AN EFFICIENT ALGORITHM FOR SOLVING THE FLEXIBLE JOB SHOP SCHEDULING PROBLEM

Karin Thörnblad [*,1]

Ann-Brith Strömberg [**,3]

Michael Patriksson [**,4]

Torgny Almgren [*,2]

*) GKN Aerospace Engine Systems, Dept. of Logistics Development, 461 81 Trollhättan, Sweden
[1]) E-mail: karin.thornblad@ gknaerospace.com, Tel: +46 520 29 22 66
[2]) E-mail: torgny.almgren@ gknaerospace.com, Tel: +46 520 29 22 62

**) Chalmers University of Technology and University of Gothenburg, Dept. of Mathematical Sciences, 421 96 Göteborg, Sweden
[3]) E-mail: anstr@chalmers.se, Tel: +46 31 772 53 78
[4]) E-mail: mipat@chalmers.se, Tel: +46 31 772 35 29

ABSTRACT

Purpose
To 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 makespan and tardiness, respectively.

Design/methodology/approach
A time-indexed mixed integer programming model is used to find solutions 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 model.

Findings
The proposed method finds significantly better solutions for the largest instances within the same time frame. Both the other models are better choices for some smaller instances, which is expected since the new method is designed for larger problems. Only our alternative model is able to solve two of the largest instances when minimizing the tardiness.

Research limitations/implications
Interesting future research topics include the introduction of constraints representing other relevant entities such as the availability of tools and fixtures, and the scheduling of maintenance activities and personnel.

Practical implications
Real cases of flexible job shop problems typically yield very large models. Since the new procedure quickly finds solutions of good quality to such instances, our findings imply that the new procedure is beneficially utilized for scheduling real flexible job shops.

Original/value
We show that real flexible job shop problems can be solved through the solution of a series of carefully formulated discretized mathematical optimization models.

Keywords: Flexible job shop scheduling, Linear integer optimization, Linear mixed integer programming, Discretization procedure, Benchmarking, 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, 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, see 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 Fattahi *et al.* (2007).

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 periods of equal length. The decision variables used in this article are valued 1 if the corresponding operation is scheduled to start at the beginning of a specific time period 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 the total weighted completion times. We obtained a similar result for a special case of the FJSP in a real production cell with the objective of minimizing a weighted sum of the completion times and 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 present 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 is an exact method, which seems to yield good

results, see e.g. Sadykov and Wolsey (2006) for an evaluation of MILP and constraint programming models for solving a so-called multimachine assignment scheduling problem. There are many so-called 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 constraint programming and an artificial immune algorithm, 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* $(\mathcal{M}_{ij} \subseteq \mathcal{K}, i \in \mathcal{N}_j, j \in \mathcal{J})$

$\mathcal{T}$ 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 a time step in the time-indexed model

M a big number, at least as big 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 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 can be scheduled before the completion of the previous operation. Therefore, and since the processing time is resource dependent, the release date is defined as $r_{ij} := \sum_{\varsigma=1}^{i-1} \min_{k \in \mathcal{M}_{\varsigma j}} \{p_{\varsigma jk}\}$, for $i = 2,\ldots,n_j, j \in \mathcal{J}$. Likewise, $\delta_{ij} := \sum_{\varsigma=i}^{n_j} \min_{k \in \mathcal{M}_{\varsigma j}} \{p_{\varsigma jk}\}$, for $i \in \mathcal{N}_j, j \in \mathcal{J}$.

3.2. The time-indexed model

The planning horizon is divided into $T+1$ intervals, each of length ℓ. The value of the parameter T has to be large enough such that an optimal schedule is contained within the time horizon $[0,(T+1)\ell]$. 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

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

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

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

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

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

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

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

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

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 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 time interval T. The constraints (1g) are redundant, but their inclusion reduces the number of variables in the model. The model (1) will henceforth be referred to as model 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 the *linear relaxation* is too long for practical purposes, then an alternative set of precedence constraints may be used:

