A strong integer programming formulation for hybrid flowshop scheduling

A. Tamer Ünal[a] and Semra Ağralı[b] and Z. Caner Taşkın[a]

[a]Department of Industrial Engineering, Boğaziçi University, 34342 Bebek, İstanbul, Turkey
[b]Department of Industrial Engineering, MEF University, 34396 Maslak, İstanbul, Turkey

ARTICLE HISTORY
Compiled July 17, 2019

ABSTRACT
We consider a hybrid flowshop scheduling problem that includes parallel unrelated discrete machines or batch processing machines in different stages of a production system. The problem is motivated by a bottleneck process within the production system of a transformer producer located in the Netherlands. We develop an integer programming model that minimises the total tardiness of jobs over a finite planning horizon. Our model is applicable to a wide range of production systems organised as hybrid flowshops. We strengthen our integer program by exploiting special properties of some constraints in our formulation. We develop a decision support system (DSS) based on our proposed optimisation model. We compare the results of our initial optimisation model with an improved formulation as well as with a heuristic that was in use at the company before the implementation of our DSS. Our results show that the improved optimisation model significantly outperforms the heuristic and the initial optimisation model in terms of both the solution time and the strength of its linear programming relaxation.

KEYWORDS
Hybrid flowshop scheduling; Decision support system; Integer programming; Conflict graph; Co-bipartite chain graph; Interval graph; Maximal clique

1. Introduction

We consider a hybrid flowshop scheduling (HFS) problem motivated by our experience in designing and implementing a decision support system for a company located in the Netherlands. This company is a leading manufacturer of industrial transformers, which are used in a wide range of applications such as residential communities, hospitals and industrial sites. Building a transformer is a complex process that requires coordination of several types of workforce, materials and resources over several months. The work centre that represents the bottleneck of the building process is called Winding. Manufacturing process in Winding is organised as a hybrid flowshop of two stages. The first stage consists of discrete processing machines whereas the second stage consists of batch processing machines. Since Winding constitutes the overall bottleneck of the manufacturing environment, optimal capacity utilisation of the work centre is crucial for efficiency of the company's operations. In this paper we propose an integer

CONTACT S. Ağralı. Email: agralis@mef.edu.tr

programming (IP) approach for multi-stage HFS problem that is applicable in a wide range of industries.

The hybrid flowshop, defined as a multistage flow line with parallel machines at some or all stages, exists in many different industries. The assembly lines in the electronics (Li et al. 2013) and automotive industries (Bellanger and Oulamara 2009); operations in chemical industry (Xuan and Li 2013) and iron&steel manufacturing (Gong, Tang, and Duin 2010) are examples of systems that are designed as hybrid flowshops. In a hybrid flowshop, a job that requires several operations enters the system at the first stage; and, after following a sequence of fixed stages, leaves the system at the last stage. Depending on the requirements of the system intermediate buffers with limited or unlimited capacities may be installed in between stages to hold the work-in-process inventory. Alternatively, the system may have no interstage storage. It is usually assumed that preemption is not allowed. In a hybrid flowshop, a stage may consist of a number of parallel identical or unrelated (nonidentical) machines. If the machines are identical, the particular machine processing a job is not significant; however, for unrelated machines, assignment of jobs to specific machines must be modelled explicitly.

While technological properties of machines (identical or unrelated) are important, processing types of machines also have significant importance in these systems. Based on the processing types we can categorise machines into two groups: discrete or Batch Processing Machines (BPM) (Ahmadi et al. 1992). Discrete machines, such as lathes process one job at a time. BPMs, such as ovens, can process a number of jobs simultaneously. We consider the batch process in the sense that once processing of a batch is started, it cannot be interrupted and other jobs cannot be introduced into the batch. Jobs need to be compatible with each other in order to be processed simultaneously at a BPM. In some cases, all jobs at a BPM require the same amount of processing time. However, if this is not the case, then the processing time of the job in the batch that requires the longest operation time determines the processing time of the whole batch (Cheng, Liu, and Yu 2001). Discrete machines or BPMs may exist at any stage of the system. All these properties affect the scheduling process in these systems.

In this study, we consider an HFS with the following properties: (i) there are multiple stages; (ii) each stage contains parallel unrelated machines, which are either discrete or batch processing; (iii) jobs must be compatible to be processed on a BPM at the same time; (iv) infinite intermediate storage capacity exists in between successive stages; (v) lag time exists between stages for each job, which depends on the stage and job; (vi) preemption is not allowed; and (vii) jobs have dynamic arrival times and due dates.

Li et al. (2015)'s study includes the most closely related problem to the one that is studied in this paper. They consider a HFS problem with unrelated parallel machines at each stage where only one stage includes BPMs. They represent the problem as a nonlinear optimisation model, and then propose a heuristic that aims to minimise maximum completion time and weighted tardiness. Although there exist some studies that consider BPMs that are unrelated (Arroyo and Leung 2017a; Shahvari and Logendran 2017a) and have compatibility restrictions (Bellanger and Oulamara 2009; Kim, Joo, and Shin 2009), they either do not have discrete machines or consider only two stage HFS problems. To the best of our knowledge, there exists no study that considers HFS with unrelated discrete machines or BPMs at any stage of the system and provides a strong MIP.

2. Literature Review

HFS problems are extensively studied in the literature. For a comprehensive review of models and solution algorithms in general HFS problems the reader is referred to Ruiz and Vazquez-Rodriguez (2010); Rossit, Tohmé, and Frutos (2017). Most of the research in HFS focuses on parallel discrete processing machine cases (Ruiz and Vazquez-Rodriguez 2010), and common objectives in these research papers are minimisation of makespan or total completion time (Liu and Karimi 2008). Only in a limited number of papers, surveyed by Ruiz and Vazquez-Rodriguez (2010), minimisation of tardiness is used as an objective function, which is very important in real-life problems.

Heuristics, such as dispatching rules, local search (Lee 2009), and metaheuristics including simulated annealing (Allahverdi and Al-Anzi 2006), tabu search (Shahvari and Logendran 2016) and population-based algorithms (Shahvari and Logendran 2018) are common solution algorithms that are applied to many different problems in HFS. Among exact solution algorithms, branch and bound is the most commonly used one (Ruiz and Vazquez-Rodriguez 2010). Since the problem we consider includes BPM, we focus on the HFS literature that includes BPMs in at least one of the stages. We divide the literature with BPMs into two separate sections and provide the details in these sections.

2.1. *Flow Shop Scheduling with only BPMs*

Scheduling of BPMs mostly focus on single machine cases (Buscher and Shen 2010). Cheng, Liu, and Yu (2001) consider the scheduling of jobs that have release dates and deadlines on a single BPM. They provide polynomial time algorithms for some special cases of this problem. Koh et al. (2005) study a scheduling problem on a single BPM with jobs belonging to different families and having different volumes. Jobs within the same family can be processed simultaneously, and the processing time depends on the family type. They provide an IP formulation and develop some heuristics to solve large problem instances. Buscher and Shen (2010) consider the scheduling problem for parallel BPMs. They develop a mixed-integer programming (MIP) model with the objective of minimising makespan. They propose MIP-based heuristics to solve large instances.

