IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 21, 2019, accepted February 5, 2019, date of publication February 15, 2019, date of current version March 7, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2899601*

# Solving the Mask Data Preparation Scheduling Problem Using Meta-Heuristics

**KUO-CHING YING[1], SHIH-WEI LIN[2,3,4], CHIEN-YI HUANG[1], MEMPHIS LIU[5], AND CHIA-TIEN LIN[6]**
[1]Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 106, Taiwan
[2]Department of Information Management, Chang Gung University, Taoyuan 333, Taiwan
[3]Department of Neurology, Linkou Chang Gung Memorial Hospital, Taoyuan 333, Taiwan
[4]Department of Industrial Engineering and Management, Ming Chi University of Technology, New Taipei 243, Taiwan
[5]Department of Mask, Nanya Technology, Taoyuan 333, Taiwan
[6]Department of Information Management, Linkou Chang Gung Memorial Hospital, Taoyuan 333, Taiwan

Corresponding author: Shih-Wei Lin (swlin@mail.cgu.edu.tw)

The work of K.-C. Ying was supported by the Ministry of Science and Technology of China, Taiwan, under Grant MOST106-2221-E-027-085. The work of S.-W. Lin was supported in part by the Ministry of Science and Technology of China, Taiwan, and in part by the Linkou Chang Gung Memorial Hospital under Grant MOST107-2410-H-182-005-MY2/MOST106-2632-H-182-001 and Grant CMRPD3G0011/CARPD3B0012.

**ABSTRACT** Mask data preparation (MDP) is a part of the mask data process for fabricating semiconductors, and its importance has commonly been neglected. This paper proposes an integer linear programming model and two meta-heuristics, a genetic algorithm (GA) and simulated annealing (SA), for solving the MDP scheduling problem (MDPSP). The proposed meta-heuristics are empirically evaluated using 768 simulation instances of MDPSP based on the characteristics of a real technology company and compared with the most commonly used first-come, first-served method. The experimental results reveal that the proposed GA and SA algorithms can critically improve the manufacturing schedule for semiconductor factories.

**INDEX TERMS** Scheduling, integer linear programming, mask data preparation, meta-heuristics, genetic algorithm, simulated annealing.

## I. INTRODUCTION

Products that are derived from semiconductors have long been an essential part of daily life. A semiconductor fabrication line involves hundreds of steps, which comprises diffusion, photoresist, exposure, development, etching, implant, chemical vapor deposition, and metallization. Optical lithographic imaging methods (photo-lithography, LITHO) are regarded as the most complicated in the semiconductor fabrication process [1]. Circuit patterns are transferred onto the wafer which is coated with a photo-resist and then baked to make the photo-resist for subsequent exposure [2], [3]. However, the LITHO workshop is commonly a bottleneck and related scheduling decisions affect the efficiency of the entire semiconductor manufacturing process [4].

The photomask is an important LITHO tool in the semiconductor manufacturing process. The alignment and exposure process in a photolithography workshop is repeated until the circuit with numerous layers has been formed. Each layer, with specific device characteristics of a layer, requires its photomask. Mask data preparation (MDP) provides mask data information to mask-shops, including the specifications and instructions to build a photomask. It is the process of translating a file that contains the intended set of polygons in an integrated circuit layout into a set of instructions that can be followed by a photomask writer to generate a physical mask [5]. When a chip design becomes tape-out, mask databases and mask tooling documents are prepared. MDP is the final stage of product design and the first stage of production. The completion of MDP represents the beginning of production. From 2010 to the present, MDP has been one of the procedures that are considered in improving the efficiency of the semiconductor manufacturing process.

As shown in Figure 1, MDP comprises three major steps, which are mask data generation (MG), optical proximity correction (OPC) and mask tape-out (TO). While the first two steps relate to design for manufacture (DFM), the third step involves outputting a document and data translation. Three steps are as follows.

The associate editor coordinating the review of this manuscript and approving it for publication was Qing Chang.

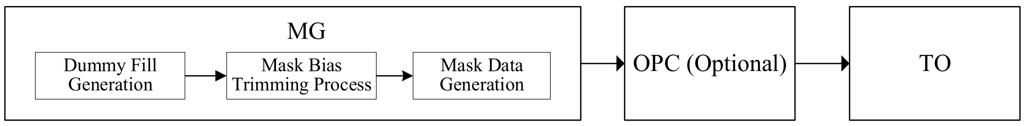

**FIGURE 1.** **The flowchart of MDP.**

(1) MG comprises three operations. The first generates the dummy fill; the second is the mask bias trimming process; the third is mask data generation. In the past, circuit design has had to meet design specifications that depend strongly on the production machine used. If the production machine is changed, then a time-consuming circuit redesign must be performed. Today, MG is performed by a simple bias trimming process to save time and workforce.

(2) OPC is a resolution enhancement method that improves pattern fidelity in lithography [6], [7]. Increasing the circuit density improves the integrated circuit (IC) performance but leads to patterning proximity issue. A layout edge or fragment migrates to its proper position to minimize edge placement error [8]. OPC can be performed using rule-based and model-based methods. Owing to the availability of the required software, and the time and cost of the process, OPC is conducted only in the critical layer.

(3) TO transforms several sets of graphic data into the electron beam lithography system (manufacturing electron beam exposure system, MEBES) format, which can be read by a mask writer. TO also performs the job deck view (JDV). The information is then passed to mask-shops for mask production after the information has been verified.

The circuit designer follows the specifications for circuit design and then hands off to the production department. In the past, whereas the production department has been responsible for manufacturing faults, the design department has been responsible for issues concerning the device and its electrical properties. Today, process scaling has reached insurmountable physical limits, so the production department is commonly unable to achieve a particular yield rate under the specifications of the circuit that are set by the design department. Design and production departments must work closely together to solve this problem [9]. Schiavone *et al.* [10] presented a novel mask proximity correction software to ensure accuracy and to reduce writing time in photomask manufacturing. The trend in the field of circuit technology to reduce the dimensions of critical features has drastically increased the size of design files. Marques *et al.* [11] presented an MDP flow that maintains a consistent delivery time to mask-shops. Also, simulation-based recent works have surveyed MDP and developed new methods to improve the productivity of OPC [12]. In the past, one product has not required the use of too many masks, and the first-come, first-served (FCFS) method is frequently used. However, with the evolution of production process technology, the monthly number of masks produced has jumped by a factor of approximately two to three. In this situation, solving MDPSP using the FCFS method tends greatly to delay delivery, and cause problems of staff allocation. Therefore, algorithms must be developed to deal with this situation.

The MDPSP that is considered herein can be elucidated as follows. Consider a set of $n$ mask data and each mask datum has three tasks to be processed in three production stages (MG, OPC, and TO). Each mask datum may have a different release time from the others. The flow of mask data through the MDP is unidirectional, but some mask data do not have to be processed in a single production stage (OPC in the MDP). Each production stage involves operations on $m_i$ machines. At each production stage $i$ ($i = 1, 2, 3$), one task of mask datum $j$ is performed by one of parallel machines and without pre-emption. Each machine can process no more than one mask datum at a time and the machines are continuously available. The processing time and the resources required are the same for all mask data in the same stages of MG or TO (and OPC if required). The objective is to find a schedule that minimizes the total weighted tardiness. Notably, the MDPSP considered herein is a special case of the hybrid flowshop scheduling problem (HFSP) with a missing operation, *i.e.*, some mask data can skip the stage of OPC. The traditional HFSP consists of a series of production stages, in which each stage has some identical parallel machines. While some stages may have only one machine, at some stage the machines are duplicated to enhance the capacity of the shop floor, or to balance the capacities between different stages. A set of independent jobs must be sequentially processed through these production stages, and each job consists of several operations to be performed by none, one or more machines on each stage. However, in most of the studies, it is assumed that there is one operation for all production stages [13].

