Expert Systems with Applications 36 (2009) 11925–11933

Contents lists available at ScienceDirect

Expert Systems with Applications

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

A GA-Tabu algorithm for scheduling in-line steppers in low-yield scenarios

Chie-Wun Chiou, Muh-Cherng Wu [*]

Department of Industrial Engineering and Management, National Chiao Tung University, 1001, Dah-Shei Road, Hsin-Chu 300, Taiwan, ROC

ARTICLE INFO

Keywords:
Scheduling
Semiconductor
Flow shop
Port capacity constraints
Meta-heuristic algorithm
Genetic algorithm
Tabu search

ABSTRACT

This paper presents a scheduling algorithm for an in-line stepper in low-yield scenarios, which mostly appear in cases when new process/production is introduced. An in-line stepper is a bottleneck machine in a semiconductor fab. Its interior comprises a sequence of chambers, while its exterior is a *dock* equipped with several ports. The transportation unit for entry of each port is a job (a group of wafers), while that for each chamber is a piece of wafer. This transportation incompatibility may lead to a capacity-loss, in particular in low-yield scenarios. Such a capacity-loss could be alleviated by effective scheduling. The proposed scheduling algorithm, called GA-Tabu, is a combination of a genetic algorithm (GA) and a tabu search technique. Numerical experiments indicate that the GA-Tabu algorithm outperforms seven benchmark ones. In particular, the GA-Tabu algorithm outperforms a prior GA both in solution quality and computation efforts.

1. Introduction

In semiconductor manufacturing, *steppers* are the most expensive machine and are usually the bottleneck of a wafer fab. Their utilization would significantly affect the throughput of a fab. One way to effectively utilize a stepper is by appropriately scheduling the jobs waiting before it. Scheduling for steppers is thus an important research issue.

Much literature on stepper scheduling has been published. In terms of problem characteristics, they vary in the inclusion of different constraints imposed on steppers—for example, mask setup (Chern & Liu, 2003; Duwayri, Mollaghasemi, Nazzal, & Rabadi, 2006), machine dedication (Wu, Huang, Chang, & Yang, 2006; Wu, Chiou, & Chen, 2007; Wu, Jiang, & Chang, 2008), rework (Sha, Hsu, Che, & Chen, 2006), and cluster tool configuration (Morrison & Martin, 2007). In terms of solution methodology, four types were mostly used: dispatching rules (Dabbas & Fowler, 2003; Ying & Lin, 2009), artificial intelligence (Wu & Chang, 2007, 2008) mathematical programming (Chung & Hsieh, 2008), and meta-heuristic algorithms (Chiang & Fu, 2008).

Significant milestones on stepper scheduling have been established by prior studies. However, most of them implicitly made a *high-yield* assumption. That is, the production yield is quite high so that each wafer job is a *full lot* (i.e., carrying 25 wafers). However, in the stage of new product/process introduction, *low-yield* is quite common. Many *small lots* (i.e., carrying less than 25 wafers) may appear.

A recent study (Wu & Chiou, 2009) indicated that an *in-line stepper*, a relatively advanced version of stepper, may loss capacity in a low-yield scenario. Such a capacity-loss would not appear in a high-yield scenario and has been rarely noticed. They proposed a genetic algorithm (GA) for scheduling an in-line stepper. Their experiment results indicated that the GA outperforms four heuristic scheduling rules widely used in practice.

This paper aims to develop a more effective scheduling algorithm for an in-line stepper. The algorithm we proposed is called GA-Tabu, which is a combination of a GA and a tabu search technique. Experiment results indicated that the GA-Tabu outperform seven other meta-heuristic methods, including the GA by Wu and Chiou (2009).

The remainder of this paper is organized as follows. Section 2 introduces the operational mechanism of an in-line stepper. Section 3 describes how to compute the makespan of a job sequence. Section 4 presents the GA-Tabu algorithm. Experiment results are reported in Section 5. Concluding remarks are in the last section.

2. Operational mechanism of an in-line stepper

An in-line stepper is a machine, whose interior comprises a group of chambers, each of which can process *one* wafer at a time. Its exterior is equipped with a dock for job entry/departure (Fig. 1). Jobs are moved from a neighboring WIP buffer to the dock. The transportation unit between the WIP buffer and the dock is a *job* (a container carrying 1–25 wafers), but that between the dock and the in-line stepper is a piece of *wafer*. A *transportation incompatibility* thus exists between the dock and the in-line stepper.

As shown in Fig. 2, an in-line stepper is composed of two modules—a *track* and an *aligner* (Quirk, 2001; Xiao, 2001). Each module

* Corresponding author. Tel.: +886 3 5731 913; fax: +886 3 5720 610.
E-mail address: mcwu@mail.nctu.edu.tw (M.-C. Wu).

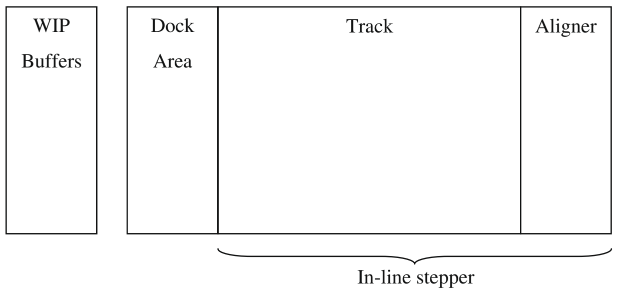

Fig. 1. Production system of stepper.

involves various types of chambers. Each type of chamber may be *more than one* in numbers. In the aligner module, a mask setup time is needed at the exposure chamber, while a wafer from a different job enters the chamber.

