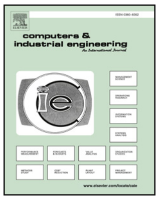
Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie



On scheduling a photolithography area containing cluster tools

Sreenath Chalil Madathil[a,*], Siddhartha Nambiar[c], Scott J. Mason[b], Mary E. Kurz[b]

[a] *The Research Foundation for State University of New York, Binghamton, NY, USA*
[b] *Department of Industrial Engineering, Clemson University, Clemson, SC 29634, USA*
[c] *Department of Industrial and Systems Engineering, North Carolina State University, Raleigh, NC 27607, USA*

ARTICLE INFO

Keywords:
Photolithography
Scheduling
Cluster tools
Optimization
Heuristics
Genetic algorithm

ABSTRACT

Photolithography is typically the bottleneck process in semiconductor manufacturing. In this paper, we present a model for optimizing the scheduling of the photolithography process in the presence of both individual and cluster tools. The combination of these individual and cluster tools that process various layers (stages) of the semiconductor manufacturing process flow is a special type of flexible flowshop. We seek separately to minimize total weighted completion time and maximize on-time delivery performance. Experimental results suggest that our solution algorithms show promise for real world implementation as they can help to improve resource utilization, reduce job completion times, and decrease unnecessary delays in a wafer fab.

1. Introduction

Scheduling and sequencing are indispensable processes in industry. A well-designed scheduling system helps the industry focus on increasing throughput by reducing the run time of machines, thereby saving money. Processing jobs on a "first-come, first-serve" basis may not be an optimal policy on the factory floor (Conway, Maxwell, & Miller, 2012). The semiconductor wafer fabrication industry is one of the largest industrial manufacturing segments. Implementing a proper scheduling system in wafer fabrication can help increase profit margins as well as reduce the time required to produce the wafers that contain integrated circuits.

In semiconductor manufacturing, photolithography is normally one of the bottleneck processes that require high capital investments (Sha, Hsu, Che, & Chen, 2006). Hence, optimizing the photolithography process by efficiently scheduling the jobs could be beneficial for the industry. Machines that perform various steps in photolithography can be organized as a flexible flowshop system. A flexible flowshop is defined as a system in which the jobs need to be processed at different sequential stages and at least one of the stages has more than one machine operating in parallel. With the advancement of technology and because of their efficiency and profitability, cluster tools were added to the wafer fabrication processes in recent years. A cluster tool combines various types of machines that perform individual processes and organizes them around a robotic wafer transport device (Yim & Lee, 1999). These tools consist of those machines that are capable of processing two or more stages and combine several processing modules into a single machine (Lee, 2008).

In this research, we develop a scheduling model for the photolithographic process, which is a special type of flexible flowshop (FFS) that has cluster tools along with the traditional individual photolithography tools. According to Chiang (2013), photolithography scheduling is more complex than tradition flexible flowshop scheduling. The author reviews several reasons for this scheduling complexity such as re-entrant job flow, a jobs' readiness, due dates, multiple machine types, multiple orders per job, and lot priorities. Each of the jobs that enter the system typically re-visits equipment visited at earlier manufacturing (*i.e.*, reentrant flow). If the proposed model is tested successfully, it could be implemented in the semiconductor industry that employs photolithography machines with advanced cluster tools. Wafer fabs will be able to schedule their machines to improve utilization of the machines, reduce the processing time for jobs, and efficiently schedule without introducing unnecessary delays in the process.

In short, the key contributions of this paper are:

- To the best of our knowledge, this is the first model that schedule a photolithographic process that consists of both cluster tools and standalone tools with reentrant job flow across multiple product types, job ready times and the continuous flow of jobs inside cluster tools.
- To develop a mixed integer programming model (MIP) to solve this special FFS.
- To implement two heuristic algorithms and compare their performances with respect to the MIP model.

* Corresponding author.
E-mail addresses: schalil@g.clemson.edu (S. Chalil Madathil), snambia@ncsu.edu (S. Nambiar), mason@clemson.edu (S.J. Mason), mkurz@clemson.edu (M.E. Kurz).

https://doi.org/10.1016/j.cie.2018.05.036
Received 8 September 2017; Received in revised form 13 April 2018; Accepted 22 May 2018
Available online 25 May 2018
0360-8352/ Published by Elsevier Ltd.

2. Literature review

Most manufacturing industries face various challenges such as processing high priority jobs, unforeseen breakdowns, scheduled maintenance, delayed processing of jobs, and meeting deadlines set by customers. Proper production planning and process scheduling help to maintain or improve the efficiency of systems and control of operations (Pinedo, 1995). The significance of proper production scheduling comes to light in this scenario when manufacturers need to satisfy customer demands with the help of a minimal number of photolithography tools missing no committed completion time. This committed completion time is the due date (Pinedo, 1995). Montazeri et al. explained and reviewed different scheduling rules, such as static and dynamic rules (Montazeri & Van Wassenhove, 1990). Static and dynamic rules depend on the time when the rule is applied. Static rules, applied at the start of the scheduling period, have a fixed schedule and dynamic rules change as the time progress. The authors also reviews various scheduling rules, compares their performance measures for different environments and conclude that performance evaluation depends on the objective under consideration (Montazeri & Van Wassenhove, 1990).

The four basic processes involved in manufacture of integrated circuits are wafer fabrication, wafer probe, assembly and packaging, and final testing (Uzsoy, Lee, & Martin-Vega, 1992). A wafer fabrication process includes complex procedures and technologies that involve high capital investments. The proper utilization of wafer fabs can lead to increased profit for a semiconductor wafer fabricator. Each time a wafer passes through photolithography, a new layer of required circuitry is formed on the wafer. For most wafers there will be at least 25 such layers. Since the photolithography process is repeated during wafer fabrication, overall performance of the systems is improved by improving the photolithography process (Arisha & Young, 2004). The high capital cost of the photolithography tools forces the wafer manufacturers to streamline the processes to utilize these machines to the fullest possible extent.

There are many literatures and textbooks that explains the machine environments like a single machine, parallel machines, flowshops, job shops, flexible flowshops, and flexible job shops found in industries (Pinedo, 1995). Many mixed-integer programming (MIP) models for scheduling FFS are explained in Sawik (2011). The book considers various scenarios of flowshop modeling with multiple machines in each stage and finite or infinite buffers between each stage. According to Floudas and Lin (2005), many scheduling problems use Mixed Integer Linear Programming (MILP) to find solutions due to their rigorousness, resilience, and flexible design capabilities. Indeed, the use of MIP models is rather popular in this regard.

Ruiz discusses the various solution approaches for the FFS problems, which includes exact methods, heuristics, and meta-heuristics (Ruiz & Vázquez-Rodríguez, 2010). In exact methods approaches such as branch and bound, algorithms solve problems to optimality. The problem with branch-and-bound algorithms is that they utilize a large amount of computer processing resources and are able to solve only problems with a few jobs and stages. Often, they are also deemed to be too complex for real world problems. Lowe and Mason (2016) proposed a deterministic MIP model to schedule weekly production quantities for semiconductor manufacturing in order to meet forecasted demand over a six-month planning horizon. MIP models are proposed in Sawik (2012) for deterministic batch or cyclic scheduling in flow shops with parallel machines and finite in-process buffers. Further, Sawik (2014) presented a new MIP formulation for cyclic scheduling in flow lines with parallel machines and finite in-process buffers, where a Minimal Part Set (MPS) in the same proportion as the overall production target is repetitively scheduled.

A simple, two-stage flexible flowshop is strongly NP-hard (Hoogeveen, Lenstra, & Veltman, 1996). According to Kyparisis and Koulamas (2001), minimizing total weighted completion time for a multiple stage flexible flowshop scheduling problem is NP hard. Hence by extension, the complexity of scheduling a larger flexible flowshop with multiple machines in almost every stage of its processing is also strongly NP hard. When compared to traditional flowshops, a photolithography system involving cluster tools, constraints for multiple wafer routes, reentrant flow, and no buffers inside the cluster tool are therefore also strongly NP hard (Yim & Lee, 1999). Since the practical-sized complex FFS problems NP-hard, we require smart heuristics to arrive at good solutions (Jungwattanakit, Reodecha, Chaovalitwongse, & Werner, 2007).

Solving FFS problems by heuristic methods like dispatching rules and variants of shifting the bottleneck procedure (SBP) (Cheng, Karuno, & Kise, 2001) are explained by Lee (2008). Sarin, Varadarajan, and Wang (2011) provides an overview of advanced dispatching rules and compares the effectiveness of the performance from various simulation studies in a wafer fab. These dispatching rules include scheduling of general wafer fab, specific operations at bay level like photolithography, batch processing, etc. The primary characteristics that make wafer fab scheduling such a different problem includes batching, reentrant flow, sequence dependent setups, and parallel machines (Mönch, Fowler, Dauzère-Pérès, Mason, & Rose, 2011).