Shahvari and Logendran (2016) study batching and scheduling problem at a HFS where unrelated BPMs exist at bottleneck stages, and the objective is to simultaneously minimise the total weighted completion time and tardiness. Then, they extend this study by considering dynamic machine availability and job release times (Shahvari and Logendran 2017b). Later, in Shahvari and Logendran (2018), they include sequence and machine-dependent family setup times and learning effect. They develop meta-heuristics based on local search and population-based structures as solution methods.

Arroyo and Leung (2017b) study a flowshop scheduling problem of arbitrary sized jobs with nonzero ready times on unrelated parallel BPMs. They develop a MIP, and propose heuristics based on first-fit and best-fit earliest job ready time rules. Then, they extend their work in Arroyo and Leung (2017a) by considering BPMs with different capacities. Li (2017) study a similar problem in which jobs with release times and arbitrary sizes are scheduled on BPMs with non-identical capacities. They propose different approximation algorithms for several special cases of the problem. Tan,

Mönch, and Fowler (2018) consider a two stage flexible flowshop where BPMs exist at each stage and jobs have unequal ready times. They develop a MIP with minimisation of the total weighted tardiness as the objective function, and then propose an iterative stage-based decomposition approach, which includes neighbourhood search techniques.

2.2. *Flow Shop Scheduling with Both Discrete Machines and BPMs*

Flow shop scheduling problems that include both discrete machines and BPMs have also attracted researchers. Among those research papers, two machine problems are studied most extensively. Ahmadi et al. (1992) is the first paper to deal with a HFS problem that includes both discrete and BPMs. They consider scheduling problems in two-machine flowshop systems where there exists at least one BPM. Lin and Liao (2012) study a scheduling problem at a two-stage assembly shop, where in the first stage jobs are assembled simultaneously on a BPM and then moved to the second stage that includes a discrete machine on which all jobs have different processing times. They develop a MIP with the aim of minimising the weighted sum of makespan, total completion time and total tardiness. For solving large instances, they propose three heuristics. Shi, Huang, and Shi (2017) study a similar setting with limited waiting time constraint. They develop a MIP with the objective of minimising makespan. Then, they propose tight lower bounds and a heuristic algorithm, which is followed by a hybrid differential evolution algorithm. For a similar setting, Li and Dai (2019) propose tight lower bounds and three heuristics. Moreover, in order to increase the efficiency, they employ a neighbourhood search algorithm. Chung, Sun, and Liao (2017) consider dynamic job arrival times where jobs are grouped into several batches. They propose two metaheuristics to solve the problem with the objective of minimising makespan.

Several studies consider a HFS with more than two machines. Bellanger and Oulamara (2009) consider a two stage HFS with parallel identical discrete machines at the first stage and parallel identical BPMs in the second one. They develop several heuristics and provide worst case analysis. Kim, Joo, and Shin (2009) study a two stage HFS subject to a product-mix ratio constraint, which requires certain job types to be kept in the same batch. The first stage includes parallel identical discrete machines and the second stage includes a single BPM. Amin-Naseri and Beheshti-Nia (2009) consider a HFS where at some stages it is possible to process jobs at the same time as a batch on some machines. The processing time of a batch is equal to the longest processing time requirement of a job in the batch, and job ready times, set up times, and transportation times between stages are zero. They develop several heuristics and give a lower bound for evaluating the performances of the heuristics. Zheng (2010) study a HFS that includes parallel BPMs in a stage and discrete machines in other stages where jobs to be processed have arbitrary sizes. They formulate the problem as a MIP; however, since it takes a very long time to solve even small sized problems to optimality, they develop heuristics inspired by the shifting bottleneck heuristic.

Our contribution to the literature can be summarised as follows: (i) we consider a multi-stage HFS with parallel unrelated discrete machines and BPMs that may exist at any stage of the system; (ii) jobs have machine eligibility restrictions; (iii) we develop a strong IP that can solve practical instances in a reasonable amount of time; and (iv) lag times exist between stages and depend on the job and the stage.

3. Problem Definition and Mathematical Model

The manufacturing system that we consider consists of a set of machines (resources) R, which process a set of jobs J, over a predetermined planning horizon consisting of a set of discrete time periods, T. Jobs follow a set of stages S, where each stage $s \in S$ includes a set of parallel unrelated machines, $R(s) \subseteq R$. Jobs have machine eligibility restrictions; job $j \in J$ at stage $s \in S$ may be processed on a set of machines, $R(j,s) \subseteq R(s)$. Each stage may include either a set of discrete machines or a set of BPMs. We denote the set of stages that include BPMs by $S_b \subseteq S$. For each BPM there is a set of configuration options, $C(s,r)$, on which that machine may operate. These configurations define operating modes of the machine (such as processing time, temperature or pressure), and restrict jobs that can be processed on these configurations. Similarly, each job requires a configuration option, c_{jr}, with a processing time of b_{cr}, to be processed on a BPM.

Each job has a release date, r_j, and a due date, d_j. Since the machines at each stage are unrelated, the processing time of a job on a machine, p_{jr}, depends on the machine to which the job is assigned. After any stage there might be a lag time defined for a job, l_j^s, that denotes the amount of time the job needs to wait before being processed at the following stage. A discrete machine may process one job at a time; however, BPMs may process multiple jobs having the same configuration requirement simultaneously. Let u_{jr} indicate the ratio of the capacity that job j allocates if it is processed on machine r. Note that u_{jr} is equal to one for discrete machines, and $0 < u_{jr} \leq 1$ for BPMs.

Our aim is to find an optimal schedule for this HFS problem such that the total tardiness is minimised. Let us define a binary variable x_{jrt} for each job $j \in J$, stage $s \in S$, machine $r \in R(j,s)$ and time $t \in T, t \geq r_j$, whose value is equal to one if job j is assigned to machine r in stage s starting at time t; and zero otherwise. We provide all parameters and decision variables, with their definitions, in Table 1.

Table 1. Symbols used in our mathematical model.

