Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

A DECOMPOSITION HEURISTIC FOR A TWO-MACHINE FLOW SHOP WITH BATCH PROCESSING

Yi Tan
Lars Mönch

Department of Mathematics and Computer Science
Universitätsstraße 1
University of Hagen
Hagen, 58097, GERMANY

John W. Fowler

Department of Supply Chain Management
Main Campus, PO BOX 874706
Arizona State University
Tempe, AZ 85287-4706, USA

ABSTRACT

In this paper, we discuss a two-stage flow shop scheduling problem with batch processing machines. The jobs belong to different incompatible job families. Only jobs of the same job family can be batched together. The performance measure is the total weighted tardiness of the jobs. A decomposition heuristic is proposed that is based on the idea to iteratively determine due dates for the jobs in the first stage and earliest start dates of the jobs in the second stage. The two resulting subproblems are solved using a time window decomposition (TWD) heuristic and a variable neighborhood search (VNS) scheme. Results of computational experiments based on randomly generated problem instances are presented. We show that the VNS-based scheme outperforms the TWD heuristic. In addition, we show that the decomposition scheme can be parallelized in a very natural way. As a result, the amount of computing time is modest, even for the computational expensive VNS scheme.

1 INTRODUCTION

Semiconductor manufacturing is among the most complex manufacturing processes. It is characterized by a diverse product mix, a large number of jobs and machines, and a mix of different process types, including single wafer processes and batch processes (cf. Mönch, Fowler, and Mason 2013). Traditionally, dispatching has been the primary approach to flow control in semiconductor manufacturing, but recently, scheduling techniques have become popular due to the increasing capabilities of computers.

A batch is a collection of jobs that have to be simultaneously processed on a single machine. Each batch machine has a capacity that is measured as the maximum number of jobs that can be batched together. In this paper, we focus on batching in semiconductor wafer fabrication facilities (wafer fabs). In contrast to batching problems in the backend stage of semiconductor manufacturing, incompatible job families arise in wafer fabs due to the different nature of the involved chemical processes. Only jobs of the same family can be used to form a batch, and the processing times of all jobs of the same family are identical. The batching problems in the backend stage are characterized by the fact that the processing time of a batch is determined by the longest processing time of the jobs that form the batch.

Because often one third of all operations in a wafer fab are performed on batch processing machines and because of the long processing times of up to 20 hours compared to the typical average processing times of less than one hour on non-batching machines (cf. Mönch et al. 2011), an appropriate scheduling of jobs on batch processing machines has a large impact on the overall wafer fab performance. While in the past single and parallel machine batching problems in wafer fabs have been extensively studied (cf. Mathirajan and Sivakumar 2006 for a survey), this is not the case for flow shop scheduling problems. But because it is well-know that, on the one hand, batch processing machines heavily influence downstream machine groups, while on the other hand information on jobs of upstream machines groups are useful

when making scheduling decisions on batch processing machines (Robinson, Fowler, and Bard 1995) investigating a two-stage flexible flow shop with batch processing machines on at least one stage is desirable. In the present paper, we study a model problem that consists of two batch processing machines in line. An iterative decomposition scheme is proposed that allows for an efficient determination of high-quality solutions.

The paper is organized as follows. The researched problem is described and analyzed in Section 2. We propose the decomposition heuristic in Section 3. Computational results are presented and analyzed in Section 4. The paper is completed by some conclusions and a discussion of future research directions.

2 PROBLEM SETTING AND ANALYSIS

2.1 Problem

We consider a two-machine flow shop scheduling problem. This problem can be described as follows:

1. In total, n jobs have to be processed on the machines of the flow shop.
2. Both stage 1 and stage 2 consist of a single batch processing machine. The maximum batch size of the machine in stage s is B_s.
3. When a batch is started on a machine no interruption is allowed.
4. The buffer between the two batch processing machines is unlimited.
5. There exist F incompatible job families. Only jobs belonging to the same family can be batched together. The family of job j is given by $1 \le f(j) \le F$.
6. All jobs of the incompatible job family f have the same processing time p_{fs} on stage s.
7. Each job j has a due date d_j, a weight w_j that is used to model the importance of the job, and a ready time r_j.

Using the three-field notation from scheduling theory, the problem being studied can be represented as follows:

$$F2 \mid r_j, p-batch, incompatible \mid TWT, \tag{1}$$

where $F2$ denotes a flow shop that consists of two stages with a single machine at each stage, $p-batch,incompatible$ refers to parallel batching with incompatible job families, and TWT abbreviates the performance measure total weighted tardiness. This measure is defined as $TWT := \sum_{j=1}^{n} w_j \left(C_j - d_j\right)^+$, where C_j is the completion time of job j. In addition, we use $x^+ := \max(x,0)$ for abbreviation. This scheduling problem (1) is strongly NP-hard since it is well-known that the special case $1 \| TWT$ is strongly NP-hard. Hence, we have to look for efficient heuristics. In order to fully define the problem under study and to determine the optimal solution for small problem instances, we present a mixed integer programming (MIP) formulation for problem (1) as follows.

