3-2022

Hybrid tabu search algorithm for unrelated parallel machine scheduling in semiconductor fabs with setup times, job release, and expired times

Changyu CHEN
Singapore Management University, cychen.2020@phdcs.smu.edu.sg

Madhi FATHI
University of North Texas

Marzieh KHAKIFIROOZ
Tecnologico de Monterrey

Kan WU
Chang Gung University

Citation

CHEN, Changyu; FATHI, Madhi; KHAKIFIROOZ, Marzieh; and WU, Kan. Hybrid tabu search algorithm for unrelated parallel machine scheduling in semiconductor fabs with setup times, job release, and expired times. (2022). *Computers and Industrial Engineering*. 165, 1-11.
Available at: https://ink.library.smu.edu.sg/sis_research/6956

Hybrid tabu search algorithm for unrelated parallel machine scheduling in semiconductor fabs with setup times, job release, and expired times

Changyu Chen[a], Mahdi Fathi[b], Marzieh Khakifirooz[c], Kan Wu[d,*]

a Singapore Management University, Singapore
b Department of Information Technology and Decision Sciences, G. Brint Ryan College of Business, University of North Texas, Denton, TX, USA
c Department of Industrial Engineering, Tecnologico de Monterrey, Monterrey, NL, Mexico
d Business Analytics Research Center, Chang Gung University, Taoyuan, Taiwan
* Corresponding author. E-mail addresses: chen1297@e.ntu.edu.sg (C. Chen), mahdi.fathi@unt.edu (M. Fathi), mkhakifirooz@tec.mx (M. Khakifirooz), kan@mail.cgu.edu.tw (K. Wu).

Published in Computers & Industrial Engineering, March 2022, 165, 107915. DOI: 10.1016/j.cie.2021.107915

Abstract: This research is motivated by a scheduling problem arising in the ion implantation process of wafer fabrication. The ion implementation scheduling problem is modeled as an unrelated parallel machine scheduling (UPMS) problem with sequence-dependent setup times that are subject to job release time and expiration time of allowing a job to be processed on a specific machine, defined as: $R|r_j, e_{ij}, ST_{sd}|C_{max}$. The objective is first to maximize the number of processed jobs, then minimize the maximum completion time (makespan), and finally minimize the maximum completion times of the non-bottleneck machines. A mixed-integer programming (MIP) model is proposed as a solution approach and adopts a hybrid tabu search (TS) algorithm to acquire approximate feasible solutions. The MIP model has two phases and attempts to achieve the first two objectives. The hybrid TS algorithm has three phases and attempts to achieve all three objectives. In a real setting, computational results demonstrate that the maximum number of processed jobs can be acquired within a short time utilizing the hybrid TS algorithm (average 8 s). By comparing the two approaches, the TS outperforms the MIP model regarding solution quality and computational time for the second objective, minimizing the makespan. Furthermore, the third phase of the hybrid TS algorithm shows the effectiveness further to enhance the utilization of the ion implantation equipment.

Keywords: Ride-hailing surge pricing, Taxi demand, Cross-price elasticity of taxi bookings, Taxi demand prediction

1. Introduction

The semiconductor industry is one of the world's most capital-intensive and time-consuming industries (Johri, 1993), with over $481 billion annual sales revenue by 2018. There are four stages in the semiconductor manufacturing process: wafer fabrication, wafer probe (sort), assembly (packaging), and final test. It costs around $4 billion to build a new wafer fabrication facility (fab), and the cost of the equipment is over 75% (Gupta & Sivakumar, 2006). On the other hand, enhancing the fabs' equipment utilization becomes increasingly important because of the enormous cost of wafer fabrication and intensive competition in the semiconductor industry. Therefore, proper utilization of the equipment via scheduling is undoubtedly critical.

In wafer fabrication, ion implantation is a process where ions of dopant elements are accelerated into the silicon substrate. Any improved efficiency in the ion implantation process will contribute to the fab's efficiency. When the previous machine completes a jobin the ion implantation process, it will release the job to the implantation machine (implanter). The processing time is dependent on product types and machines which are processed in an implant step. Moreover, between two jobs processed on the same implanter, a setup time should be considered regarding the product types and the gap used to implant these two jobs. The ion implantation process can be modeled as a parallel machine scheduling problem (PMS) with sequence-dependent setup times. In the ion implantation process, the process times are expected to be different for the same job on different machines. Thus, this problem should be categorized as an unrelated PMS (UPMS) problem. Every job has a release time, and every machine has an available time. As the wafer should be processed at a specified time, every job has an expired time. Each job will be expired and be scrapped if it cannot be processed before the expired time. Therefore, a job can only be processed on a set of machines rather than all the machines.

Only a few studies in this field look into the UPMS problem with sequence-dependent setup time. For instance, Vallada and Ruiz, 2011, and Tamimi & Rajan (1997) presented a hybrid heuristic approach using the genetic algorithm (GA). Kim et al. (2003) proposed a four-phase heuristic model to minimize the total weighted tardiness with tabu search (TS).

However, only one paper has investigated this problem for the ion implantation process to the best of our knowledge. Horng et al. (2000) dealt with the ion implantation process as a UPMS problem with sequence-dependent setup times and job release times. However, the study does not consider the job expired times, available machine times, and different process times on different machines.

In this study, we focus on the problem of scheduling unrelated parallel machines with sequence-dependent setup times subject to job release times and expired times for allowing a job processed on a particular machine. According to the practical requirement of the ion implantation, a three-phase objective problem is set to firstly maximize the number of processed jobs, then minimize the maximum completion time (makespan), and finally minimize the maximum completion times of the non-bottleneck machines. We implement the mixed integer programming (MIP) model, and a hybrid TS algorithm is utilized to solve the problem.

The remainder of the paper is organized as follows. A literature review on ion implantation and popular approaches to solving scheduling problems is discussed in Section 2. The problem is described in detail, and assumptions are stated in Section 3. In Section 4, a two-phase MIP model is proposed, and the efficiency considerations for this model are presented. Section 5 discusses the hybrid TS algorithm and the rules for generating new solutions and algorithm procedures. In Section 6, empirical data is employed to test the proposed approaches, and a computational comparison between the proposed methods and other studies is undertaken. Finally, the conclusion and future work are presented in Section 7.

2. Literature review

In wafer fabrication literature, many studies mainly focus on scheduling the whole wafer fab, generally modeled as a job shop problem (Chiang, 2013; Wein, 1988). As the only evidence in modeling the ion implantation process, Horng et al. (2000)'s approach is a very simplified model, while the production environment is very complicated. Job scheduling for ion implantation has salient features of machine aging and maintenance, different from typically formalized problems in the scheduling research literature. Compared with research problems in job scheduling literature, the scheduling problem for ion implantation has the following unique features (Chou & Huang, 2013):

1. Maintenance is achieved by running complementary part types. Complementary part types have a self-healing effect on the machine. Maintenance is not treated as a separate activity. Instead, maintenance is online, and the health of the machine is gradually restored during production. Therefore, job scheduling, and maintenance schedules are integral tasks.
2. Filament aging increases the probability of machine failure. When a machine fails, repair time is incurred. In contrast, when a decision is made to change the ion type, setup changeover time is incurred, but the filament age is renewed. There are trade-offs between saving setup time and saving machine repair time.

Therefore, more constraints should be considered given the above circumstances, and a more precise model should be proposed.

According to Mönch et al. (2011), except for rework, most of the flow in the fab is deterministic instead of being probabilistic as it is in a job shop. In addition, the processing time per wafer or lot is very nearly deterministic. Therefore, the focus of this study is on the area of deterministic scheduling problems (DSP), attempting to reduce the complexity in the problem formulation.