Set	**Description**
J	set of jobs
R	set of machines
S	set of stages
$\bar{S}$	set of all stages except the final stage
S_b	subset of stages where the production process is batch-based, $S_b \subseteq S$
$R(s)$	subset of machines on stage $s \in S$, $R(s) \subseteq R$
$R(j,s)$	subset of machines to which job $j \in J$ can be assigned on stage $s \in S$, $R(j,s) \subseteq R(s)$
T	set of discrete time periods in the planning horizon
$C(r)$	set of configuration options that BPM $r \in R(s), s \in S_b$ can operate
Parameter	**Description**
r_j	release date of job $j \in J$
d_j	due date of job $j \in J$
p_{jr}	processing time of job $j \in J$ if it is processed on stage $s \in S$ by machine $r \in R(j,s)$
u_{jr}	capacity usage ratio of job $j \in J$ if it is processed on stage $s \in S$ on machine $r \in R(j,s)$
l_j^s	minimum lag time after processing job $j \in J$ on stage $s \in S$
b_{cr}	processing time of configuration option $c \in C(r)$ associated with BPM $r \in R(s)$ at stage $s \in S_b$
c_{jr}	configuration option that job $j \in J$ requires on stage $s \in S_b$ if it is processed by BPM $r \in R(j,s)$ (note that $c_{jr} = \hat{c}$ implies that $p_{jr} = b_{\hat{c}r}$)
Variable	**Description**
x_{jrt}	binary variable that represents if job $j \in J$ is assigned to machine $r \in R(j,s)$ on stage $s \in S$ such that its processing starts at time $t \in T, t \geq r_j$
y_{crt}	binary variable that represents if configuration $c \in C(r)$ is operated on BPM $r \in R(s)$ at stage $s \in S_b$ such that its processing starts at time $t \in T$

A job must be processed by a single machine at any stage of the process. Constraint

set (1) ensures that each job is assigned to exactly one machine at each stage

$$\sum_{\substack{t\in T\\ t\geq r_j}} \sum_{r\in R(j,s)} x_{jrt} = 1 \quad \forall j \in J, s \in S. \tag{1}$$

Each machine has a limited capacity. Constraint set (2) ensures that the capacity of resources are not exceeded

$$\sum_{j\in J} \sum_{t-p_{jr}<t'\leq t} u_{jr}x_{jrt'} \leq 1 \quad \forall t \in T, r \in R. \tag{2}$$

Recall that if a machine is discrete, then u-parameters take the value of 1, which provides the assignment of at most one job to that machine during the processing time of that job. For BPMs u-parameters take value between 0 and 1 representing the percentage of the capacity used by that job.

Since we consider a multi-stage production environment, there is a precedence relationship between stages. We assume that a job is processed on each stage at a machine, and processing at a stage can only start if the process in the previous stage has finished and a lag time has passed (if a lag time exists). Therefore, we need to write for every $(\bar{r}, \hat{r})$ pair a constraint that ensures that a job is not scheduled to a machine at a stage if it is not scheduled to any machine in the predecessor stage and that processing plus lag time in that stage is finished. These constraints are given as

$$x_{j\hat{r}\hat{t}} + x_{j\bar{r}\bar{t}} \leq 1 \quad \forall s \in \bar{S}, \hat{t} \in T, \bar{t} \in T, \bar{t} \leq \hat{t} + p_{j\hat{r}} + l_j^s, \hat{r} \in R(j,s), \bar{r} \in R(j,s+1), \tag{3}$$

where $\bar{S}$ includes all stages except the final one. The number of such constraints is $O(|J||T|^2|R|^2)$, which grows at a quadratic rate with the number of time periods $|T|$ and the number of machines $|R|$, and can reach hundreds of thousands for a medium-sized problem instance. However, it is possible to write tighter constraints that can handle the precedence relationship by using *conflict graphs*, which represent logical relationships between binary variables, to improve solvability of IP problems (Atamturk, Nemhauser, and Savelsbergh 2000). Let us create an undirected graph $G(j,s)$ corresponding to stages s and $s+1$ of job j. We represent each assignment variable $x_{j\hat{r}\hat{t}}$ at stage $s \in S$ and $x_{j\bar{r}\bar{t}}$ at stage $s+1 \in S$ by a vertex in $G(j,s)$. We add an edge $(x_{j\bar{r}\bar{t}}, x_{j\hat{r}\hat{t}})$ if and only if assigning stage s of job j to machine $\bar{r}$ to start at time $\bar{t}$ and its stage $s+1$ to machine $\hat{r}$ to start at time $\hat{t}$ violates constraints (3). Moreover, since a job must be assigned to a single machine at each stage (Constraints (1)), we add an edge between every pair of $x_{j\hat{r}t}$ vertices (and also add an edge between every pair of $x_{j\bar{r}t}$ vertices).

As an example consider a job at stages s and $s+1$, and two resources $\hat{r} \in R(s)$ and $\bar{r} \in R(s+1)$. The possible starting and finishing times of the job considered at each stage and resource pair is given in Figure 1, assuming that the processing times at resources in both stages take single time period and there is no lag time between stages. If the job is not finished processing in stage s by the end of time period t_2 for the example, then it cannot be assigned to be processed on resource $\bar{r}$ in stage $s+1$ at time periods t_1 and t_2. For this example we give the conflict graph representation in Figure 2. Solid edges between vertices correspond to constraints (3). Moreover, since a job must be assigned to a single resource at each stage (constraints (1)), we add an edge between every pair of $x_{j\hat{r}t}$ vertices (and also add an edge between every pair

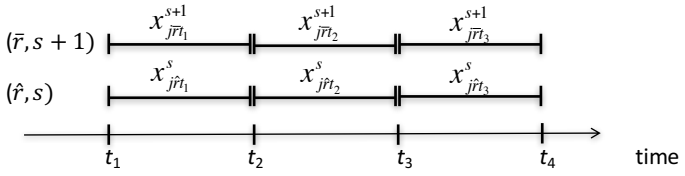

Figure 1. Example showing possible assignments of a job in two successive stages.

of $x_{j\bar{r}t}$ vertices). Dashed edges between vertices in Figure 2 correspond to constraints (1).

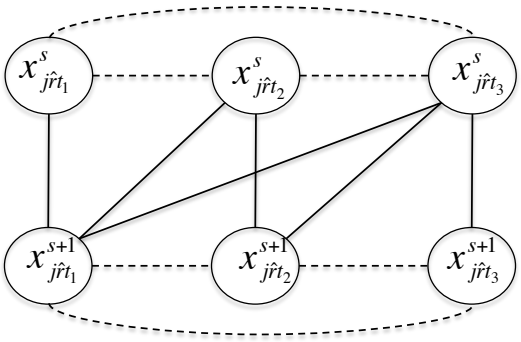

Figure 2. Conflict graph representation of the example given in Figure 1.

Given an undirected graph, $G(j, s)$ corresponding to stages s and $s+1$ of job j, an *independent set* is defined as a set of vertices that are pairwise non-adjacent. We observe that any feasible assignment of j to machines in stages s and $s+1$ given by constraints (1) and (3) corresponds to an independent set (also called *node packing* or *stable set*) on $G(j, s)$. The problem of finding an independent set having maximum cardinality is NP-Hard in general (Garey and Johnson 1979). However, we observe that $G(j, s)$ is a special kind of graph called *co-bipartite chain* (Boyacı, Ekim, and Shalom 2015), and its maximum independent set size is 2. The maximum independent set problem has been studied extensively from graph theoretical and MIP points of view (e.g., Beigel 1999; Johnson, Yannakakis, and Papadimitriou 1988; Padberg 1973). Constraints (3) correspond to edge-based formulation of the independent set polytope given in Padberg (1973). Then, constraint set (3) can be replaced by