The dock of an in-line stepper typically involves four ports. Each port could accommodate only one job container. A job container on the dock must stay at the port until all its wafers have completed processing. This implies that any other jobs cannot access the in-line stepper while all the ports have been occupied. As a result, a *capacity-loss* may appear in a low-yield scenario, due to the limit of *port number*.

An example of such a *capacity-loss* is given below. Consider an in-line stepper that has four ports and 22 chambers. Four jobs (*A*, *B*, *C*, and *D*) are now on the dock. Job *A* contains 25 wafers while jobs *B*, *C*, *D* in total carry only 16 wafers. Suppose the processing sequence is*A*, *B*, *C* and *D*. When Job *A*'s last wafer leaves the stepper for the dock, the 16 wafers of jobs *B*, *C*, and *D* would occupy the last 16 chambers of the stepper. This results in a capacity-loss—the first six (22–16) chambers have no new wafers to host. Such a capacity-loss would not occur in a high-yield scenario, in which the total number of wafers in *B*, *C*, and *D* is usually more than 22.

3. Makespan evaluation for job sequences

A method, adapted from Ruiz and Maroto (2006), is presented for evaluating the *makespan* of a job sequence. In the evaluation, we virtually sent each wafer in sequence into the in-line stepper and looked for an available chamber that can finish the wafer at the earliest time. We first give notation and proceed to the makespan evaluation procedure, where a group of functionally identical chambers is called a *stage*.

3.1. Notation

j	index of job
k	index of wafer
i	index of stage
l	index of chamber
a	index of the exposure chamber
ρ	total number of ports in the dock
n	total number of jobs to be processed by the in-line stepper
M	total numbers of stages in the in-line stepper
m_i	total number of chambers at stage i
p_{iljk}	processing time required for chamber l at stage i to process wafer k in job j
π	a job sequence for n jobs, $\pi = [\pi(1),\ldots,\pi(n)]$
$\pi(j)$	the job in the jth position of a sequence π
$w(j)$	total number of wafers in job j
t_u	transportation time for uploading a job to the dock
t_d	transportation time for downloading a job from the dock
$S_{i,l,\pi(j),k}$	setup time required for chamber l in stage i to process wafer k in job $\pi(j)$ if $i\neq a$ or $k\neq 1$, then $S_{i,l,\pi(j),k} = 0$, otherwise, $S_{i,l,\pi(j),k} = \delta_{\pi(j),\pi(j-1)}$
$\delta_{\pi(j),\pi(j-1)}$	setup time required for the exposure chamber to switch production from job $\pi(j-1)$to job $\pi(j)$; $\delta_{\pi(j),\pi(j-1)} = s_0$ if $\pi(j-1)$ and $\pi(j)$ use different masks, and $\delta_{\pi(j),\pi(j-1)} = 0$, otherwise
$A_{i,l,t}$	the time epoch when chamber l in stage i just turns to be available. That is, while the chamber (i,l) is free at t, $A_{i,l,t}$ is the last *wafer-completion-epoch* before t; while the chamber (i,l) is in operation at t, $A_{i,l,t}$ is the *first wafer-completion-time* after t
$C_{i,\pi(j),k}$	the completion-time of wafer k in job $\pi(j)$ at stage i
$C_{\max}(\pi)$	makespan of job sequence π

3.2. Evaluation procedure

The makespan evaluation procedure is governed by the following four equations:

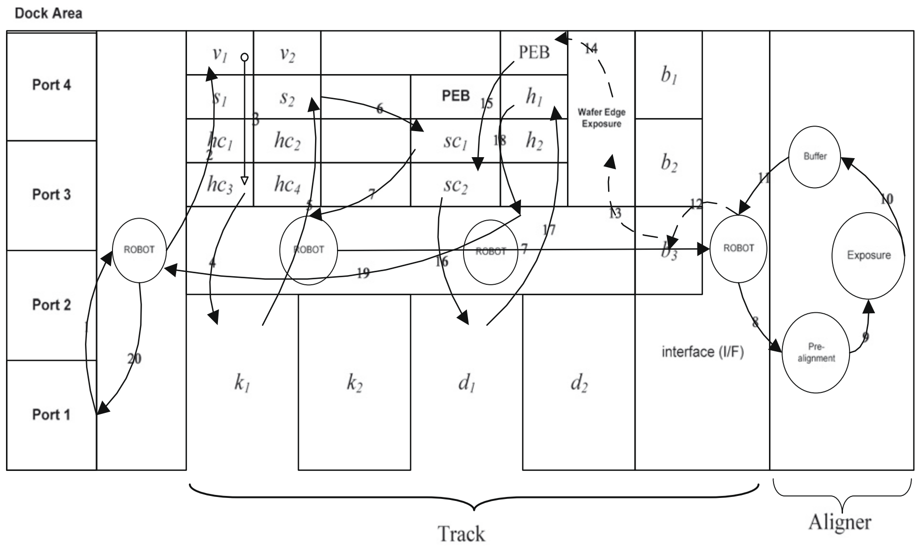

Fig. 2. Configuration of an in-line stepper. (a) Vapor prime: v_1–v_2, (b) cooling: hc_1–hc_4, (c) Coater: k_1–k_2, (d) softbake: s_1–s_2, (e) cooling: sc_1–sc_2, (f) buffer b_1–b_3, (g) cooling: pc_1–pc_2, (h) develop: d_1–d_2, (i) hard bake: h_1–h_2.