This paper focuses on the UPMS problem with sequencing-dependent setup times subject to the job release time and expired time of allowing a job to be processed on a machine, defined as: $R|r_j, e_{ij}, ST_{sd}|C_{max}$, where e_{ij} means the expired time depends on the job J_j and the machine M_i. To the best of our knowledge, no literature has studied this specific problem so far. Therefore, the problem can be regarded as the basic $P|ST_{sd}|C_{max}$ problem with additional constraints, and the approaches to the $P|ST_{sd}|C_{max}$ problem can be considered and adapted to solve the $R|r_j, e_{ij}, ST_{sd}|C_{max}$ problem. The UPMS is very complex and considered an NP-hard problem (Baker, 1974). Therefore, in recent years, different heuristic and *meta*-heuristics approaches have been applied to solve the complexity of UPMS, such as the firefly algorithm (Ezugwu & Akutsah, 2018), simulated annealing (Hamzadayi & Yildiz, 2016), artificial bee colony (Lei et al., 2021), smart pool (Cota et al., 2021), TS algorithm (Wang & Alidaee, 2019), particle swarm optimization (PSO) (Salehi & Rezaeian, 2016), and ant colony optimization (ACO) (Behnamian et al., 2009).

According to the results in the literature, the MIP model can yield a high-quality solution if the computational time is long enough. Nevertheless, the long computational time is not allowed in the production environment. In this study, for the exact algorithms, the MIP model is considered based on the research on $R|ST_{sd}|\sum(e_jE_j + t_jT_j)$ (Zhu & Heady, 2000), $R|r_j, ST_{sd}|\sum w_jT_j$ (Lin & Hsieh, 2014), and $P|r_j, ST_{sd}|C_{max}$ (Kurz & Askin, 2001). Table 1 summarizes the new trends in the UPMS problem with sequencing-dependent setup times considering their objective function and some specific features and exact and *meta*-heuristics solution approaches. In the search results presented in Table 1, we focus on literature that at least one of their objective functions is to minimize the makespan. As shown in Table 1, regardless of the application of proposed methods in the literature, the third objective goal in this study for minimizing the makespan of non-bottleneck machines is a novel approach and unique contribution among UPMS literature. In addition, only in few studies, the number of processed jobs has been considered in the objective function, which mainly they plan to minimize the load of each machine instead of maximizing the total processed job, which is one of the goals in our study.

According to our search among literature in the UPMS discipline, out of many possible choices of *meta*-heuristic approaches (as shown in Table 1), in this study, we employed the TS algorithm as a local search-based solution to deal with the NP-hardness of the UPMS problem. The main reasons for selecting the TS algorithm are summarized as follows:

- Metaheuristic algorithms are commonly divided into two groups: local search-based and population-based algorithms. While local search-based algorithms usually start with a random initial solution and iteratively try to improve it using a fitness function, population-based algorithms start with more than one solution and search a more extensive area of the search space, combining current solutions to obtain new ones. Population-based solutions suffer from a lack of precision in comparison with local search. In addition, local-search components and constraint handling flexibility making the algorithm attractive for problems with many constraints (Jaeggi et al., 2008).
- Selecting the TS algorithm generally proceeds more aggressively to the local optimum, relying on the premise that a slow descent will lead to a local optimum closer to a global one. This rationale of the TS algorithm derives from two considerations: (i) optimization problems can be solved by making the best available move at each iteration; (ii) rather than spending more time in regions with less appealing solutions, the TS algorithm spends more time exploring regions with reasonable solutions (Glover, 1989).

Table 1
Summary of some recent trends in UPMS problems with sequencing-dependent setup times.

No	Source	Objective	Main Features	Solution Method
1	Behnamian et al. (2009)	1. Minimize of makespan	–	ACO, SA, VNS
2	Kuo et al. (2011)	1. Minimize the total absolute deviation of makespan 2. Minimize the total load on all machines	Learning effects	MIP
3	Salehi & Rezaeian (2016)	1. Minimize total machine load	Release dates, deteriorating jobs, learning effects	PSO, GA
4	Fanjul-Peyro et al. (2019)	1. Minimize makespan	Renewable resources, machine-dependent setup times	MIP
5	Santos et al. (2019)	1. Minimize makespan	Machine-dependent processing time with additional setup-time	SA, ILS, LAHC, SCHC
6	Ezugwu (2019)	1. Minimize makespan	Non-pre-emptive machine, machine-dependent setup times	GA
7	Yunusoglu & Topaloglu Yildiz (2021)	1. Minimize makespan	Precedence constraint, machine eligibility, release dates, renewable resources	CP
8	Zhang et al. (2021)	1. Minimize makespan 2. Minimize setup time	Worker resources, learning effect	LS
9	Bitar et al. (2021)	1. Maximize the number of completed products 2. Minimize number of auxiliary resources 3. Minimize makespan	Auxiliary resources, machine-dependent setup times	MIP
10	Yepes-Borrero et al. (2020)	1. Minimize makespan	Renewable resource	GRASP
11	Khanh Van & Van Hop (2021)	1. Minimize total weighted earliness, tardiness, and maximum completion time	Capacity constraint of the machines	GA
12	Cota et al. (2021)	1. Minimize the makespan 2. Minimize total consumption of electricity	Independent and non-preemptible jobs	Smart Pool
13	Current study	1. Maximize the number of processed jobs 2. Minimize the makespan 3. Minimize the makespan of the non-bottleneck machines	Job release time and expiration time	TS

VNS: variable neighborhood search; GRASP: Greedy randomized sdaptive search procedure; LS: List scheduling; CP: Constraint programming; ILS: Iterated Local Search; LAHC: Late Acceptance Hill-Climbing; SCHC: Step Counting Hill-Climbing.

- The iterative nature of the TS algorithm may allow it to be stopped after a feasible solution is found. However, methods like the SA algorithm may not be stopped at any desired moment since the control parameter has to converge to a value close to zero to obtain a meaningful implementation (Michiels et al., 2007).
- The TS algorithm, considering neighboring moves, does not need objective function gradient information as some methods like GA (Kulturel-Konak et al., 2003).
- The TS algorithm uses deterministic moves, which reduce variability due to initial solutions and other parameters. In addition, the TS algorithm is designed to prevent repetition rather than a reversal of moves that seem not to work well. Also, the deterministic nature of the TS algorithm may make it feasible to reproduce the results (Brandão & Eglese, 2008).
- Our problem has constraints like feasible job-machine pairs, expired time. The TS algorithm enables us to generate solutions with domain knowledge. For example, we use heuristics 1 and 2 in section 5.1 of our paper to generate solutions. However, other algorithms that exploit information from other solutions may not even generate feasible solutions. For instance, if we use the GA algorithm, crossover or mutation may assign a job to an infeasible machine as operations in GA do not consider domain knowledge. Although some can use unique encoding methods to solve this problem, the TS algorithm seems to have advantages.
- Some algorithms necessitate encoding techniques, such as modeling the key concept velocity in PSO, which adds complexity to the solution approach. However, TS does not have requirements for the encoding of solutions.

According to the aforementioned reasons, we could not find an alternative solution approach that could be compatible with the TS algorithm; therefore, the TS algorithm has been selected for constructing the hybrid *meta*-heuristic solution in this study.

3. Problem description and assumptions

3.1. Description of the problem

The DSPs with setup times can be divided into four classifications in terms of batch (family) or non-batch setup times and independent or dependent setup times (Allahverdi et al., 2008). The schedule of unrelated parallel machines is considered with sequence-dependent setup times subject to job release times and expired times of allowing a job processed on a certain machine, defined as: $R|r_j, e_{ij}, ST_{sd}|C_{max}$. The three-phase objectives are defined to firstly maximize the number of processed jobs, then minimize the maximum completion time (makespan), and finally minimize the maximum completion times of the non-bottleneck machines. It is noted that the objective is to satisfy the requirements of ion implantation.

The number of processed jobs should be maximized because the machines may not necessarily process all the jobs due to the constraint of the expired time. Minimizing the makespan is adopted to improve the utilization of the equipment. As a result of available times of the machine, we find that a distinct bottleneck usually appears on the same machine, leading to high total setup times for non-bottleneck machines. From the perspective of the whole ion implantation process, this may reduce the utilization of the equipment. Thus, minimizing the maximum completion times of the non-bottleneck machines is set as the third-phase objective.