$$\sum_{\hat{t}\geq t-p_{j\hat{r}}-l^s_j} \sum_{\hat{r}\in R(j,s)} x_{j\hat{r}\hat{t}} + \sum_{\bar{t}\leq t} \sum_{\bar{r}\in R(j,s+1)} x_{j\bar{r}\bar{t}} \leq 1 \quad \forall j \in J, t \in T, s \in \bar{S}, \tag{4}$$

which dominates the corresponding constraint in (3). This reformulation decreases the number of constraints from $O(|J||T|^2|R|^2)$ to $O(|S||J||T|)$. Note that for the example given in Figures 1 and 2, clique-based constraints (4) can be expressed for the example given in Figures 1 and 2 as:

$$x_{j\hat{r}t_1} + x_{j\hat{r}t_2} + x_{j\hat{r}t_3} + x_{j\bar{r}t_1} \leq 1$$
$$x_{j\hat{r}t_2} + x_{j\hat{r}t_3} + x_{j\bar{r}t_1} + x_{j\bar{r}t_2} \leq 1$$
$$x_{j\hat{r}t_3} + x_{j\bar{r}t_1} + x_{j\bar{r}t_2} + x_{j\bar{r}t_3} \leq 1.$$

Recall that each BPM r has a number of configuration options that it can operate, $C(r)$. We require a binary variable, y_{crt}, for each BPM $r \in R(s), s \in S_b$, configuration option, $c \in C(r)$ and time period, $t \in T$, that takes value one if at stage s BPM r is set to start operation on configuration option c at time period t; and zero otherwise. Constraint set (5) ensures that at any time a BPM can be set to operate at a single configuration option

$$y_{\hat{c}r\hat{t}} + y_{\bar{c}r\bar{t}} \leq 1 \quad \forall s \in S_b, r \in R(s), \hat{t} \in T, \bar{t} \in \{\hat{t}, \ldots, \hat{t} + b_{\hat{c}r} - 1\}, \hat{c} \in C(r), \bar{c} \in C(r). \tag{5}$$

Constraints (5) are written for every BPM, configuration pair and time period, resulting in $O(|T||C|^2)$ number of constraints. By using a similar logic that we used above, it is possible to have tighter constraints. For every BPM $r \in R(s)$, where $s \in S_b$, let us represent each configuration option assignment variable y_{crt} $c \in C(r), t \in T$ by a vertex (c, t) in an undirected graph $G(r)$. We add an edge $(\bar{c}t, \hat{c}t')$ if and only if assigning both $(\bar{c}, t)$ and $(\hat{c}, t')$ to machine r violates constraints (5) (i.e., the two configuration option assignments overlap).

As an example consider a job $j \in J$ at a batch processing stage $s \in S_b$. Let there be two different configuration options: c_1 and c_2 with durations of 2 and 3 time periods, respectively. Assume that the planning horizon we consider includes 5 periods. Figure

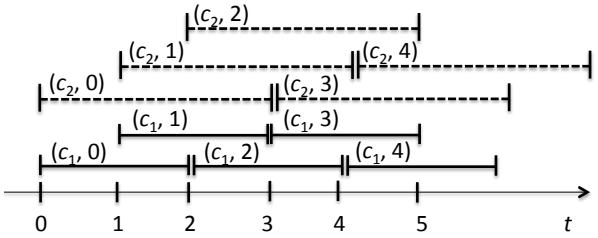

Figure 3. Example showing possible assignments of configuration options on a batch processing resource.

3 shows all possible assignment of these configuration options to a resource, r. For this example $G(r)$ can be given in Figure 4.

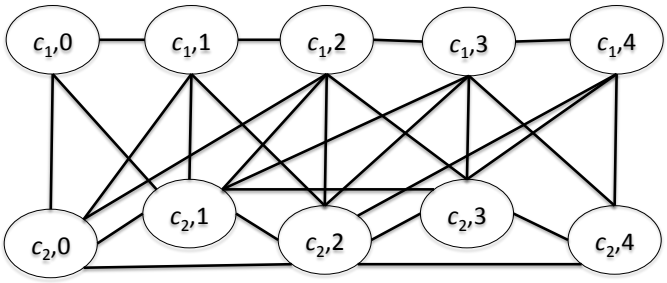

Figure 4. Conflict graph representation of the example given in Figure 3.

Similar to our discussion above, we observe that any feasible assignment of configuration options on machine r corresponds to an independent set on $G(r)$. As before, constraints (5) correspond to edge-based formulation of the independent set problem, and the formulation can be improved by considering maximal cliques of $G(r)$. Note that $G(r)$ is a special type of graph called *interval graph* since each vertex represents a time interval and two vertices are adjacent if and only if their time intervals overlap

(Golumbic 1980). Maximal cliques of interval graphs can be enumerated in polynomial time (Golumbic 1980). However, an explicit search for maximal cliques is not needed in our case since a maximal clique on $G(r)$ corresponds to a time period t, and the set of configuration option - time pairs (c, t) overlapping with t in our problem. Thus, constraints (5) can be replaced by

$$\sum_{c \in C(r)} \sum_{t - b_{cr} + 1 \leq \hat{t} \leq t} y_{cr\hat{t}} \leq 1 \quad \forall s \in S_b, r \in R(s), t \in T. \tag{6}$$

This reformulation decreases the number of constraints from $O(|T||C|^2)$ to $O(|T||R|)$ while also tightening the formulation. Note that clique-based constraints (6) can be written for the example represented in Figures 3 and 4 as:

$$\begin{aligned} y_{c_1 r0} + y_{c_1 r1} + y_{c_2 r0} + y_{c_2 r1} &\leq 1 \\ y_{c_1 r1} + y_{c_1 r2} + y_{c_2 r0} + y_{c_1 r1} + y_{c_2 r2} &\leq 1 \\ y_{c_1 r2} + y_{c_1 r3} + y_{c_2 r1} + y_{c_2 r2} + y_{c_2 r3} &\leq 1 \\ y_{c_1 r3} + y_{c_1 r4} + y_{c_2 r2} + y_{c_2 r3} + y_{c_2 r4} &\leq 1. \end{aligned}$$

Constraint set (7) ensures that a job can be assigned to a BPM only if the configuration option of that BPM at the time of assignment is the same as the configuration requirement of that job

$$x_{jrt} \leq y_{\hat{c}rt} \quad \forall j \in J, s \in S_b, r \in R(j, s), \hat{c} \in C(r), \hat{c} = c_{jr}, t \in T. \tag{7}$$

Our objective is to minimize the total tardiness of the jobs, which is calculated by the finishing time of a job less its due date if the finishing time is greater than the due date of the job. If the job is finished before its due date, then the tardiness of that job is zero. Our integer linear program can be written as follows.

