A competitive iterative procedure using a time-indexed model for solving flexible job shop scheduling problems

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, Dep. of Logistics Development, SE-461 81 Trollhättan, Sweden*

Abstract

We investigate the efficiency of a discretization procedure utilizing a time-indexed mathematical optimization model for finding accurate solutions to flexible job shop scheduling problems considering objectives comprising the makespan and the tardiness of jobs, respectively. The time-indexed model is used to find solutions to these problems by iteratively employing time steps of decreasing length. The solutions and computation times are compared with results from a known benchmark formulation and an alternative, slightly enhanced version of the same. For the largest instances—considering both objectives—the proposed method finds significantly better solutions than the other models within the same time frame, although there is a large difference in the performance of the models depending on which objective is considered. This implies that the evaluation of scheduling algorithms must be performed with respect to an objective that is suitable for the real application for which they are intended. The minimization of the makespan is no such objective, although it is the most widely used objective in research. We propose an objective incorporating tardiness. The iterative procedure for solving the time-indexed model outperforms the other models regarding the time to find the best feasible solution. We conclude that our iterative procedure with the time-indexed model is competitive with state-of-the-art mathematical optimization models. Since the proposed procedure quickly finds solutions of good quality to large instances, our findings imply that the new procedure is beneficially utilized for scheduling real flexible job shops.

Keywords: Flexible job shop scheduling, Time-indexed formulation, Mixed integer linear programming (MILP), Discretization procedure, Benchmark, Minimize makespan, Tardiness

1. Introduction

The job shop scheduling problem is defined as that to find the optimal sequences of a given set of jobs on a given set of machines. Each job consists of a number of operations which must be processed in a given order; this is modeled by so-called *precedence constraints*. Associated with each operation is a machine and a processing time. The flexible job shop problem (FJSP) is an extension of the job shop problem in which each operation may be scheduled in more than one of the machines (Brucker and Knust, 2012, Chapter 4).

The purpose of this article is to investigate the competitiveness of an iterative discretization procedure utilizing a time-indexed mixed integer linear programming (MILP) model in finding accurate solutions to flexible job shop scheduling problems. The MILP models include both binary and continuous variables, and all the relations between the variables in the objective and constraints are linear; see Nemhauser and Wolsey (1988). Our iterative solution procedure is compared with a benchmark model presented by Özgüven et al. (2010), which yielded the best results in the evaluation by Demir and İşleyen (2013). In the comparison we have also included a similar alternative model developed during the work with this article.

2. Related Work

In Manne (1960), the problem of sequencing jobs with precedence constraints on a single machine is studied. The jobs' starting times are represented by continuous variables, and the decision variables are defined as y_{jq} equals 1, if job j precedes job q, and 0 otherwise. In the operations research literature, there are many examples of models for job shops and flexible job shops employing this type of variables; see, e.g., Özgüven et al. (2010) and Low et al. (2006).

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, Dep. of Logistics Development, 9510KT, SE-461 81 Trollhättan, Sweden. Tel. +46 520 29 22 66.

An alternative means to formulating a MILP model for the flexible job shop problem is to utilize discrete time-indexed variables. The planning period is then divided into a number of time steps of equal length. The decision variables in the time-indexed model are valued 1 if the corresponding operation is scheduled to start at the beginning of a specific time step in a specific resource, and 0 otherwise. The resulting formulation results in very large models in terms of numbers of both variables and constraints, but it typically yields better optimistic estimates of the optimal objective value (see Section 3.2) than other MILP formulations of scheduling problems; see van den Akker et al. (2000). In Berghman (2012), a time-indexed formulation outperformed three other MILP models for the problem of parallel machine scheduling when the objective was to minimize a total weighted sum of the completion times. We obtained a similar result for a special case of the FJSP in a real production cell with the objective to minimize a weighted sum of the completion times and the tardiness (Thörnblad, 2011).

The objective that is the most often utilized for scheduling problems is the minimization of the makespan (Jain and Meeran, 1999). Other common objectives are related to the jobs' earliness/tardiness and completion times, and/or inventory holding costs associated with the jobs. Out of the 22 articles listed by Demir and İşleyen (2013), which presents mathematical models (some of the models being non-linear) concerning the FJSP, 16 considered the objective of minimizing the makespan; only ten considered other objectives, whereof five involving due dates. See Section 4 for a discussion of the makespan objective and an objective including tardiness, and their respective suitability for real applications.

Besides MILP, there are many methods devoted to finding good feasible solutions to scheduling problems. Constraint programming (CP) is an exact method which seems to yield good results, see, e.g., Sadykov and Wolsey (2006) for an evaluation of MILP and CP models for solving a multimachine assignment scheduling problem. There are many metaheuristics proposed for flexible job shop scheduling problems, such as simulated annealing, tabu search, and genetic algorithms amongst others, and combinations of these; see, e.g., Wang et al. (2012), Al-Hinai and ElMekkawy (2011), and Baykasoglu and Özbakir (2010). In Section 6, we compare the makespan found by our models with those found by Behnke and Geiger (2012) and Bagheri et al. (2010), who employ CP and an artificial immune algorithm (AIA), respectively.

3. Mathematical formulations

3.1. Indices, sets, and parameters

The notation used throughout this article is as follows:

Sets

$\mathcal{J}$	the set of jobs; $j \in \mathcal{J} := \{1, \ldots, n\}$
$\mathcal{N}_j$	the set of operations; $i \in \mathcal{N}_j := \{1, \ldots, n_j\}$
$\mathcal{K}$	the set of resources; $k \in \mathcal{K} := \{1, \ldots, m\}$
$\mathcal{M}_{ij}$	the set of resources allowed for operation i of job j ($\mathcal{M}_{ij} \subseteq \mathcal{K}$)
$\mathcal{T}$	the set of time steps; $u \in \mathcal{T} := \{0, \ldots, T\}$

Parameters

p_{ijk}	the processing time of operation i of job j in resource k
r_{ij}	the release date of operation i of job j, i.e., its earliest possible starting time
δ_{ij}	the shortest possible remaining time from the starting time of operation i of job j to the completion of job j
ℓ	the length of the time steps in the time-indexed model
M	a big number, at least as large as the makespan of the solution to be computed

In all test instances considered in this article, the parts to be processed are assumed to be present in the job shop at the beginning of time step 0, i.e., the job release dates are $r_{1j} = 0$. Due to the precedence relations between the operations within a job, no operation may be scheduled before the completion of the previous operation. Therefore, and since the processing time is resource dependent, the release dates are defined as $r_{ij} := r_{i-1,j} + \min_{k \in \mathcal{M}_{i-1,j}}\{p_{i-1,j,k}\}$, for $i = 2, \ldots, n_j,\ j \in \mathcal{J}$. Similarly, the shortest possible remaining time from the start of operation i to the completion of job j is defined as $\delta_{ij} := \delta_{i+1,j} + \min_{k \in \mathcal{M}_{ij}}\{p_{ijk}\}$, for $i = n_j - 1, \ldots, 1,\ j \in \mathcal{J}$, and $\delta_{n_j j} := \min_{k \in \mathcal{M}_{n_j j}}\{p_{n_j jk}\}$.