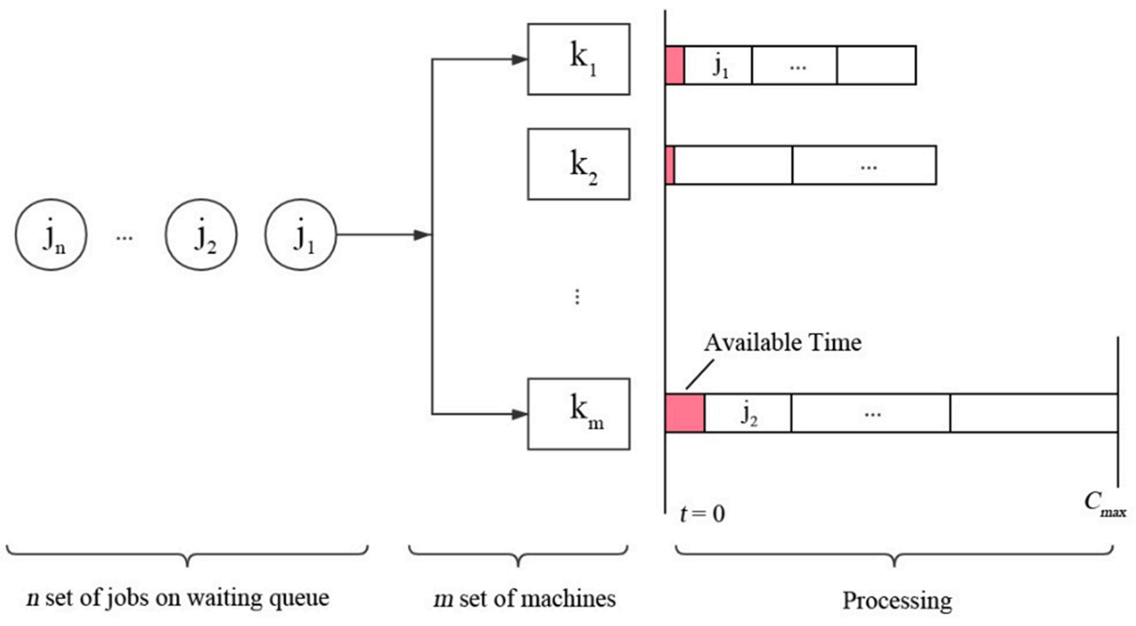

Fig. 1. The Scheduling Process.

The scheduling process can be represented in Fig. 1.

3.2. *Assumptions*

The assumptions of the problem are as follows:

1. Each job can only be processed on a maximum of one machine (for models with multiprocessor tasks, this assumption can be relaxed).
2. No preemption is allowed. In other words, a machine must complete a current job before it processes other jobs.
3. Processing times are independent of sequencing.
4. Processing times for a job can be varied on different machines.
5. The job waiting zone is allowed. Jobs can stay in the waiting zone before all the machines are busy.
6. No cancellation. Every job must be completed.
7. Machine idle time is allowed.
8. The failure and breakdown of the machines are ignored.
9. Each machine can only process one job at a time.
10. Every job has a release time.
11. Every machine has available time.
12. Every match of a job and a machine has an expired time.
13. Every machine can only process part of a job.
14. There is no randomness, and all specifications are known and fixed, including the number of jobs (n), the number of machines (m), and the processing times.

4. Mixed integer programming

In this study, the problem of unrelated parallel machines with sequence-dependent setup times is tackled by a three-phase optimization. The MIP model is applied to tackle the first two objectives, maximizing processed jobs, and minimizing the makespan. Furthermore, the last objective, minimizing the maximum completion times of the non-bottleneck machines, is satisfied by changing the sequence of the jobs on every single machine, which is a classic single-machine scheduling problem and has been investigated by many researchers. For every non-bottleneck machine, a MIP model must be formulated to solve the single-machine scheduling problem according to the result from phase 2, which is complicated. Therefore, a hybrid TS algorithm is proposed to solve the problem with all three objectives, and the MIP model will only tackle the problem with the first two objectives.

4.1. *The* $R|r_j, e_{ij}, ST_{sd}|C_{max}$ *formulation*

The MIP model uses notations as follows:

Sets and indexes:	
i	Index of a machine ($i = 1,\ldots,m$), where m is the number of machines.
j,k	Index of job ($j = 1,\ldots,n$), where n is the number of jobs. Subscripts j and k stand for two adjacent J_j, and J_k. In this study, index J denotes the job.
$\mathscr{M}_j$	The set of machines, $\mathscr{M}$, that can process J_j, $j \in \mathscr{J}$.
$\mathscr{J}_i$	The set of jobs, $\mathscr{J}$ that can be processed on the machine M_i, i.e., $\{j : i \in \mathscr{M}_j, j \in \mathscr{J}\}$.
Parameters	
P_{ij}	The processing time of J_j, on the machine M_i, $i \in \mathscr{M}, j \in \mathscr{J}_i$.
S_{ijk}	The setup time of switching from J_j to J_k on the machine M_i, $i \in \mathscr{M}, j, k \in \mathscr{J}_i, j \neq k$.
S^0_{ij}	The setup time of switching from an existing job to J_j on the machine M_i, $i \in \mathscr{M}, j \in \mathscr{J}_i$.
S_{jk}	The setup time of switching from job J_j to job J_k, $j, k \in \mathscr{J}, j \neq k$.
A_i	The available date of the machine M_i, $i \in \mathscr{M}$.
R_j	The release date of the job J_j, $j \in \mathscr{J}$.
e_{ij}	The expired time of allowing job J_j to be processed on the machine M_i, $i \in \mathscr{M}, j \in \mathscr{J}_i$.
L	A large number.
Variables:	
y_{ij}	Binary variable, 1 if J_j is assigned to the machine M_i and 0 otherwise. $i \in \mathscr{M}, j \in \mathscr{J}_i$.
x^0_{ij}	Binary variable, 1 if J_j is assigned to the machine M_i as the first job and 0 otherwise. $i \in \mathscr{M}, j \in \mathscr{J}_i$.
x_{ijk}	Binary variable, 1 if J_j precedes J_k on the machine M_i and 0 otherwise. $i \in \mathscr{M}, j, k \in \mathscr{J}_i, j \neq k$.
b_j	Continuous variable, the starting setup time of J_j.
c_j	Continuous variable, the completion time of J_j.
C_{max}	Continuous variable, the maximum completion time for all jobs.

The mathematical model for phase 1 is as follows:

$$\textbf{Phase 1 Model}: \text{Max} \sum_{j \in \mathscr{J}} \sum_{i \in \mathscr{M}_j} y_{ij} \tag{1}$$

Subject to :

$$\sum_{i \in \mathscr{M}_j} y_{ij} \leq 1 \quad \forall j \in \mathscr{J} \tag{2}$$

$$x_{ik}^0 + \sum_{j\in\mathscr{J}_i, j\neq k} x_{ijk} = y_{ik} \quad \forall i \in \mathscr{M}, k \in \mathscr{J}_i \tag{3}$$

$$\sum_{k\in\mathscr{J}_i, k\neq j} x_{ijk} \le y_{ij} \quad \forall i \in \mathscr{M}, j \in \mathscr{J}_i \tag{4}$$

$$\sum_{j\in\mathscr{J}_i} x_{ij}^0 \le 1 \quad \forall i \in \mathscr{M} \tag{5}$$

$$b_j - R_j + L\left(1 - \sum_{i\in M_j} y_{ij}\right) \ge 0 \quad \forall j \in \mathscr{J} \tag{6}$$

$$b_j - \max(R_j, A_i) + L\left(1 - x_{ij}^0\right) \ge 0 \quad \forall i \in \mathscr{M}, j \in \mathscr{J}_i \tag{7}$$

$$c_j - b_j + L\left(1 - x_{ij}^0\right) \ge P_{ij} + S_{ij}^0 \quad \forall i \in \mathscr{M}, j \in \mathscr{J}_i \tag{8}$$

