An Algorithm of Multi-Subpopulation Parameters With Hybrid Estimation of Distribution for Semiconductor Scheduling With Constrained Waiting Time

Hung-Kai Wang, Chen-Fu Chien, *Member, IEEE*, and Mitsuo Gen

***Abstract*—Scheduling for wafer fabrication of advanced technology nodes entails complicated constraints such as limited waiting times. Focusing on real settings, this paper aims to develop a novel genetic algorithm of multi-subpopulation parameters with hybrid estimation of distribution (MSPHEDA) to solve the present problem effectively and efficiently. To estimate the validity of this approach, ten scenarios were simulated on the basis of empirical data as the basis to compare the performance of MSPHEDA and other heuristic methods for minimizing makespan and reducing the total exceeded limited waiting time. The results have shown practical viability of the proposed approach.**

***Index Terms*—Hybrid estimation of distribution, limited waiting time constraint, multi-subpopulation, semiconductor manufacturing scheduling.**

I. Introduction

Semiconductor critical dimension shrinks rapidly via advanced equipment driven by Moore's Law [1]. Thus, semiconductor manufacturing continuously migrates to advanced technology nodes that are increasingly complicated [2]–[4]. Effective job scheduling to enhance tool productivity and reduce production cycle time is critical for maintaining competitive advantage [5], [6]. Mönch *et al.* [7] classified scheduling problems into six types: batching problems, problems with auxiliary resources, multiple orders per job problems, scheduling of cluster tools, scheduling approaches for individual work areas, and job-shop problems. Indeed, the semiconductor manufacturing scheduling problem (SMSP) is an expanded model of the flexible job-shop scheduling problem (FJSP) that is more complicated owing to additional constraints such as limited waiting times [6]–[8]. However, few studies have addressed the present problem for advanced wafer fabrication [7], [8].

As manufacturing contexts change rapidly, the scheduling solution must be able to obtain a near-optimal solution within a short time. This study aims to propose an efficient algorithm that can obtain an effective solution with shorter limited time than previous studies [6]–[8]. Conventional mixed-integer linear programming (MILP) model cannot obtain a solution quickly in practice [9]. Metaheuristics have been developed to solve the scheduling problems. However, as the constraints increase, the efficiency of metaheuristics may be affected [10].

Considering limited waiting time constraints in real settings, this study aims to develop a novel genetic algorithm of multi-subpopulation parameters with hybrid estimation of distribution to solve the SMSP effectively. The present SMSP is a single stage scheduling problem for unrelated parallel machines with dedications. The proposed approach is validated via simulations of ten scenarios on the basis of realistic ion implantation process data and practical manufacturing situations including those related to process defects, new products, ramping up, urgent jobs from customers, and abnormal events in advanced fabs. The results have shown practical viability of the proposed approach.

The rest of this paper is organized as follows: Section II reviews related studies on the SMSP and evolutionary algorithms. Section III formulates an MINLP model of the present problem. Section IV presents the proposed algorithm. Section V compares the proposed method with existing metaheuristics via simulations of different scenarios for validation. Section VI concludes with a discussion of contributions and future research directions.

II. Literature Review

A. SMSP

Mathematical programming models have been developed to solve scheduling problems. For instance, Jia *et al.* [11] proposed a slack-based mixed-integer programming model

Manuscript received October 11, 2014; revised April 12, 2015; accepted May 25, 2015. Date of publication June 1, 2015; date of current version July 31, 2015. This work was supported in part by the Ministry of Science and Technology under Grant NSC 100-2628-E-007-017-MY3 and Grant MOST 103-2218-E-007-023, in part by the Semiconductor Technologies Empowerment Partners Consortium under Grant NSC 102-2622-E-007-013, in part by the Advanced Manufacturing and Service Management Research Center under Grant 104N2074E1, in part by the Taiwan Semiconductor Manufacturing Company, Hsinchu Science Park, Taiwan under Grant 102A26, and in part by the Japan Society of Promotion of Science: Grant-in-Aid for Scientific Research under Grant 24510219.0001.

H.-K. Wang and C.-F. Chien are with the Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 30013, Taiwan (e-mail: cfchien@mx.nthu.edu.tw).

M. Gen is with both Fuzzy Logic Systems Institute and Research Institute for Science and Technology, Tokyo University of Science, Japan.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSM.2015.2439054

0894-6507 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

for the scheduling of reentrant batch-processing machine with incompatible job families in wafer fab. However, their study considered only up to eight job families. Klemmt and Mönch [12] addressed the SMSP as a flow-shop scheduling problem (FSP) with waiting time constraints and proposed a mixed-integer programming model with a scheduling and decomposition approach. They considered the time constraints across several stages that can be nested. However, it is hard to estimate the waiting time between different stages and thus accurately handle the SMSP between several stages for modern Giga-fabs. For a medium-size problem, Aguirre *et al.* [13] developed an MILP model for wet-etch operation scheduling that can achieve near-optimal solutions within reasonable time. Yan *et al.* [14] used a branch-and-cut method with convex hull analyses for two-phase lithography machine scheduling that can obtain near-optimal solutions within reasonable time. Jung *et al.* [15] integrated an MILP model into a real-time dispatcher for diffusion process scheduling with time window constraint. Since it will take several hours for MILP model to obtain an optimal solution for the full size problem, they firstly assigned a small number of runs to the tools to reduce the problem scope and then extended the solutions during several iterations. However, this study only compared the proposed MILP with historical dispatching rule, without comparing with other heuristics that can schedule all the jobs at one time.

Simulation has been employed to determine scheduling rules. Qiao *et al.* [16] simulated a minifab and employed a hierarchical colored timed Petri net and an extended genetic algorithm for scheduling. However, simulation approaches have limitations in real settings of modern Giga-fabs.

Evolutionary algorithms [17] have been developed to address the large-size SMSP. For example, Wu and Chien [18] developed a GA for the scheduling of semiconductor final testing and implement it in practice. Song *et al.* [19] developed an ant colony optimization to minimize the total unsupported demand and machine conversion time of a bottleneck station in semiconductor assembly and testing facilities. Driessel and Mönch [20] considered the constraints of parallel machines, sequence-dependent setup times, and the precedence and ready time of jobs, and proposed a variable neighborhood search approach to minimize the total weighted tardiness. Chien *et al.* [21] developed an evolutionary approach for the rehabilitation patient scheduling that is a hybrid shop. Dauzere-Peres and Mönch [22] used a random key genetic algorithm to solve a single-machine batch scheduling for diffusion.

Attar *et al.* [23] solved a multi-objective hybrid flexible flow shop (HFFS) problem considering limited waiting times and machine setup time constraints. However, their study assumed no machine breakdowns and maintenance and negligible transportation time between stages and unlimited intermediate storages that are not realistic in a practice. Moreover, unlike HFFS, the SMSP is an FJSP in which each job has different routes with reentrant processes. Indeed, few studies have addressed the SMSP with practical constraints such as limited waiting time constraints. Chien and Chen [8] developed a batch sequencing genetic algorithm to solve the scheduling of oxide-nitride-oxide (ONO) stacked film fabrication processes, with consideration of waiting time constraints, frequency-based setups, and capacity preoccupation.

B. EDA

Unlike the crossover and mutation in conventional GA, the EDA [24] produces a population over multiple generations to obtain improved solutions, via acquiring the probability distribution from current elitist solutions and sampling new solutions based on the probability distribution. After sampling, the probability distribution evolves with previous probability distribution and current elitist solutions. According to variable dependency, the EDA can be classified as univariate, bivariate, and multivariate distribution models. The population-based incremental learning [25] and univariate marginal distribution algorithm [26] belong to the univariate distribution model. Mutual information maximization for input clustering [27] and bivariate marginal distribution algorithm [28] belong to the bivariate distribution model. Bayesian optimization algorithm [29] belongs to the multivariate distribution model.

Since EDA adapts and evolves from improved solutions and probability distribution, EDA is more statistically valid than conventional GA to obtain superior solutions [30], [31]. However, the EDA has limited exploitation ability [32]. Thus, a number of studies have integrated EDA with local search methods to enhance its exploitation ability. For example, Wang *et al.* [32] divided one population into two subpopulations to perform a local search independently, and combined two subpopulations when no improvement occurred for specific generations. Gao *et al.* [33] used two types of bottleneck shifting methods for neighborhood search. Gao *et al.* [34] proposed the variable neighborhood descent algorithm for the FJSP. The EDA has been applied to various problems such as FJSP [30], FSP [35], single-machine scheduling problem [36], traveling salesman problem [37], and multiobjective resource-constrained project scheduling [38]. However, few studies have employed the EDA to address the SMSP.

III. Mixed-Integer Nonlinear Programming Model

A. Characteristics of the SMSP

An MINLP model is formulated for the present problem with the following constraints:

1) Limited Waiting Time Constraints: The advanced wafer fabrication consists of processing stages with limited waiting time constraints. For example, the nitride layer should be deposited within a strictly defined waiting time after the polysilicon oxide layer is deposited since the surfaces of polysilicon oxide and nitride layers are unstable [7].

2) Machine Status and Recovery Time Constraints: The machine status changes rapidly due to scheduled preventive maintenance or abnormal shutdown, while equipment engineers will repair and recover it for rescheduling.

3) Different Processing Times for Jobs in Different Machines: One job contains at most 25 wafers in one lot with a specific recipe by a machine containing the conditions such as temperature, pressure, humidity, and the chemical elements.

TABLE I
DATA FORM OF THE SMSP

Job	Pieces	Recipe	Weight	Arrival time (hours)	Limited waiting time (hours)	Available machine (processing time /hours)
J1	13	ABC123	5	0.3	0.5	M3 (1.5) M4 (0.8) M5 (2.6) M6 (1.3) M8 (0.7) M9 (1.8)
J2	5	LMN456	1	0	1.5	M1 (0.3) M2 (1.6) M4 (1.9)
J3	25	XYZ789	7	1.8	3.6	M3 (0.9) M6 (1.4) M10 (2.1)
J4	21	ABC518	3	0.2	5.5	M7 (3.2)
J5	18	QRS576	10	3	4.5	M8 (1.7) M10 (2.5)

Different jobs fabricated with the corresponding recipes in different machines have different processing times.

4) Different Setup Time Between Jobs: The machine requires a setup time between the jobs with different recipes.

5) Different Arrival Time: The jobs being processed in the previous stage that will arrive in the near future and the jobs queued in this stage are considered in the forthcoming schedule.

Table I lists the data of an example in fab. One job represents a number of wafers in a lot in the front opening unified pod (FOUP). A FOUP may contain different numbers of wafers up to the full capacity of 25 wafers. The processing time of each job is affected by the number of wafers, the recipes, and the machines. The more wafers in a job, the longer processing time required. Each job is conducted by a recipe. The weighting of a job is assigned based on its importance for production control. The arrival time indicates the job arriving time at this stage.

B. MINLP Formulation

Keha *et al.* [39] modeled a single machine scheduling problem and Unlu and Mason [40] modeled a parallel machine scheduling problems as both sequence position based and time indexed based. The sequence position based formulation with binary decision variables x_{ijk} that are 1 if job j is on position i on machine k, to avoid the nonlinearity with two multiplied decision variable. This study considers real constraints such as job arrival time, machine recovery time, and limited waiting time, in which the time indexed based formulation with two multiplied decision variable, job weight and job completion time can handle these constraints directly. To estimate the validity of the proposed approach, an MINLP model was formulated considering the constraints of operation allocation, process time, machine recovery time, machine capacity, job arrival time, limited waiting time, and setup time to optimize the allocation and sequencing of jobs. The indices, parameters, and decision variables are listed as follows:

Indices:

i, k	index of jobs, $i, k = 1, 2, \dots, n$
j	index of machines, $j = 1, 2, \dots, m$

Parameters:

n	total number of jobs
m	total number of machines
ω_i	priority of job i
p_{ij}	processing time of job i in machine j
s_{ik}	setup time between job i and job k
a_i	arrival time of job i
t_j^{r}	recovery time of machine j
t_i^{w}	limited waiting time of job i
P	penalty function when exceeding the limited waiting time
w	coefficient of penalty function
M	a large positive integer number
$\boldsymbol{E}_j$	the set of jobs that can be processed in machine j, i.e., the available jobs for machine j.