Dispatching rules include certain rules of thumb for the priority assignment of jobs onto machines. Some examples of dispatching rules include Shortest Processing Time (SPT), Longest Processing Time (LPT), and Shortest Remaining Processing Time (SRPT). The SBP uses a divide-and-conquer strategy and has been proven very effective when used in combination with exact methods for solving problems. The scheduling of a flexible flowshop with cluster tools is performed via simulated annealing (Yim & Lee, 1999) to obtain a near-optimal solution. However, the study does not consider the re-entry of jobs to previous stages. Pan et al. provide a recent comprehensive literature review of the scheduling of cluster tools in semiconductor manufacturing (Pan, Zhou, Qiao, & Wu, 2018).

Genetic Algorithms (GA) are a popular tool used in a number of papers focused on applications in real-world problems (Oduguwa, Tiwari, & Roy, 2005). GAs have been adapted to solve problems involving sequence-dependent setup times, several production stages with unrelated parallel machines at each stage, and machine eligibility (Ruiz & Maroto, 2006). The choice of how the GA solution is represented is an important facet in the design of a GA, as representation affects other design choices, such as crossover and mutation functions. A commonly employed representation scheme is the topological ordering of the tasks. Ramachandra and Elmaghraby (2006) minimize the weighted sum of completion times in a flexible flowshop by representing the chromosomes as topological orderings of jobs, the schedules of which are obtained using a first-available machine rule for machine assignments.

Table 1 summarizes the relevant literatures. Even though most of the papers reviewed have mentioned either the scheduling of flowshops, the scheduling of flexible flowshops, and/or scheduling of cluster tools separately, there exist no efficient models that analyze a flexible flowshop that contains cluster tools and reentrant job flow across multiple product types. We will also consider job ready times and the continuous flow of jobs inside cluster tools. In this research, we develop a scheduling model for the photolithographic process, that has cluster tools along with traditional photolithography tools, and considers reentrant job flow across multiple product types. Additionally, we use two heuristic algorithms to provide numerical results.

3. Problem description

The photolithography FFS system is arranged in such a way that the individual machines at each stage are organized as a general FFS with a few sets of cluster tools included. As jobs routed through the various stages of the photolithography process could return to one or more of these stages during their processing path, photolithography is a reentrant

Table 1
Summary of relevant literature.

Reference	Objective	Methodology	Features	Complexity
Yim and Lee (1999)	C_{max}	Heuristic Simulated Annealing Candidate list	Re-entry jobs No stand-alone tools No ready times	NP hard
Kyparisis and Koulamas (2001)	$\sum W_j Cj$	Approximation WSPT heuristic Worst-case analysis	Parallel machines No ready time No cluster tools	NP hard
Kock et al. (2007)	C_{max} Throughput	Discrete event simulation Aggregate modeling approach	No ready time no re-entry	N/A
Zhou et al. (2014)	C_{max}	Heuristic	Re-entrant flow Cluster tools	NP hard
Park et al. (2017)	C_{max}	Fab-level simulation	Lot cycle-time Lot residency time Lot throughput time	N/A
Zhang et al. (2018)	C_{max}	Heuristic Imperialist competitive algorithm Rolling horizon	Ready times Re-entry jobs No cluster tools	NP hard

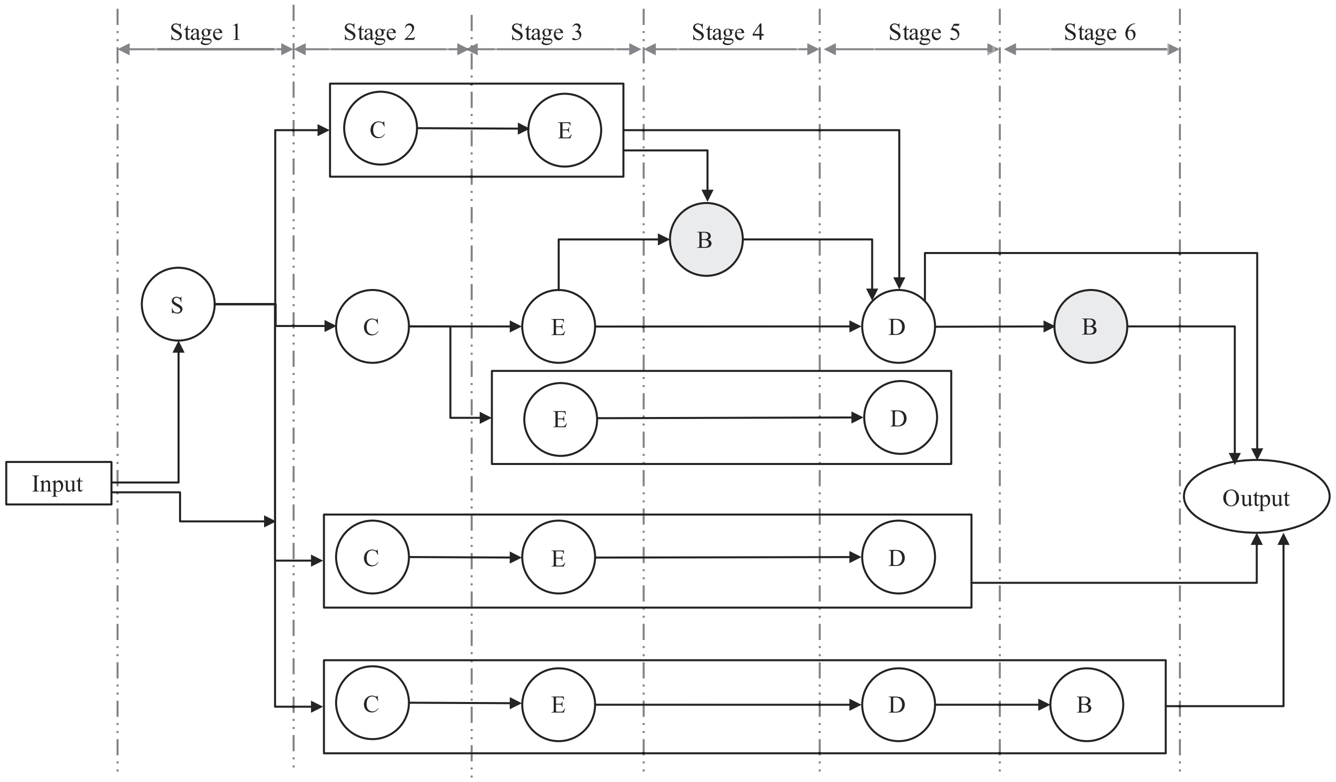


Fig. 1. Schematic diagram for the photolithography process.

flexible flowshop (Graves, Meal, Stefek, & Zeghmi, 1983). The multistep "photo" process is now described in detail and is shown in Fig. 1. The baking process in stage 4 and stage 6 can use the same bake ovens (tools) and hence represent a reentrant stage (*i.e.*, reentrant flow).

3.1. Photolithography stages

In the first step of the photolithography process, a semiconductor wafer may be cleansed in a sink (tool "S" in Fig. 1) (McGuigan, 1992). The wafer is coated (tool "C" in Fig. 1) with photosensitive resist and is exposed ("E" in Fig. 1) to light. Wafers are exposed to light with the help of a pattern mask that controls the wafer areas that receive light exposure. This helps to define the required circuit functionality. The exposed wafer is then developed ("D" in Fig. 1) so that the required patterns are imprinted onto the wafer by removing the exposed photoresist. The final photolithography stage is baking ("B" in Fig. 1). Sometimes, wafers are baked before and/or after the developing stage.

The flow diagram of the photolithography process for a single layer of wafer fabrication is illustrated in Fig. 1. Normally, a wafer repeats this process 20–30 times during its process flow. Fig. 1 also depicts cluster tools that are used in the photolithography process. Cluster tool "CEDB" processes the Coat, Expose, Develop, and Bake steps in order. Similarly cluster tools "CED," "CE", and "ED" process Coat-Expose-Develop (CED), Coat-Expose (CE), and Expose-Develop (ED), respectively. A job can have multiple routes based on the process flow. For example, a job that requires a coat, expose, develop and bake processes can flow through any of the 4 paths using the combinations of tools like $C \rightarrow E \rightarrow D \rightarrow B$, $CE \rightarrow D \rightarrow B$, $CED \rightarrow B$ or CEDB.