$$\text{Minimize} \quad \sum_{j \in J} \sum_{r \in R(j,|S|)} \sum_{\substack{t \in T \\ t > d_j - p_{jr}}} (t + p_{jr} - d_j) x_{jrt} \tag{8}$$

$$\text{subject to} \quad \sum_{\substack{t \in T \\ t \geq r_j}} \sum_{r \in R(j,s)} x_{jrt} = 1 \quad \forall j \in J, s \in S, \tag{9}$$

$$\sum_{j \in J} \sum_{t - p_{jr} < t' \leq t} u_{jr} x_{jrt'} \leq 1 \quad \forall t \in T, r \in R, \tag{10}$$

$$\sum_{\hat{t} \geq t - p_{j\hat{r}} - l_j^s} \sum_{\hat{r} \in R(j,s)} x_{j\hat{r}\hat{t}} + \sum_{\bar{t} \leq t} \sum_{\bar{r} \in R(j,s+1)} x_{j\bar{r}\bar{t}} \leq 1 \quad \forall j \in J, t \in T, s \in \bar{S}, \tag{11}$$

$$\sum_{c \in C(r)} \sum_{t - b_{cr} + 1 \leq \hat{t} \leq t} y_{cr\hat{t}} \leq 1 \quad \forall s \in S_b, r \in R(s), t \in T, \tag{12}$$

$$x_{jrt} \leq y_{\hat{c}rt} \quad \forall j \in J, s \in S_b, r \in R(j, s), \hat{c} \in C(r), \hat{c} = c_{jr}, t \in T, \tag{13}$$

$$x_{jrt} \in \{0, 1\} \quad \forall j \in J, s \in S, r \in R(j, s), t \in T, \tag{14}$$

$$y_{crt} \in \{0, 1\} \quad \forall s \in S_b, r \in R(s), c \in C(r), t \in T. \tag{15}$$

4. Application and Computational Analysis

In this section, we first provide some information about the process that motivates the problem studied in this paper, and explain the planning heuristic used in the company before our study. We then discuss the results of our computational study to compare our formulations and the company heuristic.

4.1. *Setting and Company's Approach*

As discussed in Introduction, the work center that represents the bottleneck for the building process is called Winding. Manufacturing process in Winding has a hybrid flowshop structure consisting of two stages. The first stage consists of 14 winding benches, which are discrete processing machines. These benches have been acquired over a number of years, and have different technical capabilities and processing speeds. After the first stage, some transformers need to wait in the buffer for up to 24 hours before they can be processed in the second stage, which consists of the drying and pressing operations. The second stage consists of two furnaces, which are batch processing machines. Transformers requiring the same configuration option can be placed together in a furnace, as long as the area required by the batch of jobs does not exceed the total processing area of the furnace.

Before our study, the planners in the company were manually planning the Winding work center using a heuristic approach as shown in Algorithm 1. We call this heuristic as Company's Approach. This approach is a constructive greedy approach based on the idea of first planning winding benches (first stage) and then furnaces (second stage).

Algorithm 1 Company Approach

$x \leftarrow 0$
t_r: a lower bound on earliest time that a job can be assigned to machine r
$t_r \leftarrow 1 \quad \forall j \in R$
for all $s \in S$ **do**
  while $\exists$ an unscheduled job j **do**
    $r^* \leftarrow argmin_{r \in R(s)} t_r$
    if $\exists$ job j that can be scheduled on r^* at t_{r^*} **then**
      schedule job j on r^* to start at t_{r^*}
      $x_{jr^*t_{r^*}} \leftarrow 1$
      update available capacity of r^*
      eliminate conflicting job and configuration option assignments (x- and y-variables)
    else
      $t_r \leftarrow t_r + 1$
    end if
  end while
end for

We observe that the company approach ensures that i) no machine stays idle while there are jobs that it can possibly process (i.e. the generated schedule is non-delay), and ii) no job can start processing earlier in a different machine. These characteristics make it easier to execute the plan, and the heuristic produces reasonable schedules in practice. However, as we demonstrate in the next section, the heuristic approach may produce suboptimal solutions in terms of conformance to due dates and capacity

utilization.

We developed a DSS based on our optimization model for the Winding work center to replace the manual planning process. In our implementation each transformer corresponds to a job $j \in J$. There are two processing stages, where the first stage is discrete and the second stage is batch. Each job has a release date r_j and due date d_j. We model the eligibility constraints, which indicate that certain transformers are not compatible with certain winding machines by defining sets $R(j,s)$ appropriately. Furthermore, we model differences between processing speeds of machines by using p_{jr} parameters. We set capacity usage ratio parameters u_{jr} to 1 for the first stage, and ratio of each job's area requirement to the furnace's area for the second stage. We use l_j^s parameters to model waiting time between the two stages, and define configuration options $C(r)$.

Our DSS for Winding was implemented using ICRON Advanced Planning and Scheduling system (www.icrontech.com). ICRON provides a visual algorithm modeling environment that is based on the object-oriented design paradigm. It has support for database, ERP systems integration and has interfaces to various MIP solvers including GLPK, Cbc, CPLEX and Gurobi (see Taşkın et al. 2015; Ağralı, Taşkın, and Unal 2017).

4.2. *Computational Analysis*

We test the performance of the initial mathematical model and the improved one given in the previous section on problems of various sizes and parameter configurations.

4.2.1. Data Generation

As discussed in Section 4.1 the Winding production process of the company includes two stages: first stage includes nonidentical discrete machines, and the second stage includes batch processing machines. The company currently has 14 discrete and 2 batch processing machines. Therefore, we use these numbers in our computational study.

We analyzed the data provided by the company and conclude that the processing time of jobs on discrete and batch machines are between 3 and 5 time periods. The lag time of jobs after stage one are between 0 and 2 time periods. The capacity usage values of jobs in batch processing machines are between 20% and 30%. Based on these values, in all problem sets, we randomly generate average processing times of jobs in discrete and batch processing stages using uniform distributions over the integers 3 and 5. Let this average processing time for job j in stage s be pr_{js}. Based on this average processing time, we generate the processing time of the job on each machine at a given stage by using a uniform distribution over the integers $pr_{js}-1$ and $pr_{js}+1$. We generate lag times of jobs using a uniform distribution over the integers 0 and 2, and the capacity usage ratios on batch processing machines using a uniform distribution between 0.2 and 0.3.

We follow the data generation process of (Schaller 2007) for generating the release date and due date of jobs. Let R and r be the parameters called due date range and tardiness factors. We generate the due dates for the jobs by using a uniform distribution over the integers $T(1-r-R/2)$ and $T(1-r+R/2)$, where T is the total number of time periods that we consider. We create three different sets: set 1 includes $R = 0.5$ and $r = 0.5$, set 2 includes $R = 1$ and $r = 0.5$, and set 3 includes $R = 1$ and $r = 0.25$. Let DD_j be the due date of job j, and MTP_j be the maximum possible

Table 2. Problem Sets.

set #	R	r	α
set1	0.5	0.5	1.5
set2	0.5	0.5	2.5
set3	0.5	0.5	3.5
set4	1.0	0.5	1.5
set5	1.0	0.5	2.5
set6	1.0	0.5	3.5
set7	1.0	0.25	1.5
set7	1.0	0.25	2.5
set9	1.0	0.25	3.5

total processing time of job j. Then for release date calculation we use the formulation $DD_j - \alpha MTP_j$. In this formulation we use three different α values: 1.5, 2.5 and 3.5. We provide R, r, and α values of data sets created in Table 2.