Decision variables:

x_{ij}	$x_{ij} = 1$ if job i is processed in machine j; otherwise, $x_{ij} = 0$
t_{ij}^{b}	process beginning time of job i in machine j
t_{ij}^{c}	process completion time of job i in machine j
z_{ikj}	$z_{ikj} = 1$ if job i precedes job k, which are both processed in machine j; otherwise, $z_{ikj} = 0$
T_i^{w}	total time of job i exceeding t_i^{w}

The objective function (1) is to minimize the weighted sum of completion time for all the jobs and the penalty of total exceeded limited waiting time for all jobs. Since different jobs have different priority, the domain experts set the objective as job weight multiplied by job completion time to ensure early processing of urgent jobs. A number of multiobjective approaches such as Non-dominated Sorting GA-II (NSGA-II) [41] and NSGA-III [42] can be employed. Based on realistic needs, this study combined two objectives into a single objective with a penalty P when limited waiting time constraints are violated [43]. In practice, minimizing the total exceeded time of the limited waiting time is more important than minimizing the weighted sum of completion time. However, the fabs need to follow a schedule, even when some jobs violate limited waiting times. This study models the penalty as soft constraint to avoid infeasible solutions.

Objective function:

$$\text{Minimize} Z = \sum_{i=1}^{n} \sum_{j=1}^{m} \omega_i t_{ij}^{\mathrm{c}} x_{ij} + P \tag{1}$$

subject to

1) Operation Allocation Constraints:

$$\sum_{j=1}^{m} x_{ij} = 1, \quad \forall i \tag{2}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \tag{3}$$

where x_{ij} is a binary variable that assigns each job i to machine j, where each job i will be processed in only one machine j.

2) Process Time Constraints:

$$t_{ij}^{\mathrm{b}} + p_{ij} x_{ij} = t_{ij}^{\mathrm{c}}, \quad \forall i \in \mathbf{E}_j \tag{4}$$

$$t_{ij}^{\mathrm{b}}, t_{ij}^{\mathrm{c}} \geq 0, \quad \forall i, j \tag{5}$$

Equation (4) is the process time constraint that determines the process beginning time of each job i, where its process beginning time plus processing time should equal to its completion time in each machine. Equation (5) restricts the process beginning time and completion time of each job i to greater than or equal to zero (i.e., nonnegative restriction).

3) Machine Recovery Time Constraints:

$$t_{ij}^{\mathrm{b}} + \mathrm{M}(1 - x_{ij}) \geq t_j^{\mathrm{r}}, \quad \forall i \in \mathbf{E}_j \tag{6}$$

Equation (6) restricts the process beginning time of each job i in each machine j to be greater than or equal to the recovery time of machine j. In other words, one job can be processed in the machines that are recovered.

4) Machine Capacity Constraints:

$$t_{ij}^{\mathrm{b}} + p_{ij}x_{ij} - \mathrm{M}(1 - z_{ikj}) \leq t_{kj}^{\mathrm{b}}, \quad \forall i, k \in \mathbf{E}_j, i \neq k \tag{7}$$

$$t_{kj}^{\mathrm{b}} + p_{kj}x_{kj} - \mathrm{M}(1 - z_{kij}) \leq t_{ij}^{\mathrm{b}}, \quad \forall i, k \in \mathbf{E}_j, i \neq k \tag{8}$$

$$z_{ikj} + z_{kij} = x_{ij} * x_{kj}, \quad \forall i, k \in \mathbf{E}_j, i \neq k \tag{9}$$

$$z_{ikj} \in \{0, 1\}, \quad \forall i, k, j \tag{10}$$

Equations (7) to (10) define all the preceding constraints for job i and job k. In other words, one machine can process only one job at a time.

5) Job Arrival Time Constraints:

$$t_{ij}^{\mathrm{b}} + \mathrm{M}(1 - x_{ij}) \geq a_i, \quad \forall i \in \mathbf{E}_j \tag{11}$$

A job can be processed in a machine only when the job has arrived. Equation (11) restricts the beginning time of each job i in machine j to be greater or equal to the arrival time of job i.

6) Limited Waiting Time Constraints:

$$T_i^{\mathrm{w}} = \max\left\{t_{ij}^{\mathrm{c}} - t_i^{\mathrm{w}}, 0\right\}, \quad \forall i \in E_j \tag{12}$$

$$P = w\sum_{i=1}^{n} T_i^{\mathrm{w}} \tag{13}$$

The total exceeded time of job i equals its completion time in machine j minus the limited waiting time of job i; otherwise, the total exceeded time of job i equals zero, as in (12), which is a soft constraint. If (12) is violated, the objective value will have a penalty to minimize the total exceeded time, as in (13). The coefficient of penalty function w can be determined based on realistic needs.

7) Setup Time Constraints:

$$t_{ij}^{\mathrm{b}} + p_{ij}x_{ij} + s_{ik} - \mathrm{M}(1 - z_{ikj}) \leq t_{kj}^{\mathrm{b}}, \quad \forall i, k \in \mathbf{E}_j, i \neq k \tag{14}$$

$$t_{kj}^{\mathrm{b}} + p_{kj}x_{kj} + s_{ki} - \mathrm{M}(1 - z_{kij}) \leq t_{ij}^{\mathrm{b}}, \quad \forall i, k \in \mathbf{E}_j, i \neq k \tag{15}$$

The setup time is different between different jobs. A job can be processed only after the machine is set up. Equations (14) and (15) show that the process completion time of job i plus the setup time between job i and subsequent job k should be less than or equal to the beginning time of job i in each machine j.

```
procedure: MSPHEDA
input: problem dataset, MSPHEDA parameters
output: the current optimal schedule
begin
  t ← 0;
  while (not terminating condition) do            // For each subpopulation
    initialize subPop($t$) by encoding and $P(t)$;   // subPop: subpopulation
                                                  // $P(t)$: initial probability
    evaluate subPop($t$) by decoding and keep the best solution;
    while (not cooperative condition) do
      parameters setting in each subpopulation;
      divide each chromosome into part I and part II;
      select superior supPop(t) from subPop($t$) by elitist routine;
      update $P(t+1)$ by supPop($t$) and $P(t)$;
      create part I of $C(t)$ by sampling from $P(t+1)$; //$C(t)$: offspring
      create part II of $C(t)$ by crossover routine from subPop($t$);
      create $C(t)$ by mutation routine from subPop($t$);
      evaluate $C(t)$ by decode routine;
      produce subPop($t+1$) by C($t$) & subPop($t$) and update best solution;
      while (not solution improved within $T_{LS}$ generations) do
        create $C(t)$ by local search routine;
      end
      while (not solution improved within $T_{ELS}$ generations) do
        create $C(t)$ by exhaustive local search routine;
      end
      $t \leftarrow t + 1$;
    end
    coordinate superior chromosomes & solutions between subpopulations;
  end
output the current optimal schedule
end
```

Fig. 1. Procedure of the MSPHEDA.

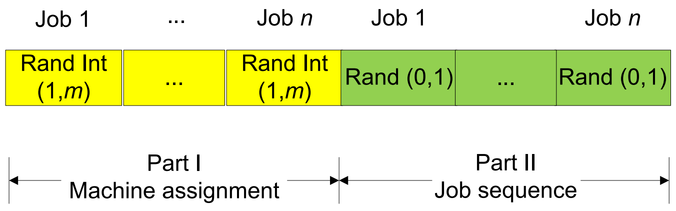

Fig. 2. Chromosome representation.

IV. Multi-Subpopulation Parameters With Hybrid Estimation of Distribution Algorithm

To solve the SMSP effectively and efficiently, the proposed algorithm of multi-subpopulation parameters with hybrid estimation of distribution (MSPHEDA) designed chromosome representation, a decoding procedure, and a univariate EDA model combining cooperative subpopulation with parameter settings and a local search mechanism as shown in Fig. 1.

A. Chromosome Representation

The designed chromosome consists of two parts, in which a random key sequencing [44] is employed in both parts. Part I represents the machine assignment, where the length is the number of jobs, and each gene represents each job on the same site. For each gene, an integer number from one to total number of machines m is used to assign each job to its available machines. The relationship between the chromosome and gene is shown in Fig. 2. However, each job is not available for every machine. To avoid infeasible solutions of machine assignment, the integer number does not represent the number of machines to be assigned, and is thus normalized between zero and one for decoding. The reason for using

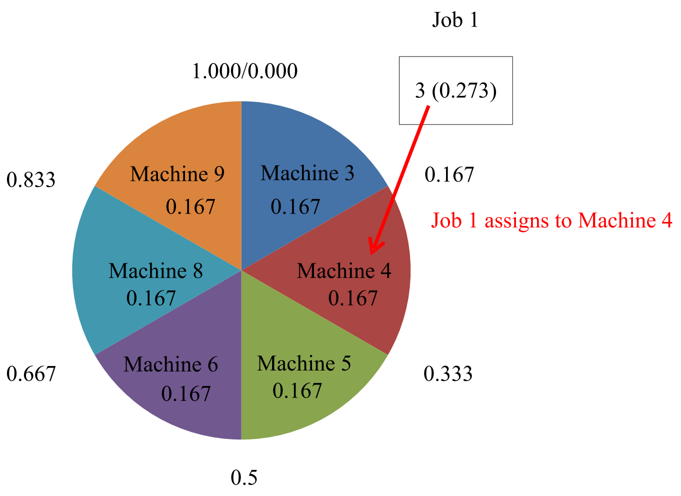

Fig. 3. Example of machine assignment.

an integer number instead of a random number between zero and one is because the appearance time of every integer in the EDA probability model is calculable. Part II represents the job sequence, where the length of the chromosome is the number of jobs, and each gene represents the sequence processing in the same machine. Each gene is produced by a random number from zero to one that determines the job sequence according to the genes in Part II.

B. Initialize the Probability Matrix and Initial Population

All genes in Part I are independent and thus univariate EDA model can be used to save computational time. An element $P_{ij}(t)$ of the probability matrix $P(t)$ represents the probability of job i assigned to machine j. Equation (16) initializes the probability matrix $P(t)$, representing the uniform distribution of job assignments. The initial population is generated as follows:

$$P_{ij}(1) = 1/m, \quad \forall i, j \tag{16}$$

C. Decoding Procedure and Evaluation

Each gene is decoded by assigning each job to its available machines. For example, there are five jobs and ten machines in Table I. For Job 1, there are six available machines; the probability of assigning Job 1 to each machine is $1/6 = 0.1667$. Equation (17) normalizes the gene number g_i in part I of the chromosome to a normalized number Nor_i between zero and one to assign its available machines. In this example, the total number of machines m is ten. If the gene number of Job 1 is three, it can be normalized to 0.272 by (17) to assign Job 1 to Machine 4 through roulette wheel selection in Fig. 3.

$$Nor_i = g_i/(m+1), \quad \forall i \tag{17}$$

When the machine assignment is complete, the genes in part II are used to prioritize the processing sequence of the jobs in the same machine. After the machine assignment and job sequencing, the jobs are arranged into a timetable to determine the process beginning time for all the jobs. The following constraints are considered to ensure that the processing time of each job is feasible.