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

The linear relaxation (or LP-relaxation) of TI-Cmax is a model with the same objective and constraints, except for the integrality constraints (1h) which are substituted by $0 \le x_{ijku} \le 1$, $i \in \mathcal{N}_j,\ j \in \mathcal{J}, k \in \mathcal{K}, u \in \mathcal{T}$. The optimal objective value of the linear relaxation of a model

is called its *LP bound*; it is a lower bound on the optimal objective value of the original model including integrality constraints. Provided that the 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 Table 3.1 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 requires fewer variables and constraints than the benchmark model, since the variables for the completion times of operations and jobs used in the latter model are redundant. The variables used in our formulation of the alternative model are

z_{ijk} = 1, if operation *i* of job *j* is processed on resource *k*; 0, otherwise,
y_{ijpqk} = 1, if operation *i* of job *j* precedes operation *p* of job *q* in resource *k*; 0, otherwise,
t_{ijk} = the starting time of operation *i* of job *j* in resource *k*, and
$C_{\max}$ = the maximum completion time over all the jobs (makespan).

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

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

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

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

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

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

$$\sum_{k\in\mathcal{M}_{i-1,j}} (t_{i-1,jk} + p_{i-1,jk} z_{i-1,jk}) - \sum_{k\in\mathcal{M}_{ij}} t_{ijk} \le 0, \qquad i=2,\dots,n_j,\ j\in\mathcal{J}, \tag{3f}$$

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

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

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

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

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 the same job are not violated. The constraints (3g) determine the makespan of the schedule. Finally, the constraints (3h)–(3j) are the nonnegativity and binary constraints on the variables.

In Table 3.1 the objective values of the solutions to the LP-relaxation 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) is clear for this instance, but the price of the better result is paid for by the longer computation time.

Table 3.1. The objective values and computation times for the solution of LP relaxation of the models for the benchmark test instance mfjs7 (Fattahi et al., 2007).

Model	Optimal objective value	Objective value of LP relaxation	Comp. time for solving the 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 model TI-Cmax is equivalent to Alt-Cmax 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 of Alt-Cmax and TI-Cmax are related according to

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

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

and

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

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 an integer programming sense) to (2), which are equivalent to (3f). The constraints (3c) are required in order to define the variables t_{ijk} for the model Alt-Cmax.

4. TARDINESS VERSUS MAKESPAN

The objective most often considered in the studies of (flexible) job shop problems is the minimization of the makespan. According to the survey (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 have 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 (i.e., takt time). 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, since if all jobs can be processed in due time, the objective equals the minimization of the weighted sum of the completion times. However, if the scheduling procedure is capable of repeatedly schedule the job shop with no tardy jobs, then the planner has the opportunity of setting more challenging due dates, which in turn will shorten even the planned production lead times. If 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 occur 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 to

$$\text{minimize} \quad \sum_{j\in\mathcal{J}} (a_j C_j + b_j T_j), \tag{5}$$

where C_j and $T_j = (C_j - d_j)_+$ denote the completion time and the tardiness of a job, respectively, and a_j and b_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 this objective. Since the completion time of a job in TI-Cmax can be expressed as

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

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

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

Note that the operator $()_+$ in this objective is applied only to parameters, and hence the objecttive remains linear. As the release dates are already considered in 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), (1a)–(1e), (1g)–(1h) will henceforth be referred to as 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, b_j, according to

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

where $B > 0$ denotes the weight employed for the jobs having due date 0, such that $0 \le b_j \le 2B$ hold for all $j \in \mathcal{J}$. In this way, the jobs that are most delayed are assigned the

highest tardiness weights. Since the main objective is to minimize the tardiness, the objective weights for the completion time, a_j, must be chosen such that $0 \le a_j \ll B,\ j \in \mathcal{J}$, holds.

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

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

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

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

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

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

The alternative model with the proposed tardiness objective is thus formulated as that to minimize (9) subject to (3b)–(3f), (3h)–(3i), (11), 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 total number of time steps. We have developed a solution procedure that solves the time-indexed model for iteratively smaller time steps, i.e., with increasingly better accuracy. In this procedure, the schedule resulting from one iteration of the procedure is transformed into a feasible starting solution for the next iteration by a *squeezing procedure*. The resulting 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 iterative procedure are described.

