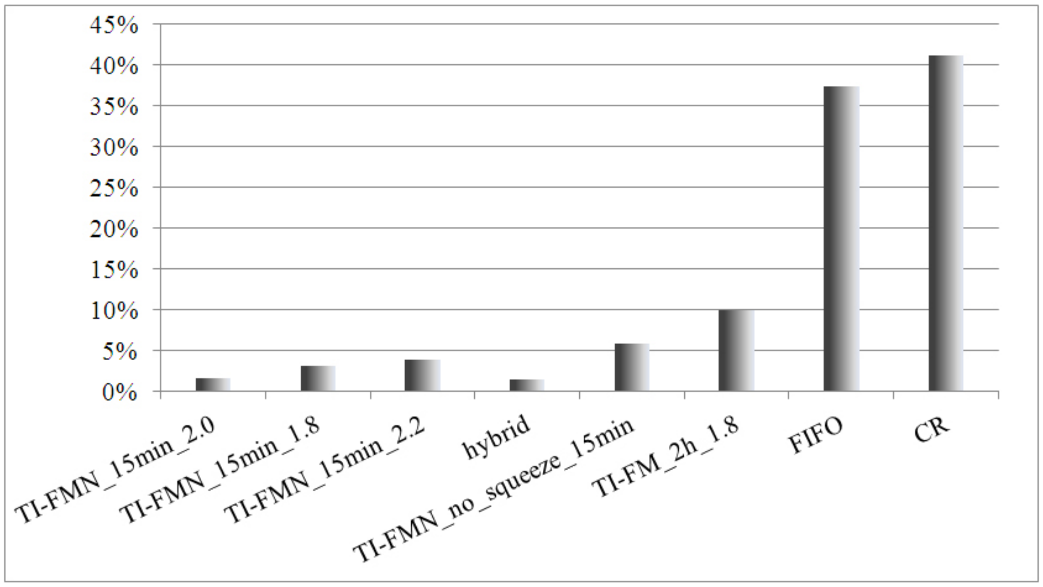


Figure 6: The relative differences, according to (16), between the objective values obtained by the different scheduling procedures and z^{best2h}.

in this article: here, in most of the iterations an improved feasible solution has been achieved. Based on the results presented in Table 5, in which half of the objective values produced by the settings TI-FMN_15min_2.0 and TI-FMN_no_squeeze_2h differ less than 0.5% from z^{best2h}, we conclude that our proposed scheduling method is able to produce near-optimal schedules for the coming shift in the multi-task cell.

8. Conclusions

We present a generic iterative procedure for the solution of time-indexed formulations. We have applied this procedure to a real flexible job shop scheduling problem originating from a production cell at GKN Aerospace Engine Systems, Sweden. A time-indexed formulation of the problem is presented including side constraints regarding preventive maintenance, fixture availability, and unmanned night shifts. We have described a (non-generic) squeezing procedure, which can be used in either each or just the last iteration of the iterative procedure. The squeezing procedure reduces the difference between the objective value of the approximate solution obtained from the time-indexed model and the sought real optimal objective value.

Computational results show that the gain of including the night shift constraints is around 10%, in terms of objective values. This gain would be even greater if the proposed scheduling principle with a hybrid event-driven policy is applied, such that the chance of fully utilizing the first unmanned night shift is increased when rescheduling is performed periodically.

The proposed scheduling procedure has been compared with two priority dispatching rules: the static FIFO (first–in, first–out) and a dynamic CR (critical ratio) rule. The choice of comparison with these two rules is based on the facts that FIFO is currently in use in most workshops in the factory and that the CR rule is an existing built-in scheduling method in the control system of the production cell specifically studied in this article. The objective values obtained after running our iterative procedure for 15 minutes were on average only 0.71 times the corresponding values computed by the FIFO and CR rules. Hence, there is a large potential of replacing these simple priority dispatching rules by a more sophisticated scheduling principle, such as our time-indexed mathematical optimization model implemented in our iterative procedure. We conclude that this proposed iterative procedure is able to produce near-optimal schedules for industrial data instances for the coming shift within an acceptable time frame.

Acknowledgements

This research was financially supported by GKN Aerospace, The Swedish Research Council, NFFP (Swedish National Aeronautics Research Programme), and VINNOVA (through Chalmers Transport Area of Advance).

References

[1] S. Azem, R. Aggoune, S. Dauzere-Peres, Disjunctive and time-indexed formulations for non-preemptive job shop scheduling with resource availability constraints, in: 2007 IEEE International Conference on Industrial Engineering and Engineering Management, pp. 787–791.

[2] J. Van den Bergh, J. Belin, P. De Bruecker, E. Demeulemeester, L. De Boeck, Personnel scheduling: A literature review, European Journal of Operational Research 226 (2013) 367–385.

[3] J.H. Blackstone, Jr., D.T. Phillips, G.L. Hogg, A state-of-the-art survey of dispatching rules for manufacturing job shop operations, International Journal of Production Research 20 (1982) 27–45.

[4] E.H. Bowman, The schedule-sequencing problem, Operations Research 7 (1959) 621–624.

[5] P. Brucker, The job-shop problem: Old and new challenges, in: P. Baptiste, G. Kendall, A. Munier-Kordon, F. Sourd (Eds.), Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2007), Paris, France, pp. 15–22.

[6] P. Brucker, B. Jurisch, A. Krämer, Complexity of scheduling problems with multi-purpose machines, Annals of Operations Research 70 (1997) 57–73.