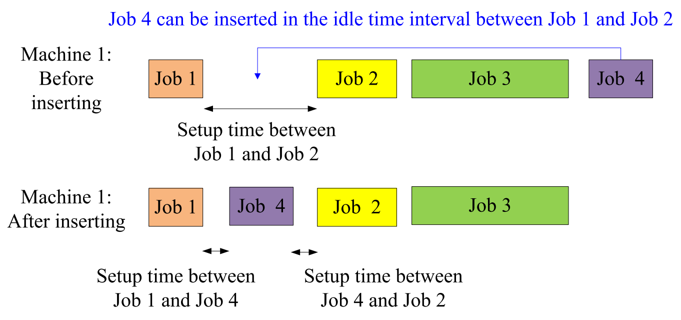
Job 4 can be inserted in the idle time interval between Job 1 and Job 2
Machine 1: Before inserting
Job 1
Job 2
Job 3
Job 4
Setup time between Job 1 and Job 2
Machine 1: After inserting
Job 1
Job 4
Job 2
Job 3
Setup time between Job 1 and Job 4
Setup time between Job 4 and Job 2

Fig. 4. Example of inserting jobs into a previous idle time interval.

1) Machine Recovery Time: All jobs can be arranged only after the machine available time, as shown in (6).

2) Arrival Time of Each Job: All jobs are arranged after their arrival time; if no jobs arrive at a specific time, the machines will be idle to wait for jobs, as shown in (11).

3) Setup Time Between Jobs: All jobs can be processed only after the machines are set up. Therefore, one job will be arranged after the completion time of its previous job plus the setup time between them, as shown in (14) and (15).

4) Inserting Jobs Into the Idle Time Interval Between Previous Jobs: As illustrated in Fig. 4, assume that a long setup time exists between Jobs 1 and 2. If the jobs after Job 2 can be inserted in this interval, the machine idle time can be reduced and the machine utilization can be enhanced. All the inserting conditions of all the jobs in all the machines are considered to complete the decoding procedure. Job 3 cannot be inserted because its process time is longer than this interval, while Job 4 can be inserted since the summation of the setup time between Jobs 1 and 4, its process time, and the setup time are shorter than this interval.

D. Parameter Settings in Each Subpopulation

The multi-subpopulation employs several subpopulations to prevent a single population from jumping into local optima. These subpopulations evolve separately and are cooperated with each other after certain generations [45].

For the present problem, efficiency is critical for implementing the algorithm in real settings. Thus, the proposed MSPHEDA employs different parameters in each subpopulation to enhance efficiency. Numerous studies have shown that a high crossover rate and high elitist selection lead to rapid convergence [10]. However, a high mutation rate and high rank selection cause the solution to diversify and converge slowly [25]. The proposed approach can balance the global search ability and local search ability in each subpopulation. For Subpopulation 1, a high crossover rate and low mutation rate with high elitist selection and low rank selection are used. For Subpopulation 2, a middle crossover rate and middle mutation rate with middle elitist selection and middle rank selection are used. For Subpopulation 3, a low crossover rate and high mutation rate with low elitist selection and high rank selection are used.

E. Select the Superior Subpopulation

The superior subpopulation is firstly selected to calculate the superior appearance matrix $A(t)$ as in (18) to calculate the

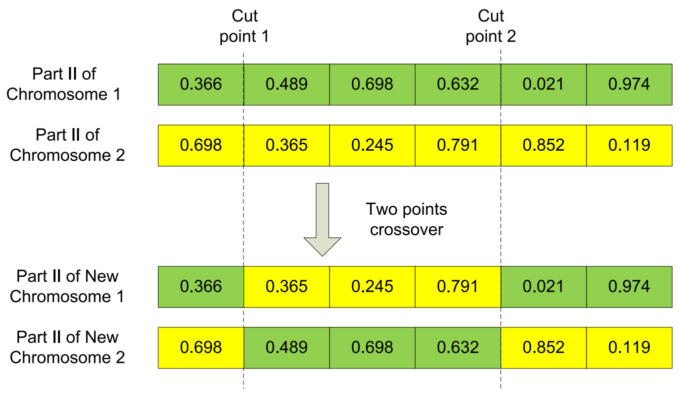

Fig. 5. Two-cut points crossover in part II of the chromosomes.

appearance time of each gene from the superior population. S_P denotes the total number of superior populations. δ_{ij}^s is an indicator vector of superior population s.

$$A_{ij} = \frac{1}{S_P}\sum_{s=1}^{S_P}\delta_{ij}^s, \quad \forall i, j$$

where

$$\delta_{ij}^s = \begin{cases} 1, & \text{if job } i \text{ assigns to position } j \text{ of total machines} \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

F. Update the Probability Matrix

Equation (19) is used to update the probability matrix for the next generation $P(t+1)$ that is the weighted sum of the probability matrix $P(t)$ and superior appearance matrix $A(t)$, where α is the weight of $A_{ij}(t)$ for the learning rate of the superior population.

$$P_{ij}(t+1) = (1-\alpha)P_{ij}(t) + \alpha A_{ij}(t), \quad \forall i, j,$$
$$\text{where}$$
$$0 \le \alpha \le 1 \tag{19}$$

G. Create New Part I and Part II Offspring

The EDA creates new offspring by sampling from the probability matrix. For each gene in part I of the chromosome, the roulette wheel is used to sample an integer number from probability matrix $P(t+1)$. However, the univariate EDA model cannot be used in part II of the chromosome since its genes are not integer numbers. Two-cut points crossover to generate new offspring that randomly select two chromosomes and two cutting points to cut each chromosome into three segments, and then exchange every gene in the middle segment between the two chromosomes, as shown in Fig. 5.

H. Create New Offspring by Mutation

A probability model and updating mechanism perform effectively in early generations. However, when the solution falls into local optima in later generations, all the populations may become identical. It is difficult to leave local optima only by sampling from a probability matrix or by crossover. Thus, two-cut point mutation is used by randomizing one gene in each part of the chromosome to assist the current solution jumping out of local optima to enhance solution quality.

I. Produce Superior Subpopulation for the Next Generation

After evaluating the new offspring, the next subpopulation is produced from the offspring and current subpopulation. The improved solutions are updated to the next generation. Percent of elitist selection P_{ES} and percent of rank selection P_{RS} are used. Elitist selection reserves the current optimal solutions from the superior subpopulations, thereby ensuring the continuous improvement of the solution and also maintaining the convergent speed. Rank selection selects the solutions from nonelitists to maintain the diversity of the subpopulations. That is, the MSPHEDA maintains the exploitation ability via elitist selection and the exploration ability via rank selection.

J. Local Search

The proposed approach designed local search mechanism to enhance the EDA exploitation ability [32] as follows.

1) Beginning Time: When the solution does not improve for consecutive T_{LS} generations, all the chromosomes become identical and thus cannot generate different new offspring. Thus, a local search mechanism will be initiated to diversify the chromosomes to possibly enhance the solutions.

2) Identifying the Critical Machine: The critical machine is the largest objective value among all the machines with the largest loading. When switching one job from the critical machine to another feasible machine, the objective value Z in (1) is likely reduced.

3) Switching the Machine Assignment and Job Sequence: Select one job from the critical machine, assign it to another feasible machine, and replace its job sequence with a random number between zero and one.

4) Stopping Criteria: Retain the obtained superior solution and select the next job in the critical machine until all the jobs have been selected.

K. Exhaustive Local Search

In later generations, all the chromosomes may converge, causing the solutions into local optima. When the superior solution is not improved for consecutive T_{ELS} generations, where T_{ELS} is greater than T_{LS} indicating a local search without improvement, an exhaustive local search should be used.

The term "exhaustive" means to select one job at one time from the first job to the last job, assign it from Machine 1 to machine m, and randomly generate its job sequence to assess whether it obtains an enhanced solution. When an enhanced solution is obtained, retain it and continue to assess other machine assignments and job sequences until all the jobs have been searched. Although an exhaustive local search is time consuming, it is an effective mechanism to find an improved solution at a later stage. When all the solutions have been searched, the current optimal solution will be replaced into other subpopulations to ensure that all the subpopulations are searched on the basis of the current optimal solution.

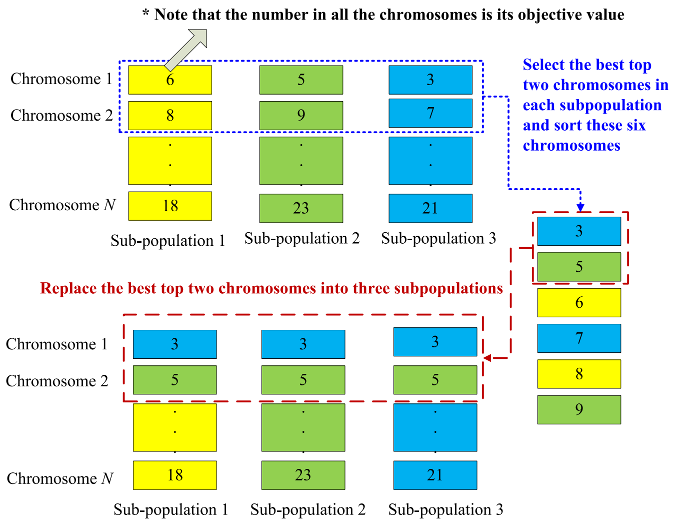
* Note that the number in all the chromosomes is its objective value
Chromosome 1
Chromosome 2
Chromosome N
6
5
3
8
9
7
18
23
21
Sub-population 1
Sub-population 2
Sub-population 3
Select the best top two chromosomes in each subpopulation and sort these six chromosomes
3
5
6
7
8
9
Replace the best top two chromosomes into three subpopulations
3
5
18
23
21

Fig. 6. Example of coordinating two chromosomes between three subpopulations.

L. Coordinate Chromosomes Between Subpopulations

In conventional metaheuristics, using only one population may converge into local optima. The proposed MSPHEDA employs a multi-subpopulation N_{pop} in which each subpopulation evolves independently to increase diversity. During specific generations T_{EX}, the best top N_{best} chromosomes in each subpopulation are selected. After sorting the solutions of $N_{\text{best}} * N_{\text{pop}}$ chromosomes, the N_{best} chromosomes replace the chromosomes of each subpopulation. Fig. 6 illustrates an example with N_{pop} equal to three and N_{best} equal to two. After incorporating the superior chromosomes, each subpopulation evolves continuously for T_{EX} generations.

V. Validation

To validate the proposed MSPHEDA, several small-scale problems and a large problem in a real setting were employed for comparison. For the small problems, the LINGO software was used to solve the formulated MINLP models to obtain the optimal solutions for comparison with those of the proposed MSPHEDA. For the large problem in real setting, the MSPHEDA was compared with other metaheuristics such as the GA, particle swarm optimization (PSO), artificial bee colony (ABC), and EDA. Each simulation was implemented using C++ program on Windows 7 X64 with Intel(R) Xeon(R) CPU E3-1230-V2 at 3.30GHz and an 8 GB RAM processor.

A. Experimental Result of Small Problems

The following five small problems were designed for comparison: Problem 1 has seven jobs and three machines; Problem 2 has ten jobs and three machines; Problem 3 has ten jobs and five machines; Problem 4 has 15 jobs and three machines; and Problem 5 has 15 jobs and five machines.