Indices and sets

$s = 1,2$: set of stages
$j = 1,\ldots,n$: set of jobs
$f = 1,\ldots,F$: set of families
$b = 1,\ldots,nb_s$: set of batches on stage s

Parameters

B_s: maximum batch size on the stage s machine

p_{fs} : processing time of family f jobs on the stage s machine

d_j : due date of job j

w_j : weight of job j

r_j : ready time of job j

$e_{jf} := \begin{cases} 1, \text{if job } j \text{ belongs to family } f \\ 0, \text{otherwise} \end{cases}$

M : big number

Decision variables

$x_{jbs} := \begin{cases} 1, \text{if job } j \text{ belongs to batch } b \text{ on the machine of stage } s \\ 0, \text{otherwise} \end{cases}$

$y_{bfs} := \begin{cases} 1, \text{if batch } b \text{ on the machine on stage } s \text{ belongs to family } f \\ 0, \text{otherwise} \end{cases}$

C_{js} : completion time of job j on stage s

T_j : tardiness of job j

s_{bs} : start time of batch b on stage s

$$\min \sum_{j=1}^{n} w_j T_j \tag{2}$$

subject to

$$\sum_{b=1}^{nb_s} x_{jbs} = 1,\ j = 1,\ldots,n, s = 1,2 \tag{3}$$

$$\sum_{j=1}^{n} x_{jbs} \le B_s, s = 1,2, b = 1,\ldots,nb_s \tag{4}$$

$$\sum_{f=1}^{F} y_{bfs} = 1, s = 1,2, b = 1,\ldots,nb_s \tag{5}$$

$$e_{jf} x_{jbs} \le y_{bfs},\ j = 1,\ldots,n, f = 1,\ldots,F, s = 1,2, b = 1,\ldots,nb_s \tag{6}$$

$$r_j x_{jb1} \le s_{b1}, j = 1,\ldots,n, b = 1,\ldots,nb_1 \tag{7}$$

$$s_{bs} + e_{jf} p_{fs} x_{jbs} \le s_{b+1,s}, s = 1,2, j = 1,\ldots,n, b = 1,\ldots,nb_s, f = 1,\ldots,F \tag{8}$$

$$s_{bs} + e_{jf} p_{fs} \le C_{js} + M\left(1 - x_{jbs}\right), j = 1,\ldots,n, s = 1,2, b = 1,\ldots,nb_s, f = 1,\ldots,F \tag{9}$$

$$C_{j1} \le M\left(1 - x_{jb2}\right) + s_{b2}, j = 1,\ldots,n, b = 1,\ldots,nb_2 \tag{10}$$

$$C_{j2} - T_j \le d_j, j = 1,\ldots,n \tag{11}$$

$$x_{jbs}, y_{bfs} \in \{0,1\}, C_{js}, T_j, s_{bs} \ge 0, j = 1,\ldots,n, f = 1,\ldots,F, s = 1,2, b = 1,\ldots,nb_s. \tag{12}$$

Our aim is to minimize the TWT value of the jobs. This is expressed by the objective function (2). The constraints (3) ensure that each job belongs to exactly one batch. The inequalities (4) model the fact that the number of jobs in each batch cannot be larger than the maximum batch size on a given stage. The constraints (5) make sure that each batch belongs to exactly one family. Constraints (6) ensure that all the jobs in a batch on a given stage belong to the same family. Constraints (7) model the fact that a given job

cannot start on the first stage before its release date. A sequencing of the batches on a given stage is represented by constraints (8). Constraints (9) relate the start time of a given batch to its completion time, while constraints (10) ensure that a batch on the second stage can only start if all the jobs that form this batch are completed on the first stage. Constraints (11) express the tardiness of each job, while constraints (12) model the fact that the decision variables are binary or nonnegative, respectively.

2.2 Related Work

We discuss related work with respect to specific two-machine flow shop scheduling problems and with respect to flow shop scheduling where batch processing machines are involved.

