This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSM.2017.2740340, IEEE Transactions on Semiconductor Manufacturing

Constraint Programming Approach for Scheduling Jobs with Release Times, Non-identical Sizes, and Incompatible Families on Parallel Batching Machines

Andy Ham, *Member, IEEE,* John Fowler, *Member, IEEE,* and Eray Cakici

***Abstract*—We study a parallel batch-scheduling problem that involves the constraints of different job release times, non-identical job sizes, and incompatible job families, is addressed. Mixed integer programming (MIP) and constraint programming (CP) models are proposed and tested on a set of common problem instances from a paper in the literature. Then, we compare the performance of the models with that of a variable neighbourhood search (VNS) heuristic from the same paper. Computational results show that CP outperforms VNS with respect to solution quality and run time by 3.4~6.8% and 47~91%, respectively. When compared to optimal solutions, the results demonstrate CP is capable of generating a near optimal solution in a short amount of time.**

***Index Terms*— parallel batching, incompatible, CP, MIP, VNS**

I. Introduction

Our ability to computerize and automate a factory has dramatically advanced with the advent of 300 mm fabrication (fab) in the semiconductor industry. Consider a daily morning operations meeting (OPS) in a leading fab. The diffusion area, which contains batch-processing machines, receives high attention during the OPS due to its vulnerability to inventory fluctuations and the impact that it has on downstream processing steps. The dashboard points out negative key performance indices (KPI): cycle time, inventory, moves, and batching size. The diffusion area manager blames a poor scheduling/dispatching decision made by a computerized system and presents Gantt chart evidence of illogical myopic decisions. The manager orders a manual job reservation based on his own experience to replace the automated dispatching system, hoping for an improvement. In fact, the diffusion process is notorious for its lowest full automation rate, which is calculated as the total count of transactions minus the total count of manual transactions divided by the total count of transactions. Why is the diffusion process troublesome for practitioners?

We now zoom in on the diffusion process. The diffusion process is used to add dopants into the wafer in order to alter electrical properties. The high temperature of a diffusion machine makes the dopants diffuse into the wafer. So, the semiconductor industry calls the machine a furnace. This diffusion process is typically extremely slow and can take up to 12 hours compared to less than an hour for most other process steps. In order to compensate for the agonizingly slow processing time and achieve economies of scale in production, diffusion furnace makers have designed the process to be batch-processing. By its nature, a batching machine can create inventory bubbles or starvation to its downstream step. Suppose there are 10 batching machines in parallel which are about to complete their operations at the same time in the worst-case scenario. A bubble comprised of 1500 wafers (=6 jobs × 25 wafers ×10 machines) can flow into the downstream steps. Similarly, it can also cause an inventory starvation.

In batch-processing, multiple jobs can be simultaneously processed as long as the total size of the batch does not exceed machine capacity. Even if several jobs can be processed at the same time as a batch, jobs which have different recipes cannot be processed together, namely, *incompatible* job families (Uzsoy 1995 and Balasubramanian *et al.* 2004). The processing time of a batch is determined by the family of jobs in the batch in the diffusion process. It should be noted that the MIP-PA model in Section 3 assumes the processing time of a batch is dependent on the individual jobs in the batch. Jobs which belong to a same family have the same processing time in the test problem instances as shown in Table 1 so all models are constrained by the same restriction.

The batch scheduling decision can be decomposed into two sub-decisions: batching and sequencing. However, the problem considered in this study has different job release times and multiple machines, which make the two decisions interrelated with each other. It will likely lead to poor solutions if the decisions are separated.

The objective function of interest is to minimize the sum of total weighted completion time (TWCT) of all jobs. This objective concentrates on the cycle time, which is a significant indicator of semiconductor manufacturing performance [11][22]. Also, the objective is weighted by job priority, job

A. Ham is with the Liberty University, Lynchburg, VA 24501 USA (e-mail: mham@liberty.edu).

J. Fowler is with the Arizona State University, Supply Chain Management, and Tempe, AZ 85287 USA (e-mail: john.fowler@asu.edu).

E. Cakici is with IBM Turk, Buyukdere Cad. Yapikredi Plaza B Blok Levent, 34330, Istanbul, Turkey (e-mail: eray.cakici@gmail.com).

0894-6507 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSM.2017.2740340, IEEE Transactions on Semiconductor Manufacturing

waiting times, length of remaining time windows, etc., in order to incorporate individual characteristics of each job [10]. As a result, the scheduling problem can be represented by $P_m|r_j$, *batch*, *incompatible* $|\sum w_j C_j$ using the $\alpha|\beta|\gamma$ notation in Graham *et al.* [7].

Dobson and Nambimadom [4] address the problem of minimising TWCT on a single batching machine with incompatible job families and jobs of different sizes and prove the problem is NP-hard. Therefore, our problem with multiple machines is also NP-hard. Among the many different approaches to the problem under study, we concentrate on constraint programming (CP) which has not been studied much in previous research.

The rest of this paper is organized as follows: a literature review is presented in Section II and the proposed MIP and CP models are developed in Section III. Computational results are reported in Section IV and finally Section V covers the conclusions and areas for future research.