The planning horizon at the company is usually 30 time periods and, on the average, total number of jobs to be scheduled within the planning horizon is around 30. Therefore, we take $|T| = 30$ and $|J| = 30$ for medium size instances. Then, we generate larger instances by taking $|T| = 30$ and $|J| = 40$, $|T| = 40$ and $|J| = 40$, and $|T| = 40$ and $|J| = 50$, resulting in four different size of test problems each having nine different configuration for release and due dates. We randomly generate a total of 36 problem sets in total, each set consisting of five problems.

4.2.2. Comparison of IP Formulations

In this section, we compare the Initial IP obtained by a model that includes constraints (3) and (5) (instead of (4) and (6)) with the Improved IP that we give at the end of Section 3. We implemented both IP formulations in Java, and performed all tests on a machine with Intel Xeon 2.27 GHz CPU, 24 GB RAM and Windows 2008 Server R2 operating system. We used CPLEX 12.6.3 as the solver and enforced a time limit of 1200 seconds. For many instances CPLEX reaches optimality within the given time limit. For those instances that CPLEX does not prove optimality, we use the best feasible solution provided by CPLEX while calculating performance measures in this section.

We provide the results of our first experiment in Table 3. In Table 3, 'unsolved/Gap' column gives the number of instances that are not solved to optimality within the given time limit and the average value of the gap for the corresponding instances that are not solved to optimality; 'IP Time' represents the average of the time that CPLEX spent to solve instances; 'Tot. Time' represents the total time spent, which is the average of total time spent to build and solve the model; 'LP Gap' column gives the average of the difference between the objective function values provided by IP and its linear programming relaxation.

Table 4 shows the percentage difference between the results obtained by Initial IP and Improved IP formulations in terms of the same metrics given in Table 3. As seen from Table 4, Improved IP significantly outperforms the Initial IP formulation for all problem sizes. The improvement in LP Gap values reveal the tightness of the Improved IP in terms of its LP relaxation. The highest difference is achieved for average IP time since the improved formulation has a tighter formulation and significantly fewer number of constraints.

Table 3. Computational Results of IP Formulations.

			Initial IP				Improved IP			
			unsolved/ Gap	IP Time	Tot. Time	LP Gap	unsolved/ Gap	IP Time	Tot. Time	LP Gap
$\|T\|$	$\|J\|$	set #	(%)	(sec)	(sec)	(%)	(%)	(sec)	(sec)	(%)
30	30	set1	-/-	505.41	531.59	80.00	-/-	27.30	42.42	38.64
30	30	set2	3/43.1	893.29	920.05	60.00	-/-	165.03	180.13	44.75
30	30	set3	4/48.9	1028.71	1055.67	100.00	-/-	88.28	104.25	71.71
30	30	set4	-/-	24.70	49.32	0.00	-/-	0.89	14.89	0.00
30	30	set5	-/-	48.37	72.50	60.00	-/-	0.90	15.27	16.67
30	30	set6	-/-	52.33	78.54	40.00	-/-	0.98	16.08	0.00
30	30	set7	-/-	103.63	128.68	72.77	-/-	4.54	19.03	4.54
30	30	set8	-/-	198.90	225.23	70.57	-/-	10.65	25.37	8.63
30	30	set9	1/4.9	416.19	444.76	74.83	-/-	14.00	28.69	11.73
30	40	set1	5/84.4	1200	1245.10	100.00	4/26.5	1075.54	1109.17	84.11
30	40	set2	5/90.2	1200	1248.72	100.00	5/58.7	1200	1234.78	84.93
30	40	set3	5/92.7	12006	1245.85	100.00	5/74.31	1200	1240.02	93.01
30	40	set4	-/-	413.03	456.10	40.00	-/-	18.02	50.01	25.27
30	40	set5	-/-	333.29	375.69	40.00	-/-	10.33	45.79	0.00
30	40	set6	1/20.0	582.80	626.25	40.00	-/-	19.93	59.27	20.00
30	40	set7	5/30.9	1200	1247.38	84.42	-/-	182.76	216.10	23.05
30	40	set8	1/5.5	861.11	906.20	78.59	-/-	84.61	120.40	13.88
30	40	set9	4/32.1	1078.49	1122.32	82.01	-/-	379.13	417.72	19.60
40	40	set1	5/100.0	1200	1277.91	100.00	3/59.7	849.88	906.53	100.00
40	40	set2	2/37.10	745.62	818.22	40.00	-/-	109.19	170.17	16.49
40	40	set3	3/60.00	1013.66	1086.55	80.00	1/20.0	270.23	332.24	37.89
40	40	set4	-/-	139.84	210.04	80.00	-/-	8.01	64.34	55.00
40	40	set5	-/-	70.70	140.89	0.00	-/-	1.29	59.63	0.00
40	40	set6	-/-	76.25	145.84	0.00	-/-	1.77	59.05	0.00
40	40	set7	-/-	225.96	300.16	76.67	-/-	9.15	67.40	5.68
40	40	set8	2/5.67	735.20	806.34	71.13	-/-	21.69	76.39	5.80
40	40	set9	-/-	271.29	342.08	65.56	-/-	7.79	66.41	2.45
40	50	set1	5/97.66	1200	299.13	100.00	5/73.6	1200	1289.40	94.95
40	50	set2	5/100.00	1200	1305.88	100.00	1/20.0	333.20	416.85	20.00
40	50	set3	5/100.00	1200	1299.83	100.00	4/80.0	983.14	1068.42	80.00
40	50	set4	-/-	368.85	462.11	95.00	-/-	152.41	235.74	43.77
40	50	set5	-/-	226.92	321.59	0.00	-/-	3.68	88.59	0.00
40	50	set6	-/-	172.70	269.11	0.00	-/-	4.53	89.30	0.00
40	50	set7	3/15.00	818.88	918.23	81.40	-/-	90.46	163.83	12.49
40	50	set8	4/19.05	1138.17	1238.91	80.42	-/-	253.40	339.76	13.84
40	50	set9	5/29.58	1200	1307.02	81.32	-/-	382.37	464.36	15.14

Table 4. Percent Difference Between Results of Initial IP and Improved IP Formulations.

		unsolved	Gap	IP Time	Tot. Time	LP Gap
$\|T\|$	$\|J\|$	(%)	(%)	(%)	(%)	(%)
30	30	100	100	94.12	84.26	67.93
30	40	60	74.77	59.92	56.89	50.66
40	40	76.67	76.74	85.38	68.10	60.48
40	50	66.67	70.77	64.39	54.90	58.26

4.2.3. Comparison of IP with Company's Approach

We compare the results obtained by applying the heuristic that was in use by the company (Algorithm 1) and the IP formulation provided in the previous section. For this experiment we also implemented the heuristic in Java and executed it on the same set of problem instances.