$$b_k - c_j + L(1 - x_{ijk}) \ge 0 \quad \forall i \in \mathscr{M}, j, k \in \mathscr{J}_i, j \neq k \tag{9}$$

$$c_k - b_k + L(1 - x_{ijk}) \ge P_{ik} + S_{ijk} \quad \forall i \in \mathscr{M}, j, k \in \mathscr{J}_i, j \neq k \tag{10}$$

$$b_j - e_{ij} - L(1 - y_{ij}) \le 0 \quad \forall i \in \mathscr{M}, j \in \mathscr{J}_i \tag{11}$$

$$y_{ij} \in \{0, 1\} \quad \forall i \in \mathscr{M}, j \in \mathscr{J}_i \tag{12}$$

$$x_j^0 \in \{0, 1\} \quad \forall i \in \mathscr{M}, j \in \mathscr{J}_i \tag{13}$$

$$x_{ijk} \in \{0, 1\} \quad \forall i \in \mathscr{M}, j, k \in \mathscr{J}_i, j \neq k \tag{14}$$

$$b_j \ge 0 \quad \forall j \in \mathrm{J} \tag{15}$$

$$c_j \ge 0 \quad \forall j \in \mathrm{J} \tag{16}$$

The decision variables y_{ij}, x_{ij}^0, $x_{ijk} \in \{0, 1\}$ are non-negative and binary variables, and the decision variables, b_j and c_j, are non-negative and continuous variables. In phase 1, the objective function in (1) maximizes the number of processed jobs. Constraint (2) indicates that one job should not be processed by any machine or can be processed by only one machine. Constraint (3)–(4) represent that if $x_{ijk} = 1$, J_k must come immediately after J_j on the machine M_i, when both y_{ij} and y_{ik} equals to 1. Constraint (5) ensures that each machine can process at most one job as the first job. Constraint (6) indicates that the starting setup time of J_j should be later than its release time. Constraint (7) ensures that if J_j is the first job processed on machine M_i, its starting setup time should be larger than both its release time and the available time of machine M_i. Constraints (8) and (10) represent the time interval between the starting setup and completion time of J_j should be larger than the sum of its setup time and processing time. Constraint (9) states that the starting setup time of successor J_k should be later than the completion time of J_j. Constraint (11) ensures that J_j can begin processing before its expired time on machine M_i. Constraints (12)–(16) indicate the non-negativity restrictions.

The mathematical model of phase 2 is as follows:

$$\text{Phase 2 Model: Min} C_{max} \tag{17}$$

Subject to :

Constraints (2)–(16) in phase 1

$$C_{max} \ge C_j \quad \forall j \in \mathscr{J} \tag{18}$$

$$\sum_{j\in\mathscr{J}} \sum_{i\in\mathscr{M}_j} y_{ij} = \textbf{Phase 1 Model}\left(\text{objective value}\right) \tag{19}$$

$$c_{max} \ge 0 \tag{20}$$

In phase 2, the objective is to minimize the makespan (equation (17)). Constraint (18) ensures that C_{max} is the maximum completion time for all the jobs. Constraint (19) indicates the number of processed jobs is from the objective value of phase 1.

4.2. *Efficiency considerations*

It is easy to implement the formulation above using commercial solvers such as CPLEX. However, some trade-offs are also considered in the development of the model. Firstly, a one-phase model has been considered to simplify the model's procedure and form, although the two-phase model is chosen at the end. Second, the time interval between the starting setup time and the completion time of a job should equal the sum of the process time and setup time, while a slack form is chosen at last (see constraints (8) and (10)). Third, the sets, $\mathscr{M}_j$ and $\mathscr{J}_i$, sharply decrease the number of binary variables, which enables us to improve the numerical efficiency of the model. These three considerations will be discussed as follows.

When formulating the problem, the efficiency and the simplification of the model are the prior factors to be considered. The model can be formulated into a one-phase model by combining two objectives in one objective function. There are two methods under consideration:

1. Minimize $(\sum_{j\in\mathscr{J}} 1 - \sum_{j\in\mathscr{J}} \sum_{i\in\mathscr{M}_j} y_{ij})C_{max}$
2. Minimize $-W\sum_{j\in\mathscr{J}} \sum_{i\in\mathscr{M}_j} y_{ij} + C_{max}$

The first objective function makes the model a mixed-integer nonlinear programming model. Compared with linear programming problems, nonlinear programming problems are intrinsically more difficult to solve (Bradley et al., 1977). Although this method simplifies the procedure, it will reduce the numerical efficiency and take more time to solve this large-scale problem. For the second function, a weight factor W is introduced to depict the relative importance of the number of the processed jobs since the number of the processed jobs is generally less than one hundred and the maximum completion time is usually greater than ten thousand. The number of processed jobs is more important than the maximum completion time, while the W is empirically determined, and it varies from case to case. Therefore, this method is challenging to implement.

In contrast, the two-phase MIP model has the advantage of high numerical efficiency and is easy to implement. The average solution times in phase 1 are about 8 s for a problem with 90 jobs and 30 machines. Moreover, it can derive an acceptable solution in 5 min in phase 2.

A slack form is introduced for the time interval between the job starting setup and completion time to simplify the model (see constraints (8) and (10)). For instance, in constraint (10), if the $x_{ijk} = 0$ then the left-hand side (LHS) should be greater than the right-hand side (RHS), while LHS should be equal to RHS if the $x_{ijk} = 1$. Nevertheless, these two constraints can be combined into one constraint, a slack form, because the time interval between the starting setup time and the completion time tend to be smaller due to the objectives, maximizing the number of processed jobs and minimizing the makespan. The strict constraints and their slack form are presented below:

Strict constraints:

$$c_k - b_k + L \ge P_{ik} + S_{ijk},$$

$$c_k - b_k = P_{ik} + S_{ijk}.$$

Slack form:

$$c_k - b_k + L(1 - x_{ijk}) \ge P_{ik} + S_{ijk}.$$

The slack form is more comfortable to implement than the strict constraints, simplifying the procedure when programming.

The sets, $\mathscr{M}_j$ and $\mathscr{J}_i$, are introduced to reduce the number of variables and improve numerical efficiency. In this problem of unrelated parallel machines with sequence-dependent setup times, a certain J_j can only be processed on several machines rather than all the machines. Similarly, a certain machine M_i can only process several jobs rather than all the jobs. Therefore, the set of jobs processed on the machine M_i and the set of machines processing J_j can directly depict this relationship rather than adding constraints to restrict unmatched jobs and machines after generating the binary variables for every job and machine.

5. Proposed hybrid tabu search

The $P|ST_{sd}|C_{max}$ is a difficult and complex combinatorial problem that can be considered an NP-hard problem because it is equivalent to the TSP when the number of machines is equal to one (Baker, 1974). In addition, the problem discussed in this study is subject to the job release times and expired times of allowing a job to be processed on a certain machine, which makes this problem more complex.

The *meta*-heuristic algorithm has the advantage of high computational efficiency, while the quality of the solutions varies from the models and the parameters. The TS algorithm has outstanding performance among the *meta*-heuristic algorithms because the TS neighborhoods are generated by the rules associated with specific problems, which leads to a smaller solution space. The proposed MIP model in this study can derive an acceptable solution in a reasonable time (within 5 min) using commercial software. In the meantime, a hybrid TS algorithm is proposed to tackle this problem with a three-phase objective.

Phase 1: An initial solution to maximize the number of processed jobs is derived from phase 1 of the MIP model.
Phase 2: A TS algorithm is conducted to achieve enhanced solutions for makespan-moving jobs between various machines. An improved sequence for the machine-implemented removal and insertion is derived by the local search heuristic for every movement.
Phase 3: A heuristic algorithm is adopted to resequence non-bottleneck machines' jobs and minimize the maximum completion times for all the non-bottleneck machines.