II. Literature review

A. Heuristics Approach

Batching problems are extensively considered in the literature so we narrow down our search to different job release times and incompatible job families. In many cases jobs enter the system at different times of a planning horizon. The batch scheduling system in a semiconductor fab typically considers jobs which arrive in the future as well as those currently waiting, mainly due to its long processing time. Glassey and Weng [6], Fowler *et al.* [5], Weng and Leachman [32], and Uzsoy [29] address dynamic job arrivals with *incompatible* job families. Dobson and Nambimadom [4] discuss a single batching machine with incompatible job families but identical job arrival times and propose a generalized assignment heuristic. Balasubramanian *et al.* [2] devise a genetic algorithm (GA) solution for parallel batch machines with incompatible job families but identical job arrival times. Similarly, Koh *et al.* [15] address a single batching machine with incompatible job families but identical job arrival times and propose a GA. Yugma *et al.* [33] suggest a simulated annealing approach for a multi-stage parallel batch machine scheduling problem with incompatible job families and different job arrival times. Mazumdar *et al.* [21] discuss a single batching machine with incompatible job families but identical job arrival times and propose a tabu search approach.

In particular, there are two papers in the literature that are most closely relevant to our study. Almeder and Mönch [1] and Cakici *et al.* [3] address the same problem involving parallel batch machines, different job release times, different job sizes, different processing times, and incompatible job families and they both suggest variable neighbourhood search (VNS) approaches. We will use the problem instances from Cakici *et al.* [3] and compare our CP results with their VNS results.

B. Constraint Programming Approach

Despite the fact that the aforementioned heuristics may generate fast and effective solutions, they are usually tailor-made. Moreover, the efficiency of these techniques strongly depends on the proper implementation and fine tuning of parameters since they combine the problem representation and the solution strategy into the same framework. In contrast, a mathematical modelling approach divides the problem representation and the solution strategy. Therefore, a general MIP model can be solved by many different solvers [16]. However, it is often too slow to solve large-size industrial scheduling problems using MIP approaches. Here, CP becomes an attractive alternative.

CP technology is well known in the artificial intelligence (AI) world owing to its success for efficiently solving many scheduling problems. CP Optimizer, the CP engine available in the IBM ILOG CPLEX Optimization Studio, provides specialized keywords and syntax for modeling detailed scheduling problems. A major benefit of the CP Optimizer approach to scheduling is that no enumeration of time, *i.e.* time buckets or time periods, is required. This means that relatively few decision variables are needed compared to MIP approaches that would require variables for each time bucket of a discretized model [8]. In CP Optimizer, both large neighbourhood search (LNS) and failure directed search (FDS) serve as solution strategies and together form the basis of the automatic search mechanism for scheduling problems [30]. Therefore, we can say CP modeling is not tailor-made for specific scheduling problems. To a modeller, it is simply a rich language of describing scheduling problems so that the remaining solution strategy solely resides in the hand of the optimizer engine, CP search algorithms are tested on a range of scheduling benchmarks: job shop, job shop with operators, flexible job shop, resource-constrained project scheduling problem, etc. Results show that the proposed search algorithms often improve best-known lower and upper bounds and closes many open instances. Readers interested in details of the search algorithms are encouraged to refer to Vilím *et al.* [30].

The main contributions of this paper can be summarized as follows. To the best of our knowledge, there is no work on CP applied to the batch scheduling problem under study herein. The closest work to be found is in Malapert *et al.* [20]. They present a CP approach for a single batch-processing machine scheduling problem to minimize the maximum lateness, but they assume identical job release times and a single machine in the context of a one-dimensional bin packing problem. Their proposed approach outperforms two exact algorithms: a MIP formulation and a branch-and-price algorithm. Similarly, a MIP model is compared with CP for the daily scheduling problem of an operating theatre in Wang *et al.* [31]. They find the MIP model provides better performance for the weighted completion time objective and the CP model provides better performance for the makespan minimization objective.

C. Methodology

As computer hardware and software have improved, practitioners have been able to solve increasingly complex problems in a reasonable amount of time [14]. Therefore, we first formulate our batch scheduling problem as two distinct MIP models and run them with the latest version of IBM CPLEX 12.6.3 to setup a baseline. We then propose a CP representation of the problem hoping for fast and effective solutions. The performance of the MIP and CP models are

0894-6507 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSM.2017.2740340, IEEE Transactions on Semiconductor Manufacturing

tested on the same problem instances in Cakici *et al.* [3] and compared with that of the VNS heuristic from the same paper.