It is demonstrated by Demirkol, Mehta, and Uzsoy (1997) that the conventional shifting bottleneck heuristic does not work well for flow shops. The problem $F2||TT$ is considered in (Koulamas 1997), where TT is used as abbreviation for the total tardiness of the jobs. An efficient decomposition procedure is proposed that exploits the relationship to the $1||TT$ problem. Mukherjee and Chatterjee (2006) examine the failure of the shifting bottleneck heuristic in a two-machine flow shop environment. An alternative decomposition that applies a Schrage-type heuristic is proposed for $F2||C_{max}$, where C_{max} is the make-span. Several MIP models and heuristics based on these formulations are proposed by Liao and Liao (2008) for the problem $F2|p-batch|C_{max}$, where the processing time of the batch is determined by the longest processing times of the jobs that form the batch. Different job sizes are possible. In addition, different buffer configurations including zero intermediate storage are considered.

The problem $FFm||TWT$ is solved using different decomposition approaches by Yang, Kreipl, and Pinedo (2000) where FFm refers to a flexible flow shop with m stages. However, batch processing machines are not included. Demirkol and Uzsoy (2000) consider the problem $FFm|s_{jk},recrc|L_{max}$, where s_{jk} refers to sequence-dependent setup times and $recrc$ to reentrant flows. The maximum lateness is denoted by L_{max}. Efficient decomposition schemes are proposed, but again batching machines are not considered. Lei and Guo (2011) consider the problem $Fm|p-batch,perm|C$, where C is the objective that is to be minimized; the TT objective, the maximum tardiness, or the weighted number of tardy jobs are used, respectively. Only permutation schedules, indicated by the $perm$ notation, are of interest. The processing time of the batch is given as the longest processing times of the jobs that form the batch. VNS is used to tackle this scheduling problem. A two-machine flow shop scheduling problem is discussed by Fu, Sivakumar, and Li (2012). The first machine is a batch processing machine, while the second machine is a non-batching machine. However, only jobs of the same incompatible job family can be batched together on the first machine. The buffer between the two machines is limited. The objective to be minimized is the mean completion time, a cycle time-related measure. Batches are formed by heuristics that are scheduled in a second step by a differential evolution algorithm. Yugma et al. (2012) discuss a two-stage flexible flow shop scheduling problem where the second stage consists of batch processing machines. Maximizing the number of performed operations, maximizing the utilization of batches, and minimizing the TT are the objectives. An iterative sampling procedure and a simulated annealing scheme based on an appropriate disjunctive graph model are proposed. The problem researched in the present paper is different because we allow for batching on both of the two stages.

To summarize the discussion, problem (1) is not considered to the best of our knowledge in the literature so far. In the present paper, we propose a decomposition procedure that can be interpreted in terms of the lead time iteration scheme proposed by Vepsalainen and Morton (1988) to determine waiting time estimates of the jobs. We use the TWD approach by Mönch et al. (2005) and the VNS scheme by Klemmt et al. (2009) to solve the resulting single machine subproblems.

3 DECOMPOSITION HEURISTIC

3.1 Iterative Scheme

The main idea of the iterative scheme consists in determining appropriate due dates for the first stage and ready times for the second stage. This situation is depicted in Figure 1.

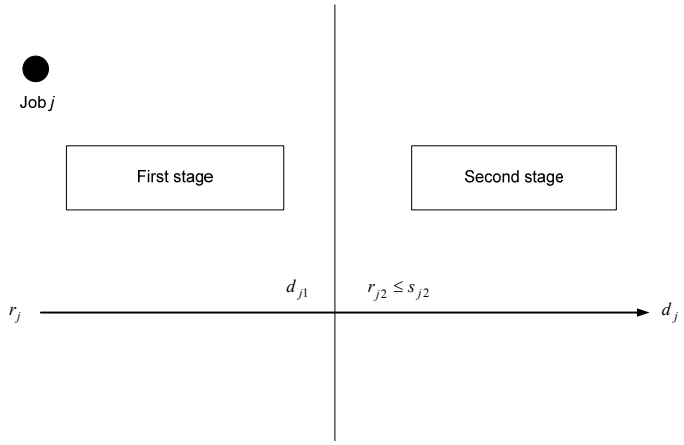

Figure 1: Decomposition of the two-machine flow shop problem.

We denote the due date for job j on the first stage that is used in iteration l by $d_{j1}^{(l)}$. The corresponding ready and start times on the second stage are $r_{j2}^{(l)}$ and $s_{j2}^{(l)}$, respectively. Once these quantities are known we are able to consider two single machine scheduling problems. The iterative approach can be summarized as follows:

1. Initialize the iteration counter by $l := 1$.
2. Set $d_{j1}^{(l)}, j = 1,\ldots,n$ using the waiting time information of the jobs from the previous iteration if any and solve the resulting scheduling problem for the first stage. The concrete setting of $d_{j1}^{(l)}$ will be discussed later.
3. Set $r_{j2}^{(l)} := C_{j1}^{(l)}, j = 1,\ldots,n$, where $C_{j1}^{(l)}$ is the completion time of job j in iteration l on the first stage. Solve the resulting scheduling problem for the second stage.
4. If the termination criterion is fulfilled then stop, otherwise set $l := l + 1$ and go to Step 2.