$$C_{i,\pi(j),k} = \min_{1\leqslant l\leqslant m_i}\{\max\{A_{i,l,t} + S_{i,l,\pi(j),k}, C_{i-1,\pi(j),k}\} + p_{i,l,\pi(j),k}\}$$

$$\text{where } t = C_{i-1,\pi(j),k} \quad \text{for } 1 \leqslant i \leqslant M \tag{1}$$

$$C_{M+1,\pi(j),w(\pi(j))} = C_{M,\pi(j),w(\pi(j))} + t_d \tag{2}$$

$$C_{M+1,\pi(j),w(\pi(j))} + t_u = C_{0,\pi(j+\rho),1} \quad \text{for } 1 \leqslant j \leqslant n-\rho \tag{3}$$

$$C_{\max}(\pi) = C_{M+1,\pi(n),w(\pi(n))} \tag{4}$$

Eq. (1) expresses the completion-time of a particular wafer at each stage i. The term $A_{i,l,t} + S_{i,l,\pi(j),k}$ denotes the time epoch when chamber l at stage i is ready for processing wafer k in job $\pi(j)$, and the term $C_{i-1,\pi(j),k}$ denotes the time epoch when the wafer is available to be processed at the chamber.

Eq. (2) describes the completion-time of job $\pi(j)$ at stage $M+1$, where we assume that a waiting-for-process wafer in the port is at stage 0; and a finished wafer in the port is at stage $M+1$.

Eq. (3) expresses the job arrival/departure relationships for the dock. The equation indicates that job $\pi(j+\rho)$ in the WIP buffer can be transported to the dock only when job $\pi(j)$ in the dock has been moved away. Eq. (4) computes the makespan $C_{\max}(\pi)$.

4. Algorithm

An algorithm called GA-Tabu, a combination of a GA (Holland, 1975) with a tabu search technique, is proposed to solve the in-line stepper scheduling problem. In the algorithm, a chromosome (a job sequence) is denoted by $\pi = [\pi(1),\ldots,\pi(n)]$ where $\pi(j)$, called a *gene*, represents the job in the jth position of sequence π. The *makespan* $C_{\max}(\pi)$ of the chromosome is called its fitness function. We first introduce the logic flow of the GA-Tabu algorithm, and proceed to describe each major component in the algorithm.

4.1. Logic flow

The logic flow of the proposed algorithm is described by a main procedure named **GA-Tabu**, in which a sub-procedure named **Tabu** will be called.

Procedure GA-Tabu

```
Step 1: Initialization
    Set $k = 0$ and $t = 0$. Randomly select $N_p$ chromosomes to form an initial population $P(t)$. Identify the best chromosome $\pi_b^0$ in $P(0)$.
    Set $\pi^o = \pi_b^0$ /*$\pi^o$ is the currently best ever solution; $k$ is the tenure of $\pi^o$ */
    Set tabu_list = $\phi$ /*tabu_list is a queue list of size $q$ */
    Set $\pi^* = \pi_b^0$ /*$\pi^*$ is a chromosome called tabu_seed */
Step 2: Update $P(t+1)$ by GA
    Use crossover operators to create a set $N_1(t)$ of $P_{cr} \cdot N_p$ new chromosomes.
    Use mutation operators to create a set $N_2(t)$ of $P_m \cdot N_p$ new chromosomes.
    Let $S(t) = P(t) \cup N_1(t) \cup N_2(t)$. From $S(t)$, use a selection strategy (the roulette wheel selection method by Michalewicz (1996) to select $N_p$ chromosomes to form $P(t+1)$).
Step 3: Update $\pi^o$ based on $\pi_{best}$, the best chromosome in $P(t+1)$
    From $P(t+1)$, identify the best chromosome $\pi_{best}$
    If $C_{\max}(\pi_{best}) \geqslant C_{\max}(\pi^o)$ set $k = k+1$, go to Step 4.
    If $C_{\max}(\pi_{best}) < C_{\max}(\pi^o)$, set $k = 0$ /*while $\pi_{best}$ is better than $\pi^o$ */
    Set $\pi^o = \pi_{best}$ /* update $\pi^o$ by $\pi_{best}$ */
    call Tabu($\pi_{best}, \pi^o$); /* further update $\pi^o$ by tabu */
Step 4: Update $\pi^o$ based on good chromosomes in $P(t+1)$ other than $\pi_{best}$
    If $k = 20$, call Tabu($\pi_{in}, \pi^o$), where $\pi_{in}$ is the 2nd best chromosome in $P(t+1)$
    If $k = 30$, call Tabu($\pi_{in}, \pi^o$); $\pi_{in}$ is the 3rd chromosome in $P(t+1)$
    If $k = 40$, call Tabu($\pi_{in}, \pi^o$); $\pi_{in}$ is the 4th chromosome in $P(t+1)$
Step 5: Update $\pi^o$ and $\pi^*$ based on the current tabu seed
    If ($k > 40$ & mod $(k, 5) = 0$), call Tabu($\pi^*, \pi^o$)
Step 6: Termination Check
    If ($k = K$ or $t = T$), then stop
    Otherwise, set $t = t+1$, go to Step 3.
```