3.2. Time-indexed model

The planning horizon is divided into T intervals (i.e., time steps), each of length $\ell > 0$. The value of the parameter T has to be large enough such that an optimal schedule is contained within the time horizon $[0, T\ell]$. Our time-indexed model is expressed in terms of the variables x_{ijku}, which are valued 1 if operation i of job j is scheduled to start processing in resource k at the start of time interval u, and 0 otherwise. Throughout the article, for any $z \in \mathbb{R}$ we

define $(z)_+ := \max\{z, 0\}$. We first consider the objective of minimizing the makespan of the schedule, represented by the variable $C_{\max} \in \mathbb{R}$; in Section 4 we present an alternative objective based on the total (weighted) tardiness. The model is thus to

$$\text{minimize} \qquad C_{\max} \tag{1a}$$

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

$$\sum_{k\in\mathcal{K}\setminus\mathcal{M}_{ij}} \sum_{u\in\mathcal{T}} x_{ijku} = 0, \qquad i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{1c}$$

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

$$\sum_{k\in\mathcal{M}_{ij}} \sum_{\mu=r_{ij}}^{u-p_{ijk}} x_{ijk\mu} - \sum_{l\in\mathcal{M}_{i+1,j}} \sum_{\nu=r_{i+1,j}}^{u} x_{i+1,jl\nu} \ge 0, \qquad \begin{array}{l} u = r_{i+1,j}, \ldots, T-\delta_{i+1,j}, \\ i = 1, \ldots, n_j - 1,\ j \in \mathcal{J}, \end{array} \tag{1e}$$

$$\sum_{k\in M_{n_j j}} \sum_{u\in\mathcal{T}} (u + p_{n_j jk}) x_{n_j jku} \le C_{\max}, \quad j \in \mathcal{J}, \tag{1f}$$

$$x_{ijku} = 0, \qquad \begin{array}{l} u \in \mathcal{T} \setminus \{r_{ij}, \ldots, T-\delta_{ij}\}, \\ k \in \mathcal{M}_{ij},\ i \in \mathcal{N}_j,\ j \in \mathcal{J}, \end{array} \tag{1g}$$

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

The constraints (1b) ensure that each operation i of job j is scheduled to be processed exactly once in an allowed resource. The constraints (1c) 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 constraints are redundant, but they are included since we discovered that the solver (AMPL-CPLEX12 (Fourer et al., 2002; IBM Corp., 2009)) was able to parallelize the computations such that they run faster (w.r.t. clocktime) when these constraints were included. The constraints (1d) ensure that at most one operation at a time is scheduled in each resource. The precedence constraints (1e) make sure that no operation starts processing before the preceding operation of the same job is completed. The makespan of the schedule is determined by

the constraints (1f). The constraints (1g) ensure that operation i of job j is scheduled neither before its release date nor such that it (or any succeeding operations) would not be completed by the end of the planning horizon, i.e., at time the $T\ell$; these constraints are redundant if all the jobs are ready to be processed at the time 0, but their inclusion reduces the number of variables in the model. The model (1) will henceforth be referred to as TI-Cmax.

The number of precedence constraints (1e) is in the order of the total number of operations times the total number of time steps. If an instance of the model is too large, such that the time to solve its *linear programming (LP) relaxation*[2] is too long for practical purposes, then an alternative set of precedence constraints may be used (its number being in the order of the total number of operations):

$$\sum_{k\in M_{ij}} \sum_{\mu=r_{ij}}^{T-\delta_{ij}} (\mu + p_{ijk}) x_{ijk\mu} \leq \sum_{l\in M_{i+1,j}} \sum_{\nu=r_{i+1,j}}^{T-\delta_{i+1,j}} \nu x_{i+1,jl\nu},\ i=1,\ldots,n_j-1,\ j\in\mathcal{J}. \quad (2)$$

The LP relaxation of TI-Cmax is a model with the same variables, objective, and constraints, except for the integrality constraints (1h), which are substituted by $0 \leq x_{ijku} \leq 1$. The optimal objective value of the LP relaxation of a model is called its *LP bound*, being a lower bound on the optimal objective value of the original model including integrality constraints. Provided that the integrality constraints (1h) are fulfilled, the constraints (2) are equivalent to (1e); they do, however, not yield as tight LP bounds as do the constraints (1e). The model (1a)–(1d), (2), (1f)–(1h) will henceforth be referred to as TI-prec-Cmax. In Section 3.3 we present the results for one test instance regarding the differences in the LP bounds of the models TI-Cmax and TI-prec-Cmax.

3.3. An alternative model

We next present an alternative model that we have developed, employing the same type of variables as the benchmark model proposed by Özgüven et al. (2010). Our model contains fewer variables and constraints than the

[2]The LP relaxation of a MILP model is a model comprising the same variables, objective, and constraints except for the integrality constraints which are substituted by the respective lower and upper bounds of the integer variables; see (Nemhauser and Wolsey, 1988, Chapter II.3, Section 1).

benchmark model, since the variables representing the completion times of operations and jobs employed in the latter model are redundant. The variables used in our alternative model are

$$z_{ijk} = \begin{cases} 1, \text{if operation } i \text{ of job } j \text{ is processed on resource } k, \\ 0, \text{otherwise,} \end{cases}$$

$$y_{ijpqk} = \begin{cases} 1, \text{if operation } i \text{ of job } j \text{ precedes operation } p \text{ of job } q \text{ in} \\ \quad \text{resource } k, \\ 0, \text{otherwise,} \end{cases}$$

$t_{ijk} =$ the starting time of operation i of job j in resource k, and

$C_{\max} =$ the completion time for the last job (makespan).