The transportation of wafer lots inside a wafer fab (interbay and intrabay) is via automated material handling systems (AMHSs). AMHSs transport lots both to stockers and to downstream production tools, as dictated by the shop floor control system (MES), as soon as the current tool the lot is on completes its processing. The availability of the downstream tool determines the destination of completed lots, whether to the stocker or to the downstream processing tool. This wafer transportation inside a wafer fab takes place asynchronously with wafer transported to the next tool as soon as current tool completes its process and the tool for the next process is free (Kock, Veeger, Etman, Lemmen, & Rooda, 2007). Another transport technique is synchronous in which transportation of the all wafers to the next process happen when the slowest wafer has finished processing. The WIP stockers in wafer fabs are sized to hold all required work-in-process within the fab (Cardarelli & Pelagagge, 1995). In this way, lots waiting to be processed do not wait in a physical queue located on/near the production equipment, but rather in a virtual queue as house in the WIP stocker. Stockers are typically located throughout the wafer fab, often at the end of each processing bay along the main center aisle.

3.2. Methodology

We propose a MILP model to schedule this photolithography stages and attribute our model's combinatorial nature to various discrete decisions such as job's assignment to machines and sequencing of jobs in a machine. We develop two heuristic algorithms, compare the solution quality and solving time for the three methods, and evaluate the heuristic's performance for larger problems.

The set of jobs entering the system for processing can be characterized by their ready times (r_j), the time at which the job is released to the shop by some external job scheduler (Conway et al., 2012). An initial step in optimizing industrial processes often includes improving the total execution time of machines, also known as makespan (C_{max}). A schedule that minimizes makespan can be obtained by applying MIP techniques (Floudas & Lin, 2005).

In this research, we develop a MIP formulation to optimize the makespan of the photolithography process containing reentrant jobs with ready times. In terms of the standard $\alpha|\beta|\gamma$ scheduling notation introduced in Graham, Lawler, Lenstra, and Kan (1979), the problem under consideration is defined as $FFm|r_j, rcrc|C_{max}$ (Pinedo, 1995). Other objectives that could be optimized in a scheduling system include total weighted completion time (WCT) and total weighted tardiness (TWT). The total weighted completion time is represented as $\sum w_k C_k$ with w_k denoting the weight or priority of job k and C_k representing the completion time of job k. In practice, WCT is a surrogate measure of the inventory or holding cost incurred by the schedule (Pinedo, 1995). The total weighted tardiness, $\sum w_k T_k$ where T_k is the tardiness of the job k, is generally an objective that relates to on-time delivery.

4. Mathematical formulation

This section explains the model formulation along with the notations used in the model, followed by an explanation of the constraint sets. The MIP formulation for scheduling flowshops with parallel machines and infinite in-process buffers (Sawik, 2011) is used as the base model for this FFS and additional constraints are added to expand this formulation to incorporate the scheduling of cluster tools and reentrant flow.

4.1. Model parameters and variables

4.1.1. Notation

Sets

I	set of processing stages indexed by $i \in I = \{1, ..., m\}$
J_i	set of processors in each stage i indexed by $j \in J_i = \{1, ..., m_i\}$
K	set of jobs that needs to be processed indexed by $k \in K = \{1, ..., n\}$
$CEDB$	set of cluster machines for $C-E-D-B$ indexed by i_1
CED	set of cluster machines for $C-E-D$ indexed by i_2
CE	set of cluster machines for $C-E$ indexed by i_3
ED	set of cluster machines for $E-D$ indexed by i_4

Parameters

m	number of processing stages
m_i	number of machines at each processing stage i
n	number of jobs
P_{ik}	processing time for job k in stage i
r_k	ready time for job k
d_k	due date for job k
w_k	priority for job k
M	sum of the processing time of all jobs in the system

Decision variables

C_{ik}	completion time of job k at stage i
C_{max}	makespan or time at which all jobs complete their operations on all stages
x_{ijk}	1, if job k is assigned to machine j in stage i; 0, otherwise
y_{kl}	1, if job k precedes job l in the processing sequence; 0, otherwise

4.2. Model formulation

$$\min C_{max} \tag{1}$$

$$C_{1k} \geqslant P_{1k} + r_k \quad \forall\, k \in K \tag{2}$$

$$C_{ik} - C_{(i-1)k} \geqslant P_{ik} \quad \forall\, k \in K,\ i \in I,\ i > 1 \tag{3}$$

$$C_{ik} + M(2 + y_{kl} - x_{ijk} - x_{ijl}) \geqslant C_{il} + P_{ik} \quad \forall\, i \in I,\ j \in J_i,\ k \in K,\ l \in K,\ l > k \tag{4}$$

$$C_{ik} + M(3 - y_{kl} - x_{ijk} - x_{ijl}) \geqslant C_{ik} + P_{il} \quad \forall\, i \in I,\ j \in J_i,\ k \in K,\ l \in K,\ l > k \tag{5}$$

$$C_{mk} \leqslant C_{\max} \quad \forall\, k \in K \tag{6}$$

$$\sum\nolimits_{j \in J_i} x_{ijk} = 1 \quad \forall\, i \in I,\ k \in K,\ P_{ik} > 0 \tag{7}$$

$$\sum\nolimits_{j \in J_i} x_{ijk} = 0 \quad \forall\, i \in I,\ k \in K,\ P_{ik} = 0 \tag{8}$$

$$\sum\nolimits_{i \in I} x_{ii_1k} = 3x_{2i_1k} \quad \forall\, i_1 \in \text{CEDB},\ k \in K,\ i_1 \neq 0,\ 3 \leqslant i \leqslant m,\ i \neq 4 \tag{9}$$

$$x_{4jk} \leqslant 1 - x_{2i_1k} \quad \forall\, i_1 \in \text{CEDB} \cup \text{CED},\ i_1 \neq 0,\ j \in J_4,\ k \in K \tag{10}$$

$$C_{2k} + M(2 + y_{kl} - x_{2i_1k} - x_{2i_1l}) \geqslant C_{ml} + P_{2k} \quad \forall\, i_1 \in \text{CEDB},\ i_1 \neq 0,\ k \in K,\ l \in K,\ l > k \tag{11}$$

$$C_{2l} + M(3 - y_{kl} - x_{2i_1k} - x_{2i_1l}) \geqslant C_{mk} + P_{2l} \quad \forall\, i_1 \in \text{CEDB},\ i_1 \neq 0,\ k \in K,\ l \in K,\ l > k \tag{12}$$

$$\sum\nolimits_{i \in I} x_{ii_2k} = 2x_{2i_2k} \quad \forall\, i_2 \in \text{CED},\ k \in K,\ i_2 \neq 0,\ 3 \leqslant i \leqslant m-1,\ i \neq 4 \tag{13}$$

$$C_{2k} + M(2 + y_{kl} - x_{2i_2k} - x_{2i_2l}) \geqslant C_{(m-1)l} + P_{2k} \quad \forall\, i_2 \in \text{CED},\ i_2 \neq 0,\ k \in K,\ l \in K,\ l > k \tag{14}$$

$$C_{2l} + M(3 - y_{kl} - x_{2i_2k} - x_{2i_2l}) \geqslant C_{(m-1)k} + P_{2l} \quad \forall\, i_2 \in \text{CED},\ i_2 \neq 0,\ k \in K,\ l \in K,\ l > k \tag{15}$$

$$\sum_{i\in I} x_{3i_3k} = x_{2i_3k} \quad \forall\, i_3 \in \text{CE},\; k \in K,\; i_3 \neq 0 \tag{16}$$

$$C_{2k} + M(2 + y_{kl} - x_{2i_3k} - x_{2i_3l}) \geqslant C_{3l} + P_{2k} \quad \forall\, i_3 \in \text{CE},\; i_3 \neq 0,\; k \in K,\; l \in K,\; l > k \tag{17}$$

$$C_{2l} + M(3 - y_{kl} - x_{2i_3k} - x_{2i_3l}) \geqslant C_{3k} + P_{2l} \quad \forall\, i_3 \in \text{CE},\; i_3 \neq 0,\; k \in K,\; l \in K,\; l > k \tag{18}$$

$$\sum_{i\in I} x_{5i_4k} = x_{3i_4k} \quad \forall\, i_4 \in \text{ED},\; k \in K,\; i_4 \neq 0 \tag{19}$$

$$C_{3k} + M(2 + y_{kl} - x_{3i_4k} - x_{3i_4l}) \geqslant C_{5l} + P_{3k} \quad \forall\, i_4 \in \text{ED},\; i_4 \neq 0,\; k \in K,\; l \in K,\; l > k \tag{20}$$

$$C_{3l} + M(3 - y_{kl} - x_{3i_4k} - x_{3i_4l}) \geqslant C_{5k} + P_{3l} \quad \forall\, i_4 \in \text{ED},\; i_4 \neq 0,\; k \in K,\; l \in K,\; l > k \tag{21}$$