Equation (20) defines the optimality gap to denote the deviation percentage of the solution of the MSPHEDA from the optimal solution of the MINLP obtained using LINGO:

$$\text{Optimality gap}(\%) = \frac{\text{MSHEDA} - \text{MINLP}}{\text{MINLP}} \times 100\% \qquad (20)$$

TABLE II
Result of LINGO and the MSPHEDA

Job #	Machine #	Objective value		Optimality gap	CPU time (unit: second)	
		MINLP (LINGO)	MSPHEDA (C++)	MSPHEDA (C++)	MINLP (LINGO)	MSPHEDA (C++)
7	3	846	846	0%	31560	0.29
10	3	1925	1925	0%	50520	0.52
10	5	911	911	0%	67841	0.61
15	3	-	4439	-	>86400	1.09
15	5	-	1196	-	>86400	1.13

TABLE III
Example Data of Ten Jobs and Three Machines

Job	Recipe	Weight	Processing time in machine 1 (minutes)	Processing time in machine 2 (minutes)	Processing time on machine 3 (minutes)
J1	A	1	6	9	M
J2	A	2	21	23	51
J3	D	3	65	M	48
J4	B	4	M	9	31
J5	B	5	M	13	76
J6	B	6	22	12	M
J7	B	7	29	23	38
J8	C	8	M	28	30
J9	A	9	31	27	33
J10	A	10	12	39	51

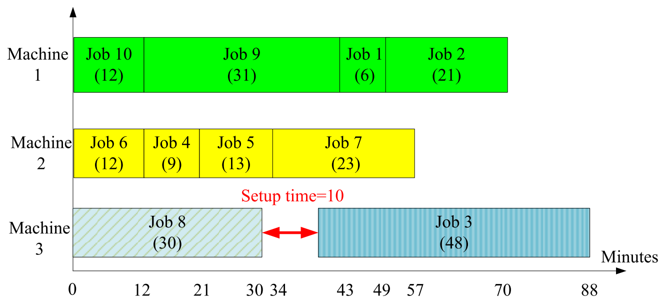

Fig. 7. Gantt chart of the example of ten jobs and three machines.

Domain experts determine that the setting of w equals to 1000 in (13) for both MINLP and MSPHEGA. The remaining parameters of MSPHEDA are identical to those listed in Table IX. Table II lists the objective value, optimality gap, and CPU time of LINGO and the MSPHEDA. The CPU time of LINGO is more than 8 hours in Problems 1, 2 and 3. The MSPHEDA obtains the same solutions within 2 seconds. In Problems 4 and 5, LINGO cannot obtain the optimal solution within 1 day.

Table III shows the example data of ten jobs and three machines. Without loss of generality, all ten jobs are 25 pieces, arriving at time 0, with the same limited waiting time of 120 minutes. M denotes a large processing time to prevent the jobs being processed in unavailable machines. The setup time between two jobs with different recipes is 10 minutes.

For the example in Table III, both the MINLP and MSPHEDA obtain the same optimal solution shown in Table II. Fig. 7 illustrates the Gantt chart of the optimal scheduling result. Identical backgrounds of the jobs indicate identical recipes. Jobs 10, 9, 1, and 2 are processed in Machine 1 because they all use Recipe A. Jobs 6, 4, 5, and 7 are processed in Machine 2 since they all use Recipe B.

A 10 minute setup time exists between Job 8 and 3 in Machine 3 with different recipes.

The job with a higher weight divided by a shorter processing time has a higher priority on the same machine. For example, in machine 1, Job 10 ($10/22 = 0.833$) has a higher priority than Job 9 ($9/31 = 0.29$); Job 9 has a higher priority than Job 1 ($1/6 = 0.167$); and Job 1 has a higher priority than Job 2 ($2/21 = 0.095$). The result is the same for Machines 2 and 3.

B. Experimental Design of Practical Problems

The proposed MSPHEDA was validated in a realistic project sponsored by a leading semiconductor manufacturing company in Taiwan. The data was collected from an ion implantation process. For confidentiality, all the continuous empirical data such as processing times, setup times, and weights are fitted and transformed with normal distribution $N(\mu, \sigma^2)$, and all the discrete empirical data such as recipes, number of pieces, and available machines are fitted and transformed with Bernoulli distribution. Without loss of generality, all the specific terms regarding the machines, jobs, and recipes are replaced by general terms. The following practical scenarios were investigated.

1) Normal Situation: Scenario 1 denotes a normal case in fab. Normal machine breakdown and maintenance rates are 10%, and these machines are repaired by equipment engineers and returned to production within a specific time.

2) Process Defect Effect: Scenarios 2, 3, and 4 exhibit a severe process defect in the previous stage, causing all the jobs to stack up and queue for a long time. The process defect effect reduces the remaining waiting time and increases the concentrated job arrival time, causing it to exceed that of the normal situation, thereby increasing constraints for the scheduling problem.

Scenario 2 assumes that the jobs that arrive at this stage have a shorter remaining waiting time than that of the normal situation. In particular, the remaining waiting time of each job is set as 60 minutes. Thus, each job should be processed within 60 minutes or it will become a defect.

Scenario 3 assumes that all the machines are prohibited producing any jobs until the defect is removed. The arrival time has half the standard deviation of that in the normal situation, causing the jobs to arrive in an increased concentrated time.

Scenario 4 combines Scenarios 2 and 3, while the remaining waiting time of each job is 60 minutes when it arrives, and each job has half the standard deviation of the arrival time compared with the normal situation.

3) Machine Process Time Effect: Scenario 5 assumes that the variation of the machine process time is double that of the normal situation. That is, advanced machines are released for production, creating a diverse process time between the existing and newly released machines. In other words, new machine selection and installation affect the scheduling and production.

4) Urgent Jobs Effect: Scenario 6 enhances the mean priority of 30% of the jobs to be five times greater, reflecting the surge in demand of critical customers. With the shortening product life cycle of consumer electronics, the proposed scheduling algorithm should address the needs of super-hot lot to reduce the time-to-market and satisfy customer needs to enlarge the market share.

5) New Product Effect: Scenario 7 extends double recipes for the ramping up of new technologies and new machines, causing the setup times to increase between jobs. The proposed scheduling algorithm should reduce the setup time to enhance the machine utilization and reduce the production cycle time.

6) New Fab Ramping Up Effect: Scenario 8 considers new fab ramping up with the mean priority of 30% of the jobs to be five times greater with double recipes for new technologies and new machines compared with the normal situation. For a new ramping fab, there is more super-hot lot to test new processes and new machines to enhance the yields of new products. Therefore, more recipes exist for these new products and new machines.

7) Abnormal Events Effect: Scenario 9 considers abnormal events such as earthquakes and power and gas supply interruptions; this causes several machines to become unavailable, and the unavailable rate increases to 20%.

8) Least Favorable Situation: Scenario 10 is the worst case that has all the considerations of Scenarios 2 to 9 to address all the possible constraints in fab to assess the performance of the proposed algorithm.

C. Experimental Result of Practical Problems

The ten aforementioned scenarios consider a practical problem of 500 jobs and 71 machines, in which more than 17.8 million decision variables are considered in the formulated MINLP model, subjected to more than 35.6 million constraints. Thus, the scenarios cannot be solved to obtain the optimal solution within a reasonable time. Since the heuristics can obtain only the near optimal solutions of the large problem within a reasonable time, each scenario was repeatedly performed ten times with the same testing data to compare the average performance of all the heuristic algorithms. Since online scheduling requires a satisfactory solution within 3 minutes, each run should stop at 3 minutes.

To estimate the validity, three conventional metaheuristics were compared: the GA [10], PSO [46], and ABC [47]. In this study, GA uses the same chromosome representation, two-cut points crossover, and two-cut points mutation as MSPHEDA in Section IV. Two-cut points mutation randomizes one gene in each part of the chromosome. PSO uses one particle to represent one solution in optimization problem. Many particles exist simultaneously to simulate the swarm path by record the current position, local best position, global best position and current velocity to determine the velocities and the positions for next iteration. ABC used the food source position to represent one solution. The onlooker bee will choose the new food by the probability value calculated from fitness value. The bees will search for new candidate food position based on current position and a random number direction.

Indeed, the parameter setting of the GA is the same as GA_H in Table VI, where P_{ES} is the percent of elitist selection, P_{RS} is the percent of rank selection, p_c is the crossover rate and p_m is the mutation rate. The chosen numbers of these

TABLE IV
MEAN OF OBJECTIVE FOR THE TEN SCENARIOS OF THE GA, PSO, AND ABC

Scenario	Mean of objectives (Standard Deviation)			GA improvement (Standard Deviation)	
	GA	PSO	ABC	GA improves PSO	GA improves ABC
S1	46277 (1218)	60446 (2008)	59822 (1658)	23% (39%)	23% (27%)
S2	410403 (128205)	558443 (151297)	565447 (122682)	27% (15%)	27% (-5%)
S3	72990 (12446)	85772 (13798)	82083 (14251)	15% (10%)	11% (13%)
S4	1178913 (188357)	1939277 (186023)	2197810 (328688)	39% (-1%)	46% (43%)
S5	38253 (1443)	49429 (3156)	47876 (1748)	23% (54%)	20% (17%)
S6	85062 (1945)	101993 (2806)	101197 (1985)	17% (31%)	16% (2%)
S7	50795 (10819)	58096 (11755)	57252 (10588)	13% (8%)	11% (-2%)
S8	87779 (1709)	103439 (2405)	104242 (2104)	15% (29%)	16% (19%)
S9	50016 (667)	62916 (11159)	65081 (21221)	21% (94%)	23% (97%)
S10	967919 (197978)	2023262 (264231)	1976478 (246899)	52% (25%)	51% (20%)
mean	298840 (54479)	504307 (64864)	525729 (75183)	41% (16%)	43% (28%)

TABLE V
MEAN OF THE TOTAL EXCEEDED LIMITED WAITING TIME OF THE GA, PSO, AND ABC

Scenario	Mean of the total exceeded limited waiting time (Standard Deviation)			GA improvement (Standard Deviation)	
	GA	PSO	ABC	GA improves PSO	GA improves ABC
S1	0 (0)	0 (1)	0 (0)	100% (100%)	-
S2	355 (128)	494 (150)	500 (122)	28% (15%)	29% (-5%)
S3	4 (12)	7 (14)	4 (14)	45% (13%)	12% (12%)
S4	1101 (188)	1859 (170)	2117 (328)	41% (-11%)	48% (43%)
S5	0 (0)	1 (3)	0 (0)	100% (100%)	-
S6	0 (0)	0 (0)	0 (0)	-	100% (100%)
S7	3 (11)	4 (12)	4 (11)	8% (8%)	3% (3%)
S8	0 (0)	0 (0)	0 (0)	-	-
S9	0 (0)	4 (11)	11 (7)	100% (100%)	100% (100%)
S10	814 (196)	1854 (263)	1808 (244)	56% (25%)	55% (20%)
mean	228 (53)	42 (262)	444 (74)	46% (14%)	49% (28%)

parameters are determined based on trial and error on a small set of problem instances. The parameter setting of PSO is that the population size equals to 100, global acceleration constant equals to 2 and local acceleration constant equals to 2 [46]. The parameter setting of ABC is that initial population equals to 50, population size equals to 100 and random number between $[-1, 1]$ [47]. Table IV shows that the GA improved the PSO by 41% and the ABC by 43% for the average objectives of the ten scenarios. In addition, the GA reduced 46% of the total exceeded limited waiting time in the PSO and 49% in the ABC, as shown in Table V. Therefore, this study attempted to improve the proposed algorithm from the GA to EDA as follows.

TABLE VI
PARAMETER SETTINGS OF GAS AND THE MSPGA

	Setting number of each algorithm					
Algorithm	GA_H	GA_M	GA_L	MSPGA		
Repetition times	10	10	10	10		
Execution time	3 minutes	3 minutes	3 minutes	3 minutes		
Subpopulation No.	-	-	-	1	2	3
w	1000	1000	1000	1000	1000	1000
Population size	50	50	50	50	50	50
P_{ES}	20%	20%	20%	20%	20%	20%
P_{RS}	80%	80%	80%	80%	80%	80%
p_c	0.9	0.5	0.1	0.9	0.5	0.1
p_m	0.1	0.5	0.9	0.1	0.5	0.9

The GA fairly outperforms PSO and ABC. Thus, this study improved GA with the proposed MSPGA (Multi-subpopulation Parameters with Genetic Algorithm) to determine the different parameters. Subsequently, GA_H was designed with a high crossover rate and low mutation rate. GA_M was designed with a medium crossover rate and medium mutation rate. GA_L was designed with a low crossover rate and high mutation rate. In the proposed MSPGA, three subpopulations exist with the same parameter settings as GA_H, GA_M, and GA_L. In GA_H, GA_M, GA_L and MSPGA, the same chromosome representation, two-cut points crossover and two-cut points mutation as MSPHEDA were used. Table VI shows the parameter settings of GAs and the MSPGA. Table VII shows the mean and standard deviation of the objective values of GAs and the MSPGA, with the improvement rates of the MSPGA compared with those of GAs. In particular, the MSPGA has more than 8% improvement in the objective mean than three GAs. The MSPGA has 71% improvement in the standard deviation than three GAs. The results showed that the proposed MSPGA can effectively improve GA with the same parameters. Table VIII shows the total exceeded limited waiting time to analyze the constraint of limited waiting time. The MSPGA obtains the least total exceeded limited waiting time than GAs, in Scenarios 2, 3, 4 and 10. The MSPGA reduces more than 16% average total exceeded limited waiting time and more than 9% standard deviation of the total exceeded waiting time.