Procedure Tabu(π_{in}, π_{out})

```
Step 1: Create a set $\Omega$ of new chromosomes.
  • Based on $\pi_{in}$, apply the steepest descent pairwise interchange (SDPI) (Armour & Buffa, 1963) technique to create a set $\Omega$ of new chromosomes. Each new chromosome $\pi$ is created by applying a particular interchange operation, represented by $move(\pi)$
Step 2: Update tabu_list
    From $\Omega$, identify the best $q$ chromosome $\pi_b^1, \ldots, \pi_b^q$
    For $i = 1, q$
        If $move(\pi_b^i) \notin$ tabu_list, then
            Place $move(\pi_b^i)$ in the tabu_list;
            Set $\pi_{out} = \pi_b^i$;
            Replace the worst chromosome in $P(t+1)$ by $\pi_{out}$ /*update $P(t+1)$ */
            Go to Step 3
        Endif
    Endfor
Step 3: Update $\pi^o$ and $\pi^*$
    If $C_{\max}(\pi_{out}) < C_{\max}(\pi^o)$, set $\pi^o = \pi_{out}$ and $k = 0$; /*update $\pi^o$ */
    If ($\pi_{in} = \pi^*$), set $\pi^* = \pi_{out}$; /*update $\pi^*$ */
    Return.
```

Procedure **GA-Tabu** is explained below. In Step 1, we create an initial population of chromosomes $P(0)$ and set the initial status of π^o and π^*, where π^o is the currently best ever chromosome and π^* is a chromosome called *tabu_seed*.

In Step 2, we update $P(t)$ by crossover operators, mutation operators and a selection strategy. Crossover and mutation operators are used to create new chromosomes. The existing and new chromosomes are then screened by the selection strategy in order to create $P(t+1)$ from $P(t)$.

In Step 3, we intend to update π^o based on π_{best}, the best chromosome in $P(t+1)$. If π_{best} is better than π^o, this implies that π_{best} is a good chromosome and its neighborhood could be exploited further. To do so, we call Procedure Tabu to create a new chromosome π_{out} from the neighborhood of π_{best}. Of the two chromosomes π_{best} and π_{out}, the better one if eligible is used to update π^o. If π^o is not updated in an iteration t, its tenure k is increased by one. The tenure k indicates how many times π^o has outperformed π_{best} while $P(t)$ is evolving. A large k value implies that update π^o based on current π_{best} in $P(t+1)$ appears to be difficult to a certain extent.

Therefore, in Step 4, some good chromosomes other than π_{best} in $P(t+1)$ are considered to update π^o. That is, we use the 2nd, 3rd, and 4th best chromosomes in $P(t+1)$ and call Procedure Tabu to exploit their neighborhood in order to create new chromosomes for possibly replacing π^o. In essence, both Steps 3 and 4 are intended to update π^o through the use of $P(t+1)$, which evolves based on the GA technique. While the value of k is too large (here we set $k > 40$), we could infer that updating π^o through the use of GA is now hard to a certain extent.

To remedy this deficiency, in Step 5, we alternatively use *tabu_seed* π^* and call Procedure Tabu to exploit its neighborhood in order to create new chromosomes for possibly replacing π^o. The evolution of π^* is independent of the GA. Therefore, π^* is likely far away from the neighborhood of the chromosomes in $P(t+1)$.

This characteristic helps the algorithm escape from a local optimal solution obtained by the GA.

Procedure **Tabu**(π_{in}, π_{out}) is designed to create a new chromosome π_{out}, which is one in the neighborhood of π_{in}. One purpose for obtaining π_{out} is for updating π^o and $P(t+1)$. The input π_{in} has two possible sources: $P(t+1)$ or *tabu_seed* π^*. The source π^* is designed to evolve through this procedure. Therefore, we additionally use π_{out} to update π^* while $\pi_{in} = \pi^*$. In summary, Procedure Tabu has three functions: updating π^o, π^* and $P(t+1)$.

4.2. Crossover and mutation operators

To create new chromosomes, we use four types of crossover operators and three types of mutation operators. A crossover operator is designed to create a new pair from an existing pair, while a mutation operator is to create a new one from an existing one. The four types of crossover operators are **C1** (one point crossover) by Reeves (1995), **LOX** (linear order crossover) by Croce, Tadei, and Volta (1995), **PMX** (partially mapped crossover) by Goldberg (1989), and **NABEL** operator by Bac and Perov (1993). The three types of mutation operators are **Swap**, **Inverse**, and **Insert** (Nearchou, 2004; Wang & Zheng, 2003; Zhang, Wang, & Zheng, 2006).

The four crossover operators are explained below, where *parent-1* and *parent-2* denote parent chromosomes while *child-1* and *child-2* denote the created ones.

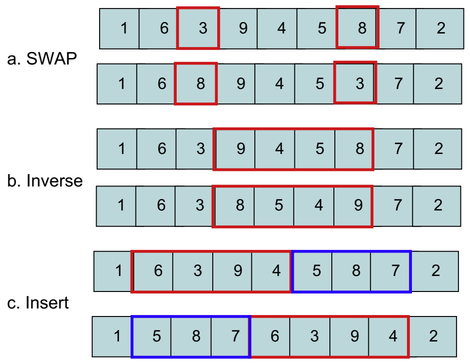

Fig. 4. Mutation operators: (a) SWAP, (b) Inverse, (c) Insert.

C1 operator: Referring to Fig. 3a, one randomly selected point is used to divide each parent into two sections (*head* and *tail*). To create an offspring (say, *child-2*), its *head* is copied from the head of *parent-2*—a string (3, 5, 6) in this case. Its *tail* is determined by sequentially referring to the genes of *parent-1*; only the gene values not in the *head* of *child-2* are eligible to appear in the tail. This results in a string (1, 9, 4, 8, 7, 2) as the *tail* of *child-2*.