[7] P. Brucker, S. Knust, Complex Scheduling, 2nd ed., Springer-Verlag, Berlin, Germany, 2012.

[8] R. Cheng, M. Gen, Y. Tsujimura, A tutorial survey of job-shop scheduling problems using genetic algorithms–I. representation, Computers & Industrial Engineering 30 (1996) 983–997.

[9] Y. Demir, S.K. İşleyen, Evaluation of mathematical models for flexible job-shop scheduling problems, Applied Mathematical Modelling 37 (2013) 977–988.

[10] M.E. Dyer, L.A. Wolsey, Formulating the single machine sequencing problem with release dates as a mixed integer program, Discrete Applied Mathematics 26 (1990) 255–270.

[11] P. Fattahi, M. Saidi Mehrabad, F. Jolai, Mathematical modeling and heuristic approaches to flexible job shop scheduling problems, Journal of Intelligent Manufacturing 18 (2007) 331–342.

[12] R. Fourer, D.M. Gay, B.W. Kernighan, AMPL: A Modeling Language for Mathematical Programming, 2$^{\text{nd}}$ ed., Brooks/Cole Publishing Company/Cengage Learning, 2002.

[13] J. Gao, M. Gen, L. Sun, Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm, Journal of Intelligent Manufacturing 17 (2006) 493–507.

[14] R. Haupt, A survey of priority rule-based scheduling, Operations-Research-Spektrum 11 (1989) 3–16.

[15] W.J. Hopp, M.L. Spearman, Factory Physics, 3^{d} ed., McGraw-Hill/Irwin, New York, NY, USA, 2008.

[16] IBM Corp., IBM ILOG CPLEX V12.1 User's Manual for CPLEX, 2009.

[17] A.S. Jain, S. Meeran, Deterministic job-shop scheduling: Past, present and future, European Journal of Operational Research 113 (1999) 390–434.

[18] T. Jansson, Resource utilization in a multitask cell, Master's Thesis, Department of Mathematical Sciences, Chalmers University of Technology, Göteborg, Sweden, 2006.

[19] S. Kedad-Sidhoum, Y. Rios-Solis, F. Sourd, Lower bounds for the earliness–tardiness scheduling problem on single and parallel machines, European Journal of Operational Research 189 (2008) 1305–1316.

[20] G.B. Mainieri, D.P. Ronconi, New heuristics for total tardiness minimization in a flexible flowshop, Optimization Letters 7 (2013) 665–684.

[21] A.S. Manne, On the job-shop scheduling problem, Operations Research 8 (1960) 219–223.

[22] MATLAB, Release 2011b, The MathWorks Inc., Natick, MA, United States, 2011.

[23] D. Ouelhadj, S. Petrovic, A survey of dynamic scheduling in manufacturing systems, Journal of Scheduling 12 (2009) 417–431.

[24] C. Özgüven, L. Özbakir, Y. Yavuz, Mathematical models for job-shop scheduling problems with routing and process plan flexibility, Applied Mathematical Modelling 34 (2010) 1539–1548.

[25] C.-H. Pan, A study of integer programming formulations for scheduling problems, International Journal of Systems Science 28 (1997) 33–41.

[26] M.L. Pinedo, Scheduling: Theory, Algorithms, and Systems, 4th ed., Springer, New York, NY, USA, 2010.

[27] S. Rahimifard, S.T. Newman, Simultaneous scheduling of workpieces, fixtures and cutting tools within flexible machining cells, International Journal of Production Research 35 (1997) 2379–2396.

[28] M. Rajkumar, P. Asokan, V. Vamsikrishna, A GRASP algorithm for flexible job-shop scheduling with maintenance constraints, International Journal of Production Research 48 (2010) 6821–6836.

[29] J. Slomp, W.H.M. Zijm, A manufacturing planning and control system for a flexible manufacturing system, Robotics and Computer-Integrated Manufacturing 10 (1993) 109–114.

[30] J.P. Sousa, L.A. Wolsey, A time indexed formulation of non-preemptive single machine scheduling problems, Mathematical Programming 54 (1992) 353–367.

[31] H. Stefansson, S. Sigmarsdottir, P. Jensson, N. Shah, Discrete and continuous time representations and mathematical models for large production scheduling problems: A case study from the pharmaceutical industry, European Journal of Operational Research (2011) 383–392.

[32] A. Syberfeldt, I. Karlsson, A. Ng, J. Svantesson, T. Almgren, A web-based platform for the simulation-optimization of industrial problems, Computers & Industrial Engineering 64 (2013) 987–998.

[33] K. Thörnblad, On the Optimization of Schedules of a Multitask Production Cell, Licentiate Thesis, Chalmers University of Technology and University of Gothenburg, Göteborg, Sweden, 2011.

[34] K. Thörnblad, A.-B. Strömberg, M. Patriksson, T. Almgren, A competitive iterative procedure using a time-indexed model for solving flexible job shop scheduling problems, Available at `http://www.optimization-online.org/DB_HTML/2013/08/3991.html`, 2013.

[35] K. Thörnblad, A.-B. Strömberg, M. Patriksson, T. Almgren, Scheduling optimization of a real flexible job shop including fixture availability and preventive maintenance, Revised for publication in European Journal of Industrial Engineering (2013).

[36] H.M. Wagner, An integer linear-programming model for machine scheduling, Naval Research Logistics Quarterly 6 (1959) 131–140.

[37] S. Wang, J. Yu, An effective heuristic for flexible job-shop scheduling problem with maintenance activities, Computers & Industrial Engineering 59 (2010) 436–447.