5.1. The squeezing procedure

The processing times and release dates for a given iteration are rounded up to multiples of the chosen length of the time step. In the order of increasing starting times for the best solution obtained in the previous iteration, the starting time of each operation is recalculated such that it is scheduled as early as possible, without violating any precedence or release date constraints. In the example illustrated in Figure 5.1, 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 short time steps, the squeezing procedure schedules the start of operation *p* of job *q* at its release date, $\tilde{r}_{pq}$.

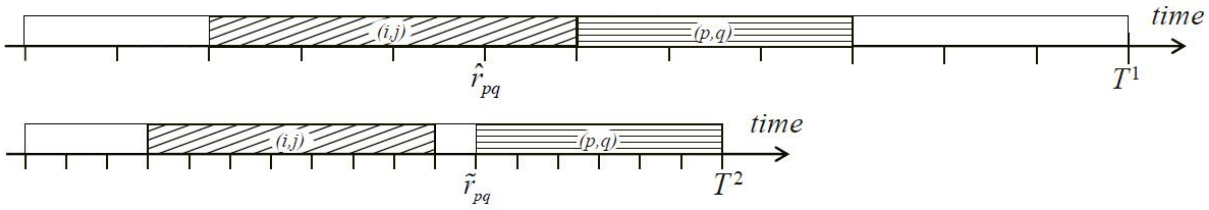

Figure 5.1 The output of the time-indexed model from one iteration is squeezed into a feasible solution for the next iteration, having shorter time steps.

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 for the next iteration, 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 processing time and the makespan obtained by the squeezing procedure, since a smaller value of the objective function may correspond to a larger makespan.

5.2. The iterative procedure

For the first iteration of the procedure, we need to assign a suitable value to the length of the time step (time horizon), here denoted ℓ^1 (T^1) for the first iteration, and ℓ^s (T^s) for iteration *s*. In order to determine a value for ℓ^1, we estimate the total number, V, of variables required for the smallest time step $\tilde{\ell}$ that we wish to use in the last iteration as $V := \left(\sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{N}_j} \bar{p}_{ij}\right)\left(\sum_{j\in\mathcal{J}} n_j\right)$, where $\bar{p}_{ij} := |\mathcal{M}_{ij}|^{-1} \sum_{k\in\mathcal{M}_{ij}} p_{ijk}$ is the average processing time over all resources in the set $\mathcal{M}_{ij}$. The threshold values of V used to determine ℓ^1 for the computations described in Section 6 are listed in Table 5.1. In our computations, T^1 is determined using a heuristic similar to that described in Thörnblad (2011, p. 35). The heuristic finds a feasible schedule, assigning the operations by order of increasing release dates in an allowed and available resource; T^1 is assigned the value of the makespan of this schedule expressed in number of time intervals of length ℓ^1.

Table 5.1. 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 is denoted by $\bar{p}$.

$0 < V < 10\,000$	$10\,000 \le V < 50\,000$	$50\,000 \le V < 100\,000$	$100\,000 \le V < 500\,000$	$500\,000 \le V$
$\ell^1 = 1$	$\ell^1 = \frac{1}{8}\bar{p}$	$\ell^1 = \frac{1}{4}\bar{p}$	$\ell^1 = \frac{1}{2}\bar{p}$	$\ell^1 = \bar{p}$

Once the value of ℓ^1 is determined, the input data for the first iteration is created. The values of the parameters are scaled and rounded, according to

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

where ~ denotes the original parameter values, and $s = 1$ for the first iteration. The iterative procedure employed for solving a time-indexed model is described in Algorithm 1. It 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 found so far. The input data needed by the model (P) is denoted *datfile* in the algorithm.

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

While $(\ell^i \geq \tilde{\ell})$ **do**

Solve model (P);
Squeeze the resulting schedule using original data;
$w^s \leftarrow$ the objective value of the squeezed schedule;
If $w^s < w^{\text{best}}$ **then** $w^{\text{best}} \leftarrow w^s$; 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** $t^{\text{root}} > t^{\text{root}}_{\max}$ **then** (P) $\leftarrow$ TI-prec-Cmax; **End if**