The first section in this chapter indicates the heuristic procedures for selecting jobs removed from a certain machine and machines receiving the removed job. After phase 1, all the processed jobs are assigned to all machines, then reassigned from one machine to another during phase 2. In Section 5.2, a detailed interpretation of these three phases will be stated.

5.1. *Insertion and removal procedures*

How jobs are inserted and removed from the machines will be described in this section. Two heuristics are adopted in the following procedure. The main heuristic algorithm applies the local neighborhood concept to generate high-quality solutions (França et al., 1996). The second heuristic algorithm employs randomness to enable some solutions to escape from the local optima.

Consider q as an input parameter to evaluate the local neighborhood of a job. The local neighbors of a J_j, relative to a machine, are its closest successors and predecessors among the jobs assigned to this machine. The closest successor of J_j means the J_k with s_{jk} less than q, and the closest predecessor is J_i with s_{ij} less than q. Now, selecting the best job to be removed is constrained to the job with the least local neighbors. Therefore, the local neighborhood is determined for each machine, i.e., every job has m local neighbors. The concept is represented through the following example. Consider the matrix $[s_{jk}]$ in Table 2. An additional column should be inserted in the matrix in which the element s_{0j} represents the setup time of J_j, the first job in the sequence. Assume that $s_{j0} = 0, I = 1, \ldots, n$.

Table 2
The Matrix of s_{jk}.

s_{ij}	0	1	2	3	4	5	6
0	–	30	32	54	9	38	42
1	0	–	35	25	47	7	89
2	0	21	–	76	38	86	25
3	0	9	38	–	30	32	54
4	0	49	49	23	–	8	45
5	0	23	29	18	36	–	61
6	0	74	82	61	25	38	–

Presume that there are two machines and the sequences assigned to them are:

$$M_1 : -0-2-5 \quad M_2 : -0--3--1--4-6$$

The local neighborhood structure for J_1 and $q = 23$ is then:

$$M_1 : successors = \{0, 5\}; predecessors = \{2, 5\}$$

$$M_2 : successors = \{0\}; predecessors = \{3\}$$

The principle of every movement (removal and insertion) is to choose a job from the busiest machine and then insert it into another machine. The maximum completion time can iteratively be reduced by conducting the movement repeatedly. The detailed procedure of the heuristics is shown below:

Heuristic 1:

- Removal: On the busiest machine, choose the job that has the least number of local neighbors.
- Insertion: For the chosen job, compute the number of local neighbors on every possible machine. Then, select the machine with the greatest number of local neighbors.

Heuristic 2:

- Removal: On the busiest machine, randomly choose a job as the removal job.
- Insertion: For the selected job, randomly insert it into a machine that can process the job.

Heuristic 1 is proposed to make the total setup time as small as possible after implementing the removal and insertion procedures. Besides, if the jobs on the machine M_i are determined, the maximum completion time of these jobs mainly depends on the total setup time, as the process times for these jobs on the machine M_i are fixed. The local neighbors for the J_j are the jobs whose setup times with the J_j are less than q. Therefore, in theory, more local neighbors every job on the machine M_i has, more likely, a sequence with less total setup time is derived.

The removal of the job with the least local neighbors from the busiest machine means the remaining jobs have more local neighbors than the removed ones. The insertion makes the removed job is taken to the machine with the largest number of its local neighbors. In other words, these two procedures are applied to reduce the total setup time for the jobs on the machine related to removal and on the machine related to insertion, which have a high number of local neighbors.

Generally, *Heuristic 1* can generate a solution with a higher possibility of approaching the optimal solution than randomly generated solutions. Nevertheless, *Heuristic 2* can avoid trapping in the local optima. Combining these two heuristics will contribute to an efficient and high-quality search.

5.2. *A tabu search algorithm*

TS, first developed by Glover (1989, 1990), is a global optimization *meta*-heuristic algorithm. One critical difference from the so-called hill-climbing algorithms is that the TS algorithm can jump out of the local minima and give better solutions. A mechanism equips the search process to allow the objective function to deteriorate. Furthermore, by doing this, the mechanism enables the solution to escape from the local optima. The movement with a minimum cost will be chosen as the next generation. If the solution is a local minimum, this means accepting an unimproved perturbation. The solution might also fall back into the local optimum where it previously escaped because the search always selects the best movement of one iteration. The recently conducted movement is forbidden (tabu) and appended into a constantly updated tabu list to avoid this situation. Over the past few years, several combinatorial problems have been well solved using the TS algorithm (Glover, 1990).

The TS algorithm implemented in this study is an adapted version of the TS algorithm for $P|ST_{sd}|C_{max}$ problems without the constraints of the job release time and the expired time of allowing a job processed on a certain machine developed by França et al. (1996). The TS algorithms are open-ended, and they should be formulated in terms of the characteristics of one specific problem. The key features which can enhance the implementation of the TS algorithm are presented as follows (França et al., 1996):

1. The neighborhood structure.
2. The initial solution.
3. The tabu tenure, i.e., how long a tabu move will be forbidden.
4. The stopping criterion.

A neighbor solution is derived by removing jobs from the busiest machine and inserting them into another machine in our approach for implementing the TS algorithm to tackle the $R|r_j, e_{ij}, ST_{sd}|C_{max}$ problem. The concepts of a local neighborhood are applied to construct all the neighbors. There are several methods to construct the neighborhood. One main method (*Heuristic 1*) is to take the J_j belonging to the busiest machine and then insert it into one of the machines from the set $\mathscr{M}(J_j)$ to form a new solution. The set $\mathscr{M}(J_j)$ includes the machines that may process the J_j and is defined as $\mathscr{M}(J_j) = \{M_i | M_i \neq \overline{M}_i, M_i \in \mathscr{M}_j\}$, where $\overline{M}_i$ is the busiest machine.

$C(M_i)$ is introduced to be the completion time for processing all the jobs assigned to the machine M_i. The movement which leads to the minimum value of $C(M_i)$ looking at every insertion/removal heuristic algorithm stated in the preceding section when referring to insertion or removal of jobs is called the optimal movement. A vector $(J_j, \overline{M}_i, M_i^*)$ is introduced to indicate a move which states the J_j that is removed from the busiest machine and inserted into the machine M_i^* that has been chosen from the candidate set $\mathscr{M}(J_j)$. After a movement is conducted, the solution is forbidden until it is removed from the tabu list. How long a solution will stay in the tabu list depends on the tabu tenure, which is the key performance indicator in most implementations of the TS algorithm (França et al., 1996). If it is too small, the TS algorithm may be cycling and trapped in the local optima.

In contrast, if it is too huge, it may lead to more neighbors being forbidden and fewer neighbors available, and the TS approach may execute poorly because the constraints are too strict to search for a better solution. We will discuss the appropriate choice for the tabu tenure and other parameters at the end of the section. The number of iterations, T, is chosen to be the stop criterion of the TS algorithm.

Phase 1 is the same as phase 1 in the MIP model. The solution will be stored and transformed to phase 2 as the initial solution. The solution contains the schedule on every machine, the completion time of every machine, and the makespan.

For ***Phase 2***, setup times are independent of the machine; therefore, suppose that $\overline{M}_i$ is the busiest machine and consider the following notations (França et al., 1996):

$C(M_i, J_j)$ = completion time associated with the machine M_i if J_j is inserted in it.

$C(\overline{M}_i, J_j)$ = completion time associated with the machine $\overline{M}_i$ if J_j is removed from it.

The detailed procedure of phase 2 is as follows:

Step 1: Set the iteration counter $t = 0$.

Step 2: Set the candidate solution counter $s = 1$.