$$C_{4k} + M(2 + y_{kl} - x_{4ik} - x_{6il}) \geqslant C_{6l} + P_{4k} \quad \forall\, i \in \text{J}_4 \cap J_6,\; i \neq 0,\; k \in K,\; l \in K,\; l > k \tag{22}$$

$$C_{6l} + M(3 - y_{kl} - x_{4ik} - x_{6il}) \geqslant C_{4k} + P_{6l} \quad \forall\, i \in \text{J}_4 \cap J_6,\; i \neq 0,\; k \in K,\; l \in K,\; l > k \tag{23}$$

$$C_{6k} + M(2 + y_{kl} - x_{6ik} - x_{4il}) \geqslant C_{4l} + P_{6k} \quad \forall\, i \in \text{J}_4 \cap J_6,\; i \neq 0,\; k \in K,\; l \in K,\; l > k \tag{24}$$

$$C_{4l} + M(3 - y_{kl} - x_{6ik} - x_{4il}) \geqslant C_{6k} + P_{4l} \quad \forall\, i \in \text{J}_4 \cap J_6,\; i \neq 0,\; k \in K,\; l \in K,\; l > k \tag{25}$$

$$C_{\max} \geqslant 0 \tag{26}$$

$$C_{ik} \geqslant 0 \quad \forall\, i \in I,\; k \in K \tag{27}$$

$$x_{ijk} \in \{0, 1\} \quad \forall\, i \in I,\; j \in J_i,\; k \in K \tag{28}$$

$$y_{kl} \in \{0, 1\} \quad \forall\, k \in K,\; l \in K \tag{29}$$

The model's objective function (1) minimizes makespan. The objective could be changed to minimizing the weighted completion time (WCT) if desired. Constraint sets (2) and (3) ensure that a job starts processing at stage 1 and processes successively on all downstream machines. The overlapping of more than one job on a single machine at a time is prevented by constraint sets (4) and (5). These constraints act as "either-or" constraints, which imply that one of the constraints will be active for a particular value of y_{kl}. When job k precedes job l, then constraint set (4) will be active and constraint set (5) will be inactive, because of the value of "M" and vice versa. The value of M is assigned as the sum of processing time of all jobs in the system. The maximum completion time, C_{max}, should be greater than or equal to the completion time of the last job in the final stage of the processing. This is achieved by including the constraint sets (6). Constraint sets (7) and (8) ensure that if the processing time for a job in any stage is a non-zero, positive number, then one of the machines in that stage must process the job (Fourer, Gay, & Kernighan, 1990). If the processing time for a job at any stage is zero, then that stage is skipped for that particular job.

The remaining sets of constraints are developed for cluster tools and reentrant flow processes. Constraint sets (9)–(12) model the cluster process of coat, expose, develop, and bake in a single machine. Constraints set (10) ensures that a job that needs to be processed on stage 4 bake process will not enter the cluster tools CEDB or CED. Constraint sets (9), (13), (16) and (19), ensure that a job that enters a cluster machine will stay inside that machine until it completes all the processes performed by the cluster tool. Pairs of constraint sets (11), (12), (14), (15), (17), (18), and (20), (21) stop jobs from entering the cluster tool if the machine is already processing some other job. Constraint set (10) updates the assignment variable for job k, for the first bake stage to 0, if job k does not require the baking. Constraint sets (22)–(25) model the re-entry of jobs in the bake process of photolithography. The photolithography process under consideration has re-entry at stages 4 and 6. Hence, the machines of the fourth and sixth stages J_4 and J_6 are referenced in these equations. These constraints guarantee that if any of the jobs is being processed in the bake oven at any of its two stages, *i.e.*, the first bake process or the reentrant second baking stage, then no other job will enter the machine. Finally, constraint sets (26)–(29) are non-negativity constraints, which imply that these variables should have a value greater than or equal to zero.

For calculating the minimum TWT, the objective that will be used is as below.

$$\text{minimize} \sum_{k\in K} w_k T_k \tag{30}$$

Constraint set (6) will be replaced by a new constraint to incorporate the tardiness value. Constraint set (31) is used when the objective function is TWT.

$$C_{mk} - d_k \leqslant T_k \quad \forall\, k \in K \tag{31}$$

5. Experimental study

5.1. Experimental plan

Our experimental plan evaluates three objective functions with the proposed MIP for photolithography scheduling: minimizing makespan (C_{max}), minimizing total weighted completion time (WCT), and minimizing total weighted tardiness (TWT). The experimental design used in the random problem generation (Mehta & Uzsoy, 1998) is given in Table 2. These parameter values were vetted with the industrial partner to be an appropriate set of factors for experimentation. Three different levels of the number of jobs to be scheduled are investigated in this problem: 5, 10, and 25. Each set of jobs tested with two levels of ready times. For the first condition, all jobs have zero ready time and for the second condition, some portion of the jobs have a ready time that is a non-zero, randomly generated value while the remaining jobs of the same sets have zero ready time.

The due date value is generated using a discrete uniform distribution (Table 2). The calculation of the estimated makespan includes the total number of jobs, the processing time of the photolithography process's bottleneck stage, the total number of machines that process the bottleneck stage, and the sum of the processing times for all non-bottleneck stages in photolithography. The parameter T is the expected percentage of tardy jobs and two cases for the value of T are considered in this experimentation: 0.3 and 0.6. Further, R is a range parameter that is studied at two levels: 0.5 and 2.5 (Mehta & Uzsoy, 1998).

The weights (w_k) are calculated based on a random distribution of all integers between 1 and 5, 1 being a low priority job and 5 being a high priority job. Considering the three levels for the number of jobs, two scenarios for job ready times, and four combinations of due date parameters T and R, 24 unique combinations of data are run for two different resource (equipment) levels (Table 3). The equipment settings in scenario 1 are based on representative data obtained from an actual wafer fab that partnered with the authors to conduct this study. In order to examine potential performance differences, scenario 2 was created by reducing equipment counts from scenario 1. As three different objective functions are investigated, a total of 24(2)(3) = 144 unique scenarios exist for investigation. Based on 10 replications for each unique scenario, a total of 1440 files are generated for analysis and comparison. Microsoft Excel 2010 is used to generate random numbers for the various cases in the experimental plan.

5.2. Model execution

The random data that was created is used to test the mathematical model. After implementing the model in AMPL (Fourer, Gay, &

Kernighan, 1993), 1338 test files were run and the objective functions are validated for proper machine assignments. The solution was produced using Gurobi 5.1.0 solver (Gurobi Optimization, 2016) within a 7200 CPU seconds time limit on a Windows 7 platform with Intel® Core™ 2 Quad CPU Q6600 @2.40 GHz with 16 GB of RAM. Although Gurobi 5.1.0 did not converge to the optimal solution within the allowed 7200 s for most of the instances, Gurobi 5.1.0 was able to obtain a good solution quickly. We explain the model size using the number of constraints and variables for each of the problems. Table 4 provides the number of constraints and variables introduced into the model based on the number of jobs.

Since the scheduling problem under study is strongly NP-Hard, heuristic and/or metaheuristic approaches may provide good, near optimal solutions that are better than the solution obtained by a time-limited MIP (Urlings, 2010). The heuristic approach for the problem under study is presented, and then its performance is compared with that of the proposed MIP model under a time limit restriction.

6. Heuristic algorithm

A review of the available literature confirmed that no heuristic is currently available for analyzing the flexible flowshop scheduling problem with cluster tools and job ready times. Therefore, in order to very quickly obtain good solutions to the research problem under study, we created our own GA-based heuristic (Holland, 1992) and a constructive heuristic.

6.1. Constructive heuristics

To this end, we generated a constructive heuristic (pseudocode 1) that takes in a topological ordering (permutation) of jobs as input and returns either C_{max}, TWT, or WCT using the pseudocode 2. The pseudocodes 1 and 2 illustrates this constructive heuristic.