End while

In Algorithm 1, a new value of ℓ^s for iteration *s* is determined by the statement "Update ℓ^s". When considering the minimization of the makespan, the algorithm was terminated either after (i) the third feasible solution was found (cplex option solutionlim = 3), or (ii) the *mipgap* was less than a pre-specified number, or (iii) a pre-specified time limit was exceeded. The *mipgap* of a problem instance is defined as

$$mipgap := \frac{z - \text{LB}}{\text{LB}} \cdot 100\%,$$

where *z* denotes the objective value of the best feasible solution and LB denotes the best lower bound found by the optimization solver during the computations. If the computations were stopped due to the condition (i), then $\ell^s := \ell^{s-1}$, otherwise, $\ell^s := round(\ell^{s-1}/\alpha)$, where $\alpha > 1$. In the computations we employed $\alpha = 1.8$. Choosing $\alpha = 2$ might result in an unfavourable data pattern in the rounding of parameters in (11). Further, in order to reduce the total number of iterations, we let $\ell^s = \tilde{\ell} = 1$ when $\ell^{s-1}/\alpha < 5$. The reason for this is that we found in the tests that the best solution is most often obtained for larger values of ℓ, and the last iteration, with $\ell^s = 1$, is most often used to establish optimality (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, the CPU time, t^{root}, required to solve the root relaxation (which is essentially the time to solve the LP-relaxation) exceeded the time limit. Therefore, the model TI-prec-Cmax was chosen if the value of t^{root} exceeded a threshold $t^{\text{root}}_{\max}$ in the previous iteration. In the computational testing, $t^{\text{root}}_{\max}$ = 1 CPU-second. For the model

TI-Tard, the precedence constraints (1e) were employed in all iterations since they yielded smaller *mipgaps* 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 have 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 through Behnke and Geiger (2012), along with other instances for the flexible job shop problem. The chosen instances range from $n = 3,\ m = 3,\ n_j \le 3,$ for the smallest instance *sfjs9* to $n = 12,\ m = 8,$ $n_j \le 4,$ 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[1] developed by Özgüven *et al.* (2010) (BM-Cmax). This model is closely related to the alternative model, which is mentioned in Section 3.3. We also used the benchmark model considering the tardiness objective (9), implemented through the constraints (10a)–(10c). This model is referred to as BM-Tard; it was chosen to be our benchmark model since it yielded the best results in the evaluation by Demir and İşleyen (2013).

The computations were carried out using AMPL-CPLEX 12.1.0 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 of the CPLEX solver from the iterative procedure was set to 7200 CPU-seconds. Since the iterative procedure calls the solver once per iteration, the total computation time may exceed 7200 CPU-seconds.

The squeezing procedure and the generation of input data files in each iteration were implemented in Matlab. The total running time for this Matlab program was approximately 15 seconds. Since the running time would only be a fraction of a second if this code was optimized and written 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 iterations of Algorithm 1.

6.1. Test results for the minimization of the makespan

Since it is only in the last iteration (with $\ell^s = 1$) that a lower bound on the optimal objective value 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* being less than 0.0005. In order to ensure a fair comparison between the models, the time limit for solving BM-Cmax and Alt-Cmax was set to the total CPU-time needed by the iterative procedure; see Table 6.1, where the results for the seven largest instances, *mfjs4–10*, are presented. All the smaller instances,

[1] We implemented the benchmark model as it is formulated in Özgüven *et al.* (2010), but for a possible typo mistake: 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, see the constraints (6), p. 1542). This mistake is also present in Demir and İşleyen (2013), where the benchmark model is presented. In this article, the constraints corresponding to our constraints (3d) and (3e) are defined for all $j \le q$ rather than for all $j < q$. This is not incorrect, but the redundancy implies longer computation times (Demir and İşleyen, 2013, see the constraints (2.5)–(2.6), p. 981).

namely *sfjs9–10*, and *mfjs1–3*, were solved to optimality within 20 CPU-seconds by all three models. The time limit was reached before the iterative procedure 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 other models. 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 best makespan found and the time required finding this solution. Our analysis showed that the iterative procedure indeed found the optimal solution for *mfjs7–8*, although optimality was not verified within the time limits.