The hybrid flowshop system (HFS) has been applied to various industries, including the semiconductor, electronics, textile and food industries [14]. In recent decades, the wide range applications of the HFS have attracted many attentions from researchers and practitioners who have addressed various HFSPs. The algorithms for solving HFSPs fall into three categories, which are exact methods [15]–[18], constructive heuristics [19]–[24], and improvement heuristics [24]–[33]. Regarding the exact methods, Naderi *et al.* [18] reviewed the shortcoming of the available models in the literature and provided a comparison of four different mathematical models. Although optimal solutions of HFSPs can be obtained using exact methods in the literature, owing to the computational complexity, it is severely painful even for moderately sized HFSPs. Therefore, most decision-makers use constructive and improvement heuristics to find approximate solutions in an acceptable computational time. A constructive heuristic usually starts with an empty permutation list and repeatedly extends the current permutation list until a complete schedule is obtained. Among the constructive heuristics available for HFSPs, the two approaches based on the Nawaz-Enscore-Ham (NEH) heuristic [24], WT1_NEH(x) and WT2_NEH(x), proposed by Kizilay *et al.* [25] are currently the best ones for the HFSP with makespan objective.

On the other hand, an improvement heuristic usually starts with an initial solution and then provides a scheme for iteratively improving the incumbent solution. Among improvement heuristics, studies with meta-heuristics have been extensively applied on the HFSPs. The meta-heuristic is a rather broad-spectrum algorithmic framework that can be applied to different combinatorial optimization problems with minor modifications. Methods of the remarkable approximation algorithms available for HFSPs including: Tabu search [26], immune algorithm [27], ant colony system algorithm [28], particle swarm optimization algorithm [29], iterated greedy algorithm [30], genetic algorithm [31], simulated annealing algorithm [32], hybrid immune algorithm [33], hybrid artificial bee colony algorithm [34], and hybrid particle swarm optimization algorithm [25]. In the past, one product has not required the use of too many masks, and the first-come, first-served (FCFS) method is frequently used. However, with the evolution of production process technology, the monthly number of masks produced has jumped by a factor of approximately two to three. In this situation, solving MDPSP using the FCFS method tends greatly to delay delivery, and causes problems of staff allocation. Therefore, algorithms must be developed to deal with this situation. Since the computational results of above studies showed that approximation algorithms have practical value for solving HFSPs in terms of solution quality, robustness and efficiency, this study presents two high-performance approximation algorithms for solving the MDPSP. The novelty and contributions of this paper are summarized as follows. The considered MDPSP, which is regarded as the most critical stage in the photo-lithography process, is mathematically modeled using an integer linear programming (ILP) model. To meet the practical requirements, two high-performance meta-heuristic algorithms, a simulated annealing (SA) and a genetic algorithm (GA), are also developed for solving this practical but computationally intractable problem. Since scheduling algorithms strongly affect the performance of the photo-lithography process, this work contributes significantly to meeting the practical requirements for solving the MDPSP problem.

The rest of this work is organized as follows. First, the ILP model of the MDPSP is formulated. Second, the two proposed meta-heuristic algorithms are elucidated. Third, the proposed algorithms are empirically evaluated using 768 simulation instances that are based on the characteristics of a real technology company, and their performance is compared with those of the ILP model and the FCFS method. Finally, conclusions are drawn, and recommendations for future research are provided.

## II. ILP MODEL

This section formulates the MDPSP of interest. To simplify the formulation, the following notation is used.

*Indices:*

| | |
|---|---|
| $i$ : | Stage |
| $j$ : | Mask datum |
| $t$ : | Time unit |

*Parameters*:

| | |
|---|---|
| $N$ : | Number of mask data |
| $K$ : | Number of stages |
| $m_i$ : | Number of identical parallel machines in stage $i$ |
| $T$ : | Planning horizon for which the schedule is to be developed |
| $B_j$ : | Release time of mask datum $j$ |
| $D_j$ : | Due date of mask datum $j$ |
| $W_j$ : | Penalty per unit time of tardiness for mask datum $j$ |
| $P_i$ : | Processing time required in stage $i$ for each mask datum |

*Decision variables*:

| | |
|---|---|
| $X_{ijt} = 1$, | if mask datum $j$ is processed in stage $i$ at time $t$; 0, otherwise |
| $S_{ij}$ : | Starting time of mask datum $j$ in stage $i$ |
| $C_{ij}$ : | Completion time of mask datum $j$ in stage $i$ |
| $T_j$ : | Tardiness of mask datum $j$ |

Let mask datum 0 be the dummy initial mask datum. The MDPSP problem can be formulated as follows.

$$Minimize \sum_{j=1}^{N} W_j T_j \tag{1}$$

$$\text{subject to} \sum_{j=1}^{N} X_{ijt} \le m_i, \quad i=1,2,\ldots,K;\ t=1,2,\ldots,T, \tag{2}$$

$$C_{(i-1)j} \le C_{ij} - P_i, \quad i = 2,\ldots,K;\ j = 1,2,\ldots,N, \tag{3}$$

$$C_{ij} - S_{ij} + 1 = P_i, \quad i = 1,2,\ldots,K;\ j = 1,2,\ldots,N, \tag{4}$$

$$\sum_{t=1}^{T} X_{ijt} = P_i, \quad i = 1,2,\ldots,K;\ j = 1,2,\ldots,N, \tag{5}$$

$$S_{ij} \le t + T(1 - X_{ijt}), \quad i = 1,2,\ldots,K;\ j = 1,2,\ldots,N;\ t = 1,2,\ldots,T \tag{6}$$

$$tX_{ijt} \le S_{ij} + P_i - 1, \quad i = 1,2,\ldots,K;\ j = 1,2,\ldots,N;\ t = 1,2,\ldots,T \tag{7}$$

$$S_{1j} \ge B_j, \quad j = 1,2,\ldots,N, \tag{8}$$

$$T_j \ge C_{Kj} - D_j, \quad j = 1,2,\ldots,N, \tag{9}$$

**TABLE 1.** An example of MDPSP data.

| Mask data no. | MG processing time | OPC processing time | TO processing time | Mask data release time | MDP due date | Unit weighted tardiness |
|---|---|---|---|---|---|---|
| 1 | 4 | 0 | 6 | 0 | 10 | 1 |
| 2 | 4 | 10 | 6 | 2 | 20 | 3 |
| 3 | 4 | 0 | 6 | 2 | 12 | 1 |
| 4 | 4 | 0 | 6 | 0 | 12 | 1 |
| 5 | 4 | 10 | 6 | 2 | 22 | 5 |

$$T_j \geq 0, \quad j = 1, 2, \ldots, N, \tag{10}$$

$$X_{ijt} \in \{0, 1\}, \quad i = 1, 2, \ldots, K; \quad j = 1, 2, \ldots, N; \; t = 1, 2, \ldots, T, \tag{11}$$

$$S_{ij} \in \{1, 2, \ldots, T\}, \quad i = 1, 2, \ldots, K; \quad j = 1, 2, \ldots, N, \tag{12}$$

$$C_{i0} = 0, i = 1, 2, \ldots, K, \tag{13}$$

$$C_{ij} \in \{1, 2, \ldots, T\}, \quad i = 1, 2, \ldots, K; \quad j = 1, 2, \ldots, N. \tag{14}$$

The objective function minimizes the total weighted tardiness. Constraint set (2) sets the maximum number of machines that are available in each stage during a period. Constraint set (3) imposes the condition that no mask data task can be begun until the preceding task has been completed. Constraint set (4) is the starting time constraint, which is obtained from the processing time requirement. Constraint set (5) represents the period that each mask data at each stage occupies. Constraint set (6) denotes that, at each stage, each mask data will occupy its required number of processors from its starting time at that stage. Constraint set (7) requires that, in each stage, each mask datum will occupy its required number of machines until its finishing time in that stage. Constraint set (8) confirms that the staring time of the first stage of each mask datum cannot earlier than its release time. Constraint (9) determines the tardiness of mask datum $j$, which is computed as the completion time of mask datum $j$ in the last stage minus its due date ($C_{Kj}-D_j$). Finally, constraints (10)-(14) define the decision variables.

## III. PROPOSED META-HEURISTIC ALGORITHMS

This section briefly describes the representation of the solution to the MDPSP of interest, the calculation of the objective function value, and the elements used in SA and GA for solving the MDPSP.