1) MIP-PA (Positional & assignment variables)

We use the following notation for the MIP model:

Set indexes:

J	jobs (j)
B	batches (b)
M	machines (m)
F	job families (f)

Parameters:

r_j	release time of job j
p_j	processing time of job j
s_j	size of job j
w_j	weight of job j
f_j	family of job j
k_m	maximum batch size of machine m
L	arbitrarily large number

Decision variables:

X_{jbm}	1 if job j is in batch b on machine m; 0 otherwise
S_{bm}	start time of batch b on machine m
C_{bm}	completion time of batch b on machine m
C_j	completion time of job j

Resultant variables:

P_{bm}	processing time of batch b on machine m
Q_{bmf}	1 if batch b on machine m consists of jobs of family f 0 otherwise

The scheduling problem under study may be formulated as follows:

$$\text{Minimize} \quad \sum_j w_j C_j \tag{1.1}$$

$$\sum_b \sum_m X_{jbm} = 1 \qquad \forall j \tag{1.2}$$

$$\sum_j s_j X_{jbm} \le k_m \qquad \forall b,\ m \tag{1.3}$$

$$P_{bm} \ge p_j X_{jbm} \qquad \forall j, b, m \tag{1.4}$$

$$S_{bm} \ge r_j X_{jbm} \qquad \forall j, b, m \tag{1.5}$$

$$S_{bm} \ge C_{b-1,m} \qquad \forall b > 1, m \tag{1.6}$$

$$C_{bm} \ge S_{bm} + P_{bm} \qquad \forall b, m \tag{1.7}$$

$$Q_{bmf} \ge X_{jbm} \qquad \forall b, m, f = f_j \tag{1.8}$$

$$\sum_f Q_{bmf} \le 1 \qquad \forall b, m \tag{1.9}$$

$$C_j \ge L\left(X_{jbm} - 1\right) + C_{bm} \qquad \forall j, b, m \tag{1.10}$$

$$X_{jbm}, Q_{bmf} \qquad \forall j, b, m, f \tag{1.11}$$

$$S_{bm,} C_{bm,} P_{bm,} C_j \qquad \forall j, b, m \tag{1.12}$$

Objective (1.1) minimizes the sum of weighted completion time of all jobs. Constraint (1.2) ensures that each job is assigned to only one batch and processed on only one machine. Constraint (1.3) ensures that the total job size of a batch cannot exceed the batch size limit of the machine. Constraint (1.4) defines the processing time of a batch on a given machine, which is represented by the longest time of all jobs in the batch. Since only jobs which belong to the same family are batched together, this constraint still holds. Constraint (1.5) ensures that the start time of a batch is greater than or equal to the release times of all jobs in a batch, which is represented by the latest release time of all jobs in a batch. Constraint (1.6) ensures that the start time of a batch must be greater than or equal to the completion time of the preceding batch on the same machine. Constraint (1.7) calculates the completion time of a batch on each machine, which is represented by the start time of a batch plus its processing time. Constraints (1.8–1.9) ensure that jobs which belong to the same job family can be processed together. Constraint (1.10) determines the completion times of each job which is equal to the completion time of the batch to which it is assigned. Constraints (1.11–1.12) impose the binary and non-negativity restrictions, respectively.

During preliminary experimentation, we found that this MIP model did not generate an optimal solution for 25-job instances after several days of CPLEX run-time so we explore other formulations. Keha *et al.* [12] discuss that in a single machine scheduling problem a less frequently used MIP formulation is computationally more efficient in practice than commonly used MIP formulations for certain problems and compare the four different types of formulation: completion time variables, time index variables, linear ordering variables, and positional and assignment variables. Our proposed MIP model is close to the positional and assignment variables formulation. Unlu and Mason [27] demonstrate the time-index variables formulation is the only formulation that produces optimal solutions in their TWC experiments for a parallel machine scheduling problem. Cakici *et al.* [3] also propose a time-indexed model for a similar problem being considered herein so we adopt their model. There is one minor difference. In the above MIP model, jobs of same family can have different processing times, whereas the following model assumes jobs of same family to have the same processing time, which is more representative of the diffusion process.

2) MIP-TI (Time-indexed)

T	time slots (t)
p_b	processing time of batch b

Decision variables:

X_{mbt}	1 if batch b starts its processing on machine m at time t; otherwise, 0
Y_{bj}	1 if job j is assigned to batch b; otherwise, 0
Z_b	time at which batch b finishes its required processing

0894-6507 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSM.2017.2740340, IEEE Transactions on Semiconductor Manufacturing

C_j completion time of job j

$$\text{Minimize} \quad \sum_j w_j C_j \tag{2.1}$$

$$\sum_b Y_{bj} = 1 \qquad \forall j \tag{2.2}$$

$$Y_{bj} = 0 \qquad \forall j \in b \tag{2.3}$$

$$\sum_j s_j Y_{bj} \le k_m \qquad \forall b \tag{2.4}$$

$$\sum_{m=1}^{M} \sum_{t=0}^{T-p_b} X_{mbt} = 1 \qquad \forall b \tag{2.5}$$

$$\sum_{b=1}^{B} \sum_{\hat{t}=max\{0,t-p_b\}}^{t-1} X_{mb\hat{t}} \le 1 \qquad \forall m,t \tag{2.6}$$

$$Z_b = \sum_{m=1}^{M} \sum_{t=0}^{T-p_b} (t + p_b) X_{mbt} \qquad \forall b \tag{2.7}$$