Algorithm 1. Procedure SchedulePhoto (SP)

```
Initialization;
Let $[O]$ denote the $o^{th}$ ordered job in a list and $n = |O|$, the cardinality of the set O;
Let $S_{itrcnt}$ denote the current schedule and $S_b$ denote the best schedule found so far;
Let $G(S_{itrcnt})$ denote the corresponding values of the objective function;
Let $P_{i,jcnt}$ denote the processing time of job $jcnt$ at stage $I$ and $maxitrcnt$ is the maximum number of iterations required;
Procedure SchedulePhoto (SP);
Sort jobs in ascending order of ready times $(r_k)$. Break any ties by sorting those jobs in ascending order of $\frac{d_k}{w_k}$. Finally, break any remaining ties by sorting the jobs in the descending order of total count of cluster tools that the job could be processed on.;
Define $X = \{k \in K | r_k = 0\}$;
Define $Y = \{k \in K | 0 < r_k <$ average ready time$\}$;
Define $Z = \{k \in K | r_k \geq$ average ready time$\}$;
for $itrcnt = 1...maxitrcnt$ do
    if $itrcnt = 1$ then
        Call Procedure CreateSchedule (CS) using $[O]$ and save the schedule in $S_{itrcnt}$;
        $S_b = S_{itrcnt}$;
    end
    else
        if $itrcnt \leq maxitrcnt/2$ then
            Use a hill-climbing approach in which we randomly select two elements and swap them;
            Update the order in $[O]$ based on swapping of elements;
        end
        else
            Generate a random number $U[0,1]$ for each element in $X, Y$ and $Z$ and sort each set in the ascending order of the random number;
            Update the order in $[O]$ based on order in $X, Y$ and $Z$;
        end
        Call Procedure CreateSchedule (CS) using $[O]$ and save the schedule in $S_{itrcnt}$.
    end
    If $G(S_{itrcnt}) < G(S_b)$, then $S_b = S_{itrcnt}$;
end
Result: Write the output results
```

Algorithm 2. Procedure CreateSchedule (CS)

```
Procedure CreateSchedule (CS);
for $jcnt = [1]...[n]$ do
    for $i = 1...6$ do
        if $P_{i,jcnt} > 0$ then
            Define set $J_{iavbl}$ = List of available machines at stage $i$ that process job $jcnt$;
            Select a machine $m$, with the largest number of cluster tools in it, from $J_{iavbl}$;
            Update $i$ with the number of stages processed by machine $m$;
            Update completion time $C_{i,jcnt}$ as ($P_{i,jcnt}$ + max [ready time, completion time of previous job on
            that machine]);
        end
    end
end
Result: Update $S_{itrcnt}$ with the current schedule
```

6.2. Genetic algorithm

Genetic algorithms use ideas borrowed from the concepts of genetics and biological evolution. The base version of the algorithm treats solutions as genomes that are iteratively combined in pairs (binary crossover) and mutated (binary mutation) to produce offsprings in every generation. This process is repeated until a specific fitness function achieves a desired threshold criteria. In our Algorithm 3, the fitness function computes the value of C_{max}, TWT, or WCT, depending on what is the desired objective function to be minimized. Further, we develop approaches to perform crossover and mutation functions that enable us to efficiently solve the flexible flowshop scheduling problem. As Algorithm 3 outlines the pseudocode for GA, we also provide a description of crossover and mutation functions in the following subsections. Our algorithm was implemented in MATLAB 8.1.0.604 (R2016a) and uses the same data as input that was generated for the mathematical model. With regards to the threshold criteria, we set the algorithm to terminate after MaxGenerations = 500 generations or if the average relative change in the best fitness function value in 50 continuous generations is less than or equal to 10^{-6}. Finally, we also set the size of the population in each generation (popsize) to 100.

6.2.1. Crossover function

In genetic algorithms, crossover functions modify the makeup of genomes from one generation to the next by taking more than one parent genome and producing a child genome. In our GA presented in Algorithm 3, we perform the crossover function in the following manner.

Let us assume two parent genomes are coded as u and v which are a vector of integers representing permutations of $\{1, 2, 3, ..., n\}$. Define u^i as the i^{th} element of u and generate a random integer r from the set of integers $\{1, 2, 3, ..., n\}$. Now, crossover is performed by swapping the

Table 2
Experimental design.

Experimental factor	Settings		
Number of jobs, n	5	15	20
Ready time	$r_k = 0$ for all k	30% of jobs, $r_k = 0$ 70% of jobs, r_k = RANDOM[1, 2/3 × C_{max}]	
Job due date d_k	d_k = Uniform $[\mu(1–0.5R), \mu(1 + 0.5R)]$ with $\mu = C_{max} \times (1-T)$ $C_{max} = 1.5 \times \left(n \times \frac{P_{BN}}{m_{iBN}} + P_{NBN}\right)$ P_{BN} = Processing time of the bottleneck stage m_{iBN} = Number of machines that processes the bottleneck stage P_{NBN} = Sum of the Processing time of all other non-bottleneck stages T = 0.3 and 0.6 R = 0.5 and 2.5		
Processing time	Stage 1	80% of jobs $P_{1j} = 40$ 20% of jobs $P_{1j} = 0$	
	Stage 2	100% of jobs $P_{2j} = 20$	
	Stage 3	100% of jobs $P_{3j} = 75$	
	Stage 4	20% of jobs $P_{4j} = 45$ 80% of jobs $P_{4j} = 0$	
	Stage 5	100% of jobs $P_{5j} = 30$	
	Stage 6	50% of jobs $P_{6j} = 45$ 50% of jobs $P_{6j} = 0$	

Table 3
Equipment settings.

Machine Type	Scenario 1	Scenario 2
Sink	4	2
Coat	2	1
Expose	4	2
Develop	2	1
Bake	3	2
CE	2	1
CED	2	1
CEDB	2	1
ED	1	1

Table 4
Model size.

Problem size	Constraints	Continuous variables	Binary variables	Total variables
5	1185	36	215	251
15	11205	106	795	901
25	31425	176	1575	1751

positions of elements u^r and v^r in both u and v. For instance, let $u = \{2, 1, 4, 3, 5\}$ and $v = \{5, 2, 3, 4, 1\}$. Let random integer $r = 2$. Therefore, we swap the positions of the second elements of u and v respectively, i.e., elements 1 and 2 in both $u = \{2, 1, 4, 3, 5\}$ and $v = \{5, 2, 3, 4, 1\}$. The newly formed children are now $\{1, 2, 4, 3, 5\}$ and $\{5, 1, 3, 4, 2\}$. One of these children is selected at random and added to the newly formed pool of children to be mutated in the next step.

6.2.2. Mutation function

During mutation in GA, one or more gene values are altered slightly so as to ensure that genetic variation is preserved. In Algorithm 3, each of the children formed after the crossover step is modified slightly by taking two random elements of the permutation and switching their positions. For instance, if one of the children formed from the crossover step is $\{4, 2, 1, 3, 5\}$, two elements are picked at random (say, 3 and 4) and their positions are swapped, to produce a new offspring with a slightly altered genetic makeup: $\{3, 2, 1, 4, 5\}$.

6.2.3. Fitness function

The proposed heuristic solution outlined in Algorithm 3 involves computing a fitness function. Procedure CS in Algorithm 3 (pseudocode 2) computes the fitness function depending on the objective function such as C_{max} (Pugazhenthi & Anthony Xavior, 2014), TWT (Liu, Abdelrahman, & Ramaswamy, 2003), or WCT (Wu, Hsu, Chen, & Wang, 2011), that we wish to minimize.

Algorithm 3. Genetic Algorithm-based Heuristic

```
Initialization;
$t \leftarrow 0$;
Define $S$ as the set containing all permutations of $(1, 2, 3, ..., n)$;
Define $P(t) \subset S$ as the initial population of permutations;
Define $F$ as a fitness function to evaluate the fitness of each individual $p \in P(t)$;
Define $arrF$ as an array to store the values of the fitness functions;
while $t < Max\ Generations$ do
    Build parent population $P_p(t)$ based on selection criterion;
    for $t = 1$ to $popSize$ do
        Randomly select two elements $(u, v) \in P_p(t)$;
        Perform crossover function $C$ to obtain offspring. $u_o = C(u, v)$;
        Apply Mutation function $M$ on offspring;
        if $F(u_o) > F_{best}$ then
            $F_{best} \leftarrow F(u_o)$;
            $S_{best} \leftarrow u_o$;
        end
        Add $u_o$ to new generation's population;
    end
    $t \leftarrow t + 1$;
    if $Average\ change\ in\ last\ 50\ elements\ in\ arrF < \epsilon$ then
        break;
    end
end
Result: Write the output results
```

7. Results

7.1. Solution time performance

We compare the algorithms' efficiency using the solving time. In order to perform a fair comparison, we classify the results in terms of the MIP's optimality status. If the MIP obtained an optimal solution, we compare the corresponding instances solution time for Algorithms 1 and 3 and is presented in Fig. 2. For instances with small jobs, our MIP model was working better in terms of solution time. For larger instances genetic algorithm (3) is better. As the size of the instances increase, the solution time also increased. The number of instances that MIP solved to optimality with 25 jobs are low when compared to the 5 job instances. The real difference in algorithms' performances can be found when we compare the results for the timed-out MIP instances. Fig. 3 compares the solution time for the two heuristics whereas the MIP was timed out after 7200 s. Our GA-based algorithm, consistently found better results with in certain time limit.