Our alternative model, Alt-Cmax, is then formulated as that to

$$\text{minimize} \qquad C_{\max}, \tag{3a}$$

$$\text{subject to} \qquad \sum_{k \in \mathcal{M}_{ij}} z_{ijk} = 1, \qquad i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{3b}$$

$$t_{ijk} - M z_{ijk} \leq 0, \qquad k \in \mathcal{M}_{ij},\ i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{3c}$$

$$t_{pqk} + p_{pqk} z_{pqk} - t_{ijk} - M y_{ijpqk} \leq 0, \qquad k \in \mathcal{M}_{ij} \cap \mathcal{M}_{pq},\ i \in \mathcal{N}_j, \tag{3d}$$
$$p \in \mathcal{N}_q,\ j, q \in \mathcal{J},\ j < q,$$

$$t_{ijk} + p_{ijk} z_{ijk} - t_{pqk} - M(1 - y_{ijpqk}) \leq 0, \qquad k \in \mathcal{M}_{ij} \cap \mathcal{M}_{pq},\ i \in \mathcal{N}_j, \tag{3e}$$
$$p \in \mathcal{N}_q,\ j, q \in \mathcal{J},\ j < q,$$

$$\sum_{k \in \mathcal{M}_{ij}} (t_{ijk} + p_{ijk} z_{ijk}) - \sum_{k \in \mathcal{M}_{i+1,j}} t_{i+1,jk} \leq 0, \qquad i = 1, \ldots, n_j - 1,\ j \in \mathcal{J}, \tag{3f}$$

$$\sum_{k \in \mathcal{M}_{n_j j}} \left(t_{n_j jk} + p_{n_j jk} z_{n_j jk} \right) \leq C_{\max}, \quad j \in \mathcal{J}, \tag{3g}$$

$$t_{ijk} \geq 0, \qquad k \in \mathcal{M}_{ij},\ i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{3h}$$

$$z_{ijk} \in \{0,1\}, \quad k \in \mathcal{M}_{ij},\ i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{3i}$$

$$y_{ijpqk} \in \{0,1\}, k \in \mathcal{M}_{ij} \cap \mathcal{M}_{pq},\ i \in \mathcal{N}_j, \tag{3j}$$
$$p \in \mathcal{N}_q,\ j, q \in \mathcal{J},\ j < q.$$

The constraints (3b) ensure that each operation is scheduled in exactly one resource. If an operation is *not* scheduled in a resource k, then its starting time in this resource is set to zero by the constraints (3c). The constraints (3d) and (3e) make sure that no two operations are processed at the same time in any common allowed resource. The constraints (3f) ensure that the precedence relations between the operations of a job are not violated. The constraints (3g) determine the makespan of the schedule, and the constraints (3h)–(3j) are the nonnegativity and binary constraints on the variables.

In Table 1 the optimal objective values and the LP bounds of the models TI-Cmax, TI-prec-Cmax, and Alt-Cmax for the benchmark instance *mfjs7* (Fattahi et al., 2007), are listed. The significance of the tightness of the precedence constraints (1e) compared to (2) is clear for this instance, but the price of the higher LP bound is paid for by the much longer computation time.

Table 1: The optimal objective values, the LP bounds, and the corresponding computation times of the models TI-Cmax, TI-prec-Cmax, and Alt-Cmax for the benchmark test instance *mfjs7* (Fattahi et al., 2007).

Model	Optimal objective value	LP bound	Time to solve LP relaxation (CPU-s)
TI-Cmax	879	773.43	1430
TI-prec-Cmax	879	765.73	80
Alt-Cmax	879	764.00	0.01

3.4. Relations between the two models

The models TI-Cmax and Alt-Cmax are equivalent for the case when all instance data are multiples of the discretization interval ℓ, in the sense that they define equivalent sets of feasible and optimal solutions, respectively. The variables in Alt-Cmax and TI-Cmax are related according to

$$z_{ijk} = \sum_{u \in \mathcal{T}} x_{ijku}, \quad t_{ijk} = \sum_{u \in \mathcal{T}} u x_{ijku}, \qquad k \in \mathcal{M}_{ij},\ i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{4a}$$

$$y_{ijpqk} = \begin{cases} 1, & \text{if } 0 < \sum_{u \in \mathcal{T}} u x_{ijku} < \sum_{u \in \mathcal{T}} u x_{pqku}, \\ 0, & \text{otherwise,} \end{cases} \qquad \begin{array}{l} k \in \mathcal{M}_{ij} \cap \mathcal{M}_{pq}, \\ i \in \mathcal{N}_j,\ p \in \mathcal{N}_q, \\ j, q \in \mathcal{J},\ j < q, \end{array} \tag{4b}$$

$$x_{ijku} = \begin{cases} z_{ijk}, & \text{if } u = t_{ijk}, \\ 0, & \text{if } u \in \mathcal{T} \setminus \{t_{ijk}\}, \end{cases} \qquad i \in \mathcal{N}_j,\ j \in \mathcal{J},\ k \in \mathcal{K}. \tag{4c}$$

The constraints (1b) are, through the definitions in (4), equivalent to the constraints (3b). Similarly, the constraints (1d) correspond to (3d)–(3e). As stated in Section 3.2, the precedence constraints (1e) are equivalent (in a MILP sense) to (2), which are equivalent to (3f). The constraints (3c) define the variables t_{ijk}.

4. Tardiness versus makespan

The objective most often considered in studies of (flexible) job shop problems is the minimization of the makespan. According to the survey by Jain and Meeran (1999), the reason is that this criterion was the first objective applied by researchers studying scheduling problems in the early 1950s and that it is easily modelled. In previous work (Thörnblad et al., 2013) we studied a real flexible job shop being a part of a longer supply chain, in which a variety of aerospace components are processed. In such a context, the production must be predictable and according to plan, since subsequent operations—as well as customers—normally require incoming material at a planned and steady pace in order to be efficient. An objective that strives to minimize the tardiness with respect to the due dates (d_j) for the respective jobs, will thus serve to control the flow and stabilize its pace. This setting is not specific for the flexible job shop studied in this article, but is present in many job shops in the manufacturing industry. The objective function (5) proposed below, targeting mainly the due dates, will also enable shorter production lead times; this is due to the fact that if all jobs can be processed in due time this objective equals the minimization of the weighted sum of the completion times. However, if the scheduling procedure is capable of repeatedly scheduling the job shop with no tardy jobs, then the planner has the opportunity of setting more challenging due dates, which in turn will shorten also the planned production lead times. If, on the other hand, the objective function targets the makespan, there is a risk that the scheduling algorithm exacerbates an already unreliable flow—instead of performing the opposite.