To enhance the conventional EDA, the authors firstly proposed Hybrid Estimation of Distribution Algorithm with Multiple Subpopulations (HEDA-MS) [48] that can outperform the EDA and GA. HEDA-MS used the same chromosome representation, initializing the probability matrix, initial population, decoding procedure, evaluation, creating new offspring and local search mechanism as MSPHEDA. HEDA-MS used fuzzy logic controller (FLC) [49] to determine the crossover rate and mutation rate, in which FLC increases the crossover rate and decreases mutation rate when the fitness from crossover offspring performs well. On the other hand, FLC decreases the crossover rate and increases mutation rate when the fitness from mutation offspring performs well. This study improves the HEDA-MS as MSPHEDA by using different parameter settings in each subpopulation to replace FLC. Also, the MSPGA was

TABLE VII
MEAN OF OBJECTIVE FOR TEN SCENARIOS BETWEEN THE GAs AND MSPGA

Scenario	Mean of objectives (Standard Deviation)				MSPGA improvement (Standard Deviation)		
	GA_H	GA_M	GA_L	MSPGA	MSPGA improves GA_H	MSPGA improves GA_M	MSPGA improves GA_L
S1	46277 (1218)	41975 (675)	41703 (445)	41686 (811)	10% (64%)	1% (34%)	0% (1%)
S2	410403 (128205)	136557 (69162)	127781 (58016)	105646 (66199)	74% (99%)	23% (99%)	17% (98%)
S3	72990 (12446)	80001 (48870)	95199 (65374)	64718 (1079)	11% (93%)	19% (98%)	32% (99%)
S4	1178913 (188357)	453442 (63384)	486374 (118650)	413757 (94719)	65% (77%)	9% (32%)	15% (64%)
S5	38253 (1443)	32457 (987)	31345 (1310)	32579 (789)	15% (3%)	0% (-42%)	-4% (-7%)
S6	85062 (1945)	78358 (2001)	76691 (1321)	77731 (2500)	9% (20%)	1% (22%)	-1% (-18%)
S7	50795 (10819)	42618 (671)	42281 (930)	43345 (752)	15% (96%)	-2% (40%)	-3% (57%)
S8	87779 (1709)	78889 (1195)	119051 (83048)	79301 (1375)	10% (-2%)	-1% (-46%)	33% (98%)
S9	50016 (667)	45963 (749)	52029 (21370)	45700 (597)	9% (2%)	1% (12%)	12% (97%)
S10	967919 (197978)	483002 (50788)	442413 (84024)	443954 (51176)	54% (91%)	8% (63%)	0% (78%)
Mean	298840 (54479)	147326 (23848)	151486 (43449)	134842 (6967)	55% (87%)	8% (71%)	11% (84%)

TABLE VIII
MEAN OF THE TOTAL EXCEEDED LIMITED WAITING TIME FOR TEN SCENARIOS BETWEEN GAs AND THE MSPGA

Scenario	Mean of the total exceeded limited waiting time (Standard Deviation)				MSPGA improvement (Standard Deviation)		
	GA_H	GA_M	GA_L	MSPGA	MSPGA improves GA_H	MSPGA improves GA_M	MSPGA improves GA_L
S1	0 (0)	0 (0)	0 (0)	0 (0)	-	-	-
S2	355 (128)	87 (69)	77 (57)	56 (66)	84% (49%)	36% (4%)	27% (-15%)
S3	4 (12)	16 (49)	31 (65)	0 (0)	100% (100%)	100% (100%)	100% (100%)
S4	1101 (188)	380 (63)	412 (118)	340 (94)	69% (50%)	11% (-50%)	17% (20%)
S5	0 (0)	0 (0)	0 (0)	0 (0)	-	-	-
S6	0 (0)	0 (0)	0 (0)	0 (0)	-	-	-
S7	3 (11)	0 (0)	0 (0)	0 (0)	100% (100%)	-	-
S8	0 (0)	0 (0)	37 (79)	0 (0)	-	-	100% (100%)
S9	0 (0)	0 (0)	7 (21)	0 (0)	-	-	100% (100%)
S10	814 (196)	348 (49)	302 (80)	304 (48)	63% (75%)	13% (1%)	-1% (40%)
Mean	228 (53)	83 (23)	87 (42)	70 (21)	69% (61%)	16% (9%)	19% (51%)

extended to the MSPHGA (Multi-subpopulation Parameters with Hybrid Genetic Algorithm) with two proposed local search mechanisms. MSPHGA used the same chromosome representation, two-cut point crossover, two-cut point mutation and three subpopulations as MSPGA. Thus, MSPHGA and MSPHEDA were compared with the same parameter settings.

Table IX shows different P_{ES}, P_{RS}, p_c, and p_m settings among HEDA-MS, MSPHGA, and MSPHEDA. In MSPHGA and MSPHEDA, a high crossover rate with low elitist selection was set for Subpopulation 1, a medium crossover rate with medium elitist selection was set for Subpopulation 2, and a low crossover rate with high elitist selection was set for Subpopulation 3. The proposed approach can maintain both exploration and exploitation abilities in each subpopulation.

Table X shows the mean objectives of the HEDA-MS, MSPHGA, and MSPHEDA among the ten scenarios. The proposed MSPHEDA improves 4% of the mean and 51% of the standard deviation comparing with the HEDA-MS, on average. The MSPHEDA also improves 31% of the mean and

TABLE IX
PARAMETER SETTINGS OF THE HEDA-MS, MSPHGA, AND MSPHEDA

	Setting number of each algorithm					
Algorithm	HEDA-MS			MSPHGA and MSPHEDA		
Repetition times	10			10		
Execution time	3 minutes			3 minutes		
Subpopulation No.	1	2	3	1	2	3
w	1000	1000	1000	1000	1000	1000
Population size	50	50	50	50	50	50
P_{ES}	20%	20%	20%	10%	20%	30%
P_{RS}	80%	80%	80%	90%	80%	70%
p_c	FLC	FLC	FLC	0.9	0.5	0.1
p_m	FLC	FLC	FLC	0.1	0.5	0.9
N_{pop}	3	3	3	3	3	3
N_{best}	3	3	3	3	3	3
T_{EX}	50	50	50	50	50	50
T_{LS}	10	10	10	10	10	10
T_{ELS}	30	30	30	30	30	30
SP	10	10	10	10	10	10
α	0.1	0.1	0.1	0.1	0.1	0.1

TABLE X
MEAN OF OBJECTIVE AND EXECUTED SOLUTIONS FOR TEN SCENARIOS AMONG THE MSPHGA, HEDA-MS, AND MSPHEDA

Scenario	Mean of objectives (Standard Deviation)			MSPHEDA improvement (Standard Deviation)	
	MSPHGA	HEDA-MS	MSPHEDA	MSPHEDA improves MSPHGA	MSPHEDA improves HEDA-MS
S1	40585 (5180)	34062 (561)	33518 (308)	17% (94%)	2% (45%)
S2	94356 (43121)	37943 (908)	37025 (319)	61% (99%)	2% (65%)
S3	59860 (3878)	55566 (480)	54696 (425)	9% (89%)	2% (12%)
S4	316268 (98426)	198162 (26048)	184826 (9143)	42% (91%)	7% (65%)
S5	30326 (6085)	24530 (842)	23432 (677)	23% (89%)	4% (20%)
S6	74672 (9941)	63240 (674)	61465 (654)	18% (93%)	3% (3%)
S7	39112 (4414)	35744 (626)	34854 (439)	11% (90%)	2% (30%)
S8	74580 (8524)	66567 (1203)	65037 (1097)	13% (87%)	2% (9%)
S9	40946 (3952)	37414 (640)	36643 (428)	11% (89%)	2% (33%)
S10	309873 (76802)	223860 (11361)	214706 (7751)	31% (90%)	4% (32%)
Mean	108058 (26032)	77709 (4334)	74620 (2124)	31% (92%)	4% (51%)
Mean of executed solutions	185600 (3659)	260776 (4617)	269465 (1971)	45% (46%)	3% (57%)

92% of the standard deviation compared with the MSPHGA, on average. The proposed MSPHEDA can generate superior solutions compared with the HEDA-MS and MSPHGA.

Table XI shows that the MSPHEDA can obtain 6% and 50% less total exceeded limited waiting time than the HEDA-MS and MSPHGA can, respectively. The proposed MSPHEDA has robust performance than the HEDA-MS and MSPHGA.

Fig. 8 shows the box plots of the mean objective values of the HEDA-MS and MSPHEDA for Scenarios 1 to 10, indicating that the proposed MSPHEDA has smaller mean values and the significantly smaller standard deviation, indicating crucial stability in practice.

D. Experimental Result of Academic Instances

Five instances are designed for medium size problems. Each instance contains 50 jobs assigning to 15 machines.

TABLE XI
MEAN OF THE TOTAL EXCEEDED LIMITED WAITING TIME FOR TEN SCENARIOS AMONG THE MSPHGA, HEDA-MS, AND MSPHEDA

	Mean of the total exceeded limited waiting time (Standard Deviation)			MSPHEDA improvement (Standard Deviation)	
Scenario	MSPHGA	HEDA-MS	MSPHEDA	MSPHEDA improves MSPHGA	MSPHEDA improves HEDA-MS
S1	0 (0)	0 (0)	0 (0)	-	-
S2	52 (39)	0 (0)	0 (0)	100% (100%)	-
S3	0 (0)	0 (0)	0 (0)	-	-
S4	250 (96)	138 (26)	126 (9)	50% (90%)	9% (64%)
S5	0 (0)	0 (0)	0 (0)	-	-
S6	0 (0)	0 (0)	0 (0)	-	-
S7	0 (0)	0 (0)	0 (0)	-	-
S8	0 (0)	0 (0)	0 (0)	-	-
S9	0 (0)	0 (0)	0 (0)	-	-
S10	197 (69)	127 (11)	122 (9)	38% (87%)	4% (22%)
Mean	50 (20)	26 (4)	25 (2)	50% (91%)	6% (51%)

Each data is generated from the same distribution of practical data. Instance 1 is the normal case. Instance 2 generates the mean of the machine process time double that of the normal situation. Instance 3 assumes that the mean of the limited waiting time is 80% that of the normal situation. Instance 4 doubles the number of recipes from the normal situation. Instance 5 combines all the considerations of Instance 1 to 4.

The parameter settings of all the algorithms are identical to the setting for practical problems in Section V-C. Tables XII and XIII show the experimental results for the nine algorithms. MSPHEDA performs the best among the others for both the mean of the objectives and the mean of the total exceeded limited waiting time. MSPHEDA improves 3% of the mean and 23% of the standard deviation compared with the HEDA-MS.

E. Discussion

The proposed approach designed the parameter settings can outperform three GAs. The effectiveness of the proposed local search mechanisms was validated. Tables VII and X showed 20% improvement from MSPGA to MSPHGA, on average.