Next, we describe the initial setting of $d_{j1}^{(l)}, j = 1,\ldots,n$. We use

$$d_{j1}^{(1)} := r_j + p_{j1} + \frac{1}{2}\left(d_j - p_{j1} - p_{j2} - r_j\right) = \frac{1}{2}\left(r_j + p_{j1} + d_j - p_{j2}\right) \qquad (13)$$

with $p_{js} := p_{f(j)s}$ for the first iteration. The earliest possible start time of job j on the second stage is $r_j + p_{j1}$. At this point of time, the slack of the job j is $d_j - p_{j1} - p_{j2} - r_j$. We add a half of this slack to the earliest possible start time of the second stage to determine the due date for the first stage.

In the general case, we update the due dates $d_{j1}^{(l)}, j = 1,\ldots,n$ in iteration $l \ge 2$ by the following procedure:

$$d_{j1}^{(l)} := (1-\alpha) r_{j2}^{(l-1)} + \alpha\left(d_j - p_{j2} - w_{j2}^{(l-1)}\right) + \beta w_{j2}^{(l-1)}, \tag{14}$$

where $\alpha, \beta \geq 0$ are parameters and $w_{j2}^{(l-1)} = s_{j2}^{(l-1)} - r_{j2}^{(l-1)}$ is the waiting time of job j on the second stage machine that is a result of the schedule in iteration $l-1$. The due date for the first stage in iteration l is a linear combination of the release date of the second stage from iteration $l-1$ and the last possible starting time of the job at the second stage such that the due date of the job is met. In addition, the due is increased by the third term that reflects the waiting time. The parameters α, β have to be varied to obtain small TWT values. It is easy to see that $\beta = 1$ leads to

$$d_{j1}^{(l)} := (1-\alpha) s_{j2}^{(l-1)} + \alpha\left(d_j - p_{j2}\right), \tag{15}$$

i.e., the due date is a linear combination of the starting time on the previous iteration and the latest possible starting if no waiting time occurs. From expression (14) we can see that our decomposition scheme can be interpreted as a specific iterative simulation procedure (cf. Mönch et al. 2013 for the principles of iterative simulation) where the evaluation of the schedule is interpreted as a deterministic forward simulation. The iterative scheme terminates when either the same schedule is obtained for a second time or when a prescribed maximum number of iterations is reached.

3.2 Solution of the Subproblems

Two subproblems of the form $1|r_j, p-batch, incomp|TWT$ have to be solved in a single iteration of the decomposition scheme. For the sake of completeness, we summarize the decisions of the TWD scheme to select a batch as follows. To simplify the notation, we assume that job j has a due date $\tilde{d}_j$, a processing time p_j, and a ready time $\tilde{r}_j$, i.e., we do not differentiate between the two stages. When the single batch processing machine becomes available at time t, only jobs with ready times smaller than $t + \Delta t$ are considered. The set $\tilde{J}_f(t) := \left\{j \mid \tilde{r}_j \leq t + \Delta t, f(j) = f\right\}$ is formed for each family. We sequence all jobs within $\tilde{J}_f(t)$ with respect to the Apparent Tardiness Cost (ATC) index (cf. Vepsalainen and Morton 1987)

$$I_j(t) := \frac{w_j}{p_j} \exp\left(-\frac{\left(\tilde{d}_j - p_j + \left(\tilde{r}_j - t\right)\right)^+}{\kappa \bar{p}}\right), \tag{16}$$

in non-increasing order. The quantity κ is a scaling parameter in the index (16), while $\bar{p}$ is the average processing time of the remaining jobs. We consider only the first *thresh* jobs from the sorted list to form batches. The resulting set is called $\hat{J}_f(t)$. Consider all batches formed by jobs of the set $\hat{J}_f(t)$. Each of these potential batches is assessed using the BATC-II batching rule (Mönch et al. 2005). The index to assess a batch of family *f* is given by

$$I_b(t) := \sum_{j \in b} \frac{w_j}{p_j} exp\left(-\frac{\left(\tilde{d}_j - p_j - t + \left(r_b - t\right)^+\right)^+}{\kappa \bar{p}}\right) \frac{|b|}{B}, \tag{17}$$

where $|b|$ is the number of jobs in b and $r_b := \max\left(\tilde{r}_j \mid j \in b\right)$ is the ready time of batch b. The batch with the largest BATC-II index is chosen among the different families. The time $t' \leq t + \Delta t$ of the next event is calculated, i.e., a machine becomes available or a new job has arrived and the entire procedure is repeated.