In Thörnblad (2011) we studied an objective whose main focus is the minimization of the total weighted tardiness. Since instances with no tardy jobs may very well appear in the real world, we chose to include the sum of

the completion times in the objective, in order to produce good schedules also for these scenarios. The objective was hence formulated as that to

$$\text{minimize} \sum_{j \in \mathcal{J}} \left(\alpha_j C_j + \beta_j T_j \right), \tag{5}$$

where C_j and $T_j = (C_j - d_j)_+$ denote the completion time and the tardiness of a job, respectively, and α_j and β_j are positive weight parameters. Note that the makespan and the completion times are related through $C_{\max} := \max_{j \in \mathcal{J}} \{C_j\}$. The models TI-Cmax and Alt-Cmax need to be altered in order to consider the objective (5). Since the completion time of a job in the model TI-Cmax can be expressed as

$$C_j = \sum_{k \in \mathcal{K}} \sum_{u \in \mathcal{T}} \left(u + p_{n_j jk} \right) x_{n_j jku}, \tag{6}$$

the objective (5) can be rewritten for the model TI-Cmax as that to

$$\text{minimize} \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} \sum_{u \in \mathcal{T}} \left(\alpha_j (u + p_{n_j jk}) + \beta_j (u + p_{n_j jk} - d_j)_+ \right) x_{n_j jku}. \tag{7}$$

Note that the operator $(\,)_+$ in this objective is applied only to parameters, and hence the objective remains linear. As the release dates are already considered in the model TI-Cmax, no other changes than removing the constraints (1f), which define the makespan, are needed in this model in order to consider the objective (7). The model (7), (1b)–(1e), (1g)–(1h) will henceforth be referred to as the model TI-Tard.

In order to prioritize the jobs that are the most delayed in the schedule resulting from our mathematical model, we define the tardiness weights, β_j, according to

$$\beta_j := B \left(1 - \frac{d_j}{\max_{j \in \mathcal{J}} \{|d_j|\}} \right)_+, \qquad j \in \mathcal{J}, \tag{8}$$

where $B > 0$ denotes the weight employed for the jobs having due date 0, such that $0 \leq \beta_j \leq 2B$ hold for all $j \in \mathcal{J}$. In this way, the jobs that are the most delayed are assigned the highest tardiness weights. Since the main objective is to minimize the tardiness, the objective weights, α_j, for the completion times must be chosen such that $0 \leq \alpha_j \ll B$, $j \in \mathcal{J}$, hold.

In order to adjust Alt-Cmax to consider the tardiness objective with a preserved linearity, the tardiness variables T_j need to be included. The objective can then be formulated as that to

$$\text{minimize} \sum_{j \in \mathcal{J}} \left(\alpha_j \sum_{k \in \mathcal{M}_{n_j j}} \left(t_{n_j j k} + p_{n_j j k} z_{n_j j k} \right) + \beta_j T_j \right). \tag{9}$$

The following constraints are added to the model Alt-Cmax in order to include the release dates and to define the tardiness objective function:

$$r_{ij} z_{ijk} \leq t_{ijk}, \quad k \in \mathcal{M}_{ij},\ i \in \mathcal{N}_j,\ j \in \mathcal{J}, \tag{10a}$$

$$\sum_{k \in \mathcal{M}_{n_j j}} \left(t_{n_j j k} + p_{n_j j k} z_{n_j j k} \right) - d_j \leq T_j, \quad j \in \mathcal{J}, \tag{10b}$$

$$T_j \geq 0, \quad j \in \mathcal{J}. \tag{10c}$$

The alternative model with the proposed tardiness objective is thus formulated as that to minimize (9) subject to (3b)–(3f), (3h)–(3j), (10), and will henceforth be referred to as the model Alt-Tard.

5. The iterative solution procedure

The major disadvantage of the time-indexed model is that the size of the model grows fast with the number of time steps. We have therefore developed a solution procedure that solves the time-indexed model for iteratively smaller time steps, i.e., with an increasingly better accuracy. In this procedure, the best schedule found in one iteration of the procedure is transformed into a feasible starting solution for the next iteration through a *squeezing procedure*. The resulting value of the makespan is used to determine the length of the next time horizon in order to keep the total number of time steps required for the model in each iteration of the procedure as small as possible. In Section 5.1 we describe the squeezing procedure; in Section 5.2 the details of the complete iterative procedure are described.

5.1. The squeezing procedure

The processing times and release dates for a given iteration s are rounded up to the nearest respective multiple of the chosen length, ℓ^s, of the time step. In the order of increasing starting times for the operations in the best

solution obtained in the previous iteration, the starting time of each operation is recalculated such that it is scheduled (on the same resource) as early as possible, without violating any precedence or release date constraints. In the example illustrated in Figure 1, the completion time of operation i of job j constrains the starting time of operation p of job q, when the time steps are large; for the shorter time steps, the squeezing procedure schedules the start of operation p of job q at its release date, $\tilde{r}_{pq}$.

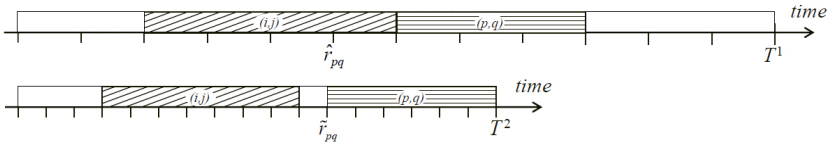

Figure 1: The output of the time-indexed model from one iteration is squeezed into a feasible solution for the next iteration possessing shorter time steps.

Since the squeezing procedure retains the ordering of the operations on each resource, and all precedence and release dates' constraints are considered by the squeezing procedure, the resulting schedule represents a feasible solution to the scheduling problem defined with the shorter time steps. The makespan of the schedule obtained by the squeezing procedure applied to the solution from the previous iteration is used as the time horizon, T^s, for the next iteration, for the case when the objective is to minimize the makespan. When minimizing the tardiness objective (7), the value assigned to the next time horizon equals the sum of the largest operation processing time and the makespan obtained by the squeezing procedure, since a smaller value of the objective function in (7) may correspond to a larger makespan.

5.2. The iterative procedure

The iterative procedure employed for solving a time-indexed model is described in Algorithm 1.