Step 3: Generate the candidate solutions using *Heuristic 1* and *Heuristic 2* as proposed in section 5.1.

a. Determine the J_j to be moved and received by the machine M_i^*, using *Heuristic 1* or *2*. If $s = 1$, *Heuristic 1* will be applied; otherwise, *Heuristic 2* will be applied.
b. Compute the minimum completion time among p randomly generating sequences for the single machine M_i^* after the insertion of J_j, defined a:

$$C^*(M_i^*, J_j) = min\{C(M_i^*, J_j) | \text{p sequences of the machine } M_i^*\}$$

c. Consider p as iteration number for re-optimization in every iteration; compute the minimum completion time among p randomly generated sequences for the busiest machine $\overline{M}_i$ after the removal of J_j, defined as:

$$C^*(\overline{M}_i, J_j) = min\{C(\overline{M}_i, J_j) | \text{p sequences of the machine } \overline{M}_i\}$$

d. Compute the maximum completion time if J_j is moved from $\overline{M}_i$ to M_i^*, defined a:

$$T_s = max\{C(M_i) | \forall i \in \mathscr{M}\}$$

e. If the yielded solution is tabu, go back to action *a* at *step 3* and generate another solution (ensure every candidate solution is not tabu).
f. Set $s = s + 1$. If $s < 100$, go back to action *a* at *step 3* and generate the next candidate solution. Otherwise, go to *step 4*.

Step 4: Select the *solution** as *solution** $= argmin\{T_s\}$, and append *solution** in the tabu list.

Step 5: Determine the busiest machine and update the best solution, optimal makespan, and incumbent solution.

Step 6: Set $t = t + 1$. If T, the number of iterations has not been satisfied, go to *Step 2*. Otherwise, complete the optimization and export the solution.

The problem in ***phase 3*** can be considered as a single-machine scheduling problem. The objective is to minimize the maximum completion time of every non-bottleneck machine separately. Based on the cases investigated in this study, the number of processed jobs on a machine is small as the optimization result shows, generally less than five jobs, barely up to 10 jobs. Therefore, a simple heuristic is applied.

For every non-bottleneck machine, consider the following steps:

Step 1: If the number of processed jobs is nonzero, implement *step 2*; otherwise, ignore this machine and consider the next machine.

Step 2: Generate p random solutions (p job sequences) and select the solution with the shortest completion time. The random solution is the randomly generated sequence of the jobs on a machine.

For the values of parameters, comments on them are shown below. The number of randomly generated sequences, p, for a single machine to yield a minimum completion time is defined as $p = min\{100, n_i!\}$, where n_i is the number of jobs on the under optimization machine. The tabu tenure can be determined according to specific cases, and 12 is set as the tabu tenure for the case investigated in this study. The number of iterations, T, is set to be $1.2n$. The proposed algorithm is tested for 90 jobs and 30 machines and has an acceptable numerical efficiency.

6. Empirical study

In this study, real instances from a wafer fab were employed to evaluate the efficiency of the proposed MIP model and the hybrid TS algorithm. The software is halted for the MIP model and exported a feasible solution if the implementation time exceeded 300 s in phase 2 since an acceptable solution derived in a reasonable time was expected. For the TS algorithm, the best solution was chosen among the five executions. Because of the same phase 1, the two algorithms derived the same maximum number of processed jobs. All problems were solved on a Surface Pro (5th Gen) with Intel Core i5-7300U, and the MIP model was solved by CPLEX (v12.7.1), a standard commercial solver developed by IBM.

6.1. Data

The data applied in the experiment is from the ion implantation process of a wafer fab. The sample period was from 8:00 AM to 12:00 PM on August 8th, 2019, and 10 cases were investigated in this study. Besides, a set of 100 synthesized data is generated based on these 10 cases to validate the performance of the proposed algorithm on a larger scale. Moreover, the data structure is organized by presenting attributes of a job processed on a machine and the machine's attributes in a row. Every combination of a job and a machine occupies a row. The attributes are as follows:

- *AppId*: the job id that can exclusively identify a job.
- *Ppid*: an attribute of the job which will be used to determine the setup time.
- *Process time*: the processing time of a job on a machine.
- *Gas*: the gas type for processing a job.
- *Job release time*: the estimated release time of a job.
- *EqpId*: the machine identification value that can exclusively identify a machine
- *Running AppId*: the identification value of the already-existing job on this machine. This attribute is employed to determine the setup time of the first job on the machine.
- *Running Gas*: the type of the already existing gas on this machine. This attribute is employed to determine the setup time of the first job on the machine.
- *Machine availability*: the amount of time a machine is available to process jobs.
- *RtdReason*: the reason why this machine-job combination is infeasible. Only the rows where RtdReason is empty will be kept in the experiment, and others will be deleted.
- *Expired time*: the expired time of this machine-job combination, which means the job will expire if it cannot be processed before this time.

The *setup time* is determined by the job existing on the machine and its successor:

- If these two jobs have the same *ppid*, the setup time is 0 s.
- If these two jobs have different *ppid* but the same *gas*, the setup time is 60 s.
- If these two jobs have different *gas*, the setup time is 900 s.

6.2. Computational results

The MIP model is applied to solve the problem with the first two objectives, and the hybrid TS algorithm is applied to tackle the problem with all three objectives. The comparison between the results of the MIP model and the hybrid TS algorithm in phase 2 is conducted. Then, the comparison between the results in phase 2 and phase 3 of the hybrid TS algorithm is undertaken. In addition, in order to validate the performance of the proposed algorithm and evaluate the adequacy of the model under a higher number of cases, relying on the empirical data, we generated 100 instances of synthesized data when randomly selected lot-machine pairs from 10 cases of empirical data were changed. The average performance of synthesized data has been calculated for comparing the maximum completion time of the MIP model and TS algorithm on a larger scale.

Table 3
MIP and TS Computational Results.

CI	NJ	T_{MIP}	T_{TS}	Z_{MIP}	Z_{TS}
1	74	305.74	19.99	13,104	12,400
2	77	307.4	23.37	11,555	11,555
3	76	305.45	25.95	20,065	14,232
4	77	305.34	18.84	13,852	9702
5	75	304.87	19.52	14,687	8795
6	72	302.25	12.49	13,460	9045
7	69	303.74	9.44	13,255	11,290
8	68	303.37	9.84	15,681	14,965
9	89	306.34	20.09	15,173	12,632
10	80	310.01	55.23	20,136	9320
Synthesized data	79	305.45	23.48	15,794	11,294

Table 3 represents the maximum number of processed jobs, the average computational time for various cases using the MIP model and TS, and the objective value of the MIP model and TS algorithm after phase 2 optimization. When implementing these two algorithms, the average computational time is the sum of the time on phase 1 and phase 2. Fig. 2 shows the maximum completion time comparison of the MIP model and TS algorithm. Fig. 3 shows the relative improvement of objective values using the TS algorithm compared with the MIP model. The relative improvement is calculated using the equation:

$$(Z_{MIP} - Z_{TS})/Z_{MIP}. \tag{21}$$

The terminology used in Table 3 is as follows:

CI	case index
NJ	maximum number of processed jobs
T_{MIP}	average computational time using the MIP model (s)
T_{TS}	average computational time using the TS algorithm (s)
Z_{MIP}	MIP solution value (s)
Z_{TS}	TS solution value (s)

The results in Table 3, Fig. 2, and Fig. 3 indicate that TS outperforms the MIP model in computational time and solution quality. If implementing more time, the MIP will give a better solution, while the longer computational time is not reasonable in the production environment. The hybrid TS algorithm yields a better solution in a shorter computational time, from 9 s to 56 s than the MIP model, more than 300 s. Fig. 3 presents that the results have improvements from 0% to 54%, on average 29%, varying from case to case. Focusing on the computational time of the TS algorithm, such short times show that the TS algorithm is quite suitable for production applications.

The problem investigated in this study has quite a bit of computational complexity. The UPMS problem, which is related to the problem in this study, is proven to be NP-hard (Baker, 1974). Therefore, solving the MIP model of this problem is difficult and time-consuming. The hybrid TS algorithm generates potential solutions for the practical problem, where the insight is to make the total setup time as short as possible; therefore, it could outperform the MIP.