Moreover, the MSPHGA can be enhanced to the MSPHEDA that can obtain 31% improvement and 45% more solutions comparing to the MSPHGA with the same parameter setting in Table X. Since the EDA samples new solutions on the basis of a probability matrix, the CPU execution speed is faster than the crossover in GA. Thus, the EDA embedded in MSPHGA can obtain better solutions in a global search than conventional GA.

Finally, the performance between the HEDA-MS and MSPHEDA was compared. The FLC in the HEDA-MS is effective in obtaining improved solutions since it can automatically set p_c and p_m. However, it is time consuming to obtain suitable p_c and p_m. The proposed MSPHEDA can set different parameters of p_c, p_m, P_{ES}, and P_{RS} in each

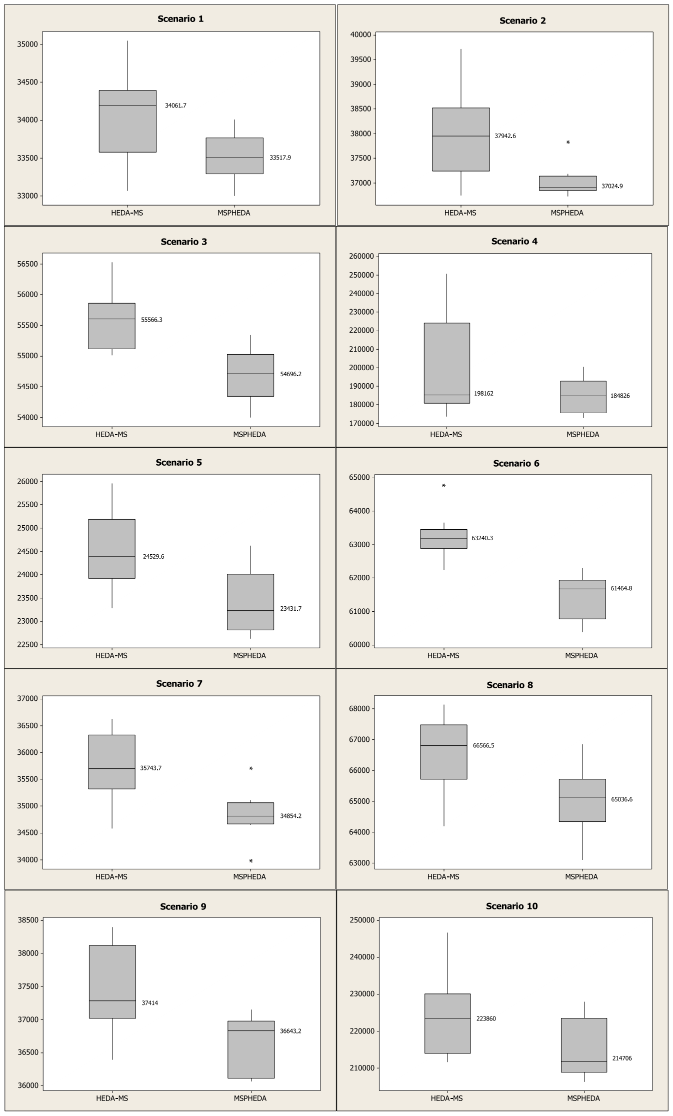

Fig. 8. Box plot of ten scenarios.

subpopulation to save the calculation time and thus effectively improve the executed solutions by 3% on average. Table X and Fig. 8 showed MSPHEDA obtains a 51% smaller standard deviation of the objective values than HEDA-MS, indicating critical robustness for online scheduling in real settings.

VI. CONCLUSION

This study presented a novel algorithm called the MSPHEDA for semiconductor manufacturing scheduling with constrained waiting times. To estimate its validity, an experiment was designed to compare it with alternative approaches in numerous scenarios for the SMSP in real settings. An MINLP model was formulated for the small-scale SMSP problems by LINGO to obtain optimal solutions for validation. The results showed that the proposed approach can obtain the optimal solution for small-scale problems. Furthermore, ten common scenarios in real settings were considered. The results showed that the average mean and standard deviation of the proposed algorithm were smaller than those of other heuristics such as the PSO, ABC, GA, MSPGA, MSPHGA, and HEDA-MS.

TABLE XII
MEAN OF OBJECTIVE FOR FIVE INSTANCES AMONG ALL THE NINE ALGORITHMS

	Mean of objectives (Standard Deviation)					
Instances	**I1**	**I2**	**I3**	**I4**	**I5**	**Mean**
ABC	14394	60568	128942	14090	268984	97396
	(638)	(21440)	(16883)	(542)	(38443)	(15589)
PSO	14447	54436	124215	13967	255438	92501
	(542)	(13263)	(22589)	(598)	(14173)	(10233)
GA_H	13653	45309	111301	13077	238241	84316
	(340)	(17235)	(31201)	(538)	(25193)	(14901)
GA_M	13375	23047	90113	12733	227760	73406
	(329)	(9503)	(28031)	(168)	(32962)	(14199)
GA_L	13417	23177	87117	12728	218947	71077
	(420)	(7336)	(19798)	(419)	(16382)	(8871)
MSPGA	13364	23440	86721	12683	200317	67305
	(361)	(6199)	(17988)	(390)	(22383)	(9464)
MSPHGA	13303	21778	84653	12643	196836	65843
	(527)	(4902)	(27574)	(388)	(28907)	(12460)
HEDA-MS	13154	22165	77238	12224	195256	64007
	(299)	(7377)	(16570)	(319)	(23284)	(9570)
MSPHEDA	13093	19460	71115	12151	194032	61970
	(316)	(5072)	(17004)	(279)	(14046)	(7343)

TABLE XIII
MEAN OF THE TOTAL EXCEEDED LIMITED WAITING TIME FOR FIVE INSTANCES AMONG ALL THE NINE ALGORITHMS

	Mean of objectives (Standard Deviation)					
Instances	**I1**	**I2**	**I3**	**I4**	**I5**	**Mean**
ABC	0	48	116	0	255	84
	(0)	(21)	(17)	(0)	(38)	(15)
PSO	0	42	112	0	241	79
	(0)	(13)	(23)	(0)	(14)	(10)
GA_H	0	33	99	0	224	71
	(0)	(17)	(31)	(0)	(25)	(15)
GA_M	0	11	78	0	214	61
	(0)	(10)	(27)	(0)	(33)	(14)
GA_L	0	11	75	0	205	58
	(0)	(7)	(20)	(0)	(16)	(9)
MSPGA	0	11	75	0	186	54
	(0)	(6)	(18)	(0)	(22)	(9)
MSPHGA	0	10	59	0	189	52
	(0)	(5)	(23)	(0)	(32)	(12)
HEDA-MS	0	10	65	0	181	51
	(0)	(7)	(17)	(0)	(23)	(9)
MSPHEDA	0	8	59	0	180	49
	(0)	(5)	(17)	(0)	(14)	(7)

The proposed algorithm has the shortest total exceeded limited waiting time.

Future research can apply the proposed algorithm to other complicated batch production problems such as those related to etching and furnaces for semiconductor manufacturing. Because the waiting time of jobs between different process stages is difficult to estimate in current Giga-fabs, most existing semiconductor manufacturing studies have focused on individual work areas such as oxidation, deposition, diffusion, photolithography, etching, ion implantation, and planarization, and have not covered the scheduling process loop that has increasingly complicated constraints such as waiting times, frequency-based setups, and capacity preoccupation. For instance, Mason *et al.* [50] developed a modified shifting bottleneck heuristic to minimize the total weighted tardiness for a complex job shop scheduling problem in a fab. Furthermore, when the waiting time between two consecutive stations can be estimated more precisely, the proposed approach can be extended to address multiple-station scheduling problems for wafer fabrication such as ONO stacked film processes.

TABLE XIV
ACADEMIC INSTANCES I

Job (Pieces)	Recipe (Weight)	Arrival time (Limited waiting time)	Available machine (Processing time)
J1(16)	B(4)	98(128)	M9(55),M3(47),M8(42),M10(65),M12(57),M4(37),M6(82), M14(20),M13(65)
J2(16)	I(1)	76(114)	M13(13),M1(34),M7(33),M11(58),M4(39),M15(25),M9(41) ,M12(63),M3(83),M6(5),M8(39)
J3(3)	E(2)	72(115)	M14(49),M2(41),M4(27),M8(20),M10(41),M3(31),M11(38)
J4(3)	B(8)	60(90)	M14(44),M2(68),M8(31),M5(30),M13(46),M4(5),M11(80), M15(48),M7(37),M6(40),M10(34),M3(11),M9(50),M1(22)
J5(10)	A(1)	0(91)	M3(30),M11(35),M6(44),M14(41),M7(35)
J6(9)	G(6)	86(116)	M13(82),M11(51),M8(21),M7(43),M3(57),M9(16),M12(49) ,M1(60)
J7(7)	G(8)	68(128)	M7(49),M9(42),M2(63),M14(72),M12(49),M13(18),M5(39) ,M6(24),M8(40),M4(48),M15(67),M1(19),M3(52)
J8(19)	A(9)	34(118)	M8(60)
J9(24)	I(5)	70(113)	M3(63),M15(41),M2(65),M11(33),M7(30),M14(23),M8(38) ,M6(5),M12(25),M9(32)
J10(13)	A(1)	88(126)	M3(23),M1(36),M12(50),M9(33),M15(49),M4(33),M8(47), M10(19),M11(33),M2(52),M6(71),M5(33)
J11(21)	F(8)	95(125)	M5(45),M14(64),M13(12),M6(82),M1(19),M15(31),M8(38) ,M4(54),M7(15),M10(33),M2(37)
J12(11)	H(7)	73(135)	M12(22),M7(25),M10(18),M2(52),M6(33)
J13(13)	J(5)	93(123)	M10(59),M8(42),M3(64),M13(54),M11(83),M2(55),M1(47) ,M5(23),M7(36),M12(56),M15(46),M9(46),M4(67),M6(50), M14(57)
J14(14)	D(2)	14(79)	M4(40),M13(52),M11(17),M8(35),M12(44),M3(39),M10(19),M9(45),M7(45),M2(93),M14(52)
J15(19)	F(1)	84(137)	M1(18),M8(35),M5(62),M10(44),M6(75),M9(35),M11(35), M3(52),M15(36),M2(38),M13(26)
J16(18)	E(7)	38(118)	M1(55),M12(57),M9(21)
J17(24)	B(8)	61(98)	M6(38),M13(21),M12(39),M7(56),M5(39),M11(22)
J18(12)	A(2)	75(105)	M5(39),M4(35),M1(5)
J19(11)	D(2)	63(119)	M14(34),M15(12),M1(29),M3(50),M6(45),M13(35),M7(5)
J20(11)	C(9)	98(149)	M5(42),M14(54),M8(52),M12(43),M3(58)
J21(1)	B(7)	46(96)	M13(44),M6(57)
J22(15)	F(3)	50(107)	M7(7),M9(50)
J23(23)	C(1)	61(143)	M10(54),M13(50)
J24(14)	J(6)	75(111)	M14(46),M11(25),M13(55),M9(91),M4(29),M12(36)
J25(7)	E(1)	56(86)	M5(55),M9(40),M10(35),M8(53)
J26(17)	I(3)	29(131)	M15(76),M9(16),M11(30),M10(24),M14(33),M1(45),M12(37),M4(71),M7(49),M5(36),M8(28),M3(49),M6(28),M2(49)
J27(9)	H(8)	70(111)	M15(54),M14(65),M6(75),M3(54),M2(31),M5(70),M13(44) ,M10(5),M11(60),M7(68),M1(23),M4(33)
J28(5)	F(5)	37(115)	M12(34),M5(67)
J29(7)	J(8)	0(132)	M5(41),M13(53),M3(20),M8(53),M9(40),M4(42),M15(42), M12(59),M11(20),M14(55),M7(80),M2(33),M10(57)
J30(19)	I(6)	104(134)	M6(51),M14(48),M13(37),M5(22),M1(55),M2(22),M11(57) ,M7(65),M8(5),M4(6),M9(26),M3(34)
J31(22)	B(7)	42(118)	M15(24),M11(21)
J32(24)	G(3)	45(206)	M6(43),M14(77),M9(35),M2(72),M1(21),M5(46),M3(42),M10(14),M8(15),M15(13),M12(18),M7(43),M13(42),M4(46), M11(23)
J33(5)	C(5)	87(117)	M3(32),M7(57),M4(57),M2(38),M6(46),M11(50),M13(37), M8(14),M9(15),M12(18),M1(48),M5(45)
J34(18)	A(8)	49(113)	M5(40)
J35(6)	E(2)	63(99)	M2(32),M13(11),M10(47),M14(10),M3(10),M7(40),M12(6) ,M15(42),M1(61),M4(27),M8(5)
J36(5)	E(7)	17(159)	M6(43),M5(16),M15(46),M4(46),M10(42),M3(69),M2(31), M14(18),M12(65),M13(58),M7(24),M11(59)
J37(25)	J(6)	37(80)	M2(28),M9(49),M12(35),M14(60),M7(77),M8(36),M5(30), M1(48),M15(5),M13(31),M3(47)
J38(21)	C(7)	62(92)	M6(43),M13(84),M15(5),M4(38),M1(50),M2(53),M11(56), M12(44),M8(50)
J39(18)	B(10)	64(149)	M15(14),M10(55),M11(37)
J40(13)	D(9)	22(107)	M3(60),M7(32),M1(52)
J41(20)	F(9)	60(156)	M10(9),M1(78),M7(44),M6(45),M8(10),M9(58),M13(47),M14(20),M5(18),M12(50),M15(23),M2(15)
J42(18)	A(7)	33(104)	M13(63),M4(35),M9(60),M14(5),M7(41),M6(53),M3(40),M5(34),M2(11),M11(30),M1(68),M8(44),M12(38),M15(25), M10(51)
J43(20)	G(3)	70(176)	M10(41),M11(6)
J44(22)	G(2)	89(163)	M14(76),M10(27),M8(52),M15(5),M11(48),M13(22),M1(5) ,M5(33),M4(63),M3(59)
J45(8)	A(6)	42(98)	M1(64),M2(56),M5(78),M13(26),M4(40),M6(17),M3(11),M15(61),M11(5)
J46(21)	G(3)	10(125)	M14(49),M10(66),M1(25),M11(45),M13(50),M3(81),M7(41),M12(31),M15(15),M8(68),M6(38),M2(7),M5(65)
J47(12)	A(6)	15(75)	M8(5),M1(19),M13(12),M11(46),M5(45),M7(14),M3(15),M6(39),M10(81),M4(5)
J48(18)	G(2)	70(102)	M6(50),M1(28),M2(25),M8(17)
J49(20)	A(5)	93(151)	M15(56),M6(37)
J50(22)	F(2)	32(132)	M1(46),M11(31),M9(46),M10(56),M3(43),M7(32),M5(37), M8(59),M13(41),M12(38),M2(15),M14(19)