For each iteration s of the procedure, suitable values are assigned to the length ℓ^s of the time steps and the time horizon T^s. In order to determine a value for ℓ^1, the total number, V, of variables required for the smallest value of the time step, denoted $\tilde{\ell}$ to use in the last iteration is estimated as $V := \big(\sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{N}_j} \bar{p}_{ij}\big)\big(\sum_{j\in\mathcal{J}} n_j\big)$, where $\bar{p}_{ij} := |\mathcal{M}_{ij}|^{-1} \sum_{k\in\mathcal{M}_{ij}} p_{ijk}$ for

Algorithm 1 Iterative_procedure(`datfile`, ℓ^1, T^1, P, w^{best}, $\tilde{\ell}$)

while $\ell^s \geq \tilde{\ell}$ **do**
  Solve model P;
  Squeeze the resulting schedule using original data;
  $w^{\text{s}} \leftarrow$ the objective value of the squeezed schedule;
  if $w^{\text{s}} < w^{\text{best}}$ **then**
    $w^{\text{best}} \leftarrow w^{\text{s}}$;
    Store the corresponding solution, x^{best};
  end if
  $s \leftarrow s + 1$;
  Update ℓ^s;
  Squeeze the best solution, x^{best}, using data defined by (11);
  Compute T^s;
  Generate a `datfile` with the best solution found as starting solution;
  if P = TI-Cmax **and** $t^{\text{root}} > t^{\text{root}}_{\text{max}}$ **then**
    P $\leftarrow$ TI-prec-Cmax;
  end if
end while

operation i of job j denotes the average processing time over all resources in the set $\mathcal{M}_{ij}$. The threshold values for V used to determine ℓ^1 for the computations described in Section 6 are listed in Table 2.

In our computations, T^1 is determined using a heuristic similar to the one described in Thörnblad (2011, p. 35). The heuristic finds a feasible schedule by assigning the operations by order of increasing release dates in an allowed and available resource; T^1 is then assigned the value of the makespan of this schedule, expressed in number of time intervals of length ℓ^1. Once the value of ℓ^1 is determined, the input data for the first iteration, $s = 1$, is created, according to

$$p_{ijk} := \left\lceil \frac{\tilde{p}_{ijk}}{\ell^s} \right\rceil,\ r_{ij} := \left\lceil \frac{\tilde{r}_{ij}}{\ell^s} \right\rceil,\ d_j := \left\lfloor \frac{\tilde{d}_j}{\ell^s} \right\rfloor, \qquad i \in \mathcal{N}_j,\ j \in \mathcal{J},\ k \in \mathcal{K}, \quad (11)$$

where the superscript $\sim$ indicates the original parameter values.

Algorithm 1 is initialized by letting the model P be TI-Cmax or TI-Tard, determined by the objective considered, and by assigning a large enough number to w^{best}, being an upper bound on the value of the best solution

Table 2: The threshold values of V used to determine ℓ^1 (the time step for the first iteration). V is the estimated total number of variables required for the smallest desired length of the time step. The median of the processing times (in multiples of $\tilde{\ell}$) is denoted by $\bar{p}$.

$V \in$ ($*1000$)	ℓ^1
$(0, 10)$	1
$[10, 50)$	$\bar{p}/8$
$[50, 100)$	$\bar{p}/4$
$[100, 500)$	$\bar{p}/2$
$[500, \infty)$	$\bar{p}$

found so far. The input data required to solve the model P is denoted `datfile` in the algorithm.

When considering the minimization of the makespan, the algorithm is terminated after either (i) the nth (with $n = 3$) feasible solution is found (cplex option solutionlim $= n$ [see (IBM Corp., 2009, Chapter 7)]), or (ii) the `mipgap`[3] is less than a pre-specified number, or (iii) a pre-specified time limit is exceeded. If the computations were stopped due to the condition (i), then $\ell^s := \ell^{s-1}$, otherwise, $\ell^s :=$ `round`(ℓ^{s-1}/ζ), where $\zeta > 1$. In the computations we employed $\zeta = 1.8$. [Choosing $\zeta = 2$ may result in an unfavourable data pattern in the rounding of parameters in (11).] Further, in order to reduce the total number of iterations, we set $\ell^s := \tilde{\ell} = 1$ when $\ell^{s-1}/\zeta < 5$. The reason for this is that we found in the tests that the best solution is most often obtained for larger values of ℓ^s, and the last iteration, with $\ell^s = 1$, is most often used to verify the optimality of a solution already found (or compute a `mipgap`) rather than to find better feasible solutions.

The value T^s of the time horizon is updated in the iterative procedure as described in Section 5.1. For some of the largest instances of TI-Cmax, the CPU time, t^{root}, required to solve the root relaxation (which is essentially the LP relaxation) exceeded the time limit. Therefore, the model TI-prec-Cmax was chosen instead of TI-Cmax if the value of t^{root} exceeded a threshold value, $t^{\text{root}}_{\max}$, in the previous iteration. In our computational testing, $t^{\text{root}}_{\max} := 1$

[3]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` $:= \frac{|z-\text{LB}|}{10^{-10}+|z|} \cdot 100\%$.

CPU-second. For the model TI-Tard, the precedence constraints (1e) were employed in all iterations since they yielded smaller `mipgap` and shorter computation times.

6. Computational results

As mentioned in Section 3.3, we have chosen to compare the models by solving the twelve largest benchmark test instances in Fattahi et al. (2007). We omitted the eight smallest instances out of the original 20, since they can be computed in just a few seconds and are thus not that interesting for comparison. All these test instances are available via Behnke and Geiger (2012), along with other instances for the flexible job shop problem. The chosen instances range from $n = 3,\ m = 3$, and $n_j \leq 3$, for the smallest instance *sfjs9* to $n = 12,\ m = 8$, and $n_j \leq 4$, for the largest instance *mfjs10*.

The models compared are the time-indexed models TI-Cmax (TI-prec-Cmax) and TI-Tard (implemented through Algorithm 1), the alternative model (Alt-Cmax and Alt-Tard), and a benchmark model[4] (BM-Cmax) developed by Özgüven et al. (2010). The latter model is closely related to the alternative model; see Section 3.3. BM-Cmax was chosen to be our benchmark model since it yielded the best results in the evaluation by Demir and İşleyen (2013). We also employed the benchmark model considering the tardiness objective (9), implemented through the constraints (10a)–(10c). This model is referred to as BM-Tard.

The computations were carried out using AMPL-CPLEX 12.1.0 (Fourer et al., 2002; IBM Corp., 2009) on a computer with two 2.66 GHz Intel Xeon X5650 processors, each with six cores (24 threads), with a total memory of 48 Gbyte of RAM. The time limit for each call to the CPLEX solver from the iterative procedure was set to 7200 CPU-seconds. Since the iterative procedure calls the solver once each iteration, the total computation time may exceed 7200 CPU-seconds.

[4]We implemented the benchmark model as it is formulated in Özgüven et al. (2010), except for a possible typo: In the precedence constraints regarding the operations of the same job, the sum over the completion times for operation $i - 1$ should, using our notation, be over $k \in \mathcal{M}_{i-1,j}$; this is unclear in Özgüven et al. (2010, constraints (6), p. 1542). This mistake is also present in Demir and İşleyen (2013), where the benchmark model is presented. In ibid., the constraints corresponding to (3d)–(3e) are defined for $j \leq q$ rather than for $j < q$. This is not incorrect, but the redundancy implies longer computation times (Demir and İşleyen, 2013, constraints (2.5)-(2.6), p. 981)).