### A. SOLUTION REPRESENTATION AND CALCULATION OF OBJECTIVE FUNCTION VALUE

In this work, a sequence of mask data is represented by a string of numbers that is a permutation of mask data. Therefore, the first entry in the solution representation is the first mask datum to be processed on the machine of the first stage (MG). Then, from left to right, other mask data are sequentially processed on the available machines of the first stage. In the other two stages, *i.e.*, OPC (if required) and TO, the mask data are dispatched to the available machine according to the First-Come First-Served (FCFS) rule. If a mask datum is scheduled to be processed before its release time, then the process is delayed until that time. Suppose that five mask data are to be processed, and two, one, and two machines are available for MG, OPC, and TO, respectively. Table 1 presents five mask data. Column 1 in Table 1 denotes the number of mask data, while columns 2-4 present the processing times of MG, OPC and TO for each mask datum which is zero if the operation does not have to be performed on the mask data. Columns 5 to 7 present the release time, due date and unit weighted tardiness of the mask data. For example, a solution that is represented as [3 2 1 4 5] can be interpreted as an operating sequence that involves five mask data on the machines of the first stage of 3-2-1-4-5, then the Gantt chart of the schedule is presented in Fig. 2. As listed in Table 2, the total weighted tardiness for this solution is 68 (6*1 + 2*3 + 0*1 + 6*1 + 10*5 = 68).

### B. NEIGHBORHOOD SOLUTION

Given a schedule $\Pi$, the neighborhood solutions of $\Pi$, denoted as $N(\Pi)$,can be randomly generated by the swap and insertion operations. For the swap operation, $N(\Pi)$ is sampled by randomly selecting two mask data and directly switching their positions; for the insertion operation, $N(\Pi)$ is sampled by randomly selecting a mask datum and inserting it immediately before another randomly chosen mask datum. For example, if a new solution $N(\Pi)$ is obtained by swapping the second mask datum and the fifth mask datum of the solution $\Pi$ = [3 2 1 4 5] revealed in Figure 2, then $N(\Pi)$ = [3 5 1 4 2]. If a new solution $N(\Pi)$ is obtained by inserting the fourth mask datum before the second mask datum in the $\Pi$ = [3 2 1 4 5], then $N(\Pi)$ = [3 4 2 1 5].

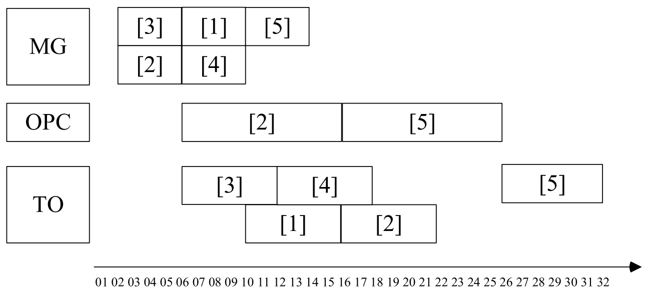

**FIGURE 2.** Gantt chart of an MDPSP solution.

**TABLE 2.** The calculation of an MDPSP solution.

| Mask data no. | MDP due date (Hour) | Complete time (Hour) | Tardiness (Hour) | Unit weighted tardiness | Weighted Tardiness |
|---|---|---|---|---|---|
| 1 | 10 | 16 | 6 | 1 | 6 |
| 2 | 20 | 22 | 2 | 3 | 6 |
| 3 | 12 | 12 | 0 | 1 | 0 |
| 4 | 12 | 18 | 6 | 1 | 6 |
| 5 | 22 | 32 | 10 | 5 | 50 |
| Total | | | | | 68 |

## C. GENETIC ALGORITHM

Holland [35] developed the first genetic algorithm (GA). Following the publication of a text book on GA by Goldberg [36], the range of applications of GAs has increased rapidly. GA has been successfully used to solve many complex combinatorial optimization problems [36]–[38]. The searching procedure of a GA is based on natural selection, which operates on multiple solutions simultaneously and evaluates a fitness function for the corresponding solutions.

```
GA ( $P_{size}$ , $p_c$ , $p_m$ , and $Max_{Time}$ )
  $t = 0$;
  Initialize $P(t)$;
  Evaluate $P(t)$;
  While (Running Time $< Max_{Time}$ )
  {
    Crossover $P(t)$ to yield $C(t)$;
    Evaluate $C(t)$;
    Mutate $C(t)$;
    Evaluate $C(t)$;
    Select $P(t+1)$ from $P(t)$ and $C(t)$;
  }
```

**FIGURE 3.** Pseudo-code of the proposed GA.

The Pseudo-code of the proposed GA is shown in Figure 3. Let $P(t)$ and $C(t)$ represents the parents and offspring in current population $t$. The initial population, chromosome representation, selection scheme, crossover operator, mutation operator, and termination condition of the proposed GA algorithm to be used to solve the MDPSP of interest are described as follows.

- **Chromosome Representation and Initial population**—The chromosome representation is the solution representation as described in Section 3.1. Initial solutions are randomly generated. A collection of $P_{size}$ such individual randomly generated solutions forms a population $P$.
- **Selection Scheme**—The probability that an individual is selected as a parent increases with its corresponding fitness function value. Given a population $P$ and the total weighted tardiness value ($WT_t$) of each individual $t \in P$, the fitness function value of $t$, $f_t$, is given by $f_t = 1/(1 + WT_t)$. The proposed GA uses tournament selection to identify two individuals and to select the one with the higher fitness function value as a parent.
- **Crossover**—Two parents have a probability $p_c$ of undergoing the crossover operation. A newly generated individual has chromosomes that comprise genes from both of its parents. If the total weighted tardiness of the newly generated individual in $C(t)$ is less than that of the parent with the larger total weighted tardiness value, then it will replace that parent; otherwise, the newly generated individual will be eliminated. The proposed GA uses the well-known position-based crossover [39]. The position based crossover is one of the simplest but powerful crossover operators for sequencing problems, and has been applied to solve many optimization problem successfully [40]–[42]. As a rule of thumb, we adapted it in this study.
- **Mutation**—New offspring have a probability $p_m$ of undergoing mutation. If the total weighted tardiness of a mutated offspring is less than the pre-mutation value, then the mutated offspring in $C(t)$ will replace the original offspring. Section 3.2 describes how to generate the mutated individual using neighborhood solution.
- **Termination Condition**—The proposed GA is terminated if the maximal computation time $Max_{Time}$ is reached. $Max_{Time}$ is set to $T_{value} \times n$ seconds, where $T_{value}$ is a parameter value controlling the maximal computation time and $n$ is the number of mask data.

## D. SIMULATED ANNEALING

The simulated annealing (SA) algorithm was introduced by Metropolis *et al.* [43] and popularized by Kirkpatrick *et al.* [44]. SA has been successfully applied to a wide variety of complex combinatorial optimization problems [45]–[48]. The SA algorithm is modeled on the process by which the slow cooling of metal causes good crystallization, while rapid cooling causes poor crystallization. The proposed SA algorithm is described in detail as follows. First, four parameters, $N_{iter}$, $T_0$, $\alpha$, and $Max_{Time}$, are set, where $N_{iter}$ represent the number of iterations at a particular temperature; $T_0$ denotes the initial temperature; $\alpha$ is the coefficient that controls the cooling schedule, and $Max_{Time}$ is the maximal computational time for executing the algorithm. The Pseudo-code of the proposed SA heuristic is shown in Figure 4. The initial current temperature $T$ is set to $T_0$. The solution is represented as described in Section 3.1. Subsequently, an initial solution $\Pi_{incumbent}$ is randomly generated. The current best solution $\Pi_{best}$ is set to $\Pi_{incumbent}$. Let $TWT(\Pi)$ be the

```
SA ( $N_{iter}$ , $T_0$ , $\alpha$ , and $Max_{Time}$ )
Step 1: Generating the initial solution $\Pi_{incumbent}$ randomly;
Step 2: Let $T = T_0$; $N = 0$; $\Pi_{best} = \Pi_{incumbent}$ ;
Step 3: Generating a solution $\Pi_{new}$ based on $\Pi_{incumbent}$ ;
Step 4: $N = N + 1$;
Step 5: If ( $\Delta = TWT(\Pi_{iew}) - TWT(\Pi_{incumbent}) \leq 0$ ) {Let $\Pi_{incumbent} = \Pi_{new}$ ;}
        ELSE {
            Generate $\gamma \sim U(0 \sim 1)$ ;
            If ( $\gamma < E^{-(\Delta/T)}$ ) {Let $\Pi_{new} = \Pi_{incumbent}$ };
        }
Step 6: If ( $TWT(\Pi_{best}) > TWT(\Pi_{incumbent})$ ){
            $\Pi_{best} = \Pi_{incumbent}$ ;
        }
Step 7: If ( $N > N_{iter}$ ){
            $T = T \times \alpha$ ;
            $N = 0$;
        }
Step 8: If (Running Time > $Max_{Time}$ ) {Terminate the SA
            procedure;}
        Else {Go to Step 3;}
```

**FIGURE 4.** Pseudo-code of the proposed SA.

total weighted tardiness of $\Pi$. Set the best objective function value $TWT(\Pi_{best})$ to $TWT(\Pi_{incumbent})$.