LOX operator: Referring to Fig. 3b, each parent is randomly divided into three sections (*head, middle* and *tail*). The *middle* for *parent-1* and *parent-2* are (3, 9, 4) and (6, 1, 8) respectively. A

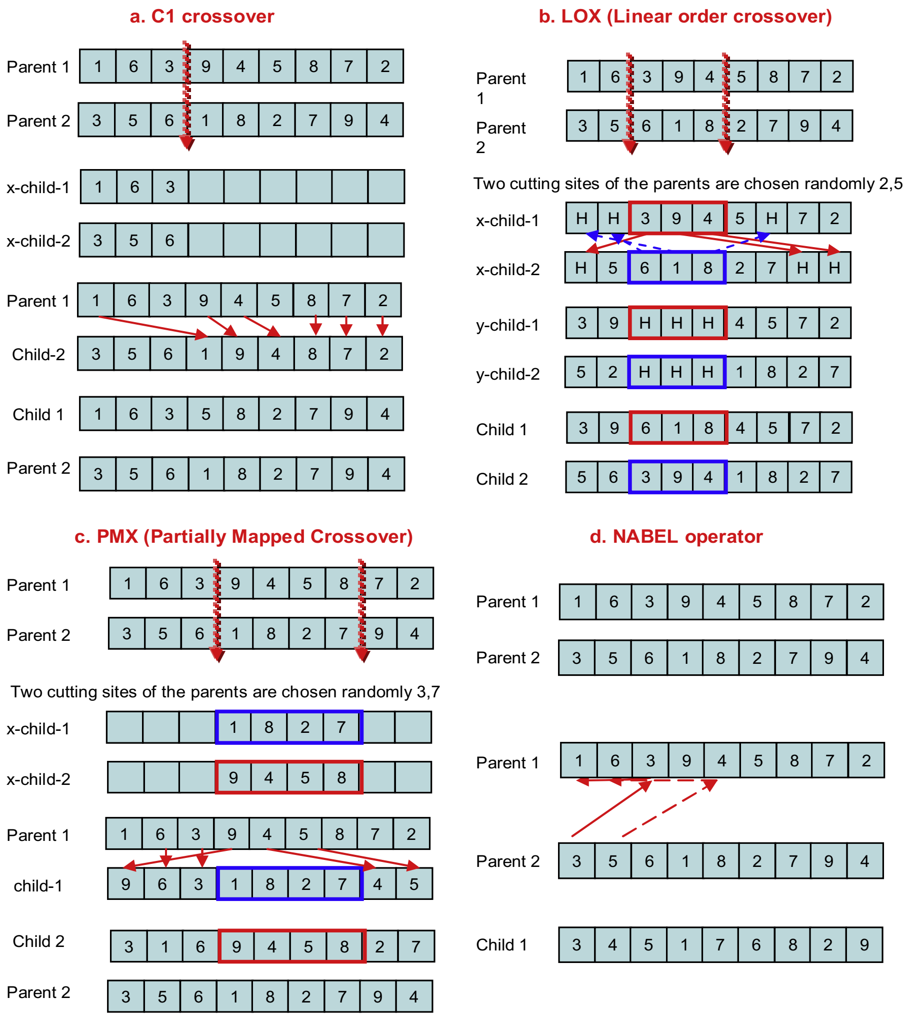

Fig. 3. Crossover operators: (a) C1, (b) PMX, (c) LOX, (d) NABEL.

chromosome *x-child-1* is created by two steps. First, its *middle* is copied from the *middle* of *parent-1*. Second, its genes values that appear in the *middle* of *parent-2* are replaced by "H". This yields *x-child-1* = (H, H, $\underline{3, 9, 4}$, 5, H, 7, 2). We then manipulate *x-child-1* by moving "H" to the middle and yield a chromosome *y-child-1* = (3, 9, $\underline{\text{H, H, H}}$, 4, 5, 7, 2). Finally, the *middle* of *y-child-1* is replaced by that of *parent-2*; this yield *child-1* = (3, 9, $\underline{6, 1, 8}$, 4, 5, 7, 2).

PMX operator: Referring to Fig. 3c, each parent is randomly divided into three sections (*head*, *middle*, and *tail*). A new offspring (e.g., *child-1*) is created by the following procedure. The *middle* of *child-1* is created by referring to that of *parent-2*, which is a string (1, 8, 2, 7). Both the head and tail of *child-1* are created by referring to *parent-1*. If the gene values in the head/tail sections of *parent-1* do not appear in *child-1*, we copy them in the exact positions of *child-1* (e.g., gene values 6 and 3). Finally, for those *vacant* genes in *child-1*, we place their values by sequentially referring to the *unassigned* genes in *parent-1*. This yields *child-1* = (9, 6, 3, $\underline{1, 8, 2, 7}$, 4, 5).

NABEL operator: Suppose the two parent chromosomes are $\pi_{p1} = [\pi_{p1}(1), \ldots, \pi_{p1}(n)]$ and $\pi_{p2} = [\pi_{p2}(1), \ldots, \pi_{p2}(n)]$; and the two children chromosomes to be created are $\pi_{c1} = [\pi_{c1}(1), \ldots, \pi_{c1}(n)]$ and $\pi_{c2} = [\pi_{c2}(1), \ldots, \pi_{c2}(n)]$. Referring to Fig. 3d, the NABEL operator is designed to set $\pi_{c1}(i) = \pi_{p1}(\pi_{p2}(i))$ and $\pi_{c2}(i) = \pi_{p2}(\pi_{p1}(i))$. For example, $\pi_{p2}(2) = 5$ and $\pi_{p1}(\pi_{p2}(2)) = 4$, and we can obtain $\pi_{c1}(2) = \pi_{p1}(\pi_{p2}(2)) = 4$.