The VNS scheme used as a subproblem solution procedure is similar to the scheme presented by Klemmt et al. (2009) for unrelated parallel machines. We consider four different neighborhood structures.

All the structures work on a sequence of batches. The first neighborhood structure moves randomly selected batches to a randomly selected position on the machine. The second one moves sequences of consecutive batches in a similar manner. Single batches or sequences of consecutive batches are swapped by the two remaining neighborhood structures, respectively. The neighborhood structures are applied in this order. The local search approach within VNS is based on job moves among batches, job swaps, and swaps of entire batches (cf. Klemmt et al. 2009 for details).

4 COMPUTATIONAL RESULTS

4.1 Design of Experiments

We expect that the performance of the decomposition heuristic depends on the number of jobs, the maximum batch size, the due date tightness, and the spread of the release dates. Therefore, we randomly generate problem instances that take into account these factors. The release dates are set using a rough makespan estimate similar to (Mönch et al. 2005), whereas the due dates are chosen considering a flow factor FF. We model a different bottleneck severity similar to (Yang, Kreipl, and Pinedo 2000). Therefore, the workload of a batch machine is measured as

$$WL_s = \frac{1}{B_s}\sum_{j=1}^{n} p_{js}, s = 1,2, \tag{18}$$

where we set $WL_{\min} := \min\{WL_1, WL_2\}$ and $R_s := WL_s / WL_{\min}$. Let machine i be the bottleneck machine, i.e. the machine with the largest workload. We modify the processing times p_{ji} of job j on the bottleneck machine i to be $\hat{p}_{ji} := p_{ji}\, g/R_i$ with a bottleneck severity factor g. As a result, we obtain the relation $WL_i = g \cdot WL_{\min}$, i.e., g is a measure of how much higher the workload on the bottleneck machine is relative to the non-bottleneck machine. The resulting design of experiments is summarized in Table 1.

Table 1: Design of experiments.

Factor	Level	Count
Number of jobs per family	10, 20, 30	3
Number of families	3, 6	2
Processing time of a job family	5 with $p = 0.2$, 10 with $p = 0.3$, 15 with $p = 0.3$, 20 with $p = 0.2$	1
Maximum batch size of a machine	2, 4, 8	9
Weight of the jobs	w_j ~U(0,1)	1
Release date of the jobs	$r_j \sim U\left(0, a\left(\frac{1}{0.75\,B_1}\sum_{j=1}^{n} p_{j1} + \frac{1}{0.75\,B_2}\sum_{j=1}^{n} p_{j2}\right)\right)$ $a = 0.25, 0.75$	2
Due date of the jobs	$d_j := r_j + FF\left(p_{j1} + p_{j2}\right)$ $FF = 1.1, 1.5$	2
Bottleneck severity	$g = 1.2, 2.0, 3.0$	3
Total		648

In addition, six small-size problem instances are generated to compare the performance of the decomposition heuristics with solutions obtained by the MIP formulation (2)-(12). This allows us to check the correctness of the implementation of the proposed heuristics. The instances have two families, and each family contains eight jobs. We also assume that $B_1 = B_2 = 4$ holds.

We consider three different variants of the decomposition heuristic. When we perform only the first iteration of the iterative scheme and use TWD as a subproblem solution procedure, we obtain H(TWD). This heuristic is used as a reference heuristic. H(TWD) serves at the same time as an initial solution for the two iterative schemes abbreviated by H(iter,TWD) and H(iter, VNS) depending on the subproblem solution procedure applied. We assess each heuristic H by taking the ratio of the TWT values of the heuristic and the TWT value of H(TWD), i.e., we compute

$$Imp(H) = TWT(H)/TWT(H(TWD)). \tag{19}$$

When a heuristic with stochastic ingredients is considered, each problem instance is solved five times with different seeds to obtain statistically meaningful results.

4.2 Parameter Setting and Implementation Details

The TWD-based subproblem solution procedures are parameterized by $\Delta t = 4$ and $thresh = 15$ because we know from (Mönch et al. 2005) that these settings lead to small TWT values. Within TWD, we apply the BATC-II batching rule with an appropriate value of the look-ahead parameter κ for sequencing the batches. We select the parameter from the interval $[0.5,5.0]$ with step size 0.5 and choose the value that corresponds to the smallest TWT value. The maximum number of iterations is denoted by $l_{\max}$. The maximum computing time per subproblem together with the $l_{\max}$ values are shown for the different heuristics in Table 2. Note that TWD requires less than five seconds per subproblem.

Table 2: Settings for the different heuristics.