The current temperature $T$ is reduced following $N_{iter}$ iterations after the previous temperature reduce using a formula $T \leftarrow \alpha T$, where $0 < \alpha < 1$. In each iteration, a feasible solution $\Pi_{new}$ is generated from $\Pi_{incumbent}$ using $N(\Pi_{incumbent})$, as described in Section 3.2. Let $\Delta$ be the difference between $TWT(\Pi_{incumbent})$ and $TWT(\Pi_{new})$, such that $\Delta = TWT(\Pi_{new}) - TWT(\Pi_{incumbent})$. The probability of replacing $\Pi_{incumbent}$ with $\Pi_{new}$, given that $\Delta > 0$, is $e^{-\Delta/T}$. Replacement is performed by generating a random number $r \in [0, 1]$ and replacing the incumbent solution $\Pi_{incumbent}$ with $\Pi_{new}$ if $r < e^{-\Delta/T}$. Conversely, if $\Delta \leq 0$, then the probability of replacing $\Pi_{incumbent}$ with $\Pi_{new}$ is 100%. The searching process is terminated when the maximal computation time is used and the maximal computation time is calculated in the same way as for the GA.

## IV. COMPUTATIONAL RESULTS AND DISCUSSION

The proposed GA and SA algorithm were coded using C and tested on a personal computer with an Intel Core 2 2.67 GHz CPU, 4GB memory, and Window 10 operating system. The ILP model of the MDPSP was solved using Gurobi 8.1 [49] on that machine.

### A. TEST PROBLEMS

To demonstrate the applicability of the two developed meta-heuristic algorithms in practice, 720 large-scaled problem instances were obtained from a technology company in Taoyuan, Taiwan. To confirm the proposed ILP and to compare with GA and SA, another 48 small-scale problem instances (reduced from the large-scale problem instances) were used. Therefore, a total of 768 test problems were solved.

The parameters of the 720 large-scaled test problems, including the processing time required in stage $i$ for each mask datum ($P_i$) and the numbers of identical parallel machines in stage $i$ ($m_i$) came from the technology company's actual data. The number of mask data ($N$), the OPC ratio (OPCR), the release time ($B_j$), the due date ($D_j$), and the penalty per unit time of tardiness for mask datum $j$ ($W_j$) are generated according to the standard time and real requirement of the company. The processing times for MG and TO are 12 and 24 hours, respectively, while the operating time for OPC, if required for mask data, is 60 hours. According to the company's actual data, the numbers of machines for MG, OPC and TO are three, five, and four, respectively. The number of mask data ($N$) has three levels, 75, 100, and 125. Two OPC ratios (*OPCR*) – 33% (*Low*) and 80% (*High*) – are used, meaning that the number of mask data to be processed in OPC is around 33% or 80% of all the mask data. The Release Time Range (*RTR*) has two levels - *Short*, which is [0, 180] hours, and *Long*, which is [0, 360] hours, where the numbers inside parentheses specify the range of discrete uniform distribution that be used to generate release times. The tightness of the due date (*TDD*) determines the

**TABLE 3. Summarized levels of the five factors for 720 test instances.**

| Factor | Level |
|---|---|
| Mask data Size ($N$) | 75 mask data |
| | 100 mask data |
| | 125 mask data |
| OPC Ratio ($OPCR$) | *Low* (1) |
| | *High* (2) |
| Release Time Range ($RTR$) | *Short* (1) |
| | *Long* (2) |
| Tightness of Due Date ($TDD$) | *Tight* (1) |
| | *Middle* (2) |
| | *Loose* (3) |
| Mask Data Weight Range ($MDWR$) | *Small* (1) |
| | *Large* (2) |

possible available time for handling mask data and is set to *Tight*(Release time + Total Processing Time), *Middle* (Release time + 1.5 × Total Processing time) or *Loose* (Release Time + 2 × Total Processing Time). The mask data weight range ($MDWR$) denotes the range of unit penalties for tardy mask data. There are two levels of $MDWR$: *Small*, which is [1, 5], and *Large*, which is [1, 15], where the numbers inside parentheses denote the range of discrete uniform distribution that be used to generate weights. The values for $B_j$, $D_j$, and $W_j$ are generated using discrete uniform distributions from the intervals of $RTR$, $TDD$, and $MDWR$, respectively. For each combination of problem factors, ten instances are generated. As a result, $3 \times 2 \times 2 \times 3 \times 2 = 72$ combinations of factors and $72 \times 10 = 720$ problem instances are used. Table 3 presents the levels of the five factors.

The 48 small-scale problems are generated as follows. The number of mask data ($N$) has six levels, 6, 8, 10, 12, 14, and 16, in which eight test instances are generated from each level. The $RTR$ is generated from discrete uniform distribution in the range [0, 40]. The operating times for MG and TO are scaled to 6 and 12 hours, respectively. The operating time for OPC, if required, is scaled to 30 hours. The numbers of machines for MG, OPC and TO are two, four, and three, respectively.

### B. PARAMETER VALUE

This subsection discusses the results of computational experiments that are designed for evaluating the performance of the proposed GA and SA. Since parameter setting may influence the performance of each algorithm, 72 new test instances (one test instance for each combination of problem factors) are randomly generated for parameter analysis. Tested parameter combinations for the GA in the analysis are $P_{size} = \{50,100,150,200\}$, $P_c = \{0.6,07,0.8,0.9\}$, and $P_m = \{0.10, 0.20, 0.30, 0.40\}$. Tested parameter combinations for SA in the analysis are $T_0 = \{1, 4, 7, 10\}$, $\alpha = \{0.90, 0.93, 0.95, 0.97\}$, and $N_{iter} = \{1500, 2000, 2500, 3000\}$. Each test instance is solved 10 times.

Since each parameter had four levels, the orthogonal array L16 ($3^4$) is applied in the design of experiment (DOE) [50]. Table 4 shows the 16 combinations of parameter values in the experiment, and the response variable is average improvement rate (AIR), which is computed as follows.

$$AIR_h = \frac{TWT_{\max}^{FCFS} - TWT_{\max}^{h}}{TWT_{\max}^{FCFS}} \times 100\%$$

where $TWT_{\max}^{FCFS}$ and $TWT_{\max}^{h}$ represent the total weighted tardiness values that are obtained using the FCFS method and heuristic $h$, respectively.

Table 5 presents the significance value of each parameter. It shows that $P_m$ and $T_0$ is the most significant parameters for GA and SA, respectively. The Analysis of Variance (ANOVA)

**TABLE 4. Orthogonal array of experiments.**

| | GA | | | SA | | |
|---|---|---|---|---|---|---|
| Experiment No. | $P_{size}$ | $P_c$ | $P_m$ | $T_0$ | $\alpha$ | $N_{iter}$ |
| 1 | 50 | 0.6 | 0.10 | 1 | 0.90 | 1500 |
| 2 | 50 | 0.7 | 0.20 | 1 | 0.93 | 2000 |
| 3 | 50 | 0.8 | 0.30 | 1 | 0.95 | 2500 |
| 4 | 50 | 0.9 | 0.40 | 1 | 0.97 | 3000 |
| 5 | 100 | 0.6 | 0.20 | 4 | 0.90 | 2000 |
| 6 | 100 | 0.7 | 0.10 | 4 | 0.93 | 1500 |
| 7 | 100 | 0.8 | 0.30 | 4 | 0.95 | 2500 |
| 8 | 100 | 0.9 | 0.40 | 4 | 0.99 | 3000 |
| 9 | 150 | 0.6 | 0.30 | 7 | 0.90 | 2500 |
| 10 | 150 | 0.7 | 0.40 | 7 | 0.93 | 3000 |
| 11 | 150 | 0.8 | 0.10 | 7 | 0.95 | 1500 |
| 12 | 150 | 0.9 | 0.20 | 7 | 0.99 | 2000 |
| 13 | 200 | 0.6 | 0.40 | 10 | 0.90 | 3000 |
| 14 | 200 | 0.7 | 0.30 | 10 | 0.93 | 2500 |
| 15 | 200 | 0.8 | 0.20 | 10 | 0.95 | 2000 |
| 16 | 200 | 0.9 | 0.10 | 10 | 0.97 | 1500 |