The squeezing procedure and the generation of input data files in each iteration were implemented in MATLAB (2011). The total running time for this Matlab program was around 15 seconds. Since the running time would only be a fraction of a second if this code was optimized and implemented in, for example, C, the total running time of the iterative procedure was calculated as the sum of the computation times used by CPLEX over all the iterations of Algorithm 1.

Since it is only in the last iteration (with $\ell^s = 1$) that a lower bound on the optimal objective value sought is guaranteed, and the `mipgap` is comparable with that of the other models, the iterative procedure was run until the last iteration was terminated due to either the computation time exceeding the time limit of 7200 seconds, or that the `mipgap` in the last iteration (with $\ell^s = \tilde{\ell} = 1$) being less than 0.0005. This is due to the fact that the optimal objective value—and the lower bound—in earlier iterations are expressed in multiples of ℓ^s, and the lower bound can not be transformed into a lower bound valid for the last iteration employing $\tilde{\ell}$. In order to ensure a fair comparison between the models, for each of the instances the time limits for solving BM-Cmax and Alt-Cmax (BM-Tard and Alt-Tard) were set to the total CPU time needed by the iterative procedure implemented with the model TI-Cmax (TI-Tard).

6.1. Test results for the minimization of the makespan

In Table 3, we present the results from employing the iterative procedure with TI-Cmax are compared with the results from solving BM-Cmax and Alt-Cmax for the seven largest instances *mfjs4-10*. All the smaller instances, namely *sfjs9-10*, and *mfjs1-3*, were solved to optimality within 20 CPU-seconds by all the three models.

The time limit was reached before the iterative procedure with TI-Cmax had established optimality for the instances *mfjs4* and *mfjs6*, although the time to find the best feasible solution ("time to best") is competitive with the models BM-Cmax and Alt-Cmax;, see Figure 2. The computation times used by the iterative procedure to verify optimality were in all cases longer than that of the other two models. It seems, however, that the Algorithm 1 employing the models TI-Cmax and TI-prec-Cmax works well for the four largest instances, *mfjs7-10*, since the best results were found by this iterative procedure—both the value of the shortest makespan found and the time required to find the corresponding feasible solution. During the computational tests we found that the iterative procedure indeed found the optimal

Table 3: Computational results for the largest instances *mfjs4-10*. The best results for each instance are written in bold and '*' in the `mipgap` column indicates that optimality is verified. The values within parentheses denote the CPU times to compute the best makespan found by that model, for the cases when a shorter makespan was found by another model for the corresponding instance.

	Iterative procedure TI-Cmax (TI-prec-Cmax)				BM-Cmax				Alt-Cmax			
Instance	CPU time (s)	Time to best (s)	$C_{\max}$	mip-gap (%)	CPU time (s)	Time to best (s)	$C_{\max}$	mip-gap (%)	CPU time (s)	Time to best (s)	$C_{\max}$	mip-gap (%)
mfjs4	7213	**3**	**554**	3.2	177	12	**554**	*	**124**	23	**554**	*
mfjs5	82	81	**514**	*	**11**	**9**	**514**	*	**11**	11	**514**	*
mfjs6	7259	54	**634**	3.2	227	**19**	**634**	*	72	51	**634**	*
mfjs7	14091	**2539**	**879**	9.3	14401	(540)	881	9.6	14423	5220	**879**	**8.7**
mfjs8	16163	**391**	**884**	13.6	16416	(1980)	886	**10.0**	16527	(6620)	887	13.5
mfjs9	25880	**3899**	**1081**	**23.1**	26318	(7010)	1113	25.0	26154	(6760)	1120	26.1
mfjs10	33325	**464**	**1208**	21.9	33881	(6870)	1236	**19.5**	33600	(2700)	1264	21.8

solutions to the instances *mfjs7-8*, although their optimality was not verified within the time limits.

The models BM-Cmax and Alt-Cmax performed equally well, which is not surprising since the respective reduced models, constructed by the solver in the presolve phase, comprises the same number of variables, and since BM-Cmax contained only 5% more constraints than Alt-Cmax. TI-Cmax, on the other hand, contained 40 times more variables and about 6 times more constraints than the benchmark model, in the reduced model for $\ell^s = 1$. [Note that the optimal value for *mfjs4* is 554, and not 564 as stated in both Özgüven et al. (2010) and Demir and İşleyen (2013).] Both Behnke and Geiger (2012), and Bagheri et al. (2010) found the makespan of 554 for this instance when applying constraint programming (CP) and an artificial immune algorithm (AIA), respectively. To our knowledge, the two latter articles present the hitherto best results for the Fattahi test instances when employing these methods.

Behnke and Geiger (2012) found and verified the optimal solution for the smaller instances *sfjs9-10*, and *mfjs1-3*. Within a time limit of 10 minutes of computing time, they found the same solutions as TI-Cmax for the instances *mfjs4–6*, *mfsj8*, and *mfjs10*. For *mfjs7* and *mfjs9*, they found the makespans of 931 and 1070, respectively; hence CP produces equally good results as TI-Cmax. Regarding the computation times, we have not been able to compare the iterative procedure implemented with the model TI-Cmax

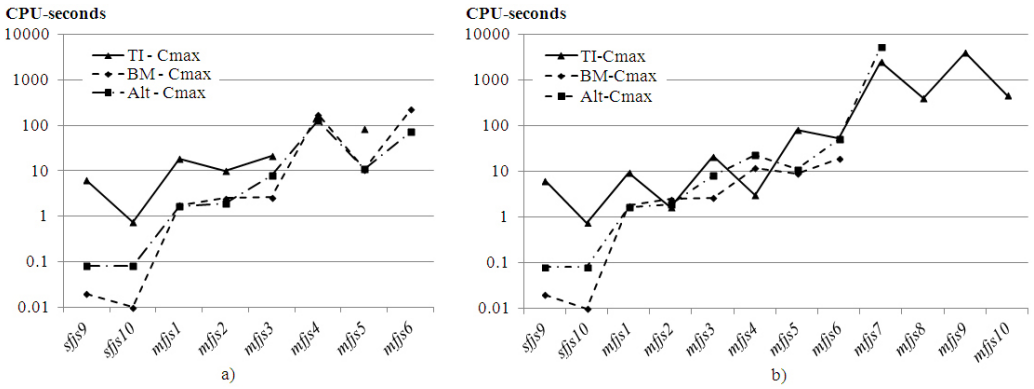

Figure 2: Computation times for the models TI-Cmax, BM-Cmax, and Alt-Cmax as applied to the Fattahi test instances. a) The times required for the respective models to find optimal solutions. For some of the larger instances optimality was not verified within 7200s; these results are hence omitted. b) The time required for the respective model to find the best feasible solution.