7.2. Experimental results

The performance of GA and the proposed mathematical model is compared by computing a performance ratio. Let performance ratio PR be defined as the ratio of the objective function value obtained by GA (OF_{GA}) for a problem instance to the optimal objective function value produced by the mathematical model (OF_{opt}) for the same problem instance (Erramilli & Mason, 2006). While the PR ratio can be computed for any objective function case of interest, it is only valid when the MIP

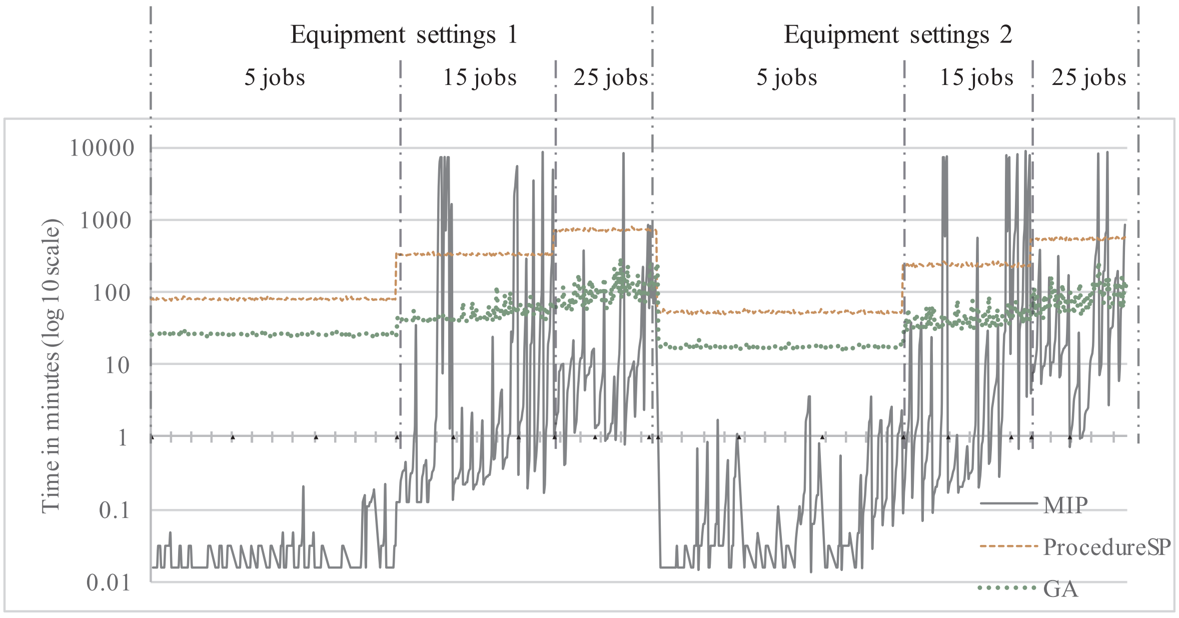


Fig. 2. Solution time comparison for optimally completed MIP instances.

model produces an optimal solution. In this way, an estimate of the quality of the GA solution is obtained in terms of its percent above the optimal solution value.

Once the results are obtained for each instance, the PR values can be averaged across all experimental instances for a given type of problem type (e.g., all instances with five jobs). The set of like problem instances is characterized in terms of (n, r_k, T, R, mc). In this expression, n is the number of jobs and $r_k = 0$ denotes all 0 job ready times while $r_k = 1$ denotes the presence of non-zero ready times. Further, T and R are the due date-related parameters described above and mc represents the machine configuration (scenario 1 or scenario 2) from Table 3. An example for this instance characterization approach is (15, 0, 0.3, 2.5, 1), which represents the average PR values for problems with 15 jobs that have zero job ready times, T and R values of 0.3 and 2.5, respectively, and machine configuration of scenario 1 from Table 3.

Table 5 presents a summary of the average performance ratios for every experimental factor of interest. This summary is segmented according to the number of jobs (n) and objective function under study. A "*" for an experimental factor denotes that all instances at all possible levels were combined. For example, Table 5 row labeled (5, 0, *, *, *) contains the average performance ratio of all instances with n = 5 jobs and no ready times (i.e., $r_k = 0$), while the (25, *, *, 2.5, *) rows contains the average performance ratio for all 25 job instances with due date range factor R = 2.5. For each objective function, the number given in parentheses denotes the number of optimal solutions that were found across the 80 instances analyzed. Finally, a Table 5 entry of "N/A (0)" denotes the case wherein no optimal solutions were found; therefore, no average performance ratio can be computed.

Next performance measure is the heuristic ratio (HR) metric, which is defined as the ratio of the objective function value obtained by GA

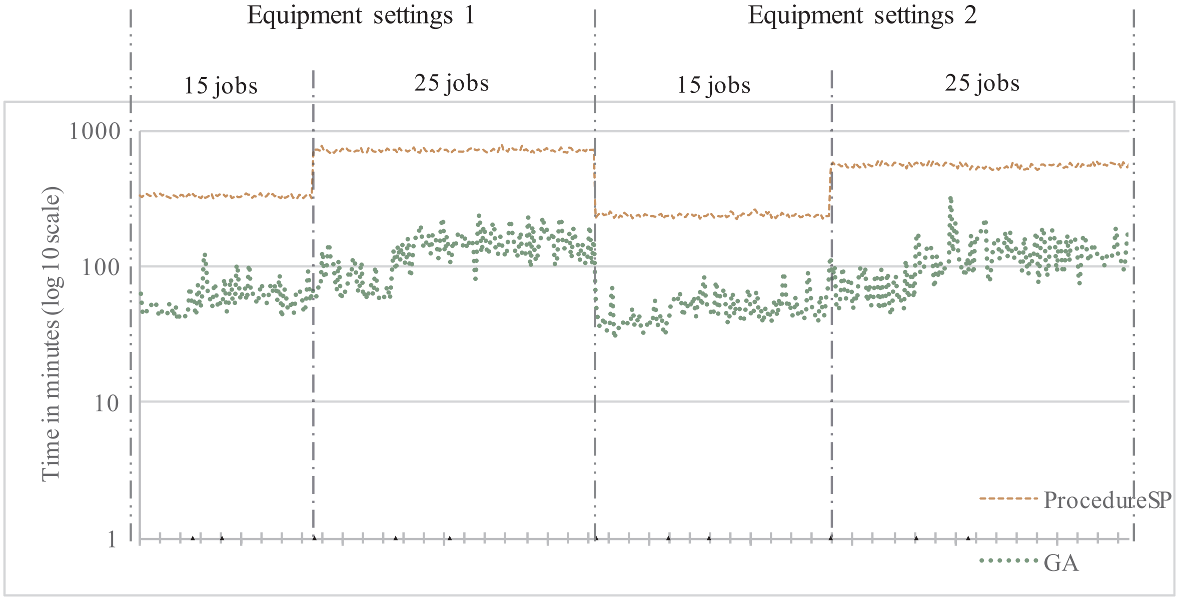


Fig. 3. Solution time comparison for time-limited MIP instances.

Table 5
Performance ratio.

	Algorithm 3			Algorithm 1		
n, r_k, T, R, mc	C_{max}	WCT	TWT	C_{max}	WCT	TWT
(5,0,*,*,*)	1.02 (80)	1.01 (80)	1.02 (80)	1.02 (80)	1.01 (80)	1.07 (80)
(5,1,*,*,*)	1.01 (80)	1.01 (80)	1.01 (80)	1.00 (80)	1.01 (80)	1.01 (80)
(5,*,0.3,*,*)	1.01 (80)	1.01 (80)	1.01 (80)	1.01 (80)	1.01 (80)	1.07 (80)
(5,*,0.6,*,*)	1.01 (80)	1.01 (80)	1.02 (80)	1.01 (80)	1.01 (80)	1.01 (80)
(5,*,*,0.5,*)	1.01 (80)	1.01 (80)	1.01 (80)	1.01 (80)	1.01 (80)	1.06 (80)
(5,*,*,2.5,*)	1.01 (80)	1.01 (80)	1.02 (80)	1.01 (80)	1.01 (80)	1.01 (80)
(5,*,*,*,1)	1.01 (80)	1.01 (80)	1.01 (80)	1.01 (80)	1.01 (80)	1.00 (80)
(5,*,*,*,2)	1.02 (80)	1.01 (80)	1.03 (80)	1.02 (80)	1.01 (80)	1.06 (80)
(15,0,*,*,*)	1.16 (20)	N/A	1.08 (45)	1.15 (20)	N/A	1.10 (45)
(15,1,*,*,*)	1.02 (79)	1.02 (55)	1.02 (80)	1.01 (79)	1.01 (55)	1.10 (80)
(15,*,0.3,*,*)	1.06 (50)	1.01 (24)	1.03 (76)	1.04 (50)	1.01 (24)	1.11 (76)
(15,*,0.6,*,*)	1.04 (49)	1.02 (31)	1.05 (49)	1.03 (49)	1.01 (31)	1.08 (49)
(15,*,*,0.5,*)	1.05 (51)	1.02 (28)	1.03 (59)	1.05 (51)	1.01 (28)	1.13 (59)
(15,*,*,2.5,*)	1.04 (48)	1.02 (27)	1.04 (66)	1.03 (48)	1.01 (27)	1.08 (66)
(15,*,*,*,1)	1.04 (55)	1.01 (35)	1.02 (64)	1.04 (55)	1.01 (35)	1.02 (64)
(15,*,*,*,2)	1.05 (44)	1.03 (20)	1.06 (61)	1.04 (44)	1.02 (20)	1.18 (61)
(25,0,*,*,*)	N/A	N/A	1.13 (31)	N/A	N/A	1.28 (31)
(25,1,*,*,*)	1.08 (78)	1.02 (8)	1.06 (77)	1.03 (78)	1.01 (8)	1.18 (77)
(25,*,0.3,*,*)	1.08 (39)	1.02 (4)	1.05 (71)	1.02 (39)	1.01 (4)	1.20 (71)
(25,*,0.6,*,*)	1.08 (39)	1.02 (4)	1.09 (37)	1.03 (39)	1.01 (4)	1.19 (37)
(25,*,*,0.5,*)	1.09 (39)	1.01 (3)	1.07 (57)	1.03 (39)	1.01 (3)	1.17 (57)
(25,*,*,2.5,*)	1.08 (39)	1.02 (5)	1.06 (51)	1.03 (39)	1.01 (5)	1.21 (51)
(25,*,*,*,1)	1.04 (40)	1.02 (8)	1.05 (53)	1.01 (40)	1.01 (8)	1.08 (53)
(25,*,*,*,2)	1.13 (38)	N/A	1.09 (55)	1.04 (38)	N/A	1.31 (55)