$$Z_b \ge (r_j + p_b) Y_{bj} \qquad \forall b,j \tag{2.8}$$

$$C_j \ge Z_b - L(1 - Y_{bj}) \qquad \forall b,j \tag{2.9}$$

$$X_{mbt}, Y_{bj} \text{ are binary} \qquad \forall m,b,t,j \tag{2.10}$$

$$Z_b, C_j \ge 0 \qquad \forall b,j \tag{2.11}$$

Objective (2.1) minimizes the sum of total weighted completion time of all jobs. Constraints (2.2–2.3) ensure that jobs are assigned to one of the available batches that are eligible to include the corresponding job family. In order to increase the efficiency of the model, every batch is pre-designated for use by a specific job family and can only include jobs of that family. By this pre-definition, the number of variables is reduced as compared with the case of introducing batch assignment variables for all job-to-batch combinations. Machine batch size is taken into consideration in Constraint (2.4). Then, Constraint (2.5) enforces that each batch can start only at exactly one particular time and Constraint (2.6) ensures that at any given time at most one batch can be processed on each machine. Constraint (2.7) calculates a completion time of each batch as its processing start time plus its processing time. Constraint (2.8) ensures that a batch cannot start its processing until all jobs assigned to the corresponding batch become ready. Constraint (2.9) determines each job's completion time is determined by the completion time of the batch to which it is assigned. Finally, constraints (2.10–2.11) impose the binary and non-negativity restrictions, respectively.

Table I shows an example of fifteen jobs with different job release times, job sizes, and job families. In addition, there are two parallel batching machines with batch size of 50. Both MIP and CP successfully find an optimal solution for this problem. Figure 1 represents an optimal solution with the TWCT of 627. Each value shows a job and its properties, for instance, j8r1p6s8f3 indicates job 8 with release time of 1, processing time of 6, size of 8, and family of 3.

3) *Constraint programming model*

The CP Optimizer provides specialized variables,

TABLE I
THE EXAMPLE OF FIFTEEN-JOB PROBLEM WITH DIFFERENT RELEASE TIMES, SIZES, AND JOB FAMILIES.

Job id	Release time	Processing time	Job size	Family	Weight
1	10	6	24	3	2
2	9	10	10	2	1
3	2	10	14	2	2
4	7	10	20	1	2
5	10	10	2	1	2
6	2	6	15	3	5
7	5	10	12	1	5
8	1	6	8	3	3
9	10	6	16	3	2
10	7	6	17	3	1
11	0	10	7	1	1
12	6	10	7	2	4
13	1	6	21	3	3
14	1	6	21	3	3
15	8	10	15	2	4

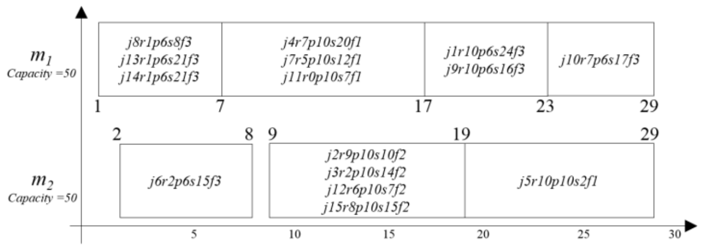

Fig. 1. An optimal solution for the fifteen-job example with different release times, sizes, and different job families.

constraints and functions designed for modelling scheduling problems. We exploit the features and develop a CP model. For a detailed discussion of CP modeling concepts, in particular using IBM CP Optimizer, please refer to Laborie and Rogerie [17], Laborie *et al*. [18], and IBM Software [8][9].

Figure 2 shows a schematic diagram representing *interval* variables, *state* functions, and *cumul* functions being used in this batch-processing machine scheduling problem. Each job is expressed as an *interval* and is represented as a box in the figure. For instance, job 14 can be declared as "*interval J14 in 1..EndMax size 6..szMax*" which indicates the earliest start time is 1 and the minimum processing time 6.

In addition to the notation used in the previous MIP models, the following notation is added to the CP model.

Sets
$I_{m,b}$ virtual set of jobs to be scheduled to during *state b* on machine *m*

Interval variables:
x_j object of job *j*

0894-6507 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSM.2017.2740340, IEEE Transactions on Semiconductor Manufacturing

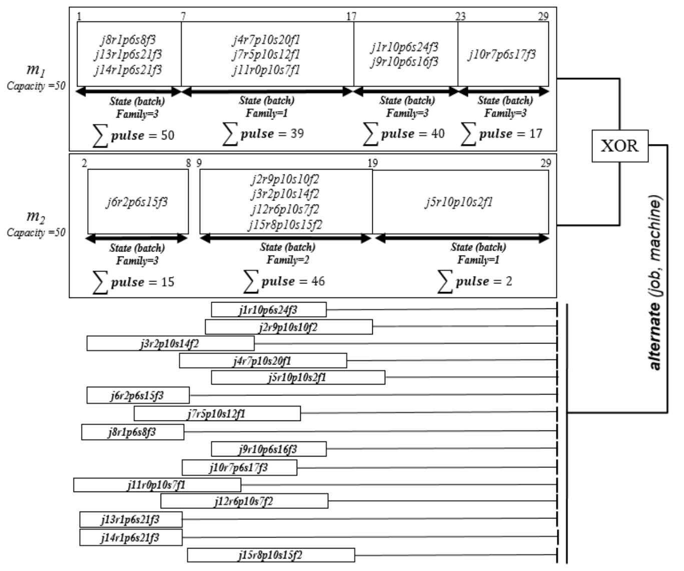

Fig. 2. A schematic diagram representing *interval* variables, *state* functions, and *cumul* functions.