We give the results of the heuristic and the Improved IP in Table 5. The 'Time' column of Table 5 shows the average of time required to solve five instances; 'Tardiness' column gives the average tardiness obtained, which is calculated by the total tardiness divided by the number of jobs; 'Earliness' column represents the average earliness, which is calculated as the average of the times that the jobs start being processed divided by the total number of jobs considered to be scheduled; and finally 'Flow' column gives the average flowtime, which is calculated as the time between start time of the job's operation on first stage and finish time of the job's operation on the second stage.

In order to compare our IP formulation with the heuristic, we calculate the percentage differences for the average tardiness, average earliness and average flowtime for different sizes of data sets. We present these percentages in Table 6. As seen from Table 6, Improved IP significantly outperforms the heuristic in terms of tardiness for all problem sizes. Heuristic schedules some instances earlier compared to Improved IP and the average flowtime is less for heuristic in most of the cases. Although IP's average flowtime is less than the heuristic's for problem size for $|T| = 30$ and $|J| = 40$, this is the result of large difference for some of the instances of this set, as seen in Table 5 for set1 and set2.

5. Conclusions

In this paper we investigated a hybrid flowshop scheduling problem motivated by a real-world planning problem at a manufacturer located in the Netherlands. There exist multiple stages with discrete and BPMs through which jobs are processed within a finite time horizon. There are predecessor-successor relationship between stages, which requires the predecessor process of a job to be finished and a lag time, if any, to pass before the process of the job in the next stage starts. BPMs operate at different configuration options that may be defined by the temperature, pressure, duration or any property of the process. Each job requires a configuration option to be processed at a BPM at a given time, and allocates some percentage of the capacity of the BPM. Therefore, multiple jobs that require the same configuration option may be processed on the same machine simultaneously as long as capacity of the machine is not exceeded. We assume that preemption is not allowed for any machine in discrete or batch stages. Our objective is to minimize the total tardiness of jobs. We developed an integer programming model for this problem, and then proposed an improved formulation based on graph representations of certain constraints. Our computational experiments show that our improved formulation reduces the number of constraints while also significantly improving its linear programming relaxation and solvability.

Table 5. Comparison of Heuristic's and Improved IP's Results.

			Heuristic				Improved IP			
$\|T\|$	$\|J\|$	set #	Time	Tardiness	Earliness	Flow	Time	Tardiness	Earliness	Flow
30	30	set1	0.01	1.31	10.54	4.31	27.30	0.15	8.71	5.74
30	30	set2	0.02	1.09	2.18	13.57	165.03	0.09	3.79	11.47
30	30	set3	0.03	0.93	2.49	13.19	88.28	0.26	2.33	11.80
30	30	set4	0.01	0.11	13.91	8.17	0.89	0.00	11.99	9.13
30	30	set5	0.01	0.07	12.02	9.18	0.90	0.03	7.99	13.43
30	30	set6	0.01	0.08	11.44	11.07	0.98	0.02	5.10	15.81
30	30	set7	0.01	2.20	6.33	9.21	4.54	1.13	7.85	8.90
30	30	set8	0.02	2.63	3.17	12.45	10.65	1.34	3.01	12.46
30	30	set9	0.02	2.39	3.87	12.68	14.00	1.47	1.67	14.02
30	40	set1	0.02	2.69	3.07	12.80	1075.54	0.61	5.64	9.47
30	40	set2	0.04	2.78	2.13	15.72	1200	0.66	3.07	12.21
30	40	set3	0.04	2.11	2.18	15.44	1200	0.62	2.92	12.98
30	40	set4	0.01	0.19	14.18	9.01	18.02	0.05	13.89	8.58
30	40	set5	0.02	0.13	10.52	11.27	10.33	0.02	9.10	13.39
30	40	set6	0.03	0.21	8.53	13.91	19.93	0.02	6.03	15.66
30	40	set7	0.02	2.17	8.04	10.40	182.76	1.47	6.45	9.39
30	40	set8	0.04	3.25	4.40	13.84	84.61	1.33	3.61	12.51
30	40	set9	0.04	2.74	3.08	14.45	379.13	1.35	2.15	14.21
40	40	set1	0.01	0.36	10.59	9.18	849.88	0.25	10.79	9.48
40	40	set2	0.03	0.28	6.61	12.79	109.19	0.05	6.77	13.01
40	40	set3	0.05	0.66	4.53	15.13	270.23	0.02	4.67	14.95
40	40	set4	0.01	0.00	20.50	8.27	8.01	0.08	21.32	7.59
40	40	set5	0.01	0.00	21.66	8.69	1.29	0.00	15.85	13.18
40	40	set6	0.02	0.01	18.01	10.93	1.77	0.00	11.87	16.91
40	40	set7	0.01	1.80	10.70	9.24	9.15	0.65	12.92	8.99
40	40	set8	0.03	1.94	9.33	11.20	21.69	1.01	7.21	13.14
40	40	set9	0.04	1.25	8.25	12.88	7.79	0.76	5.21	16.21
40	50	set1	0.01	1.55	8.36	11.27	1200	0.59	10.00	10.14
40	50	set2	0.04	1.40	5.44	15.21	333.20	0.04	7.72	12.84
40	50	set3	0.05	1.50	3.27	17.69	983.14	0.10	4.36	15.37
40	50	set4	0.01	0.13	20.32	8.50	152.41	0.06	22.51	7.67
40	50	set5	0.02	0.00	20.63	9.27	3.68	0.00	17.58	12.71
40	50	set6	0.04	0.18	15.60	12.34	4.53	0.00	12.32	17.07
40	50	set7	0.02	2.66	9.90	10.33	90.46	1.02	11.60	9.06
40	50	set8	0.04	2.22	8.63	13.21	253.40	1.25	7.59	13.64
40	50	set9	0.05	2.42	6.30	16.22	382.37	1.15	4.00	15.95

Table 6. Comparison of Improved IP Formulation with the Heuristic (% Differences).

$\|T\|$	$\|J\|$	Tardiness (%)	Earliness (%)	Flowtime (%)
30	30	68.33	4.44	-7.20
30	40	68.41	-5.50	6.52
40	40	65.94	9.78	-15.85
40	50	70.76	-4.28	-1.78

Acknowledgements

We would like to thank Anita Graumans (DDCgroup B.V.), Taner Sezgin, Mahir Yüksel, Seda Lafcı (ICRON Technologies) for their valuable contributions in the implementation project. We also thank Tınaz Ekim (Boğaziçi University) for helpful discussions.

References

Ahmadi, J.H., R.H. Ahmadi, S. Dasu, and C.S. Tang. 1992. "Batching and scheduling jobs on batch and discrete processors." *Operations Research* 39 (4): 750–763.

Allahverdi, A., and F.S. Al-Anzi. 2006. "Scheduling multi-stage parallel-processor services to minimize average response time." *Journal of the Operational Research Society* 57 (1): 101–110.

Amin-Naseri, Mohammad Reza, and Mohammad Ali Beheshti-Nia. 2009. "Hybrid flow shop scheduling with parallel batching." *International Journal of Production Economics* 117 (1): 185–196.