The three mutation operators are explained below, where π_a denotes the parent chromosome while π_b denotes the child one.

Swap operator: Referring to Fig. 4a, we randomly choose two distinct genes in π_a, and then swap their gene values to create π_b.

Inverse operator: Referring to Fig. 4b, we randomly select two cut-off points in π_a to divide it into three sections. Represent $\pi_a = \{\pi_{a1}, \pi_{a2}, \pi_{a3}\}$ and $\pi_{a2} = [\pi_{a2}(1), \ldots, \pi_{a2}(m)]$. The inverse operator is designed to create a new chromosome $\pi_b = \{\pi_{a1}, \pi_{b2}, \pi_{a3}\}$ where $\pi_{b2}(i) = \pi_{a2}(m + 1 - i)$.

Insert operator: Referring to Fig. 4c, we randomly select an *insert point* and a *segment* of genes in π_a. This would divide π_a into four sections $\pi_a = \{\pi_{a1} \mid \pi_{a2}, \boxed{\pi_{a3}}, \pi_{a4}\}$ which denote that the insert point is between π_{a1} and π_{a2}, and the selected segment is π_{a3}. To create a new chromosome, the insert operator moves the selected segment to the insert point position. This would yield a new chromosome $\pi_b = \{\pi_{a1}, \boxed{\pi_{a3}} \mid \pi_{a2}, \pi_{a4}\}$.

4.3. Procedure Tabu

In Step 1 of Procedure Tabu, a set Ω of new chromosomes is created by the SDPI technique. Given a chromosome π_{in}, the technique creates a new one by choosing a pair of genes from π_{in} and interchanges their values. With n genes inπ_{in}, Ω would include $C(n, 2)$ new chromosomes. For a new chromosome $\pi = [\pi(1), \ldots, \pi(i) \ldots, \pi(j), \ldots, \pi(n)]$, where $\pi(i)$and $\pi(j)$ are the two genes that are interchanged. Then, its interchange operation $move(\pi)$ is represented $\{\pi(i), \pi(j)\}$.

In Step 2 of the procedure, out of Ω, only the best possible chromosome created by a "new" move is eligible for updating π^o, π^*, and $P(t + 1)$. Herein, a "new" move is one, currently not in the tabu_list. This implies that the tabu_list is designed to record "good" and "new" moves. Such a design is to avoid a cyclic creation of the same chromosome. This would keep π^o and π^* away from being trapped into a local optima.

5. Numerical experiments

By numerical experiments, we justify the effectiveness of the proposed GA-Tabu algorithm. The in-line stepper is assumed to have four ports, 14 stages and 21 chambers. The operation time at each chamber i follows a uniform distribution $[a_i, b_i]$ (Table 1). A mask setup is always required for the exposure chamber while it turns to process a new job's wafer, and the mask change time is a constant (1.0 min). Parameters in the GA-Tabu are set as follows: $P = 100$, $P_{cr} = 0.8$, $P_{mu} = 0.2$, $q = 7$, $K = 3000$, $T = 100{,}000$.

5.1. Test cases

To model the process yield in experiments, we use a *truncated binomial distribution* (TBD). The TBD implies that the job size is governed by a *binomial distribution*; however, the jobs carrying no wafer are moved away from the fab.

We use (N, Y) to represent a test case, where N represents the number of jobs and Y represents the average yield. In our experiments, N involves five options ranging from 20 to 100 jobs while Y involves 10 options ranging from 15% to 90% (Table 2–6). That is, each algorithm has 50 test cases, and each test case is justified by 15 replicates.

5.2. Benchmark algorithms

For each test case, we compare the GA-Tabu with seven other algorithms: optimum heuristic rule (OHR), simulated annealing (SA) by Osman and Potts (1989), GA by Wu and Chiou (in press), tabu search (TS) by Widmer and Hertz (1989), two ant colony algorithms (ACO) by Rajendran and Ziegler (2004), and particle swarm optimization (PSO) by Liaoa, Tseng, and Luarnb (2007). The OHR algorithm denotes taking the best result out of three heuristic scheduling rules: SPT (shortest job processing time), LPT (longest job processing time), and NEH (Nawaz, Enscore, & Ham, 1983). The two ACOs are respectively called MMAX and PACO.

For each algorithm in each test case, the average makespan of the 15 replicates is taken as the performance. Define the average makespan of each algorithm as follows: $C_{GA\text{-}tabu}$ for the GA-Tabu, C_{OHR} for the OHR, C_{SA} for the SA, C_{GA} for the GA, C_{TS} for the TS, C_{MMAX} for the MMAX, C_{PACO} for the PACO, and C_{PSO} for the PSO. The computation time for each algorithm is defined accordingly; for example, that for the GA-Tabu is defined as $t_{GA\text{-}tabu}$.

5.3. Experiment results

To compare the solution quality between the GA-Tabu and a benchmark algorithm (say, x), a performance metric is so defined: $\gamma_x = (C_x - C_{GA\_Tabu})/C_{GA\_Tabu}$. A positive γ_x indicates that the GA-Tabu

Table 1
Process times of in-line stepper chambers.