Table 6
Heuristic ratio.

	Algorithm 3			Algorithm 1		
n, r_k, T, R, mc	C_{max}	WCT	TWT	C_{max}	WCT	TWT
(15,0,*,*,*)	1.09 (60)	1.03 (80)	1.15 (34)	1.08 (60)	1.05 (80)	1.26 (34)
(15,1,*,*,*)	1.01 (1)	1.04 (25)	N/A	1.01 (1)	1.05 (25)	N/A
(15,*,0.3,*,*)	1.09 (30)	1.03 (56)	1.10 (3)	1.09 (30)	1.05 (56)	1.28 (3)
(15,*,0.6,*,*)	1.08 (31)	1.03 (49)	1.15 (31)	1.08 (31)	1.06 (49)	1.25 (31)
(15,*,*,0.5,*)	1.08 (29)	1.03 (52)	1.15 (20)	1.08 (29)	1.05 (52)	1.23 (20)
(15,*,*,2.5,*)	1.09 (32)	1.03 (53)	1.14 (14)	1.09 (32)	1.06 (53)	1.29 (14)
(15,*,*,*,1)	1.10 (25)	1.02 (45)	1.10 (15)	1.09 (25)	1.04 (45)	1.13 (15)
(15,*,*,*,2)	1.08 (36)	1.04 (60)	1.18 (19)	1.08 (36)	1.06 (60)	1.35 (19)
(25,0,*,*,*)	1.16 (80)	1.02 (80)	1.16 (48)	1.17 (80)	1.10 (80)	1.36 (48)
(25,1,*,*,*)	1.21 (2)	1.05 (72)	1.37 (3)	1.16 (2)	1.07 (72)	1.52 (3)
(25,*,0.3,*,*)	1.15 (41)	1.04 (76)	1.19 (8)	1.17 (41)	1.08 (76)	1.27 (8)
(25,*,0.6,*,*)	1.16 (41)	1.03 (76)	1.17 (43)	1.17 (41)	1.08 (76)	1.38 (43)
(25,*,*,0.5,*)	1.16 (41)	1.03 (77)	1.14 (22)	1.17 (41)	1.07 (77)	1.31 (22)
(25,*,*,2.5,*)	1.15 (41)	1.04 (75)	1.20 (29)	1.17 (41)	1.09 (75)	1.41 (29)
(25,*,*,*,1)	1.19 (40)	1.03 (72)	1.14 (26)	1.19 (40)	1.07 (72)	1.27 (26)
(25,*,*,*,2)	1.13 (42)	1.03 (80)	1.20 (25)	1.15 (42)	1.10 (80)	1.46 (25)

(OF_{GA}) for a problem instance to the non-optimal objective function value produced by the mathematical model in 7200 s (OF_{7200}) for the same problem instance. The average heuristic ratio for each experimental factor level by objective function is shown in Table 6. A Table 5 entry of "N/A (0)" denotes the case wherein optimal solutions were found for all instances; therefore, no average heuristic ratio can be computed. The GA produced results that are 20% above the time-limited MIP model solution for the makespan objective function. However, the 20% above optimal performance is obtained in less than 5 min. One other observation that was obtained from Table 6 is that when the problem instance is small, the mathematical problem solved the instances to optimality for all scenarios and hence an HR is not available for those instances that had five jobs.

7.3. Analysis

From the results it is clear that the sets of instances, that have ready times, performed better than those sets of instances whose ready times are zero. This pattern is consistent irrespective of the number of jobs or the type objective function. The performance of the heuristic is also compared based on the two equipment scenarios mentioned in Table 3. As the number of tools are reduced, the performance ratios is seen to be increased which is as expected. As the number of tools is reduced, the tightness of the resources increases which results in an increased value for average performance ratios. The parameters T and R have negligible impact on the average performance.

The results for the average heuristic ratio for set of instances with ready times and without ready times follow similar pattern as the average performance ratio. Similar to the performance ratio, heuristic ratio also seems to perform better for weighted completion time (WCT) and makespan (C_{max}) followed by the performance for total weighted tardiness (TWT). The GA fared better than Algorithm 1 for the objective function WCT, TWT and most instances of C_{max}.

8. Conclusion and future work

A mixed-integer programming (MIP) formulation for the photolithography process with individual and cluster tool was developed for improved job scheduling. Due to this problem's complexity, a heuristic was developed to analyze our experimental cases. When comparing the solution approaches, the MIP model provides better results but took a considerable amount of time. The heuristic approach achieved some good results in a very short span of time. The two heuristics' performance comparison showed the GA-based algorithm is efficient to schedule a photolithography process that involves larger number of jobs. The heuristic method could be employed to scenarios when there is an unexpected machine downtime, shift changes, and changes in due dates. The heuristic seems to perform well for the objective function WCT and C_{max}. A better heuristic for TWT can be developed that can produce better results when compared to the current GA algorithm. Future work includes developing improved heuristic solutions to obtain better results by considering the job availability at every instant. The research can be extended to deal with finding a solution for minimizing other objectives like minimizing the total number of tardy jobs, minimizing maximum lateness, or to extend the research to investigate solutions for multiple objective problems.

References

Arisha, A., & Young, P. (2004). Intelligent simulation-based lot scheduling of photolithography toolsets in a wafer fabrication facility. *Proceedings of the 2004 Winter Simulation Conference, 2004: Vol. 2*, (pp. 1935–1942). . http://dx.doi.org/10.1109/WSC.2004.1371552.

Cardarelli, E., & Pelagagge, P. M. (1995). Simulation tool for design and management optimization of automated interbay material handling and storage systems for large wafer fab. *IEEE Transactions on Semiconductor Manufacturing, 8*(1), 44–49. http://dx.doi.org/10.1109/66.350756.

Cheng, J., Karuno, Y., & Kise, H. (2001). A shifting bottleneck approach for a parallel-machine flowshop scheduling problem. *Journal of the Operations Research Society of Japan, 44*(2), 140–156. http://dx.doi.org/10.15807/jorsj.44.140.

Chiang, T.-C. (2013). Enhancing rule-based scheduling in wafer fabrication facilities by evolutionary algorithms: Review and opportunity. *Computers & Industrial Engineering, 64*(1), 524–535. http://dx.doi.org/10.1016/j.cie.2012.08.009.

Conway, R. W., Maxwell, W. L., & Miller, L. W. (2012). *Theory of scheduling.* Courier Corporation.

Erramilli, V., & Mason, S. J. (2006). Multiple orders per job compatible batch scheduling. *IEEE Transactions on Electronics Packaging Manufacturing, 29*(4), 285–296. http://dx.doi.org/10.1109/TEPM.2006.887355.

Floudas, C. A., & Lin, X. (2005). Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Annals of Operations Research, 139*(1), 131–162. http://dx.doi.org/10.1007/s10479-005-3446-x.

Fourer, R., Gay, D. M., & Kernighan, B. W. (1990). A modeling language for mathematical programming. *Management Science, 36*(5), 519–554. http://dx.doi.org/10.1287/mnsc.36.5.519.