$x_{j,m}$ object of job *j* at machine *m*

State variables:
b_m *state* value of machine *m*

We formulate the parallel batch-processing machines problem into a CP as follows.

Interval $x_j\ in\ [r_j)\ size\ p_j\ \ \forall j$ (3.1)

Interval $x_{jm}\ optional\ \ \forall j,m$ (3.2)

State $b_m\ \forall m$ (3.3)

Cumul $load_{m,b} = \sum_j pulse\,(x_{jm}, s_j)\forall j \in I_{m,b}, m$ (3.4)

Objective $Minimize \sum_j (w_j) endOf(x_j)$ (3.5)

Constraint $Alternate(x_j, \{x_{j,m}\})\ \forall j,m$ (3.6)

Constraint $AlwaysEqual(b_m, \{x_{j,m}\}, f_j)\ \forall j,m$ (3.7)

Constraint $load_{m,b} \le k_m\ \ \forall m$ (3.8)

Statement (3.1) declares the *interval* variables for the jobs. Note there is no *optional* flag, these *interval* variables are necessarily present. Therefore, each job must be assigned to a machine. Statement (3.2) declares the *interval* variables for the job to machine assignment as *optional* because not all machines are necessarily required to complete each job: interval variable $x_{j,m}$ will be present if and only if job *j* is allocated to machine *m*. Statement (3.3) declares the *state* function which implicitly divides the schedule horizon into multiple segments (batches). Statement (3.4) defines the *cumul* function which determines the total job size of a batch. Note that this *cumul* function is interwoven with the *state* function via the virtual set $I_{m,b}$. The objective function (3.5) minimizes the sum of total weighted completion time of all jobs. Constraint (3.6) forces each job to be assigned to exactly one machine. Constraint (3.7) specifies that each batch (*state*) can be represented by only one job family which models the incompatible job families. Finally, constraint (3.8) ensures that the total job size of a batch cannot exceed the batch size limit of the machine.

4) VNS model

We next briefly introduce the VNS model suggested by Cakici *et al.* [3]. For an extensive review on the heuristic, we refer readers to the original paper.

Different local search procedures based on insertion and/or swap moves of the jobs and/or batches are examined by Cakici *et al.* [3]. After an initial solution is found and neighbourhood distance is set as 1, a local search is performed until the stopping criteria is reached. In VNS, every time before performing local search, a random solution is selected from the defined neighbourhood of the best solution so far. Because release times and incompatible job families exist in the problem, they claim that achieving good solutions by starting the search from a random solution is quite difficult. In order to explore neighbourhoods effectively, they investigate four different local search procedures, both separately and sequentially and they are named LS1, LS2, LS3, and LS4. The first two procedures are based on the insertions and swaps of the jobs. A job insert move removes a job from one batch and inserts it into another. A job swap move selects two jobs from the same family and switches their batch assignments. Swap and insert moves of the batches are also examined in the last two procedures. Job swap moves always yield the same number of jobs assigned to batches and machines. Similarly, the number of batches processed on each machine remains constant when batch swaps are applied. On the other hand, any improvement involving more than a single job's batch re-assignment or a single batch's repositioning is not easily found with an insert move. Therefore, they jointly apply local search procedures in a sequential manner to overcome these weaknesses and come up with a total of 22 different VNS heuristics. They found H20 (LS2 + LS1 + LS4 + LS3) and H22 (LS4 + LS3 + LS2 + LS1) are the best performing heuristics.

III. Computational Experiments

In this section, we test the effectiveness of our CP model. We compare it with the MIP model as well as the VNS heuristics from Cakici *et al.* [3]. MIP and CP models are generated by IBM OPL and solved by CPLEX 12.6.3 on a personal computer with an Intel Core i5-3470 @ 3.2 Ghz processor and 16 GB RAM.

A. Problem instances

To test our model, we borrow the same test problem instances used by Cakici *et al.* [3]. They consider four levels of the number of jobs: 15, 25, 50, and 100, and also consider two levels of the number of machines: 2 and 3. Two different levels of number of job families are investigated: 3 and 5 as shown in Table II. The maximum batch size is set as 50, and job sizes are generated from a discrete uniform distribution of

0894-6507 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

[1, 50]. Processing times are generated from a discrete uniform distribution of [1, 15]. Job weights are randomly generated from a discrete uniform distribution of [1, 10]. For each combination of the levels, one-hundred-sixty problem instances are generated yielding a total of 2560 (4 × 2 × 2 × 160) problem instances.

TABLE II
Factors and levels

Factors	Levels
Jobs	15, 25, 50, 100
Families	3, 5
Machines	2, 3

We limit the computational time of CP to 180 seconds because a diffusion scheduling system in the semiconductor industry is expected to generate a Gantt-chart schedule every few minutes.

TABLE III
PERFORMANCE COMPARED TO OPTIMALITY (15-JOB INSTANCE).

15 jobs	MIP-TI	MIP-PA	H20	H22	CP
Avg. TWCT	1347.87 (640)	1349.75 (493)	1394.60 (95)	1395.77 (91)	1347.87 (639)
Avg. PR	1.0000	1.0014	1.0347	1.0355	1.0000
Avg. Run-time	22.2282	299.02	19.1417	19.0769	1.7858

TABLE IV
PERFORMANCE COMPARED TO BEST SOLUTIONS (25-JOB INSTANCES).