**TABLE 5. The average relative percent of deviations for each parameter of GA and SA.**

| Heuristic | Level | $P_{size}$ | $P_c$ | $P_m$ |
|---|---|---|---|---|
| GA | 1 | 51.0729 | 51.0066 | 50.6750 |
| | 2 | 50.9841 | 51.0030 | 50.9651 |
| | 3 | 50.9448 | 50.9728 | 51.1097 |
| | 4 | 50.9811 | 51.0005 | 51.2333 |
| | Range | 0.1281 | 0.0338 | 0.5583 |
| | Rank | 51.0729 | 51.0066 | 50.6750 |
| | Level | $T_0$ | $\alpha$ | $N_{iter}$ |
| SA | 1 | 51.9017 | 51.8794 | 51.9684 |
| | 2 | 51.8195 | 51.9173 | 51.9461 |
| | 3 | 51.9967 | 51.9442 | 51.9523 |
| | 4 | 52.0742 | 52.0510 | 51.9252 |
| | Range | 0.2547 | 0.1716 | 0.0432 |
| | Rank | 1 | 2 | 3 |

of parameters is shown in Tables 6 and 7. Table 6 reveals that both $P_m$ and $P_{size}$ has an impact on the solution quality of GA, while Table 7 shows that $T_0$ has an impact on the solution quality of SA. According to the parameter analysis, in this study, the parameter values of GA are set as $P_{size} = 50$, $P_c = 0.6$, and $P_m = 0.4$, and the parameter values of SA are set as $T_0 = 10$, $N_{iter} = 1500$, and $\alpha = 0.97$.

### C. COMPARING RESULTS OF ILP SOLVED BY GUROBI SOLVER, GA AND SA

The proposed ILP model is optimally solved using the Gurobi 8.1, a state-of-the-art mathematical programming solver. In this study, the planning horizon ($T$) is set to $T = \max_{j=1,\ldots,N}\{B_j\} + NP_1/m_1 + N_{opc}P_2/m_2 + NP_3/m_3$, where $N_{opc}$ is the number of mask data required OPC. Notably, for the ILP model, the initial lower bound (LB) of the objective function value (total weighted tardiness) is zero, and the initial upper bound (UB) of the objective function value is infinite. When the ILP model is solved using Gurobi 8.1, the LB and UB will be replaced according to the solver and the incumbent solution. Once the objective function value of the incumbent solution equals the current LB, the optimal solution is obtained and the solver is terminated. The solver is terminated after one hour if it had not found the optimal solution.

Table 8 presents the solutions and the computing times for 48 small problem instances. The Gurobi solver finds optimal solutions to 29 out of the 48 problems, as shown in bold in Table 8. The GA and SA also yield the same 29 optimal solutions. For the other 19 small-scale problems, which do not have known optimal solutions, both the GA and SA find better solutions than the Gurobi solver. The reason is that the addressed MDPSP is NP-hard in the strong sense, so the performance of the ILP model is significantly reduced when the number of musk data is greater than 12. To measure the degree of the improvement, the overall average improving rate ($AIR$) is used. $AIR$ is computed as $(TWT^{MILP}_{\max} - TWT^{h}_{\max})/TWT^{MILP}_{\max} \times 100\%$ where $TWT^{MILP}_{\max}$ and $TWT^{h}_{\max}$ denote the total weighted tardiness values that are obtained by Gurobi solver and the heuristic $h$, respectively. The $AIR$s of the GA and SA are 8.293% and 8.238%, respectively. In summary, the proposed GA and SA heuristics yield better solutions in less time than the Gurobi solver. Many factors

**TABLE 6. ANOVA for parameters of GA.**

| Source | DF | SS | MS | F | P |
|---|---|---|---|---|---|
| $P_{size}$ | 3 | 26 | 8.667 | 6.032 | <0.001 |
| $P_c$ | 3 | 2 | 0.667 | 0.487 | 0.693 |
| $P_m$ | 3 | 499 | 166.333 | 117.397 | <0.001 |
| *Block* | 71 | 1381364 | 19455.831 | 11733.962 | <0.001 |
| Error | 11439 | 16206 | 1.4167 | | |
| Total | 11519 | | | | |

**TABLE 7. ANOVA for parameters of SA.**

| Source | DF | SS | MS | F | P |
|---|---|---|---|---|---|
| $T_0$ | 3 | 106 | 35.333 | 5.268 | 0.001 |
| $\alpha$ | 3 | 47 | 16.667 | 2.321 | 0.073 |
| $N_{iter}$ | 3 | 3 | 1.000 | 0.137 | 0.938 |
| *Block* | 71 | 1798346 | 25328.827 | 3760.158 | <0.001 |
| Error | 11439 | 76750 | 6.710 | | |
| Total | 11519 | | | | |

**TABLE 8.** Computational results for the 48 small problem instances.

| No | N | Gurobi 8.1 | | | GA | | SA | | No | N | Gurobi 8.1 | | | GA | | SA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Obj. | LB | Time | Obj. | Time | Obj. | Time | | | Obj. | LB | Time | Obj. | Time | Obj. | Time |
| 1 | 6 | **36** | **36** | **1.8** | **36** | 1.2 | **36** | 1.2 | 25 | 12 | **156** | **156** | **2683.8** | 156 | 2.4 | 156 | 2.4 |
| 2 | 6 | **99** | **99** | **2.8** | **99** | 1.2 | **99** | 1.2 | 26 | 12 | 661 | 404 | 3600.2 | 641 | 2.4 | 641 | 2.4 |
| 3 | 6 | **16** | **16** | **0.6** | **16** | 1.2 | **16** | 1.2 | 27 | 12 | **122** | **122** | **1330.9** | 122 | 2.4 | 122 | 2.4 |
| 4 | 6 | **39** | **39** | **1.3** | **39** | 1.2 | **39** | 1.2 | 28 | 12 | **278** | **278** | **69.6** | 278 | 2.4 | 278 | 2.4 |
| 5 | 6 | **138** | **138** | **34.1** | **138** | 1.2 | **138** | 1.2 | 29 | 12 | **311** | **311** | **674.6** | 311 | 2.4 | 311 | 2.4 |
| 6 | 6 | **11** | **11** | **0.4** | **11** | 1.2 | **11** | 1.2 | 30 | 12 | 1305 | 747 | 3600.3 | 1131 | 2.4 | 1131 | 2.4 |
| 7 | 6 | **42** | **42** | **1.9** | **42** | 1.2 | **42** | 1.2 | 31 | 12 | 382 | 270 | 3600.3 | 370 | 2.4 | 370 | 2.4 |
| 8 | 6 | **81** | **81** | **0.9** | **81** | 1.2 | **81** | 1.2 | 32 | 12 | **421** | **421** | **2406.1** | 421 | 2.4 | 421 | 2.4 |
| 9 | 8 | **73** | **73** | **15.9** | **73** | 1.6 | **73** | 1.6 | 33 | 14 | 252 | 178 | 3600.3 | 252 | 2.8 | 252 | 2.8 |
| 10 | 8 | **61** | **61** | **1.9** | **61** | 1.6 | **61** | 1.6 | 34 | 14 | 501 | 253 | 3600.2 | 493 | 2.8 | 469 | 2.8 |
| 11 | 8 | **9** | **9** | **1.1** | **9** | 1.6 | **9** | 1.6 | 35 | 14 | 175 | 109 | 3600.2 | 171 | 2.8 | 171 | 2.8 |
| 12 | 8 | **0** | **0** | **0.8** | **0** | 1.6 | **0** | 1.6 | 36 | 14 | **433** | **433** | **1583.5** | 433 | 2.8 | 433 | 2.8 |
| 13 | 8 | **43** | **43** | **15.4** | **43** | 1.6 | **43** | 1.6 | 37 | 14 | 844 | 396 | 3600.4 | 697 | 2.8 | 697 | 2.8 |
| 14 | 8 | **168** | **168** | **20.6** | **168** | 1.6 | **168** | 1.6 | 38 | 14 | 1745 | 825 | 3600.3 | 1407 | 2.8 | 1407 | 2.8 |
| 15 | 8 | **10** | **10** | **0.8** | **10** | 1.6 | **10** | 1.6 | 39 | 14 | 197 | 157 | 3600.3 | 195 | 2.8 | 195 | 2.8 |
| 16 | 8 | **109** | **109** | **4.5** | **109** | 1.6 | **109** | 1.6 | 40 | 14 | 1900 | 761 | 3600.4 | 1260 | 2.8 | 1260 | 2.8 |
| 17 | 10 | **148** | **148** | **146.1** | **148** | 2.0 | **148** | 2.0 | 41 | 16 | 640 | 198 | 3600.2 | 491 | 3.2 | 491 | 3.2 |
| 18 | 10 | **115** | **115** | **21.1** | **115** | 2.0 | **115** | 2.0 | 42 | 16 | 1591 | 597 | 3600.2 | 1384 | 3.2 | 1410 | 3.2 |
| 19 | 10 | **24** | **24** | **2.2** | **24** | 2.0 | **24** | 2.0 | 43 | 16 | 230 | 110 | 3600.2 | 209 | 3.2 | 209 | 3.2 |
| 20 | 10 | **638** | **638** | **80.5** | **638** | 2.0 | **638** | 2.0 | 44 | 16 | 524 | 263 | 3600.2 | 524 | 3.2 | 524 | 3.2 |
| 21 | 10 | 484 | 270 | 3600.3 | 466 | 2.0 | 466 | 2.0 | 45 | 16 | 1029 | 284 | 3600.4 | 622 | 3.2 | 622 | 3.2 |
| 22 | 10 | **601** | **601** | **59.9** | **601** | 2.0 | **601** | 2.0 | 46 | 16 | 1036 | 602 | 3600.2 | 892 | 3.2 | 978 | 3.2 |
| 23 | 10 | **71** | **71** | **72.4** | **71** | 2.0 | **71** | 2.0 | 47 | 16 | 228 | 194 | 3600.3 | 216 | 3.2 | 216 | 3.2 |
| 24 | 10 | **732** | **732** | **279.2** | **732** | 2.0 | **732** | 2.0 | 48 | 16 | 2289 | 2045 | 3600.4 | 2146 | 3.2 | 2146 | 3.2 |