Heuristic	Maximum computing time (in seconds)	$l_{\max}$
H(TWD)	-	1
H(iter,TWD)	-	100
H(iter, VNS)	5	20

We also performed some additional experiments with more iterations for H(iter,VNS). However, a larger number of iterations did not improve the results. The same is true for using more than 100 iterations for H(iter,TWD). The α values in expression (14) are chosen as $\alpha = 0.25k, k = 0,\ldots,8$. In our preliminary experiments, we simply set $\beta = 1$ to limit the computational burden. The setting $\beta = 0$ is slightly outperformed by $\beta = 1$.

The heuristics are coded using the C++ programming language. The MIP (2)-(12) is implemented using ILOG CPLEX 12.1. All the tests are performed on an Intel Core i7-4770 3.40 GHz, 8GB computer with Windows 7 64-Bit operating system. The grid search for the α values can be parallelized. If m different α values are to be assessed for one of the three heuristics, we use m processes with m different α values. Each process independently solves an instance for a given α value. Nine processes of one of the heuristics are created and deployed on a multi-core computer in our experiments. We use the smallest TWT value and the longest computing time obtained from these nine processes.

4.3 Results and Discussion

We start by looking at the small-size problem instances. CPLEX was not able to prove the optimality of the solution for any of these instances within 72 hours of computing time, whereas H(iter,VNS) and

H(iter, TWD) terminate within one minute. Note that H(iter,TWD) always stops before $l_{\max}$ is reached, i.e., more iterations are not beneficial. The results of the six small-size instances are summarized in Table 3. The smallest TWT values for each instance are presented in bold. We see that CPLEX always obtains the smallest TWT value, followed by H(iter,VNS), while H(iter,TWD) is often outperformed by the two remaining approaches. As a result, we know that H(iter,VNS) is able to provide near to optimal solutions for small-size instances.

Table 3: Comparison of H(iter,VNS) and H(iter,TWD) with CPLEX.

Instance	CPLEX	H(iter,VNS)	H(iter,TWD)
1	**37.437**	**37.437**	**37.437**
2	**125.804**	**125.804**	**125.804**
3	**45.099**	**45.099**	60.7461
4	**134.949**	**134.949**	150.648
5	**33.860**	34.194	40.6813
6	**43.608**	**43.608**	75.6269

Next, we focus on the experimental results for the 648 instances from the design in Table 1. The computational results are shown in Table 4. Instead of comparing all problem instances individually, the instances are grouped according to factor levels such as the number of jobs, the maximum batch size etc.

We report the improvements of H(iter,TWD) and H(iter,VNS) compared to the initial solution. In addition, we show the average computing time per problem instance and the average number of required iterations. In total, H(iter,VNS) outperforms the H(iter,TWD) with an improvement of 5.80%. The computational most expensive heuristic H(iter,VNS) only requires an average time of 125.39 seconds per problem instance due to the parallelization approach. Even for the largest instances with 180 jobs the computing time of H(iter,VNS) is only around four minutes and is suitable for real-world applications.

When the number of families and the number of jobs increase, H(iter,VNS) achieves less improvement with respect to H(iter,TWD). This behavior is caused by the fact that H(iter,VNS) needs more computing time to solve these hard instances. The average number of iterations required by H(iter,VNS) for the instances with 180 jobs is 14.72 and is often close to $l_{\max} = 20$. We can see that the combination of the two maximum batch sizes does not lead to major performance differences. We do not see any specific pattern for the (B_1, B_2) pairs.

A larger *a* value leads to wide-spread releases of the jobs. In this case, the improvement of H(iter,VNS) compared to H(iter,TWD) increases significantly because there is more room for optimization. A larger *FF* value results in wider due dates for each job. In this case, both H(iter,TWD) and H(iter,VNS) perform better. Larger values of *g* reveal a reduced performance of H(iter,VNS) and H(iter,TWD) because in this situation the scheduling of the bottleneck machine is more important compared to scheduling jobs on the non-bottleneck machine.

5 CONCLUSIONS AND FUTURE RESEARCH

In this paper, we proposed a decomposition technique for a two-machine flow shop with batch processing. The technique is based on the idea of iteratively improving the internal due dates for the first stage and the internal ready times of the second stage. The resulting subproblems for each single machine are either solved by the TWD approach or by a VNS scheme. We demonstrated that the VNS approach outperforms the TWD approach. The decomposition scheme can be parallelized. This leads to moderate computing times even when the computationally expensive VNS scheme is applied.

There are several directions for future research. First of all, we think that more computational experiments are required to better understand the performance of the proposed heuristics.

Table 4: Computational results.