Process sequence	WIP buffers to dock area	Dock area to track	HMDS	Cooling	Coater	Softbake	Cooling	Aligner	Wafer edge exposure	PEB	Cooling	Develop	Hard bake	High cooling
Chamber number	1	1	2	2	2	2	2	1	1	2	2	2	2	1
Process time (min)	2.5	0.1	1.2	1.2	1.2	[1.2,2.8]	1	[0.75,1.65]	1	[1.2,2.8]	1	[1.2,2.8]	[1.2,2.8]	0.5

Table 2
Makespan comparison at different binomial yield scenarios for job 20.

Jobs	20																								
	GA_Tabu		OHR					GA			SA			Tabu			MMAX			PACO			PSO		
Yield (%)	$C_{GA\text{-}tabu}$ (min)	$t_{GA\text{-}tabu}$ (s)	Random (min)	SPT (min)	LPT (min)	NEH (min)	C_B (min)	C_{GA} (min)	γ_{GA} (%)	t_{GA} (s)	C_{SA} (min)	γ_{SA} (%)	t_{SA} (s)	C_{tabu} (min)	γ_{tabu} (%)	t_{tabu} (s)	$C_{\max}$ (min)	$\gamma_{\max}$ (%)	$t_{\max}$ (s)	C_{paco} (min)	γ_{paco} (%)	t_{paco} (s)	C_{PSO} (min)	γ_{PSO} (%)	T_{pso} (s)
90	562.0	109	572.6	571.6	565.1	565.2	565.1	562.1	0.01	251	563.2	0.21	9	562.0	0.00	70.3	562.5	0.08	31	562.6	0.10	23	566.7	0.84	2
80	529.8	100	539.9	537.1	532.9	534.2	532.9	529.8	0.00	230	530.8	0.18	8	529.8	0.00	70.0	530.0	0.03	29	530.2	0.08	21	534.2	0.82	2
70	448.0	101	458.0	454.2	450.9	453.1	450.9	448.0	0.00	215	448.8	0.18	7	448.0	0.00	69.7	448.3	0.08	24	448.4	0.09	18	452.4	0.99	1
60	391.2	89	400.5	398.4	395.1	394.4	394.4	391.3	0.01	175	392.2	0.26	7	391.2	0.00	69.4	391.6	0.09	21	391.6	0.10	16	396.0	1.22	1
50	349.8	84	359.1	356.5	353.3	353.6	353.3	349.8	0.00	214	350.9	0.30	5	349.8	0.00	68.9	350.0	0.05	19	350.1	0.09	14	354.4	1.31	1
40	268.4	62	277.0	276.6	272.4	272.6	272.4	268.4	0.00	114	269.6	0.43	5	268.4	0.00	68.2	268.8	0.13	14	268.8	0.14	11	272.9	1.65	1
30	215.4	59	224.8	222.9	220.0	219.2	219.2	215.4	0.00	137	216.3	0.41	4	215.4	0.00	67.0	215.7	0.12	11	215.6	0.06	8	220.6	2.41	1
25	161.8	47	174.9	170.9	168.9	168.5	168.5	161.9	0.05	132	163.9	1.29	3	161.8	0.00	67.0	162.0	0.11	8	162.2	0.24	6	167.9	3.74	1
20	138.6	44	159.2	150.1	151.3	149.2	149.2	139.4	0.54	131	144.1	3.98	3	138.6	0.01	66.1	139.3	0.50	7	140.4	1.28	5	148.3	6.99	1
15	125.8	42	146.3	143.4	141.0	139.7	139.7	127.4	1.28	131	132.0	4.92	3	126.3	0.37	66.0	127.7	1.49	6	127.8	1.57	5	136.4	8.38	1

Table 3
Makespan comparison at different binomial yield scenarios for job 40.

Jobs	40																								
	GA_Tabu		OHR					GA			SA			Tabu			MMAX			PACO			PSO		
Yield (%)	$C_{GA\text{-}tabu}$ (min)	$t_{GA\text{-}tabu}$ (s)	Random (min)	SPT (min)	LPT (min)	NEH (min)	C_B (min)	C_{GA} (min)	γ_{GA} (%)	t_{GA} (s)	C_{SA} (min)	γ_{SA} (%)	t_{SA} (s)	C_{tabu} (min)	γ_{tabu} (%)	t_{tabu} (s)	$C_{\max}$ (min)	$\gamma_{\max}$ (%)	$t_{\max}$ (s)	C_{paco} (min)	γ_{paco} (%)	t_{paco} (s)	C_{PSO} (min)	γ_{PSO} (%)	T_{pso} (s)
90	1155.6	381	1170.0	1170.6	1158.7	1160.6	1158.7	1155.6	0.00	722	1158.0	0.21	30	1155.6	0.00	160	1155.9	0.03	261	1155.8	0.01	185	1163.5	0.68	7
80	1001.0	333	1015.7	1011.9	1004.4	1008.0	1004.4	1001.0	0.00	635	1001.o	0.00	31	1001.0	0.00	155	1001.2	0.02	226	1001.4	0.04	160	1008.8	0.78	7
70	901.3	294	914.7	910.7	905.4	906.8	905.4	901.3	0.00	674	901.2	−0.01	30	901.3	0.00	151	901.8	0.06	203	901.8	0.06	143	909.5	0.91	6
60	773.7	276	787.9	781.5	778.3	781.4	778.3	773.7	0.00	553	775.1	0.19	28	773.7	0.00	146	774.0	0.04	173	773.9	0.03	122	781.8	1.04	6
50	677.8	261	691.0	685.7	681.8	683.1	681.8	677.8	0.00	488	685.6	1.15	26	677.8	0.00	143	678.0	0.02	150	677.9	0.02	106	685.7	1.17	6
40	519.2	182	531.9	527.2	523.8	525.7	523.8	519.2	0.00	359	526.2	1.35	24	519.2	0.00	138	519.6	0.08	116	519.4	0.05	81	526.9	1.49	6
30	405.9	150	419.0	413.8	411.3	411.9	411.3	405.9	0.01	334	405.8	−0.01	22	405.9	0.01	133	406.2	0.08	89	406.0	0.05	63	413.5	1.90	5
25	348.6	150	367.3	358.8	356.1	357.3	356.1	348.6	0.02	419	351.6	0.86	21	348.6	0.00	130	348.9	0.10	76	348.9	0.09	54	358.9	2.97	5
20	309.7	128	332.8	319.7	319.3	314.4	314.4	309.7	0.00	386	292.6	−5.52	20	309.7	0.01	129	309.8	0.05	69	309.8	0.05	48	313.6	1.25	5
15	247.3	104	290.7	268.2	267.2	272.9	267.2	251.4	1.65	649	261.4	5.69	19	247.9	0.21	124	251.2	1.55	48	251.7	1.78	34	273.0	10.37	5