**TABLE 9.** Performance comparison between GA and SA.

| | MAX($N, OPTR, RTR, TDD, MDWR$) | | AVE($N, OPTR, RTR, TDD, MDWR$) | | MIN($N, OPTR, RTR, TDD, MDWR$) | | BS($N, OPTR, RTR, TDD, MDWR$) | | CPU time (s) |
|---|---|---|---|---|---|---|---|---|---|
| Problem class | *GA* | *SA* | *GA* | *SA* | *GA* | *SA* | *GA* | *SA* | *GA and SA* |
| $(75,*,*,*,*)$ | 56.64/56.93/57.15 | 57.92/58.00/58.03 | 55.74/56.13/56.41 | 56.29/56.45/56.53 | 54.78/55.30/55.65 | 54.22/54.56/54.71 | 114/138/162 | 162/175/186 | 15/30/60 |
| $(100,*,*,*,*)$ | 51.04/51.22/51.45 | 51.90/51.99/52.10 | 50.46/50.66/50.94 | 50.66/50.79/50.97 | 49.76/50.02/50.31 | 49.34/49.45/49.73 | 68/91/145 | 107/136/168 | 20/40/80 |
| $(125,*,*,*,*)$ | 47.26/47.37/47.40 | 47.63/47.69/47.70 | 46.82/47.00/47.05 | 47.01/47.10/47.13 | 46.33/46.59/46.67 | 46.30/46.44/46.49 | 24/73/120 | 75/124/158 | 25/50/100 |
| $(*,Low,*,*,*)$ | 55.03/55.41/55.73 | 56.74/56.89/56.99 | 53.83/54.32/54.72 | 54.57/54.80/54.99 | 52.55/53.17/53.62 | 52.08/52.43/52.72 | 28/51/130 | 115/185/251 | 20/40/80 |
| $(*,High,*,*,*)$ | 48.26/48.27/48.27 | 48.23/48.23/48.23 | 48.18/48.21/48.22 | 48.07/48.09/48.1 | 48.03/48.10/48.13 | 47.83/47.87/47.90 | 178/251/297 | 229/250/261 | 20/40/80 |
| $(*,*,Short,*,*)$ | 46.94/47.01/47.07 | 47.03/47.05/47.06 | 46.65/46.75/46.83 | 46.63/46.65/46.66 | 46.29/46.44/46.56 | 46.14/46.16/46.18 | 87/141/205 | 170/217/244 | 20/40/80 |
| $(*,*,Long,*,*)$ | 56.35/56.67/56.94 | 57.93/58.07/58.16 | 55.36/55.78/56.10 | 56.01/56.24/56.43 | 54.29/54.82/55.19 | 53.77/54.13/54.44 | 119/161/222 | 174/218/268 | 20/40/80 |
| $(*,*,*,Tight,*)$ | 46.22/46.29/46.37 | 46.08/46.14/46.17 | 45.67/45.79/45.89 | 44.69/44.76/44.84 | 45.07/45.27/45.39 | 43.08/43.15/43.31 | 68/100/156 | 105/134/153 | 20/40/80 |
| $(*,*,*,Middle,*)$ | 52.36/52.59/52.76 | 53.36/53.50/53.56 | 51.69/51.94/52.15 | 52.18/52.36/52.48 | 50.92/51.23/51.54 | 50.82/51.10/51.24 | 60/92/138 | 115/149/170 | 20/40/80 |
| $(*,*,*,Loose,*)$ | 56.35/56.64/56.88 | 58.00/58.05/58.10 | 55.66/56.06/56.36 | 57.09/57.22/57.31 | 54.88/55.4/55.71 | 55.97/56.19/56.38 | 78/110/133 | 124/152/189 | 20/40/80 |
| $(*,*,*,*,Small)$ | 48.49/48.67/48.83 | 49.47/49.53/49.57 | 47.78/48.08/48.29 | 48.46/48.55/48.62 | 46.97/47.42/47.70 | 47.16/47.33/47.42 | 120/160/212 | 200/240/276 | 20/40/80 |
| $(*,*,*,*,Large)$ | 54.81/55.00/55.18 | 55.49/55.59/55.65 | 54.23/54.45/54.64 | 54.18/54.34/54.47 | 53.61/53.85/54.05 | 52.75/52.97/53.20 | 86/142/215 | 144/195/236 | 20/40/80 |
| $(*,*,*,*,*)$ | 51.65/51.84/52.00 | 52.48/52.56/52.61 | 51.00/51.26/51.47 | 51.32/51.44/51.55 | 50.29/50.63/50.88 | 49.96/50.15/50.31 | 206/302/427 | 344/435/512 | 20/40/80 |

may influence the computing time; they include, for example, CPU speed, memory size, operating system, compiler, and the computer program. The proposed GA and SA heuristics take no more than 3.2 seconds to solve small-scale problems, which the Gurobi solver takes much longer to solve.

## D. COMPARING RESULTS OF FCFS, SA AND GA

To validate the performance of the GA and SA, computational experiments were carried out on the large problem set. Given the computational complexity of the MDPSP, optimal solutions to large problems are typically not readily obtainable. Therefore, for each benchmark instance of the large problem set, the *AIR* values of the GA and SA algorithms were calculated.

Table 9 presents the statistical average *AIR*s that were obtained for the large problem set using the GA and SA. This table presents the instances grouped by the values of Mask data Size ($N$), OPC Ratio ($OPCR$), Release Time Range ($RTR$), Tightness of Due Date ($TDD$) and Mask Data Weight Range ($MDWR$). The statistical results that correspond to the three $T_{value}$ values are separated by a slash ($T_{value}$ = 0.2/0.4/0.8). Since each problem is solved five times, the maximum (MAX), average (AVE) and minimum (MIN) *AIRs* are computed for each problem.

The performances of the proposed GA and SA meta-heuristics were compared using the four indices: MAX($N$, $OPTR$, $RTR$, $TDD$, $MDWR$), AVE($N$, $OPTR$, $RTR$, $TDD$, $MDWR$), MIN($N$, $OPTR$, $RTR$, $TDD$, $MDWR$), and BS($N$, $OPTR$, $RTR$, $TDD$, $MDWR$) which represents the maximum, average, minimum values of *AIRs* for the problem classes that are characterized by *N, OPCR, RTR, TDD*, and *MDWR,* while *BS* denotes the number of best solutions that is identified by the GA and SA heuristics to problems in the class that is characterized by *N, OPCR, RTR, TDD*, and *MDWR*.