*Table 6.1. Computational results for the largest instances mfjs4–10. The best results in each category are written in bold and a * in the mipgap column denotes that optimality is proven.*

	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

The models BM-Cmax and Alt-Cmax performed equally well, which is not surprising since the reduced models, constructed by the solver in the presolve phase, comprised the same number of variables for these two models, and 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 problem for $\ell^s = 1$. [Note that the optimal value for *mfjs4* is 554 and not 564, as stated both in Özgüven *et al.* (2010) and in 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 and an artificial immune algorithm (AIA), respectively. To our knowledge, these two articles present the best results for the Fattahi test instances employing these methods.

Behnke and Geiger (2012) found and verified the optimal solution for the smaller instances *sfjs9–10*, and *mfjs1–3*. Within the time limit of 10 minutes of running 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 constraint programming produces equally good results as TI-Cmax. We cannot make any comparisons between the Algorithm 1 employing model TI-Cmax (TI-prec-Cmax) and the constraint programming model by Behnke and Geiger (2012) regarding the computation times, since the tests are not run using the same computer.

Bagheri *et al.* (2010) propose an AIA for the FJSP. In six out of the twelve benchmark instances, the best makespan found after 10 runs of AIA was larger than that found by TI-Cmax, and in no cases they found a better result. As an example, the best makespan for *mfjs9* and *mfjs10* was 1088 and 1267, respectively. One run of AIA is performed in only a few CPU-seconds, but as it is a meta-heuristic there is no guarantee that a better result would be achieved if it was run repeatedly during the same CPU times as we have used.

6.2. Test results for the minimization of the tardiness objective

In order to compare the models TI-Tard, 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.* (2010). 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 smallest makespan found for each instance. The choice of this range is motivated by our experience from a real flexible job shop, namely that some jobs have negative due dates and are hence late already from the start, and some jobs may be scheduled with zero tardiness. These instances are denoted by a "d-" in front of the original name of each instance.

The objective weights for the completion times used in the computations are $a_j = 1,\ j \in \mathcal{J}$. The tardiness weights $b_j,\ j \in \mathcal{J},$ are computed according to (8), with $B = 10$, such that $0 \le b_j \le 20$, hold for all $j \in \mathcal{J}$. The largest values are assigned to the objective weights of the jobs being the most delayed.

In Figure 6.1, the computation times required for the three models to solve the test instances generated are plotted. For the medium size instances, all 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*.