Table 4 indicates the differentiation of the results of non-bottleneck machines between phase 2 and phase 3 using the TS algorithm. Table 4 shows the maximum completion times of non-bottleneck machines in phase 2 and phase 3 using the TS algorithm, respectively. It is noted that the non-bottleneck machines processing no jobs will not be presented in

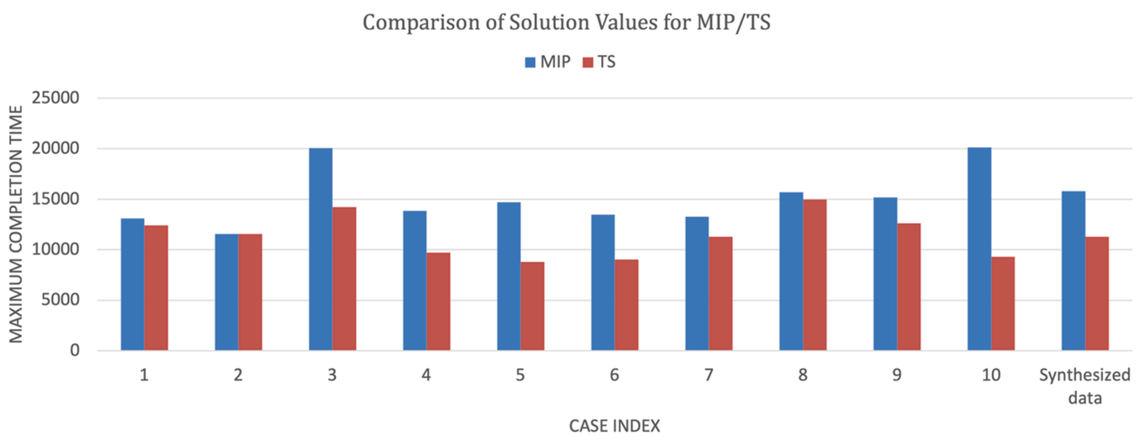

Fig. 2. Comparison of the Solution Values for MIP/TS.

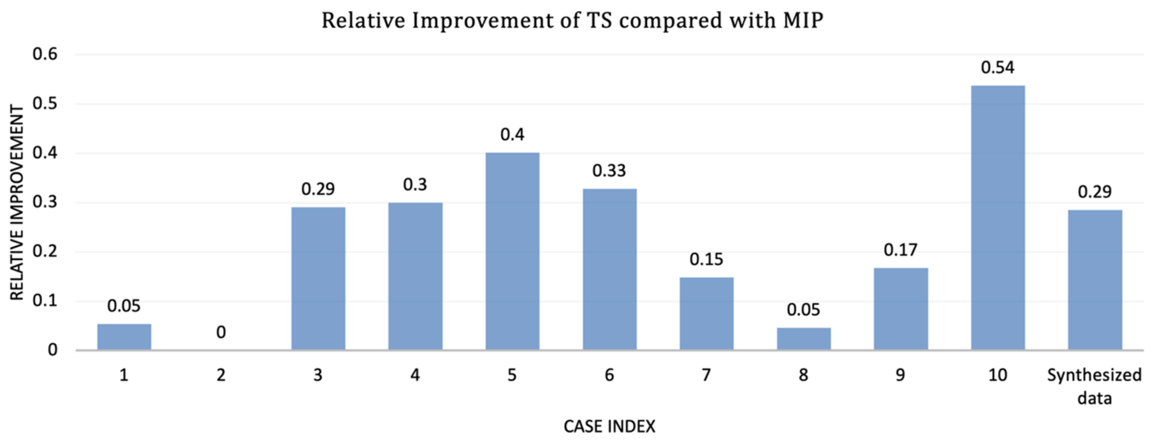

Fig. 3. Relative improvement of the TS algorithm compared with the MIP model.

Table 4
The Computational Results in Phase 2 and Phase 3 Using TS.

MI	Z_2	Z_3	R_{phase3}
2	7477	6552	0.124
3	1045	1045	0
4	4317	4317	0
12	11,080	11,080	0
13	10,646	10,646	0
14	10,325	10,325	0
15	8643	7803	0.097
16	5755	4300	0.253
17	9470	9470	0
20	8666	8666	0
22	6142	5025	0.182
28	8621	8621	0
29	5340	4940	0.075
32	5362	5135	0.042

Table 4. The relative improvement is calculated using the equation:

$$R_{phase3} = \frac{Z_2 - Z_3}{Z_2}. \tag{22}$$

The terminology used in Table 4 is as follows:

MI	machine index
Z_2	maximum completion time of the machine in phase 2
Z_3	maximum completion time of the machine in phase 3
R_{phase3}	relative improvement from phase 2 to phase 3

According to Table 4, the heuristic applied in phase 3 of the TS algorithm effectively enhances the utilization of the equipment based on a minimum makespan in phase 2, while this heuristic is also efficient as the computational cost is within 1 s. Meanwhile, it is noted that the maximum completion times on several machines cannot be reduced. Two possible reasons are considered: the solution yielded from phase 2 of the TS algorithm is optimal. It is possible because the strict constraints of this problem make a few feasible solutions on certain machines. In addition, in phase 2 of the TS algorithm, the insertion or removal is implemented on certain machines, and the heuristic in phase 3 is used in the insertion and removal procedures. Therefore, the heuristic becomes ineffective after it has been applied in phase 2.

More complex methods are not used in phase 3 of the TS algorithm because the optimization problem is simple (single-machine-scheduling with the number of jobs less than 5 for most non-bottleneck machines). At the same time, computational efficiency should be considered, and this heuristic can be implemented within 1 s.

7. Conclusion

This study investigates the scheduling of ion implantation in wafer fabrication. To the best of our knowledge, only one paper has studied this problem and modeled it as a UPMS problem with sequence-dependent setup times and job release times. This paper proposes a more precise model by considering the job expired times, available machine times, and different process times on different machines and two approaches to solve this problem. The contributions of this study can be concluded as follows:

1. Model the scheduling of the ion implantation as a UPMS problem with sequence-dependent setup times subject to job release times and expired times of allowing a job processed on a certain machine, defined as:$R|r_j, e_{ij}, ST_{sd}|C_{max}$.
2. Propose a MIP model to satisfy the first two objectives of the problem and solve the MIP model by CPLEX. Some considerations are introduced to simplify the model, and at the same time, enhance computational efficiency.
3. Propose a hybrid TS algorithm to satisfy all the three objectives of the problem. The effectiveness and efficiency are tested to be higher than that of the MIP model.

The two algorithms have tested real cases from a wafer fab to compare their performance. The hybrid TS algorithm outperforms the MIP model in computational time and solution quality for the first two optimization phases. The makespan yielded from the hybrid TS algorithm shortens from 0% to 54%, compared to the MIP model's makespan. The hybrid TS algorithm shows a shorter computational time, from 9 to 56 s, than the MIP model, with more than 300 s. The heuristic applied in phase 3 of the hybrid TS algorithm effectively enhances the utilization of the equipment based on a minimum makespan in phase 2. Phase 3 optimization can be completed in a short time (within 1 s). The proposed hybrid TS algorithm effectively and efficiently solves the ion implantation's scheduling problem from the above discussion. The proposed MIP model in this study can be employed to generate the benchmark and evaluate other algorithms.

On the other hand, there are still some limitations in this study:

1. Distinct bottleneck machine. This study proposes a heuristic in phase 3 of the hybrid TS algorithm to reduce the total setup times of the non-bottleneck machines caused by the machine's available times. Nevertheless, the effectiveness of the heuristic heavily relies on the practical situation where the number of jobs on the non-bottleneck machines is generally less than 5 in the tested cases.
2. Limited complexity. The approaches proposed in this study are mainly tested by the cases of 30 machines and 90 jobs. It remains to be investigated if the approaches are effective and efficient when the problem gets more complex.
3. The hybrid TS algorithm can be further improved. For the hybrid TS algorithm, only one job will move in one iteration. It remains to be investigated if an algorithm can move more jobs in an iteration and more effectively generate solutions.