APPENDIX

Five instances are designed for the medium size problems in Section V-D. Tables XIV and XV illustrate two of the instances. All the 15 machines are available at time zero. The setup time are set as 10 minutes between different recipes and no setup time between the same recipes.

TABLE XV
ACADEMIC INSTANCES II

Job (Pieces)	Recipe (Weight)	Arrival time (Limited waiting time)	Available machine (Processing time)
J1(8)	I(1)	20(77)	M7(32),M8(48),M6(47),M12(47),M9(68),M4(58),M10(24), M3(48),M2(57),M11(30),M15(36),M1(50)
J2(4)	G(4)	23(63)	M6(48),M12(35)
J3(12)	F(5)	47(68)	M1(52),M9(21),M3(45),M13(52),M11(54),M12(39),M10(40),M7(50),M5(55),M14(18),M15(52),M4(50),M8(44)
J4(1)	B(2)	33(50)	M11(52),M14(35),M1(32),M15(35),M3(48),M13(28),M6(44)
J5(9)	I(5)	27(37)	M13(35),M8(42),M6(28),M3(28),M11(28),M5(52)
J6(17)	B(6)	18(50)	M10(41),M1(22),M14(24),M4(39),M15(33)
J7(13)	F(7)	27(37)	M4(40),M2(41)
J8(15)	B(5)	15(66)	M7(33),M2(41),M3(40),M10(34),M6(30),M12(37)
J9(17)	B(1)	16(26)	M12(33),M8(43),M10(39),M4(35),M6(32),M2(50),M11(24),M3(29),M5(45),M14(45),M13(52)
J10(18)	F(1)	35(51)	M13(44),M14(34),M15(25),M11(20),M4(40),M9(26),M6(46),M8(51),M5(36)
J11(24)	G(4)	4(58)	M5(32),M8(46),M12(29),M3(49),M9(58),M15(37),M6(48), M7(41),M4(34),M14(53),M11(40)
J12(16)	A(4)	31(44)	M10(24),M5(35),M7(59),M6(32),M8(36),M13(41),M2(27), M4(31),M9(40),M12(35),M3(43)
J13(11)	B(9)	25(60)	M12(26),M3(36),M5(56),M7(46),M14(41),M13(43),M11(29),M8(49),M1(36),M2(35),M6(30),M15(31),M10(48),M4(44)
J14(9)	I(8)	18(57)	M9(50),M10(41),M6(46)
J15(16)	I(9)	44(54)	M13(55),M14(39),M11(43),M9(33),M8(46),M15(55),M4(53),M3(44),M2(42),M10(39),M1(49),M7(35)
J16(3)	A(7)	41(54)	M9(34),M1(30),M5(42),M10(39),M15(31),M12(25)
J17(19)	G(4)	30(64)	M2(49),M8(47),M13(46),M10(35),M9(37),M1(26),M12(39),M5(37),M6(51),M4(36),M15(30)
J18(20)	E(4)	37(81)	M1(38),M8(52),M14(35),M15(33),M12(33)
J19(15)	F(9)	25(40)	M14(34),M15(62),M2(45),M4(31),M7(33),M11(47),M10(37),M5(45),M13(44),M9(41),M8(31),M1(39)
J20(15)	G(1)	45(55)	M7(57),M11(30),M12(48),M10(55),M13(43),M3(33),M14(44),M8(46),M6(35),M15(37),M4(54),M5(34),M9(35),M2(31)
J21(6)	I(1)	33(53)	M7(48),M1(38),M3(33),M14(50),M4(44),M12(37),M9(32), M2(43),M15(23),M5(40),M8(24),M10(19)
J22(4)	C(4)	36(56)	M3(38),M10(38),M5(51),M13(41)
J23(3)	E(6)	27(67)	M8(34),M1(40),M6(39),M5(31),M14(41),M12(32),M4(44), M13(60),M11(31),M2(21)
J24(9)	I(4)	19(59)	M13(35),M14(24),M10(34),M2(34)
J25(11)	F(1)	16(99)	M8(23),M6(42),M15(35),M13(33),M5(28)
J26(1)	F(10)	21(106)	M15(76),M9(16),M11(30),M10(24),M14(33),M1(45),M12(37),M4(71),M7(49),M5(36),M8(28),M3(49),M6(28),M2(49)
J27(11)	G(1)	35(46)	M15(54),M14(65),M6(75),M3(54),M2(31),M5(70),M13(44)
			,M10(5),M11(60),M7(68),M1(23),M4(33)
J28(10)	E(4)	40(81)	M12(34),M5(67)
J29(4)	A(1)	23(62)	M5(41),M13(53),M3(20),M8(53),M9(40),M4(42),M15(42), M12(59),M11(20),M14(55),M7(80),M2(33),M10(57)
J30(9)	H(1)	35(83)	M6(51),M14(48),M13(37),M5(22),M1(55),M2(22),M11(57),M7(65),M8(5),M4(6),M9(26),M3(34)
J31(6)	F(8)	0(65)	M15(24),M11(21)
J32(5)	F(5)	34(64)	M6(43),M14(77),M9(35),M2(72),M1(21),M5(46),M3(42),M10(14),M8(15),M15(13),M12(18),M7(43),M13(42),M4(46), M11(23)
J33(12)	G(1)	32(44)	M3(32),M7(57),M4(57),M2(38),M6(46),M11(50),M13(37), M8(14),M9(15),M12(18),M1(48),M5(45)
J34(17)	A(9)	30(66)	M5(40)
J35(10)	C(3)	38(48)	M2(32),M13(11),M10(47),M14(10),M3(10),M7(40),M12(6),M15(42),M1(61),M4(27),M8(5)
J36(10)	I(10)	28(38)	M6(43),M5(16),M15(46),M4(46),M10(42),M3(69),M2(31), M14(18),M12(65),M13(58),M7(24),M11(59)
J37(3)	F(3)	17(67)	M2(28),M9(49),M12(35),M14(60),M7(77),M8(36),M5(30), M1(48),M15(5),M13(31),M3(47)
J38(17)	G(7)	27(76)	M6(43),M13(84),M15(5),M4(38),M1(50),M2(53),M11(56), M12(44),M8(50)
J39(1)	A(1)	35(45)	M15(14),M10(55),M11(37)
J40(24)	D(1)	23(49)	M3(60),M7(32),M1(52)
J41(12)	E(7)	23(57)	M10(9),M1(78),M7(44),M6(45),M8(10),M9(58),M13(47),M14(20),M5(18),M12(50),M15(23),M2(15)
J42(19)	B(7)	14(40)	M13(63),M4(35),M9(60),M14(5),M7(41),M6(53),M3(40),M5(34),M2(11),M11(30),M1(68),M8(44),M12(38),M15(25), M10(51)
J43(13)	F(5)	31(83)	M10(41),M11(6)
J44(14)	I(2)	40(73)	M14(76),M10(27),M8(52),M15(5),M11(48),M13(22),M1(5),M5(33),M4(63),M3(59)
J45(13)	B(8)	29(84)	M1(64),M2(56),M5(78),M13(26),M4(40),M6(17),M3(11),M15(61),M11(5)
J46(10)	A(10)	22(51)	M14(49),M10(66),M1(25),M11(45),M13(50),M3(81),M7(41),M12(31),M15(15),M8(68),M6(38),M2(7),M5(65)
J47(13)	J(2)	37(68)	M8(5),M1(19),M13(12),M11(46),M5(45),M7(14),M3(15),M6(39),M10(81),M4(5)
J48(1)	A(1)	32(73)	M6(50),M1(28),M2(25),M8(17)
J49(22)	F(7)	28(62)	M15(56),M6(37)
J50(19)	G(2)	25(57)	M1(46),M11(31),M9(46),M10(56),M3(43),M7(32),M5(37), M8(59),M13(41),M12(38),M2(15),M14(19)

REFERENCES

[1] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 114–117, 1965.

[2] R. C. Leachman, S. Ding, and C.-F. Chien, "Economic efficiency analysis of wafer fabrication," *IEEE Trans. Autom. Sci. Eng.*, vol. 4, no. 4, pp. 501–512, Oct. 2007.

[3] C.-F. Chien *et al.*, "Modeling and analysis of semiconductor manufacturing in a shrinking world: Challenges and successes," *Eur. J. Ind. Eng.*, vol. 5, no. 3, pp. 254–271, 2011.

[4] C.-F. Chien, Y.-J. Chen, C.-Y. Hsu, and H.-K. Wang, "Overlay error compensation using advanced process control with dynamically adjusted proportional-integral R2R controller," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 2, pp. 473–484, Apr. 2014.

[5] C. Kuo, C.-F. Chien, and J. Chen, "Manufacturing intelligence to exploit the value of production and tool data to reduce cycle time," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 1, pp. 103–111, Jan. 2011.