(TI-prec-Cmax) with the CP model by Behnke and Geiger (2012), since the tests were run using different computers.

Bagheri et al. (2010) propose an AIA for the FJSP. In six out of the twelve benchmark instances, the shortest makespan found after 10 runs of AIA was larger than that found by TI-Cmax; the optimal makespan was found for four instances. As an example, the shortest makespan for *mfjs9* and *mfjs10* was 1088 and 1267, respectively. The corresponding values found by our iterative procedure are 1081 and 1208; see Table 3. One run of AIA is performed in only a few CPU-seconds, but since it is a meta-heuristic there is no guarantee that a better result would be achieved if it was run repeatedly during the same amount of CPU time as we have used.

6.2. Test results for the minimization of the tardiness objective

In order to compare the models TI-Tard (implemented in the iterative procedure), BM-Tard, and Alt-Tard, we generated due dates for all the test instances, since due dates are not given for the original instances by Fattahi et al. (2007). Due dates for all jobs were randomly generated in the range $[0.5C_{\max},1.5C_{\max}]$, where the value of $C_{\max}$ equals the shortest makespan found for each instance. The choice of this range is motivated by our experience from a real flexible job shop (Thörnblad et al., 2013). The test instances including due dates are denoted by the prefix '*d-*' of the original

name of each instance. The objective weights for the completion times used in the computations are $\alpha_j = 1,\ j \in \mathcal{J}$. The tardiness weights $\beta_j,\ j \in \mathcal{J}$, are computed according to (8), with $B = 10$, such that $0 \leq \beta_j \leq 20$ holds for all $j \in \mathcal{J}$. In Figure 3, the computation times required for the three models to solve the test instances generated are plotted. For the medium

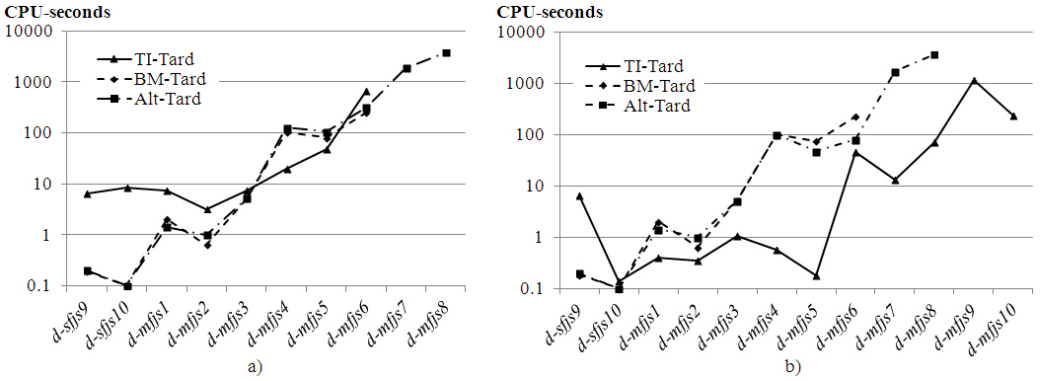

Figure 3: a) The computation times required for solving the models TI-Tard, BM-Tard, and Alt-Tard to optimality for the Fattahi test instances including due dates. For some of the larger instances optimality was not verified within the time limit of 7200s and are therefore not shown.
b) The time required to find the best feasible solution for the models TI-Tard, BM-Tard, and Alt-Tard.

sized instances, all three models require almost the same computation times. As expected, TI-Tard is not competitive for the smallest instances, since it is solved in several steps. Only Alt-Tard reaches optimality for the instances *d-mfjs7–8*.

In Table 4, the results are listed for the four largest instances. As for the case with the makespan objective, the best objective values were found by our iterative procedure implemented with TI-Tard. It outperforms the other models regarding the time to find the best solution for all instances except for the two smallest ones, see Figure 3b); the iterative procedure with TI-Tard was on average more than 60 times faster than BM-Tard and Alt-Tard with respect to the time to find the best solution.

When considering the tardiness objective, the reduced models constructed by the solver in the presolve phase contain on average 19% more constraints and 14% more variables for the BM-Tard model compared with the Alt-Tard model. Hence the presolve phase is not able to reduce all the redundant

Table 4: Computational results for the largest instances *d-mfjs7-10*. The best results for each instance are written in bold and '*' in the `mipgap` column indicates that optimality is verified. The values within parentheses denote the CPU times to compute the best makespan found by that model, for the cases when a shorter makespan was found by another model for the corresponding instance.

	Iterative procedure TI-Tard				BM-Tard				Alt-Tard			
Instance	CPU time (s)	Time to best (s)	Obj value	mip-gap (%)	CPU time (s)	Time to best (s)	Obj value	mip-gap (%)	CPU time (s)	Time to best (s)	Obj value	mip-gap (%)
d-mfjs7	7592	**14**	**35111.1**	3.9	7593	(795)	35244.9	2.8	**1894**	1740	**35111.1**	*
d-mfjs8	7292	**70**	**25225.3**	1.4	7593	(6870)	25240.0	4.2	**3801**	3710	**25225.3**	*
d-mfjs9	15568	**1142**	**60159.5**	**4.4**	15569	(7200)	63175.4	24.7	15690	(7200)	60420.3	21.9
d-mfjs10	10260	**236**	**45568.4**	**2.6**	10261	(9600)	46532.4	21.0	10344	(6850)	47215.7	21.1

variables of the BM-Tard model, as was the case for the BM-Cmax model. On the other hand, TI-Tard contains 17 times more constraints and 77 times more variables than BM-Tard in the reduced problem for $\ell^s = 1$. Nevertheless, TI-Tard is competitive with the Alt-Tard and BM-Tard regarding the computation time to verify optimality and outperforms these models regarding the time to find the best solution for all instances tested, except for the two smallest ones; see Figure 3.

In Figure 4, the best feasible solution found by each of the models is compared with its best lower bound, for the instances *mfjs7–10* and *d-mfjs7–10*, respectively. For each instance, the values are divided by the objective value of the best solution found (by TI-Cmax and TI-Tard, respectively). In Figure 4, we note that there is a large difference between the lower bounds found by employing the makepan objective and those found by employing the tardiness objective. This indicates that the performance of a model is dependent on the objective used. TI-Cmax performs equally good (or bad) regarding the lower bounds as do the models BM-Cmax and Alt-Cmax, while TI-Tard outperforms the models BM-Tard and Alt-Tard for the instances *d-mfjs9-10* by finding significantly better lower bounds.