As seen in Table 9, the total MAX($N$, $OPTR$, $RTR$, $TDD$, $MDWR$), AVE($N$, $OPTR$, $RTR$, $TDD$, $MDWR$), MIN($N$, $OPTR$, $RTR$, $TDD$, $MDWR$), and BS($N$, $OPTR$, $RTR$, $TDD$, $MDWR$) obtained by the proposed GA heuristic were 51.65/51.84/52, 50.29/50.63/50.88, 50.29/50.63/50.88, and 202/285/387, respectively. The corresponding values obtained by SA were 52.48/52.56/52.61, 51.32/51.44/51.55, 49.96/50.15/50.31, and 344/435/512, so both the proposed GA and SA heuristics significantly outperform the FCFS method.

Evidently, a greater computing time enable larger *AIRs* to be obtained by both the GA and SA. As the number of mask data increases, the rates of improvement that are provided by both the GA and SA are reduced, perhaps because of the NP-hard property of MDPSP, only the linear time used in the GA and the SA are not enough. Accordingly, more computing time is required as the number of mask data is increased.

The *AIRs* at a lower OPC ratio are better, perhaps because the bottleneck in a schedule is the OPC operation with a high OPC ratio, so mask data are prone to delay. Evidently, the *AIRs* with a shorter release time are better. Since more mask data must wait for their release times when the mask data have a longer range of release times, the processing of these mask data are more likely to be finished later, and the total weighted penalties (the denominator of *AIR*) is larger, so the *AIR* divided by FCFS is smaller.

The tightness of due date affects the *AIR*. With a looser due date, more mask data are unlikely to be delayed, and smaller total weighted penalties (the denominator of *AIR*) are obtained. Therefore, the *AIR* with a loose *TDD* is better than that with a tight and a middle *TDD*. Evidently, the improvement rates achieved using a smaller *MDWR* are better. Since the weight values are large, the total weighted penalties (the denominator of *AIR*) are also large and the *AIR* divided by FCFS is smaller. These computational results clearly demonstrate that the proposed GA and SA heuristics improve upon the solutions that are obtained using the traditional (FCFS) method. The experimental results clearly reveal that this work reduces the gap between theoretical progress and industrial practice.

## V. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

Mask data preparation is part of the mask data processing in the production of semiconductors, and its importance has commonly been neglected. This work concerns the scheduling problem in mask data processing, which is regarded as the most critical stage in the photo-lithography process. To reduce the gap between theoretical progress and industrial practice, this work develops an ILP model of the mask data processing scheduling problem and develops two meta-heuristic algorithms - a simulated annealing (SA) algorithm and a genetic algorithm (GA) - for solving the MDPSP with total weighted tardiness. Experimental results reveal that the GA and SA produce higher-quality MDPSP solutions than the ILP model and the traditional FCFS method.

The MDPSP is a challenging issue with practical applications. Related problems, such as those with different objective functions or a different number of machines in each stage, can be solved in the future. Future research may also attempt to use other meta-heuristics or to hybridize other algorithms to solve the MDPSP.

## REFERENCES

[1] Y.-J. Park and H.-R. Hwang, "A rule-based simulation approach to scheduling problem in semiconductor photolithography process," in *Proc. IEEE 8th Int. Conf. Intell. Syst., Theories Appl. (SITA)*, May 2013, pp. 1–4.

[2] E. Akcalt, K. Nemoto, and R. Uzsoy, "Cycle-time improvements for photolithography process in semiconductor manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 14, no. 1, pp. 48–56, Feb. 2001.

[3] Y.-J. Sha, L.-F. Hsieh, and S.-H. Lin, "Study on wafer rework strategies and dispatching rules at the photolithography stage," *J. Chin. Inst. Ind. Eng.*, vol. 20, no. 5, pp. 457–464, 2003.

[4] A. Bitar, S. Dauzère-Pérès, C. Yugma, and R. Roussel, "A memetic algorithm to solve an unrelated parallel machine scheduling problem with auxiliary resources in semiconductor manufacturing," *J. Scheduling*, vol. 19, no. 4, pp. 367–376, 2016.

[5] S. H. Kim and Y. H. Lee, "Synchronized production planning and scheduling in semiconductor fabrication," *Comput. Ind. Eng.*, vol. 96, pp. 72–85, Jun. 2016.

[6] X. Ma and Y. Li, "Resolution enhancement optimization methods in optical lithography with improved manufacturability," *Proc. SPIE*, vol. 10, no. 2, 2011, Art. no. 023009.

[7] C.-Y. Hung *et al.*, "Integrated post tape outflow for fast design to mask turn-around time," *Proc. SPIE* vol. 5992, Nov. 2005, Art. no. 599251.

[8] E. Yang, C. H. Li, S. J. Park, Y. Zhu, and E. Guo, "An optimized OPC and MDP flow for reducing mask write time and mask cost," *Proc. SPIE*, vol. 7823, Sep. 2010, Art. no. 78233E.

[9] R. Morgan, M. Chacko, D. Hung, J. Yeap, and M. Boman, "Integration of OPC and mask data preparation for reduced data I/O and reduced cycle time," *Proc. SPIE*, vol. 6730, Nov. 2007, Art. no. 67304E.

[10] P. Schiavone *et al.*, "A novel mask proximity correction software combining accuracy and reduced writing time for the manufacturing of advanced photomasks," *Proc. SPIE*, vol. 8441, Jun. 2012, Art. no. 84411F.

[11] A. Marques, C. Miramond, and O. Ndiaye, "Masks data preparation flow for advanced technology nodes," *Proc. SPIE*, vol. 8441, Jun. 2012, Art. no. 84410S.

[12] J. Wan *et al.*, "Building block modular OPC recipes for productivity improvement in modern IC manufacturing flows," in *Proc. China Semiconductor Technol. Int. Conf. (CSTIC)*, Mar. 2016, pp. 1–3.

[13] R. H. Huang, T. H. Yu, and C. L. Yang, "Multiprocessor flow shop scheduling problem with common due window," *Appl. Mech. Mater.*, vols. 284–287, pp. 3712–3716, Jan. 2013.

[14] I. Ribas, R. Leisten, and J. M. Framiñan, "Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective," *Comput. Oper. Res.* vol. 37, no. 8, pp. 1439–1454, 2010.

[15] S. A. Brah and J. L. Hunsucker, "Branch and bound algorithm for the flow shop with multiple processors," *Eur. J. Oper. Res.*, vol. 1, no. 1, pp. 88–99, 1991.

[16] T. Kis and E. Pesch, "A review of exact solution methods for the non-preemptive multiprocessor flowshop problem," *Eur. J. Oper. Res.*, vol. 164, no. 3, pp. 592–608, 2005.

[17] R. Ruiz and J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *Eur. J. Oper. Res.*, vol. 205, no. 1, pp. 1–18, 2010.