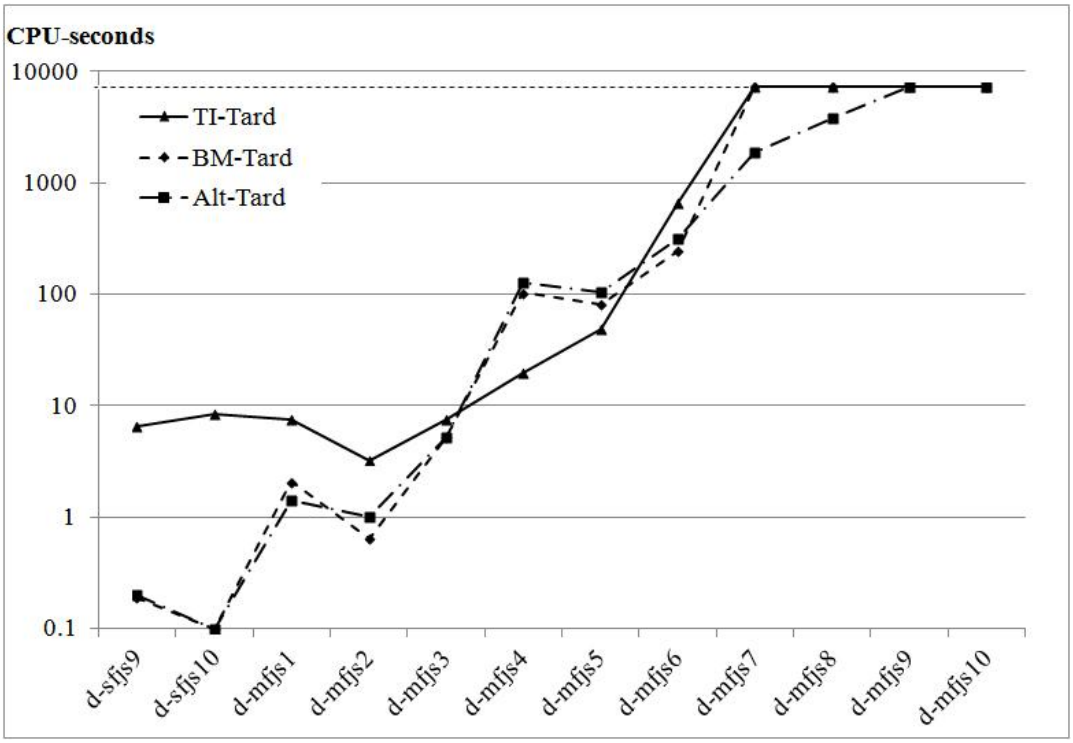

Figure 6.1. The computation times required for solving the models TI-Tard, BM-Tard, and Alt-Tard to optimality for the Fattahi test instances including due dates. For some of the larger instances optimality was not verified within the time limit of 7200s.

In Table 6.2, the results are listed for the four largest instances. As for the makespan objective, the best objective values were found by our iterative procedure. The model TI-Tard outperforms the other models regarding the time to find the best solution for all four instances, and regarding the mipgap for the instances *d-mfjs9–10*.

*Table 6.2. Computational results for the largest instances d-mfjs7–10. The best results in each category are written in bold and a * in the mipgap column denotes that optimality is verified.*

	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

When considering the tardiness objective, the reduced models constructed by the solver in the presolve phase contain 19% more constraints and 14% more variables for the BM-Tard model compared with the Alt-Tard model. Hence the presolve phase is no longer able to reduce all the redundant 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 $\ell^s = 1$.

7. CONCLUSIONS

We have noted 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 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 targeting due dates will serve to control the flow and stabilize its pace, while the objective of minimizing the makespan risk to exacerbate an already unreliable flow.

We have shown that the time-indexed model, combined with the iterative algorithm proposed, is able to find significantly better feasible solutions for the largest instances than both the alternative model and the benchmark model, considering both objectives studied, despite its large number of variables and constraints. 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. This model is recommended for small and medium size instances, since it is solved with only one launch of the optimization solver. It seems, however, worthwhile to implement the algorithm proposed for large instances, since it typically finds good feasible solutions in an early iteration, employing long time steps; after squeezing, these solutions are, actually, often optimal or near-optimal. The verification of optimality is, however, typically done in a later iteration.

ACKNOWLEDGEMENTS

A special thanks to Thomas Eriksson 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, The Swedish Research Council, NFFP (Swedish National Aeronautics Research Programme), and VINNOVA through Chalmers Transport Area of Advance.

REFERENCES

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

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

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

Behnke, D. and Geiger, M. J. (2012), "Test Instances for the Flexible Job Shop Scheduling Problem with Work Centers", available at: http://www.nbn-resolving.de/urn:nbn:de:gbv:705-opus-29827 (Accessed 13 March 2013), Helmut-Schmidt-Universität, Hamburg, Germany.

Berghman, L. (2012), *Machine scheduling models for warehousing docking operations*, Phd thesis, Katholieke Universiteit Leuven, Belgium.

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

Demir, Y. and İşleyen, S. K. (2013), "Evaluation of mathematical models for flexible job-shop scheduling problems", *Applied Mathematical Modelling*, Vol. 37, pp. 977 – 988.

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

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

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

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

Sadykov, R. and 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*, Vol. 18, pp. 209 – 217.

Thörnblad, K. (2011), *On the optimization of schedules of a multitask production cell*, Licentiate thesis, Chalmers University of Gothenburg, Gothenburg, Sweden.

Thörnblad, K., Strömberg, A.-B., Patriksson, M., and Almgren, T. (2013), "Scheduling optimization of a real flexible job shop including fixture availability and preventive maintenance", under review for journal publication, 8 May.

van den Akker, J., Hurkens, C., and Savelsberg, M. (2000), "Time-indexed formulations for machine scheduling problems: Column generation", *INFORMS Journal on Computing*, Vol. 12, pp. 111 – 124.

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

Özgüven, C., Özbakir, L., and Yavuz, Y. (2010), "Mathematical models for jobshop scheduling problems with routing and process plan flexibility", *Applied Mathematical Modelling*, Vol. 34, pp. 1539 – 1548.