7. Conclusions

We have demonstrated that there is a large difference in the performance of the scheduling models depending on which objective is considered. Hence, it is important to evaluate scheduling models with respect to objectives that are well suited for the real applications for which they are intended. Since

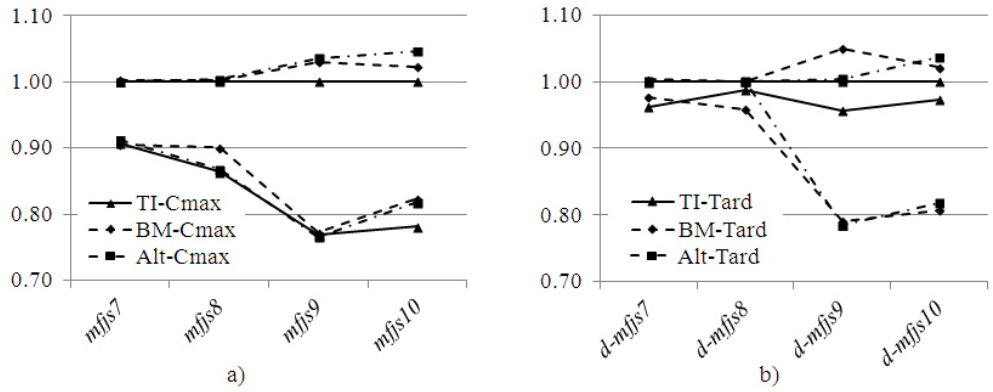

Figure 4: The values of the best feasible solutions (upper marker) and the best lower bounds (lower marker) found. a) Values for the instances *mfjs7–10* found by the models TI-Cmax, BM-Cmax, and Alt-Cmax, normalized by the value of the best solution found by TI-Cmax. b) Values for the instances *d-mfjs7–10* found by the models TI-Tard, BM-Tard, and Alt-Tard, normalized by the value of the best solution found by TI-Tard.

most real flexible job shops are parts of longer supply chains with ongoing production, it must be prioritized that production should be predictable and according to plan. In Section 4, we argue that an objective that targets due dates will serve to control the flow and stabilize its pace, while the objective of minimizing the makespan risk to exacerbate a possibly already unreliable flow.

We have shown that the time-indexed model (despite its large numbers of variables and constraints), combined with the iterative algorithm proposed is able to find significantly better feasible solutions for the largest instances than both the other models, considering both objectives studied. The scheduling method proposed outperforms the other models regarding the time required to find the best feasible solution. We have also presented an alternative model, similar to the benchmark model but with fewer variables and constraints. This model is the only one that is able to solve two of the large instances when minimizing the tardiness objective. It seems, however, worthwhile to implement the iterative algorithm proposed for medium and large sized instances, since it is able to quickly find solutions of high quality. The proposed scheduling method typically finds good feasible solutions in an early iteration, employing relatively long time steps; after squeezing, these solutions are, actually, often optimal or near-optimal. The verification of op-

timality can, however, only be made in a later iteration when all parameter data can be expressed as multiples of the length of the time steps. Our findings imply that our new procedure can be beneficially utilized for scheduling real flexible job shops.

8. Acknowledgements

A special thanks to Thomas Ericsson at the Department of Mathematical Sciences, for all support regarding the computations. Furthermore, we would like to gratefully acknowledge the financial support from GKN Aerospace Engine Systems (formerly Volvo Aero), The Swedish Research Council (grant no. 621-2007-4716), NFFP (Swedish National Aeronautics Research Programme), and VINNOVA (through Chalmers Transport Area of Advance).

References

Al-Hinai, N., ElMekkawy, T. Y., 2011. An efficient hybridized genetic algorithm architecture for the flexible job shop scheduling problem. Flexible Services and Manufacturing Journal 23, 64–85.

Bagheri, A., Zandieh, M., Mahdavi, I., Yazdani, M., 2010. An artificial immune algorithm for the flexible job-shop scheduling problem. Future Generation Computer Systems 26 (4), 533–541.

Baykasoglu, A., Özbakir, L., 2010. Analyzing the effect of dispatching rules in the scheduling performance through grammar based flexible scheduling system. International Journal of Production Economics 124 (2), 269–381.

Behnke, D., Geiger, M. J., 2012. Test instances for the flexible job shop scheduling problem with work centers. `http://www.nbn-resolving.de/urn:nbn:de:gbv:705-opus-29827` (Accessed on August 13, 2013).

Berghman, L., 2012. Machine scheduling models for warehousing docking operations. PhD Thesis, Katholieke Universiteit Leuven, Belgium.

Brucker, P., Knust, S., 2012. Complex Scheduling, 2nd Edition. Springer-Verlag, Berlin, Germany.

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

Fattahi, P., Saidi Mehrabad, M., Jolai, F., 2007. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. Journal of Intelligent Manufacturing 18 (3), 331–342.

Fourer, R., Gay, D. M., Kernighan, B. W., 2002. AMPL: A Modeling Language for Mathematical Programming, 2nd Edition. Brooks/Cole Publishing Company/Cengage Learning.

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

Jain, A. S., Meeran, S., 1999. Deterministic job-shop scheduling: Past, present and future. European Journal of Operational Research 113 (2), 390–434.

Low, C., Yip, Y., Wu, T., 2006. Modelling and heuristics of FMS scheduling with multiple objectives. Computers & Operations Research 33 (3), 674–694.

Manne, A. S., 1960. On the job-shop scheduling problem. Operations Research 8 (2), 219–223.

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

Nemhauser, G. L., Wolsey, L. A., 1988. Integer and Combinatorial Optimization. John Wiley & Sons, Inc., New York, NY, USA.

Özgüven, C., Özbakir, L., Yavuz, Y., 2010. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Applied Mathematical Modelling 34 (2), 1539–1548.

Sadykov, R., Wolsey, L. A., 2006. Integer programming and constraint programming in solving a multimachine assignment scheduling problem with deadlines and release dates. INFORMS Journal on Computing 18, 209–217.

Thörnblad, K., Sep. 2011. On the optimization of schedules of a multitask production cell. Licentiate Thesis, Chalmers University of Technology and University of Gothenburg, Göteborg, Sweden.

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

van den Akker, J. M., Hurkens, C. A. J., Savelsberg, M. W. P., 2000. Time-indexed formulations for machine scheduling problems: Column generation. INFORMS Journal on Computing 12, 111–124.

Wang, L., Zhou, G., Xu, Y., Wang, S., Liu, M., 2012. An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. The International Journal of Advanced Manufacturing Technology 60, 303–315.