Arroyo, José Elias C, and Joseph Y-T Leung. 2017a. "An effective iterated greedy algorithm for scheduling unrelated parallel batch machines with non-identical capacities and unequal ready times." *Computers & Industrial Engineering* 105: 84–100.

Arroyo, José Elias C, and Joseph Y-T Leung. 2017b. "Scheduling unrelated parallel batch processing machines with non-identical job sizes and unequal ready times." *Computers & Operations Research* 78: 117–128.

Atamturk, Alper, George L. Nemhauser, and Martin W.P. Savelsbergh. 2000. "Conflict graphs in solving integer programming problems." *European Journal of Operational Research* 121 (1): 40–55.

Ağralı, Semra, Z. Caner Taşkın, and A. Tamer Unal. 2017. "Employee Scheduling in Service Industries with Flexible Employee Availability and Demand." *Omega* 66: 159–169.

Beigel, R. 1999. "Finding maximum independent sets in sparse and general graphs." In *Proceedings of the tenth annual ACM-SIAM symposium on discrete algorithms*, .

Bellanger, Adrien, and Ammar Oulamara. 2009. "Scheduling hybrid flowshop with parallel batching machines and compatibilities." *Computers & Operations Research* 36 (6): 1982–1992.

Boyacı, Arman, Tınaz Ekim, and Mordechai Shalom. 2015. "The maximum cardinality cut problem in co-bipartite chain graphs." *Journal of Combinatorial Optimization* 1–16.

Buscher, Udo, and Liji Shen. 2010. "MIP formulations and heuristics for solving parallel batching problems." *Journal of Systems Science and Complexity* 23 (5): 884–895.

Cheng, T.C.E., Z. Liu, and W. Yu. 2001. "Scheduling jobs with release dates and deadlines on a batch processing machine." *IIE Transactions* 33: 685–690.

Chung, Tsui-Ping, Heng Sun, and Ching-Jong Liao. 2017. "Two new approaches for a two-stage hybrid flowshop problem with a single batch processing machine under waiting time constraint." *Computers & Industrial Engineering* 113: 859–870.

Garey, M. R., and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, New York: W. H. Freeman & Co.

Golumbic, Martin Charles. 1980. "Chapter 8 - Interval Graphs." In *Algorithmic Graph Theory and Perfect Graphs*, edited by Martin Charles Golumbic, 171 – 202. Academic Press.

Gong, Hua, Lixin Tang, and CW Duin. 2010. "A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times." *Computers & Operations Research* 37 (5): 960–969.

Johnson, D.S., M. Yannakakis, and C.H. Papadimitriou. 1988. "On generating all maximal independent sets." *Information Processing Letters* 27 (3): 119–123.

Kim, Yeong-Dae, Byung-Jun Joo, and Jong-Ho Shin. 2009. "Heuristics for a two-stage hybrid flowshop scheduling problem with ready times and a product-mix ratio constraint." *Journal of Heuristics* 15 (1): 19–42.

Koh, S.-G., P.-H. Koo, D.-C. Kim, and W.-S. Hur. 2005. "Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families." *International Journal of Production Economics* 98: 81–96.

Lee, G.C. 2009. "Estimating order lead times in hybrid flowshops with different scheduling rules." *Computers & Industrial Engineering* 56 (4): 1668–1674.

Li, Dongni, Xianwen Meng, Qiqiang Liang, and Junqing Zhao. 2015. "A heuristic-search genetic algorithm for multi-stage hybrid flow shop scheduling with single processing machines and batch processing machines." *Journal of Intelligent Manufacturing* 26 (5): 873–890.

Li, Shuguang. 2017. "Approximation algorithms for scheduling jobs with release times and arbitrary sizes on batch machines with non-identical capacities." *European Journal of Operational Research* .

Li, Xiaolin, YanLi Huang, Qi Tan, and HuaPing Chen. 2013. "Scheduling unrelated parallel batch processing machines with non-identical job sizes." *Computers & Operations Research* 40 (12): 2983–2990.

Li, Yonglin, and Zhenhua Dai. 2019. "A two-stage flow-shop scheduling problem with incompatible job families and limited waiting time." *Engineering Optimization* 1–23.

Lin, Rock, and Ching-Jong Liao. 2012. "A case study of batch scheduling for an assembly shop." *International Journal of Production Economics* 139 (2): 473–483.

Liu, Y., and I.A. Karimi. 2008. "Scheduling multistage batch plants with parallel units and no interstage storage." *Computers and Chemical Engineering* 32: 671–693.

Padberg, M. W. 1973. "On the facial structure of set packing polyhedra." *Mathematical Programming* 5: 199–215.

Rossit, Daniel Alejandro, Fernando Tohmé, and Mariano Frutos. 2017. "The non-permutation flow-shop scheduling problem: a literature review." *Omega* .

Ruiz, R., and J.A. Vazquez-Rodriguez. 2010. "The hybrid flow shop scheduling problem." *European Journal of Operational Research* 205: 1–18.

Schaller, J. 2007. "Scheduling on a single machine with family setups to minimize total tardiness." *International Journal of Production Economics* 105: 329–344.

Shahvari, Omid, and Rasaratnam Logendran. 2016. "Hybrid flow shop batching and scheduling with a bi-criteria objective." *International Journal of Production Economics* 179: 239–258.

Shahvari, Omid, and Rasaratnam Logendran. 2017a. "A bi-objective batch processing problem with dual-resources on unrelated-parallel machines." *Applied Soft Computing* 61: 174–192.

Shahvari, Omid, and Rasaratnam Logendran. 2017b. "An enhanced tabu search algorithm to minimize a bi-criteria objective in batching and scheduling problems on unrelated-parallel machines with desired lower bounds on batch sizes." *Computers & Operations Research* 77: 154–176.

Shahvari, Omid, and Rasaratnam Logendran. 2018. "A comparison of two stage-based hybrid algorithms for a batch scheduling problem in hybrid flow shop with learning effect." *International Journal of Production Economics* 195: 227–248.

Shi, Zhongshun, Zewen Huang, and Leyuan Shi. 2017. "Two-stage scheduling on batch and single machines with limited waiting time constraint." *Frontiers of Engineering Management* 4 (3): 368–374.

Taşkın, Z. Caner, Semra Ağralı, A. Tamer Unal, Vahdet Belada, and Filiz Gokten-Yilmaz. 2015. "Mathematical Programming-Based Sales and Operations Planning at Vestel Elec-

tronics." *Interfaces* 45 (4): 325–340.
Tan, Yi, Lars Mönch, and John W Fowler. 2018. "A hybrid scheduling approach for a two-stage flexible flow shop with batch processing machines." *Journal of Scheduling* 21: 209–226.
Xuan, H, and B Li. 2013. "Scheduling dynamic hybrid flowshop with serial batching machines." *Journal of the operational research society* 64 (6): 825–832.
Zheng, Yanming. 2010. "Minimizing Makespan for Hybrid Flowshops with Batch, Discrete Processing Machines and Arbitrary Job Sizes." PhD diss., Florida International University.