The following future research directions are recommended as follows:

1. *Other ways to model the problem*: this paper proposes to model the ion implantation process as a $R|r_j, e_{ij}, ST_{sd}|C_{max}$ problem and to achieve three objectives: first, maximize the number of processed jobs; second, minimize the maximum completion time (makespan); and finally, minimize the maximum completion times of the non-bottleneck machines. Moreover, it is encouraged to study modeling the ion implantation process in other ways.
2. *The heuristic in phase 3 of the hybrid TS algorithm*: other heuristics for phase 3 of the hybrid TS algorithm can be investigated to solve problems with more jobs on non-bottleneck machines.
3. *More complex cases*: In the practical situation, the problem may become more complex. More jobs or more machines should be considered. How to effectively solve the problem with a large size is another research direction.
4. *Solution generation strategy:* The hybrid TS algorithm is suggested to move multiple jobs in an iteration to yield a better solution and increase efficiency.

CRediT authorship contribution statement

Changyu Chen: Data curation, Writing – original draft, Visualization, Validation. **Mahdi Fathi:** Writing – review & editing, Supervision. **Marzieh Khakifirooz:** Writing – review & editing, Validation. **Kan Wu:** Conceptualization, Methodology, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research is supported in part by the Ministry of Science and Technology, Taiwan, R.O.C. under Grant no. MOST 110-2622-8-182-002.

References

Allahverdi, A., Ng, C., Cheng, T. E., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research, 187*(3), 985–1032.
Baker, K. R. (1974). *Introduction to sequencing and scheduling.* John Wiley & Sons.
Behnamian, J., Zandieh, M., & Ghomi, S. F. (2009). Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm. *Expert Systems with Applications, 36*(6), 9637–9644.
Bitar, A., Dauzère-Pérès, S., & Yugma, C. (2021). Unrelated parallel machine scheduling with new criteria: Complexity and models. *Computers & Operations Research, 132*, Article 105291.
Bradley, S. P., Hax, A. C., & Magnanti, T. L. (1977). *Applied mathematical programming.* Addison-Wesley.
Brandão, J., & Eglese, R. (2008). A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research, 35*(4), 1112–1126.
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.
Chou, Y. C., & Huang, P. H. (2013). Integrating machine scheduling and self-healing maintenance by job-mix pull control. *International Journal of Production Research, 51* (20), 6194–6208.
Cota, L. P., Coelho, V. N., Guimarães, F. G., & Souza, M. J. (2021). Bi-criteria formulation for green scheduling with unrelated parallel machines with sequence-dependent setup times. *International Transactions in Operational Research, 28*(2), 996–1017.
Ezugwu, A. E. (2019). Enhanced symbiotic organisms search algorithm for unrelated parallel machines manufacturing scheduling with setup times. *Knowledge-Based Systems, 172*, 15–32.
Ezugwu, A. E., & Akutsah, F. (2018). An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times. *IEEE Access, 6*, 54459–54478. https://doi.org/10.1109/ACCESS.2018.2872110
Fanjul-Peyro, L., Ruiz, R., & Perea, F. (2019). Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. *Computers & Operations Research, 101*, 173–182.
França, P. M., Gendreau, M., Laporte, G., & Müller, F. M. (1996). A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *International Journal of Production Economics, 43*(2–3), 79–89.
Glover, F. (1989). Tabu search—part I. *ORSA Journal on Computing, 1*(3), 190–206.
Glover, F. (1990). Tabu search—part II. *ORSA Journal on Computing, 2*(1), 4–32.
Gupta, A. K., & Sivakumar, A. I. (2006). Job shop scheduling techniques in semiconductor manufacturing. *The International Journal of Advanced Manufacturing Technology, 27*(11–12), 1163–1169.
Hamzadayi, A., & Yildiz, G. (2016). Event driven strategy based complete rescheduling approaches for dynamic m identical parallel machines scheduling problem with a common server. *Computers & Industrial Engineering, 91*, 66–84.
Horng, S.-M., Fowler, J. W., & Cochran, J. K. (2000). A genetic algorithm approach to manage ion implantation processes in wafer fabrication. *International Journal of Manufacturing Technology and Management, 1*(2–3), 156–172.
Jaeggi, D. M., Parks, G. T., Kipouros, T., & Clarkson, P. J. (2008). The development of a multi-objective Tabu Search algorithm for continuous optimisation problems. *European Journal of Operational Research, 185*(3), 1192–1212.
Johri, P. K. (1993). Practical issues in scheduling and dispatching in semiconductor wafer fabrication. *Journal of Manufacturing Systems, 12*(6), 474.
Kim, S.-S., Shin, H., Eom, D.-H., & Kim, C.-O. (2003). A due date density-based categorising heuristic for parallel machines scheduling. *The International Journal of Advanced Manufacturing Technology, 22*(9–10), 753–760.
Khanh Van, B., & Van Hop, N. (2021). Genetic algorithm with initial sequence for parallel machines scheduling with sequence dependent setup times based on earliness-tardiness. *Journal of Industrial and Production Engineering, 38*(1), 18–28.

Kulturel-Konak, S., Smith, A. E., & Coit, D. W. (2003). Efficiently solving the redundancy allocation problem using tabu search. *IIE Transactions, 35*(6), 515–526.

Kuo, W. H., Hsu, C. J., & Yang, D. L. (2011). Some unrelated parallel machine scheduling problems with past-sequence-dependent setup time and learning effects. *Computers & Industrial Engineering, 61*(1), 179–183.

Kurz, M., & Askin, R. (2001). Heuristic scheduling of parallel machines with sequence-dependent set-up times. *International Journal of Production Research, 39*(16), 3747–3769.

Lei, D., Yuan, Y., & Cai, J. (2021). An improved artificial bee colony for multi-objective distributed unrelated parallel machine scheduling. *International Journal of Production Research, 59*(17), 5259–5271.

Lin, Y.-K., & Hsieh, F.-Y. (2014). Unrelated parallel machine scheduling with setup times and ready times. *International Journal of Production Research, 52*(4), 1200–1214.

Michiels, W., Aarts, E., & Korst, J. (2007). *Theoretical aspects of local search*. Springer Science & Business Media.

Mönch, L., Fowler, J. W., Dauzere-Peres, 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.

Salehi, M. S., & Rezaeian, J. (2016). A robust hybrid approach based on particle swarm optimization and genetic algorithm to minimize the total machine load on unrelated parallel machines. *Applied Soft Computing, 41*, 488–504.

Santos, H. G., Toffolo, T. A., Silva, C. L., & Vanden Berghe, G. (2019). Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem. *International Transactions in Operational Research, 26*(2), 707–724.

Tamimi, S. A., & Rajan, V. N. (1997). Reduction of total weighted tardiness on uniform machines with sequence dependent setups. *Paper presented at the Industrial Engineering Research-Conference Proceedings*.

Vallada, E., & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research, 211*(3), 612–622.

Wein, L. M. (1988). Scheduling semiconductor wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing, 1*(3), 115–130.

Wang, H., & Alidaee, B. (2019). Effective heuristic for large-scale unrelated parallel machines scheduling problems. *Omega, 83*, 261–274.

Yepes-Borrero, J. C., Villa, F., Perea, F., & Caballero-Villalobos, J. P. (2020). GRASP algorithm for the unrelated parallel machine scheduling problem with setup times and additional resources. *Expert Systems with Applications, 141*, Article 112959.

Yunusoglu, P., & Topaloglu Yildiz, S. (2021). Constraint programming approach for multi-resource-constrained unrelated parallel machine scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, 1–18.

Zhang, L., Deng, Q., Lin, R., Gong, G., & Han, W. (2021). A combinatorial evolutionary algorithm for unrelated parallel machine scheduling problem with sequence and machine-dependent setup times, limited worker resources and learning effect. *Expert Systems with Applications, 175*, Article 114843.

Zhu, Z., & Heady, R. B. (2000). Minimizing the sum of earliness/tardiness in multi-machine scheduling: A mixed integer programming approach. *Computers & Industrial Engineering, 38*(2), 297–305.