25 jobs	H20	H22	CP
Avg. TWCT	3085.2484 (24)	3086.4219 (18)	2925.7250 (633)
Avg. PR	1.0545	1.0549	1.0001
Avg. Run-time	39.8255	40.0332	21.0010

B. Results

Table III summarizes the computational results of the small 15-job problem instances, a total of 640 instances. Columns 2–6 contain the performance measures of each model. Cakici *et al.* [3] found that H20 and H22 are the best performing VNS heuristics so we benchmark them. Row 1 contains the average TWCT of each model. The number of times that a given model produced an optimal solution for each problem instance is reported in parentheses. Row 2 reports the performance ratio, $\frac{TWCT(model)}{TWCT(optimal)}$, to assess the solution quality of each model. The last row reports the average run-time of each model. All the test instances, log files, and summary are located at http://schedulingworld.com/

The MIP-TI from Cakici *et al.* [3] successfully reaches at an optimal for all instances within 22.2 seconds on average. On the other hand, MIP-PA finds an optimal for 493 out of 640 instances within 299.02 seconds on average. We confirm the effectiveness of the time-indexed MIP model.

CP finds an optimal for all problem instances but one problem instance (3m5f_12116). The optimal objective function value is 719 whereas CP returns 720. However, it impressively finds optimal solutions on average within 2 seconds compared to 22 seconds by MIP-TI.

Table IV summarizes the performance measures for the 25-job problem instances. CP finds the best solutions in 633 out of 640 instances and is twice as fast as the VNS heuristics. At this time, both MIP models could not reach at an optimal even after several hours of runtime so the table only reports CP and VNS results. However, we are still interested in the solution quality against an optimal so we let CPLEX run for an unlimited time.

TABLE V
PERFORMANCE COMPARED TO **OPTIMAL** SOLUTIONS (25-JOB INSTANCES).

Instances	MIP-TI	Best of H20 & H22		CP	
	TWCT (Runtime)	TWCT (Runtime)	PR	TWCT (Runtime)	PR
2m3f_1_1_1_1_1	1527* (194164.6)	1580 (14.7)	1.0347	1562 (0.9)	1.0229
2m5f_1_1_1_1_1	1675* (1213.88)	1741 (38.2)	1.0394	1706 (14.2)	1.0185
3m3f_1_1_1_1_1	1146* (4029.02)	1155 (14.3)	1.0079	1152 (7.5)	1.0052
3m5f_1_1_1_1_1	1290* (1938.50)	1299 (59.6)	1.0070	1290 (11.2)	1.0000

* indicates the optimal solution

TABLE VI
PERFORMANCE COMPARED TO BEST SOLUTIONS (50-JOB INSTANCES).

50 jobs	H20	H22	CP
Avg. TWCT	9670.4203 (9)	9675.4391	9051.0359
Avg. PR	1.0697	1.0702	1.0001
Avg. Run-time	118.1377	117.4761	57.4407

TABLE VII
PERFORMANCE COMPARED TO BEST SOLUTIONS (100-JOB INSTANCES).

100 jobs	H20	H22	CP
Avg. TWCT	33051.5438	33044.1906	30798.3297
Avg. PR	1.0732	1.0729	1.0005
Avg. Run-time	218.2115	230.7303	109.8975

Table V reports a comparison against optimal solutions of four individual instances after the exhaustive experimentation. We find our CP model is capable of generating solutions within about 2% of optimality in these 25-job instances.

Tables VI–VII summarize the performance measures for the 50 and 100 job problem instances. In this experimentation, CPLEX crashed with an error of “out of memory” after several days of runtime so we cannot compare to an optimal. When we compare CP against the VNS heuristics, CP consistently finds the best solutions faster than the VNS heuristics.

Tables VIII–IX summarize the performance measures by the number of machines and the number of job families respectively. The results do not show a significant difference between the different levels.

It is worth mentioning that the proposed CP model always finds a feasible solution within 1 *s* for all test problem instances. This remarkable run-time advantage of CP model may bring a paradigm shift to real-time scheduling, from real-time dispatching, that is still prevalent in practice.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSM.2017.2740340, IEEE Transactions on Semiconductor Manufacturing

TABLE VIII
PERFORMANCE COMPARED TO BEST SOLUTIONS
(ALL JOB SIZES: FACTOR = NUMBER OF MACHINES).

Machines	Indexes	H20	H22	CP
2	Avg. TWCT	13807.14 (71)	13807.43 (70)	12915.33 (1256)
	Avg. PR	1.0691	1.0691	1.0000
	Avg. Run-time	122.21	56.06	44.71
3	Avg. TWCT	9793.77 (71)	9793.48 (65)	9146.15 (1264)
	Avg. PR	1.0708	1.0708	1.0000
	Avg. Run-time	75.45	147.60	50.35

TABLE IX
PERFORMANCE COMPARED TO BEST SOLUTIONS
(ALL JOB SIZES: FACTOR = NUMBER OF FAMILIES).