[6] C.-F. Chien and C. Chen, "Using GA and CTPN for modeling the optimization-based schedule generator of a generic production scheduling system," *Int. J. Prod. Res.*, vol. 45, no. 8, pp. 1763–1789, 2007.

[7] L. Mönch, J. W. Fowler, S. Dauzere-Peres, S. J. Mason, and O. Rose, "A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations," *J. Schedul.*, vol. 14, no. 6, pp. 583–599, 2011.

[8] C.-F. Chien and C.-H. Chen, "A novel timetabling algorithm for a furnace process for semiconductor fabrication with constrained waiting and frequency-based setups," *OR Spectr.*, vol. 29, no. 3, pp. 391–419, 2007.

[9] Y. Demir and S. K. Isleyen, "Evaluation of mathematical models for flexible job-shop scheduling problems," *Appl. Math. Model.*, vol. 37, no. 3, pp. 977–988, 2013.

[10] M. Gen, R. Cheng, and L. Lin, *Network Models and Optimization: Multiobjective Genetic Algorithms*. London, U.K.: Springer, 2008.

[11] W. Jia, Z. Jiang, and Y. Li, "Combined scheduling algorithm for re-entrant batch-processing machines in semiconductor wafer manufacturing," *Int. J. Prod. Res.*, vol. 53, no. 6, pp. 1–14, 2015.

[12] A. Klemmt and L. Mönch, "Scheduling jobs with time constraints between consecutive process steps in semiconductor manufacturing," in *Proc. Winter Simulat. Conf.*, Berlin, Germany, 2012, pp. 1–10.

[13] A. M. Aguirre, C. A. Mendez, and P. M. Castro, "A novel optimization method to automated wet-etch station scheduling in semiconductor manufacturing systems," *Comput. Chem. Eng.*, vol. 35, no. 12, pp. 2960–2972, 2011.

[14] B. Yan, H. Y. Chen, P. B. Luh, S. Wang, and J. Chang, "Litho machine scheduling with convex hull analyses," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 4, pp. 928–937, Oct. 2013.

[15] C. Jung, D. Pabst, M. Ham, M. Stehli, and M. Rothe, "An effective problem decomposition method for scheduling of diffusion processes based on mixed integer linear programming," *IEEE Trans. Semicond. Manuf.*, vol. 27, no. 3, pp. 357–363, Aug. 2014.

[16] F. Qiao, Y. M. Ma, L. Li, and H. X. Yu, "A Petri net and extended genetic algorithm combined scheduling method for wafer fabrication," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 1, pp. 197–204, Jan. 2013.

[17] X. Yu and M. Gen, *Introduction to Evolutionary Algorithm*. Berlin, Germany: Springer, 2010, p. 430.

[18] J.-Z. Wu and C.-F. Chien, "Modeling semiconductor testing job scheduling and dynamic testing machine configuration," *Expert Syst. Appl.*, vol. 35, nos. 1–2, pp. 485–496, 2008.

[19] Y. Song, M. T. Zhang, J. Yi, L. Zhang, and L. Zheng, "Bottleneck station scheduling in semiconductor assembly and test manufacturing using ant colony optimization," *IEEE Trans. Autom. Sci. Eng.*, vol. 4, no. 4, pp. 569–578, Oct. 2007.

[20] R. Driessel and L. Mönch, "Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times," *Comput. Ind. Eng.*, vol. 61, no. 2, pp. 336–345, 2011.

[21] C.-F. Chien, F. Tseng, and C. Chen, "An evolutionary approach to rehabilitation patient scheduling: A case study," *Eur. J. Oper. Res.*, vol. 189, no. 3, pp. 1234–1253, 2008.

[22] S. Dauzere-Peres and L. Mönch, "Scheduling jobs on a single batch processing machine with incompatible job families and weighted number of tardy jobs objective," *Comput. Oper. Res.*, vol. 40, no. 5, pp. 1224–1233, 2013.

[23] S. F. Attar, M. Mohammadi, R. Tavakkoli-Moghaddam, and S. Yaghoubi, "Solving a new multi-objective hybrid flexible flowshop problem with limited waiting times and machine-sequence-dependent set-up time constraints," *Int. J. Comput. Integr. Manuf.*, vol. 27, no. 5, pp. 450–469, 2014.

[24] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. New York, NY, USA: Springer, 2002.

[25] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-94-163, Jun. 1994.

[26] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I: Binary parameters," in *Parallel Problem Solving From Nature—PPSN IV* (LNCS 1141). Berlin, Germany: Springer, 1996.

[27] J. S. De Bonet, C. L. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," in *Proc. Conf. Adv. Neural Inf. Process. Syst*, vol. 9. Denver, CO, USA, 1997, pp. 424–430.

[28] M. Pelikan and H. Mühlenbein, *Advances in Soft Computing: Engineering Design and Manufacturing*. London, U.K.: Springer, 1999.

[29] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz, "BOA: The Bayesian optimization algorithm," in *Proc. Genet. Evol. Comput.*, San Francisco, CA, USA, 1999, pp. 525–532.

[30] J. Ceberio, E. Irurozki, A. Mendiburu, and J. A. Lozano, "A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems," *Progr. Artif. Intell.*, vol. 1, no. 1, pp. 103–117, 2012.

[31] X. C. Hao, J.-Z. Wu, C.-F. Chien, and M. Gen, "The cooperative estimation of distribution algorithm: A novel approach for semiconductor final test scheduling problems," *J. Intell. Manuf.*, vol. 25, no. 5, pp. 867–879, 2014.

[32] L. Wang, S. Y. Wang, Y. Xu, G. Zhou, and M. Liu, "A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem," *Comput. Ind. Eng.*, vol. 62, no. 4, pp. 917–926, 2012.

[33] J. Gao, M. Gen, L. Y. Sun, and X. H. Zhao, "A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems," *Comput. Ind. Eng.*, vol. 53, no. 1, pp. 149–162, 2007.

[34] J. Gao, M. Gen, L. Y. Sun, and X. H. Zhao, "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems," *Comput. Oper. Res.*, vol. 35, no. 9, pp. 2892–2907, 2008.

[35] B. Jarboui, M. Eddaly, and P. Siarry, "An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems," *Comput. Oper. Res.*, vol. 36, no. 9, pp. 2638–2646, 2009.

[36] S. H. Chen, M. C. Chen, P. C. Chang, Q. F. Zhang, and Y. M. Chen, "Guidelines for developing effective estimation of distribution algorithms in solving single machine scheduling problems," *Expert Syst. Appl.*, vol. 37, no. 9, pp. 6441–6451, 2010.

[37] V. Robles, P. D. Miguel, and P. Larrañaga, "Solving the traveling salesman problem with EDAs," *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. New York, NY, USA: Springer, 2002.

[38] L. Wang, C. Fang, C.-D. Mu, and M. Liu, "A Pareto-archived estimation-of-distribution algorithm for multiobjective resource-constrained project scheduling problem," *IEEE Trans. Eng. Manage.*, vol. 60, no. 3, pp. 617–626, Aug. 2013.

[39] A. B. Keha, K. Khowala, and J. W. Fowler, "Mixed integer programming formulations for single machine scheduling problems," *Comput. Ind. Eng.*, vol. 56, no. 1, pp. 357–365, 2009.

[40] Y. Unlu and S. J. Mason, "Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems," *Comput. Ind. Eng.*, vol. 58, no. 4, pp. 785–800, 2010.

[41] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[42] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.

[43] Ö. Yeniay, "Penalty function methods for constrained optimization with genetic algorithms," *Math. Comput. Appl.,* vol. 10, no. 1, pp. 45–56, 2005.

[44] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA J. Comput.*, vol. 6, no. 2, pp. 154–160, 1994.

[45] J. K. Cochran, S. M. Horng, and J. W. Fowler, "A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines," *Comput. Oper. Res.*, vol. 30, no. 7, pp. 1087–1102, 2003.

[46] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4. Piscataway, NJ, USA, 1995, pp. 1942–1948.

[47] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Glob. Optim.*, vol. 39, no. 3, pp. 459–471, 2007.

[48] H.-K. Wang, C.-F. Chien, and M. Gen, "Hybrid estimation of distribution algorithm with multiple subpopulations for semiconductor manufacturing scheduling problem with limited waiting-time constraint," in *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, Taipei, Taiwan, 2014, pp. 101–106.

[49] Y. Yun and M. Gen, "Performance analysis of adaptive genetic algorithms with fuzzy logic and heuristics," *Fuzzy Optim. Decis. Making,* vol. 2, no. 2, pp. 161–175, 2003.

[50] S. J. Mason, J. W. Fowler, and W. M. Carlyle, "A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops," *J. Schedul.*, vol. 5, no. 3, pp. 247–262, 2002.

Hung-Kai Wang received the M.S. degree in industrial engineering and engineering management from National Tsing Hua University (NTHU), Taiwan, in 2010, where he is currently pursuing the Ph.D. degree with the IEEM, NTHU. His research interests include advanced process control, genetic algorithms, modeling and analysis for semiconductor manufacturing, and intelligent manufacturing. His research works appear in the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING. He was a recipient of the Best Thesis Award.

Chen-Fu Chien (M'03) received the B.S. degree (with Phi Tao Phi Hons.) in industrial engineering and electrical engineering from National Tsing Hua University (NTHU), Taiwan, in 1990, the M.S. and Ph.D. degrees from the University of Wisconsin-Madison, USA, in 1994 and 1996, respectively. He was a Fulbright Scholar with the University of California, Berkeley, from 2002 to 2003 and received PCMPCL Training with Harvard Business School in 2007. He is a Tsing Hua Chair Professor and the Director of the NTHU-Taiwan Semiconductor Manufacturing Company (TSMC) Center for Manufacturing Excellence. He is the Principal Investigator for the Semiconductor Technologies Empowerment Partners Consortium funded by the Ministry of Science and Technology, Taiwan. From 2005 to 2008, he was on-leave as the Deputy Director of Industrial Engineering Division in TSMC. He holds eight USA invention patents and published three books, over 120 journal papers, and a number of case studies in Harvard Business School. His research efforts center on decision analysis, big data analytics, modeling and analysis for semiconductor manufacturing, and advanced intelligent manufacturing systems. He was a recipient of the National Quality Award, the Distinguished Research Award and the Tier 1 Principal Investigator Award (Top 3%) from NSC, the Distinguished University–Industry Collaborative Research Award from the Ministry of Education, the University Industrial Contribution Award from the Ministry of Economic Affairs, the Distinguished University–Industry Collaborative Research Award and the Distinguished Young Faculty Research Award from NTHU, the Distinguished Young Industrial Engineer Award, the Best IE Paper Award, and the IE Award from CIIE, the Best Engineering Paper Award and the Distinguished Engineering Professor by the Chinese Institute of Engineers in Taiwan, and the 2012 Best Paper Award from the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.

Mitsuo Gen received the B.E., M.E., and Ph.D. degrees in electronic engineering from Kogakuin University in 1969, 1971, and 1975, respectively, the Ph.D. degree in informatics from Kyoto University, Japan, in 2006. He is a Senior Research Scientist with Fuzzy Logic Systems Institute and a Visiting Professor at Tokyo University of Science, Japan. He was a Professor with the Graduate School of Information, Production and Systems, Waseda University, Japan. He was a Visiting Professor with the IEEM, National Tsing Hua University, Taiwan, from 2012 to 2013, a Hanyang Chair Professor with Hanyang University, Korea, from 2011 to 2012, and a Visiting Professor with the University of California, Berkeley, from 1999 to 2000 and Texas A&M University in 2000. His research interest includes evolutionary algorithms, manufacturing scheduling, logistics network, and decision making. He has co-authored five books such as *Introduction to Evolutionary Algorithms* in 2010 and *Network Models and Optimization: Multiobjective Genetic Algorithm Approach* (Springer) in 2008. He is an Area Editor of *Computers & Industrial Engineering* and an Editorial Board Member of several international journals.