Fourer, R., Gay, D. M., & Kernighan, B. (1993). *Ampl, Vol. 117.* MA: Boyd & Fraser Danvers.

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. http://dx.doi.org/10.1016/S0167-5060(08)70356-X.

Graves, S. C., Meal, H. C., Stefek, D., & Zeghmi, A. H. (1983). Scheduling of re-entrant flow shops. *Journal of Operations Management, 3*(4), 197–207. http://dx.doi.org/10.1016/0272-6963(83)90004-9.

Gurobi Optimization, I. (2016). Gurobi optimizer reference manual. < http://www.gurobi.com > .

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biologyControl and artificial intelligence*Cambridge, MA, USA: MIT Press.

Hoogeveen, J. A., Lenstra, J. K., & Veltman, B. (1996). Preemptive scheduling in a two-stage multiprocessor flow shop is np-hard. *European Journal of Operational Research, 89*(1), 172–175. http://dx.doi.org/10.1016/S0377-2217(96)90070-3.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, & Werner, F. (2007). Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *The International Journal of Advanced Manufacturing Technology, 37*(3), 354–370. http://dx.doi.org/10.1007/s00170-007-0977-0.

Kock, A., Veeger, C., Etman, L., Lemmen, B., & Rooda, J. (2007). Cycle time and throughput performance analysis of a litho cell using an aggregate modeling approach. *Proc. adv. semi. manuf. conf.(ASMC)* (pp. 65–70). .

Kyparisis, G. J., & Koulamas, C. (2001). A note on weighted completion time minimization in a flexible flow shop. *Operations Research Letters, 29*(1), 5–11. http://dx.doi.org/10.1016/S0167-6377(01)00072-4.

Lee, T. E. (2008). A review of scheduling theory and methods for semiconductor manufacturing cluster tools. *2008 Winter simulation conference* (pp. 2127–2135). . http://dx.doi.org/10.1109/WSC.2008.4736310.

Liu, N., Abdelrahman, M. A., & Ramaswamy, S. (2003). A genetic algorithm for the single machine total weighted tardiness problem. *Proceedings of the 35th southeastern symposium on system theory, 2003* (pp. 34–38). . http://dx.doi.org/10.1109/SSST.2003.1194525.

Lowe, J. J., & Mason, S. J. (2016). Integrated semiconductor supply chain production planning. *IEEE Transactions on Semiconductor Manufacturing, 29*(2), 116–126. http://dx.doi.org/10.1109/TSM.2016.2544202.

McGuigan, T. C. (1992). Modeling the lot selection process in semiconductor photolithography processing. *Proceedings of the 24th conference on winter simulation, WSC '92* (pp. 885–889). New York, NY, USA: ACM. http://dx.doi.org/10.1145/167293.167763.

Mehta, S. V., & Uzsoy, R. (1998). Minimizing total tardiness on a batch processing machine with incompatible job families. *IIE Transactions, 30*(2), 165–178. http://dx.doi.org/10.1023/A:1007466101115.

Mönch, L., Fowler, J. W., Dauzère-Pérès, S., Mason, S. J., & Rose, O. (2011). A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *Journal of Scheduling, 14*(6), 583–599. http://dx.doi.org/10.1007/s10951-010-0222-9.

Montazeri, M., & Van Wassenhove, L. (1990). Analysis of scheduling rules for an FMS. *The International Journal of Production Research, 28*(4), 785–802. http://dx.doi.org/10.1080/00207549008942754.

Oduguwa, V., Tiwari, A., & Roy, R. (2005). Evolutionary computing in manufacturing industry: An overview of recent applications. *Applied Soft Computing, 5*(3), 281–299. http://dx.doi.org/10.1016/j.asoc.2004.08.003.

Pan, C., Zhou, M., Qiao, Y., & Wu, N. (2018). Scheduling cluster tools in semiconductor manufacturing: Recent advances and challenges. *IEEE Transactions on Automation Science and Engineering, 15*(2), 586–601. http://dx.doi.org/10.1109/TASE.2016.2642997.

Park, J. Y., Park, K., & Morrison, J. R. (2017). Models of clustered photolithography tools for fab-level simulation: From affine to flow line. *IEEE Transactions on Semiconductor Manufacturing, 30*(4), 547–558. http://dx.doi.org/10.1109/TSM.2017.2752755.

Pinedo, M. L. (1995). Scheduling: Theory, algorithms and systems. http://dx.doi.org/10.1007/978-1-4614-2361-4.

Pugazhenthi, R., & Anthony Xavior, M. (2014). A genetic algorithm applied heuristic to minimize the makespan in a flow shop. *Procedia Engineering, 97*, 1735–1744. http://dx.doi.org/10.1016/j.proeng.2014.12.325.

Ramachandra, G., & Elmaghraby, S. E. (2006). Sequencing precedence-related jobs on two machines to minimize the weighted completion time. *International Journal of Production Economics, 100*(1), 44–58. http://dx.doi.org/10.1016/j.ijpe.2004.10.014.

Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research, 169*(3), 781–800. http://dx.doi.org/10.1016/j.ejor.2004.06.038.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research, 205*(1), 1–18. http://dx.doi.org/10.1016/j.ejor.2009.09.024.

Sarin, S. C., Varadarajan, A., & Wang, L. (2011). A survey of dispatching rules for operational control in wafer fabrication. *Production Planning and Control, 22*(1), 4–24. http://dx.doi.org/10.1080/09537287.2010.490014.

Sawik, T. (2011). *Scheduling in supply chains using mixed integer programming.* John Wiley & Sons.

Sawik, T. (2012). Batch versus cyclic scheduling of flexible flow shops by mixed-integer programming. *International Journal of Production Research, 50*(18), 5017–5034. http://dx.doi.org/10.1080/00207543.2011.627388.

Sawik, T. (2014). A mixed integer program for cyclic scheduling of flexible flow lines. *Bulletin of the Polish Academy of Sciences: Technical Sciences, 62*(1), 121–128. http://dx.doi.org/10.2478/bpasts-2014-0014.

Sha, D., Hsu, S., Che, Z., & Chen, C. (2006). A dispatching rule for photolithography scheduling with an on-line rework strategy. *Computers & Industrial Engineering, 50*(3), 233–247. http://dx.doi.org/10.1016/j.cie.2006.04.002.

Urlings, T. (2010). *Heuristics and metaheuristics for heavily constrained hybrid flowshop problems* (Ph.D. thesis)Universitat Politècnica de València.

Uzsoy, R., Lee, C.-Y., & Martin-Vega, L. A. (1992). A review of production planning and scheduling models in the semiconductor industry part I: System characteristics, performance evaluation and production planning. *IIE Transactions, 24*(4), 47–60. http://dx.doi.org/10.1080/07408179208964233.

Wu, C.-C., Hsu, P.-H., Chen, J.-C., & Wang, N.-S. (2011). Genetic algorithm for minimizing the total weighted completion time scheduling problem with learning and release times. *Computers & Operations Research, 38*(7), 1025–1034. http://dx.doi.org/10.1016/j.cor.2010.11.001.

Yim, S. J., & Lee, D. Y. (1999). Scheduling cluster tools in wafer fabrication using candidate list and simulated annealing. *Journal of Intelligent Manufacturing, 10*(6), 531–540. http://dx.doi.org/10.1023/A:1008904604531.

Zhang, P., Lv, Y., & Zhang, J. (2018). An improved imperialist competitive algorithm based photolithography machines scheduling. *International Journal of Production Research, 56*(3), 1017–1029. http://dx.doi.org/10.1080/00207543.2017.1346320.

Zhou, B.-h., Gao, Z.-s., & Chen, J. (2014). Scheduling algorithm of dual-armed cluster tools with residency time and reentrant constraints. *Journal of Central South University, 21*(1), 160–166. http://dx.doi.org/10.1007/s11771-014-1927-2.

Sreenath Chalil Madathil completed his Ph.D. from Department of Industrial Engineering at Clemson University, Clemson SC. He completed his MS in Industrial Engineering from Clemson University and B.Tech from Mahatma Gandhi University, Kottayam.

Scott J. Mason is the Fluor Endowed Chair in Supply Chain Optimization and Logistics and a Professor of Industrial Engineering at Clemson University. Prior to joining Clemson, Dr. Mason spent 10 years in the Department of Industrial Engineering at the University of Arkansas. He received his Ph. D. in Industrial Engineering from Arizona State University after earning Bachelor's and Master's degrees from The University of Texas at Austin. He is a senior member of the Institute for Industrial Engineers and a member of INFORMS.

Mary E. Kurz is an Associate Professor of Industrial Engineering at Clemson University. She received her Ph. D. in Systems and Industrial Engineering from University of Arizona as well as her Master's and Bachelor's in Systems engineering. She is a senior member of the Institute for Industrial Engineers and a member of INFORMS.