Families	Indexes	H20	H22	CP
3	Avg. TWCT	11718.84 (55)	11711.44 (50)	10905.86 (1270)
	Avg. PR	1.0745	1.0739	1.0000
	Avg. Run-time	132.91	70.98	47.17
5	Avg. TWCT	11882.07 (87)	11889.48 (85)	11155.63 (1250)
	Avg. PR	1.0651	1.0658	1.0000
	Avg. Run-time	64.75	132.68	47.89

IV. CONCLUSION

We address the parallel batch-scheduling problem which involves the constraints of different job release times, non-identical job sizes, and incompatible job families which can be represented as $P_m|r_j$, *batch, incompatible* $|\sum w_j C_j$. We first represent the problem with an MIP formulation based on positional and assignment variables and later adopt a time-indexed MIP model for a faster computational time. Then, we propose a CP representation for the first time. Computational results demonstrate that the CP model has a computational advantage over the VNS heuristic as well as over the MIPs. It further reveals that CP is able to find an optimal for 639 out of 640 instances for 15-job instances in less than 2 seconds on average. For the 25-jobs instances, CP is capable of generating solutions within 2% of an optimal solution. When we compare CP against the VNS heuristics, 3–7% reduction of TWC and 47–91% reduction of computational time are achieved overall.

This research can be further extended by considering multi-stage batch operations. In the semiconductor industry, researchers have investigated the performance of local functional areas such as lithography, diffusion, etch, and implanter for the last two plus decades. Now, there is a growing need of orchestrating a whole factory to seek global optimization. One of strong use-cases is to schedule jobs having a time constraint between consecutive process steps (Sun *et al.* [24], Klemmt and Mönch 2012 [13], Sadeghi *et al.* [23]). Another use-case is to schedule urgent (hot, rocket, or ultra) jobs in order to meet a due date. Due to the non-preemptive nature of machines in a fab, a floor supervisor often takes an extreme measure letting some load ports of a machine empty as urgent jobs are approaching from an upstream operation which results in a productivity loss. A similar application seems to be prevalent in the production of pharmaceuticals (Sundaramoorthy and Maravelias [25], Kopanos *et al.* [16]). However, their batch stays unchanged throughout the floor compared to the dynamic nature of semiconductor industry where jobs are repetitively grouped for a batch-processing and un-batched for single processing, as jobs travel routes which are comprised of 500–1000 steps. Lastly, it will be interesting to compare CP against other metaheuristics in terms of run-time and solution-quality.

ACKNOWLEDGMENTS

The author would like to thank Philippe Laborie (IBM CPLEX Optimization Studio Team) for the valuable comments and technical support.

REFERENCES

[1] Almeder, C., and Mönch, L. 2011. Metaheuristics for scheduling jobs with incompatible families on parallel batching machines. *Journal of the Operational Research Society*, 62(12), 2083-2096.
[2] Balasubramanian, H., Mönch, L., Fowler, J., and Pfund, M., 2004. Genetic algorithm based scheduling of parallel batch machines with incompatible job families to minimize total weighted tardiness. *International Journal of Production Research*, 42(8), 1621–1638.
[3] Cakici, E., Mason, S. J., Fowler, J. W., and Geismar, H. N. 2013. Batch scheduling on parallel machines with dynamic job arrivals and incompatible job families. *International Journal of Production Research*, 51(8), 2462–2477.
[4] Dobson, G., and Nambimadom, R. S. 2001. The batch loading and scheduling problem. *Operations Research*, 49, 52–65.
[5] Fowler, J. W., Phillips, D. T., and Hogg, G. L. 1992. Real-time control of multiproduct bulk-service semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing*, 5(2), 158–163.
[6] Glassey, C. R., & Weng, W. W. (1991). Dynamic batching heuristic for simultaneous processing. *IEEE Transactions on Semiconductor Manufacturing*, 4(2), 77-82.
[7] Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5, 287-326.
[8] IBM Software, 2010. *Modeling with IBM ILOG CPLEX CP Optimizer – Practical Scheduling Examples* [White paper]. Retrieved from http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/practical-scheduling-examples/
[9] IBM Software, 2015. IBM ILOG CPLEX Optimization Studio V12.6.3.
[10] Jung, C., Pabst, D., Ham, M., Stehli, M., & Rothe, M. (2014). An effective problem decomposition method for scheduling of diffusion processes based on mixed integer linear programming. *IEEE Transactions on Semiconductor Manufacturing*, 27(3), 357-363.
[11] Kalir, A., & Bouhnik, S. (2006). Achieving reduced cycle times in semiconductor manufacturing via effective control of the PK equation factors. IFAC Proceedings Volumes, 39(3), 65-69.
[12] Keha, A. B., Khowala, K., & Fowler, J. W. (2009). Mixed integer programming formulations for single machine scheduling problems. *Computers & Industrial Engineering*, 56(1), 357-367.
[13] Klemmt, A., & Mönch, L. (2012, December). Scheduling jobs with time constraints between consecutive process steps in semiconductor manufacturing. *In Proceedings of the Winter Simulation Conference*, 194.
[14] Klotz, E., and Newman, A. M., 2013. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18(1), 18–32.
[15] Koh, S. G., Koo, P. H., Kim, D. C., and Hur, W. S. (2005). Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families. *International Journal of Production Economics*, 98(1), 81–96.
[16] Kopanos, G. M., Méndez, C. A., & Puigjaner, L. (2010). MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry. *European journal of operational research*, 207(2), 644-655.
[17] Laborie, P., and Rogerie, J. 2008. Reasoning with Conditional Time-Intervals. *In FLAIRS conference*, 555–560.