Factor	**Imp**			**Computing time (in s)**		**Number of iterations**	
	H(TWD)	H(iter,TWD)	H(iter,VNS)	H(iter,TWD)	H(iter,VNS)	H(iter,TWD)	H(iter,VNS)
(B_1,B_2)							
(2,2)	1.000	0.963	0.891	5.09	117.84	6.92	10.23
(2,4)	1.000	0.944	0.878	7.32	147.95	11.49	13.13
(2,8)	1.000	0.934	0.897	12.81	169.15	20.06	14.86
(4,2)	1.000	0.937	0.885	4.36	111.12	6.60	9.75
(4,4)	1.000	0.969	0.906	3.43	113.72	6.94	9.93
(4,8)	1.000	0.956	0.899	4.11	128.99	8.42	11.00
(8,2)	1.000	0.963	0.910	2.04	87.73	3.76	7.15
(8,4)	1.000	0.958	0.912	2.50	114.76	5.39	9.31
(8,8)	1.000	0.938	0.863	1.69	134.91	4.82	8.63
n/F							
10	1.000	0.950	0.870	1.17	79.84	4.99	7.82
20	1.000	0.950	0.898	4.06	119.03	8.30	10.58
30	1.000	0.954	0.912	9.22	177.30	11.51	12.97
F							
3	1.000	0.948	0.866	2.12	95.97	6.90	8.72
6	1.000	0.954	0.920	7.51	154.81	9.63	12.19
n							
30	1.000	0.942	0.827	0.46	61.57	3.90	6.09
60	1.000	0.953	0.897	1.88	95.65	6.70	9.19
90	1.000	0.953	0.890	4.03	133.16	9.48	11.22
120	1.000	0.951	0.914	6.23	144.88	9.28	12.31
180	1.000	0.955	0.934	14.42	221.44	13.54	14.72
a							
0.25	1.000	0.968	0.938	5.54	136.82	8.45	10.42
0.75	1.000	0.934	0.849	4.09	113.96	8.08	10.49
FF							
1.1	1.000	0.973	0.913	2.66	93.37	4.89	7.68
1.5	1.000	0.929	0.873	6.97	157.42	11.64	13.23
g							
1.2	1.000	0.933	0.867	6.40	127.70	10.92	10.51
2	1.000	0.952	0.901	4.15	128.00	7.38	10.63
3	1.000	0.969	0.912	3.90	120.48	6.50	10.22
Overall	**1.000**	**0.951**	**0.893**	**4.81**	**125.39**	**8.26**	**10.45**

It seems especially reasonable to consider a grid of (α, β) pairs instead of varying only the α values. In addition, it is worthwhile to extend the problem setting to a situation where unrelated parallel machines are on each stage, i.e. where we consider a two-stage flexible flow shop. This setting can be found in wafer fabs. We believe that it is worthwhile to consider situations where the machines of the upstream and downstream stage are non-batching machines because then we can investigate the interplay of the two stages and can derive information on the importance of scheduling on each of the two stages.

Secondly, we are interested in researching the performance of solution approaches for problem (1) that are not based on decomposition. We expect that neighborhood search-based approaches as used by Liao and Huang (2010) for a non-permutation flow shop scheduling problem or by Yugma et al. (2012) for a two-stage flexible flow shop can be useful for the problem studied in the present paper. However, carrying out all the necessary details is part of future research.

ACKNOWLEDGMENTS

The authors would like to thank Carsten Eilks for interesting discussions on the topic of this paper and for his programming efforts.

REFERENCES

Demirkol, E., and R. Uzsoy. 2000. "Decomposition Methods for Reentrant Flow Shops with Sequence-dependent Setup Times." *Journal of Scheduling* 3:155-177.

Demirkol, E., S. Mehta, and R. Uzsoy. 1997. "A Computational Study of Shifting Bottleneck Procedures for Shop Scheduling Problems." *Journal of Heuristics* 3:111-137.

Fu, Q., A. I. Sivakumar, and K. Li. 2012. "Optimization of Flow-shop Scheduling with Batch Processor and Limited Buffer." *International Journal of Production Research* 50(8):2267-2285.

Klemmt, A., G. Weigert, C. Almeder, and L. Mönch. 2009. "A Comparison of MIP-based Decomposition Techniques and VNS Approaches for Batch Scheduling Problems." In *Proceedings of the 2009 Winter Simulation Conference*, 1684-1694.

Koulamas, C. 1998. "A Guaranteed Accuracy Shifting Bottleneck Algorithm for the Two-machine Flowshop Total Tardiness Problem." *Computers & Operations Research* 25(2):83-89.

Lei, D., and X. Guo. 2011. "Variable Neighborhood Search for Minimizing Tardiness Objectives on Flow Shop with Batch Processing Machines." *International Journal of Production Research* 49(2): 519-529.