[18] B. Naderi, S. Gohari, and M. Yazdani, "Hybrid flexible flowshop problems: Models and solution methods," *Appl. Math. Model.*, vol. 38, pp. 5767–5780, Dec. 2014.
[19] J. L. Hunsucker and J. R. Shah, "Comparative performance analysis of priority rules in a constrained flow shop with multiple processors environment," *Eur. J. Oper. Res.* vol. 72, no. 1, pp. 102–114, 1994.
[20] C. Oğuz and M. F. Ercan, "Scheduling multiprocessor tasks in a two-stage flow-shop environment," *Comput. Ind. Eng.*, vol. 33, nos. 1–2, pp. 269–272, 1997.
[21] C. Oğuz, M. F. Ercan, T. C. E. Cheng, and Y. F. Fung, "Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop," *Eur. J. Oper. Res.* vol. 149, no. 2, pp. 390–403, 2003.
[22] G.-C. Lee, Y.-D. Kim, J.-G. Kim, and S. H. Choi, "A dispatching rule-based approach to production scheduling in a printed circuit board manufacturing system," *J. Oper. Res. Soc.*, vol. 54, no. 10, pp. 1038–1049, 2003.
[23] K.-C. Ying and S.-W. Lin, "Scheduling multistage hybrid flowshops with multiprocessor tasks by an effective heuristic," *Int. J. Prod. Res.* vol. 47, no. 13, pp. 3525–3538, 2009.
[24] M. Nawaz, Jr., E. E. Enscore, Jr., and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.
[25] D. Kizilay, M. F. Tasgetiren, Q.-K. Pan, and L. Wang, "An iterated greedy algorithm for the hybrid flowshop problem with makespan criterion," in *Proc. IEEE Symp. Comput. Intell. Prod. Log. Syst. (CIPLS)*, Dec. 2014, pp. 16–23.
[26] C. Oğuz, Y. Zinder, V. H. Do, A. Janiak, and M. Lichtenstein, "Hybrid flow-shop scheduling problems with multiprocessor task systems," *Eur. J. Oper. Res.*, vol. 152, no. 1, pp. 115–131, 2004.
[27] M. Zandieh, S. M. T. F. Ghomi, and S. M. M. Husseini, "An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times," *Appl. Math. Comput.*, vol. 180, no. 1, pp. 111–127, 2006.
[28] K.-C. Ying and S.-W. Lin, "Multiprocessor task scheduling in multistage hybrid flow-shops: An ant colony system approach," *Int. J. Prod. Res.*, vol. 44, no. 16, pp. 3161–3177, 2006.
[29] C.-T. Tseng and C.-J. Liao, "A particle swarm optimization algorithm for hybrid flow-shop scheduling with multiprocessor tasks," *Int. J. Prod. Res.*, vol. 46, no. 17, pp. 4655–4670, 2008.
[30] K.-C. Ying, "An iterated greedy heuristic for multistage hybrid flowshop scheduling problems with multiprocessor tasks," *J. Oper. Res. Soc.*, vol. 60, no. 6, pp. 810–817, 2009.
[31] W. Besbes, J. Teghem, and T. Loukil, "Scheduling hybrid flow shop problem with non-fixed availability constraints," *Eur. J. Ind. Eng.*, vol. 4, no. 4, pp. 413–433, 2010.
[32] H.-M. Wang, F.-Der Chou, and F.-C. Wu, "A simulated annealing for hybrid flow shop scheduling with multiprocessor tasks to minimize makespan," *Int. J. Adv. Manuf. Technol.*, vol. 53, nos. 5–8, pp. 761–776, 2011.
[33] K.-C. Ying, "Minimising makespan for multistage hybrid flowshop scheduling problems with multiprocessor tasks by a hybrid immune algorithm," *Eur. J. Ind. Eng.* vol. 6, no. 2, pp. 199–215, 2011.
[34] S.-W. Lin, K.-C. Ying, and C.-Y. Huang, "Multiprocessor task scheduling in multistage hybrid flowshops: A hybrid artificial bee colony algorithm with bi-directional planning," *Comput. Oper. Res.*, vol. 40, no. 5, pp. 1186–1195, 2013.
[35] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI, USA: Univ. of Michigan Press, 1975.
[36] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.
[37] J. W. Kim, "Candidate order based genetic algorithm (COGA) for constrained sequencing problems," *Int. J. Ind. Eng.*, vol. 23, no. 1, pp. 1–12, 2016.
[38] M.-C. Wu, C.-S. Lin, C.-H. Lin, and C.-F. Chen, "Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems," *Comput. Oper. Res.*, vol. 80, pp. 101–112, Apr. 2017.
[39] G. Syswerda, "Uniform Crossover in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. Schaffer, Ed. San Mateo, CA, USA: Morgan Kaufmann, 1989, pp. 2–9.
[40] S.-W. Lin, K.-C. Ying, and Z.-J. Lee, "Metaheuristics for scheduling a non-permutation flowline manufacturing cell with sequence dependent family setup times," *Comput. Oper. Res.*, vol. 36, no. 4, pp. 1110–1121, 2009.
[41] K.-C. Ying, Z.-J. Lee, C.-C. Lu, and S.-W. Lin, "Metaheuristics for scheduling a no-wait flowshop manufacturing cell with sequence-dependent family setups," *Int. J. Adv. Manuf. Technol.*, vol. 58, nos. 5–8, pp. 671–682, 2012.
[42] P. Błażej, M. Wnętrzak, and P. Mackiewicz, "The role of crossover operator in evolutionary-based approach to the problem of genetic code optimization," *Biosystems*, vol. 150, pp. 61–72, Dec. 2016.
[43] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J. Chem. Phys.* vol. 21, no. 6, pp. 1087–1092, 1953.
[44] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[45] C.-C. Wu, J.-C. Chen, W.-H. Wu, P.-H. Hsu, and W.-H. Wu, "Simulated annealing algorithms for the two-machine makespan flowshop scheduling with truncated learning consideration," *Int. J. Ind. Eng.*, vol. 18, no. 8, pp. 432–443, 2011.
[46] S.-W. Lin and V. F. Yu, "A simulated annealing heuristic for the multiconstraint team orienteering problem with multiple time windows," *Appl. Soft Comput.* vol. 37, pp. 632–642, Dec. 2015.
[47] S.-W. Lin and K.-C. Ying, "A multi-point simulated annealing heuristic for solving multiple objective unrelated parallel machine scheduling problems," *Int. J. Prod. Res.*, vol. 53, no. 4, pp. 1065–1076, 2015.
[48] S.-W. Lin, C.-Y. Huang, K.-C. Ying, and D.-L. Chen, "Decreasing the system testing makespan in a computer manufacturing company," *IEEE Access*, vol. 6, pp. 16464–16473, 2018.
[49] *Gurobi Optimization*. Accessed: Aug. 17, 2016. [Online]. Available: http://www.gurobi.com
[50] D. C. Montgomery, *Design and Analysis of Experiments*, 8th ed. Hoboken, NJ, USA: Wiley, 2012.

**KUO-CHING YING** is currently a Distinguished Professor with the Department of Industrial Engineering and Management, National Taipei University of Technology. His research interests focus on the operations scheduling and combinatorial optimization, in which he has published over 100 academic research papers in refereed international journals, such as *Applied Intelligence*, *Applied Soft Computing*, *Computers and Operations Research*, *Computers and Industrial Engineering*, the *European Journal of Industrial Engineering*, the *European Journal of Operational Research*, the IEEE ACCESS, the *International Journal of Production Economics*, the *International Journal of Advanced Manufacturing Technology*, the *International Journal of Innovational Computing, Information and Control*, the *International Journal of Production Research*, the *Journal of the Operational Research Society*, *OMEGA–The International Journal of Management Sciences*, *Production Planning and Control*, and *Transportation Research Part E: Logistics and Transport Review*, among others. He is the (Senior) Editor of ten international journals.

**SHIH-WEI LIN** received the bachelor's, master's, and Ph.D. degrees in industrial management from the National Taiwan University of Science and Technology, Taiwan, in 1996, 1998, and 2000, respectively. He is currently a Professor with the Department of Information Management, Chang Gung University, Taiwan. He is also with the Department of Neurology, Linkou Chang Gung Memorial Hospital, Taoyuan, Taiwan, and with the Department of Industrial Engineering and Management, Ming Chi University of Technology, Taipei, Taiwan. His papers have appeared in *Computers and Operations Research*, the *European Journal of Operational Research*, the *Journal of the Operational Research Society*, the *European Journal of Industrial Engineering*, the *International Journal of Production Research*, the *International Journal of Advanced Manufacturing Technology*, *Knowledge and Information Systems*, *Applied Soft Computing*, *Applied Intelligence*, and *Expert Systems with Applications*. His current research interest includes meta-heuristics and data mining.

**CHIEN-YI HUANG** received the Ph.D. degree from the State University of New York, Binghamton, in 1996. He served as the Chief Process Technology Head of Wistron Corporation, where he was responsible for new process technology enabling and materials characterization. He is currently a Professor with the National Taipei University of Technology, Taipei, Taiwan. His research interests include process optimization and electronics reliability.

**MEMPHIS LIU** received the E.M.B.A. degree from the School of Business, Chang Gung University, in 2015. He has more than 12 years of working experience in streamlining operations and increasing efficiency. He currently serves as the CAD Leader of the Department of Mask, Nanya Technology, Taiwan. His current research interest includes scheduling.

**CHIA-TIEN LIN** received the E.M.B.A. degree from the School of Business, Chang Gung University, in 2012. He has more than 20 years of working experience in information system development and management. He currently serves as the Senior Manager of the Department of Information Management, Linkou Chang Gung Memorial Hospital, Taiwan. His current research interests include project management and scheduling.