0894-6507 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

[18] Laborie, P., Rogerie, J., Shaw, P., and Vilim, P. 2009. Reasoning with Conditional Time-Intervals. Part II: An Algebraical Model for Resources. *In FLAIRS conference*, 201–206.

[19] Lawler EL, Labetoulle J., 1978. On preemptive scheduling of unrelated parallel processors by linear programming. *Journal of the ACM*, 25(4), 612–619.

[20] Malapert, A., Guéret, C., and Rousseau, L. M. 2012. A constraint programming approach for a batch processing problem with non-identical job sizes. *European Journal of Operational Research*, 221(3), 533–545.

[21] Mazumdar, C. S., Mathirajan, M., Gopinath, R., and Sivakumar, A. I. 2008. Tabu Search methods for scheduling a burn-in oven with non-identical job sizes and secondary resource constraints. *International Journal of Operational Research*, 3(1-2), 119–139.

[22] Mönch, L., Fowler, J.W., and Mason, S.J. 2013. Production Planning and Control for Semiconductor Wafer Fabrication Facilities: Modeling, Analysis, and Systems. Vol. 52. Springer.

[23] Sadeghi, R., Dauzere-Peres, S., Yugma, C., & Lepelletier, G. (2015, May). Production control in semiconductor manufacturing with time constraints. *IEEE/SEMI Advanced Semiconductor Manufacturing Conference* (*ASMC*), 29-33.

[24] Sun, D. S., Choung, Y. I., Lee, Y. J., & Jang, Y. C. (2005, September). Scheduling and control for time-constrained processes in semiconductor manufacturing. *IEEE International Symposium on Semiconductor Manufacturing*, 295-298.

[25] Sundaramoorthy, A., & Maravelias, C. T. (2008). Simultaneous batching and scheduling in multistage multiproduct processes. *Industrial & Engineering Chemistry Research*, 47(5), 1546-1555.

[26] Timpe, C., 2002. Solving planning and scheduling problems with combined integer and constraint programming. *OR spectrum*, 24(4), 431–448.

[27] Unlu, Y., & Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers & Industrial Engineering*, 58(4), 785-800.

[28] Uzsoy, R., 1994. A single batch processing machine with non-identical job sizes. *International Journal of Production Research*, 1615–1635.

[29] Uzsoy, R., 1995. Scheduling batch processing machines with incompatible job families. *International Journal of Production Research*, 33(10), 2685–2708.

[30] Vilím, P., Laborie, P., and Shaw, P. 2015. Failure-Directed Search for Constraint-Based Scheduling. *In Integration of AI and OR Techniques in Constraint Programming*, 437–453.

[31] Wang, T., Meskens, N., and Duvivier, D. 2015. Scheduling operating theatres: Mixed integer programming vs. constraint programming. *European Journal of Operational Research*, 247(2), 401–413.

[32] Weng, W. W., & Leachman, R. C. (1993). An improved methodology for real-time production decisions at batch-process work stations. *IEEE Transactions on Semiconductor Manufacturing*, 6(3), 219-225.

[33] Yugma, C., Dauzère-Pérès, S., Derreumaux, A., and Sibille, O. 2008. A Batch Optimization Sofware for diffusion area scheduling in semiconductor manufacturing. *IEEE/SEMI Advanced Semiconductor Manufacturing Conference* (*ASMC*), 327–332.

BIOGRAPHIES

Andy Ham was born in Suwon, South Korea. He received Ph.D. in industrial engineering from Arizona State University in 2009, and M.S. in OR/IE from University of Texas at Austin in 2000.

He is currently working as an associate professor in Industrial and Systems Engineering, Liberty University, Virginia. Prior to the current position, he worked for Samsung Electronics, Samsung Austin Semiconductor, Globlafoundries, AMD, and ILOG, in the areas of modeling, real-time dispatching, real-time scheduling, supply chain management, and decision analysis. His research has been shifted into scheduling of drones and robots in a smart factory, and scheduling of a fleet of self-driving cars, a fleet of human-driving cars, a set of drivers, and a set of orders, in a future taxi company. His research has been published in peer-reviewed journals such as *IEEE Transactions on Semiconductor Manufacturing*, *IEEE Transactions on Automation Science and Engineering*, *Applied Mathematical Modelling*, *International Journal of Production Research*, *Computers & Industrial Engineering*, etc.

John W. Fowler is the Motorola Professor of International Business 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 120 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 recently served as Editor-in-Chief for *IIE Transactions on Healthcare Systems Engineering*. He is also an Editor of the *Journal of Simulation* and an Associate Editor of *IEEE Transactions on Semiconductor Manufacturing.* He is a Fellow of the Institute of Industrial and Systems Engineers (IISE) and recently served as the IISE Vice President for Continuing Education, is a former INFORMS Vice President, and served on the WSC Board of Directors 2005-2017.