Liao, L.-M., and C.-J. Liao. 2008. "Improved MILP Models for Two-machine Flowshop with Batch Processing Machines." *Mathematical and Computer Modeling* 48:1254-1264.

Liao, L.-M., and C.-J. Huang. 2010. "Tabu Search for Non-permutation Flowshop Scheduling Problem with Minimizing Total Tardiness. *Applied Mathematics and Computation* 217:557-567.

Mathirajan, M., and A. Sivakumar. 2006. "A Literature Review, Classification and Simple Meta-analysis on Scheduling of Batch Processors in Semiconductor." *International Journal of Advanced Manufacturing Technology* 29: 990-1001.

Mönch, L., H. Balasubramanian, J. W. Fowler, and M. E. Pfund. 2005. "Heuristic Scheduling of Jobs on Parallel Batch Machines with Incompatible Job Families and Unequal Ready Times." *Computers & Operations Research* 32: 2731-2750.

Mönch, L., J. W. Fowler, S. Dauzère-Pérès, S. J. Mason, and O. Rose. 2011. "A Survey of Problems, Solution Techniques, and Future Challenges in Scheduling Semiconductor Manufacturing Operations." *Journal of Scheduling* 14:583-599.

Mönch, L., J. W. Fowler, and S. J. Mason. 2013. *Production Planning and Control for Semiconductor Wafer Fabrication Facilities: Modeling, Analysis, and Systems.* Springer, New York.

Mukherjee, S., and A. K. Chatterjee. 2006. "Applying Machine-based Decomposition in 2-machine Flow Shops." *European Journal of Operational Research* 169:723-741.

Robinson, J., J. W. Fowler, and J. F. Bard. 1995. "The Use of Upstream and Downstream Information in Scheduling Semiconductor Batch Operations." *International Journal of Production Research* 33(7):1849-1869.

Vepsalainen, A. P. J., and T. E. Morton. 1987. "Priority Rules for Job Shops with Weighted Tardiness Cost." *Management Science* 33(8): 1035 – 1047.

Vepsalainen, A. J. P., and T. E. Morton. 1988. "Improving Local Priority Rules with Global Lead-time Estimates: a Simulation Study." *Journal of Manufacturing and Operations Management* 1:102-118.

Yang, Y., S. Kreipl, and M. Pinedo. 2000. "Heuristics for Minimizing Total Weighted Tardiness in Flexible Job Shops." *Journal of Scheduling* 3:89-108.

Yugma, C., S. Dauzère-Pérès, C. Artigues, A. Derreumaux, and O. Sibille. 2012. "A Batching and Scheduling Algorithm for the Diffusion Area in Semiconductor Manufacturing." *International Journal of Production Research* 50(8): 2118-2132.

AUTHOR BIOGRAPHIES

YI TAN is a research associate at the Chair of Enterprise-wide Software Systems, University of Hagen. He is a Ph.D. candidate in Production Engineering at the University of Bremen, Germany. He got his MS degree in Computer Science from the University of Paderborn, Germany. His research interests are in planning and scheduling of production and logistics systems. He can be reached by email at <Yi.Tan@fernuni-hagen.de>.

LARS MÖNCH is professor of Computer Science at the Department of Mathematics and Computer Science, University of Hagen where he heads the Chair of Enterprise-wide Software Systems. He holds MS and Ph.D. degrees in Mathematics from the University of Göttingen, Germany. After his Ph.D., he obtained a habilitation degree in Information Systems from Technical University of Ilmenau, Germany. His research and teaching interests are in information systems for production and logistics, simulation, scheduling, and production planning. He is an Associate Editor of *European Journal of Industrial Engineering* and of *Business & Information Systems Engineering*. He can be reached by email at <lars.moench@fernuni-hagen.de>.

JOHN W. FOWLER is the Motorola Professor and Chair of the Supply Chain Management department at ASU. His research interests include discrete event simulation, deterministic scheduling, and multi-criteria decision making. He has published over 100 journal articles and over 100 conference papers. He was the Program Chair for the 2002 and 2008 Industrial Engineering Research Conferences and the 2008 Winter Simulation Conference (WSC). He is currently serving as Editor-in-Chief for *IIE Transactions on Healthcare Systems Engineering*. He is also an Editor of the *Journal of Simulation* and Associate Editor of *IEEE Transactions on Semiconductor Manufacturing* and *TOMACS*. He is a Fellow of the Institute of Industrial Engineers (IIE) and currently serves as the IIE Vice President for Continuing Education, is a former INFORMS Vice President, and was an SCS representative on the WSC Board of Directors from 2005-2013. His email address is <john.fowler@asu.edu>.