Table 4
Makespan comparison at different binomial yield scenarios for job 60.

Jobs	60																								
	GA_Tabu		OHR					GA			SA			Tabu			MMAX			PACO			PSO		
Yield (%)	$C_{GA\text{-}tabu}$ (min)	$t_{GA\text{-}tabu}$ (s)	Random (min)	SPT (min)	LPT (min)	NEH (min)	C_B (min)	C_{GA} (min)	γ_{GA} (%)	t_{GA} (s)	C_{SA} (min)	γ_{SA} (%)	t_{SA} (s)	C_{tabu} (min)	γ_{tabu} (%)	t_{tabu} (s)	$C_{\max}$ (min)	$\gamma_{\max}$ (%)	$t_{\max}$ (s)	C_{paco} (min)	γ_{paco} (%)	t_{paco} (s)	C_{PSO} (min)	γ_{PSO} (%)	T_{pso} (s)
90	1707.1	961	1726.0	1730.6	1710.2	1714.3	1710.2	1707.1	0.00	1670	1710.9	0.23	25	1707.1	0.00	527	1707.2	0.00	876	1707.3	0.01	618	1718.7	0.68	17
80	1503.7	838	1521.8	1519.5	1508.0	1510.8	1508.0	1503.7	0.00	1245	1513.3	0.64	23	1503.7	0.00	511	1503.9	0.02	773	1503.7	0.00	542	1514.4	0.71	17
70	1323.1	792	1341.1	1335.8	1328.1	1329.8	1328.1	1323.1	0.00	1549	1334.5	0.87	22	1323.1	0.00	495	1323.1	0.00	679	1323.3	0.02	475	1334.3	0.85	16
60	1192.7	694	1210.2	1201.3	1197.3	1199.6	1197.3	1192.7	0.00	1091	1203.0	0.86	19	1192.7	0.00	483	1192.9	0.02	612	1192.8	0.00	427	1203.6	0.92	16
50	960.6	517	977.2	967.4	964.8	967.3	964.8	960.6	0.00	980	970.8	1.06	18	960.6	0.00	466	960.9	0.04	487	960.8	0.02	339	971.5	1.14	15
40	794.1	433	811.1	801.4	798.6	800.5	798.6	794.1	0.00	771	804.8	1.35	16	794.1	0.00	452	794.4	0.05	401	794.3	0.03	278	805.5	1.44	15
30	566.7	324	586.4	578.0	575.5	577.2	575.5	566.7	0.00	726	570.1	0.60	14	566.7	0.00	431	566.7	0.01	282	566.9	0.04	197	578.9	2.15	15
25	472.7	284	502.6	486.1	482.2	489.4	482.2	472.8	0.03	858	485.2	2.66	13	472.7	0.01	421	472.8	0.04	240	473.1	0.10	166	490.9	3.86	15
20	417.3	281	462.3	445.8	444.1	452.3	444.1	419.2	0.44	1384	434.8	4.18	11	417.8	0.11	417	419.8	0.60	216	421.1	0.90	149	450.0	7.82	14
15	367.7	212	454.9	400.2	399.4	397.2	397.2	370.0	0.61	1595	386.7	5.16	10	368.7	0.28	411	373.7	1.62	162	374.5	1.83	112	389.8	6.00	14

Table 5
Makespan comparison at different binomial yield scenarios for job 80.

Jobs	80																								
	GA_Tabu		OHR					GA			SA			Tabu			MMAX			PACO			PSO		
Yield (%)	$C_{GA\text{-}tabu}$ (min)	$t_{GA\text{-}tabu}$ (s)	Random (min)	SPT (min)	LPT (min)	NEH (min)	C_B (min)	C_{GA} (min)	γ_{GA} (%)	t_{GA} (s)	C_{SA} (min)	γ_{SA} (%)	t_{SA} (s)	C_{tabu} (min)	γ_{tabu} (%)	t_{tabu} (s)	$C_{\max}$ (min)	$\gamma_{\max}$ (%)	$t_{\max}$ (s)	C_{paco} (min)	γ_{paco} (%)	t_{paco} (s)	C_{PSO} (min)	γ_{PSO} (%)	T_{pso} (s)
90	2274.0	1851	2297.0	2305.3	2277.7	2282.8	2277.7	2274.0	0.00	2502	2288.4	0.63	49	2274.0	0.00	1210	2274.4	0.02	2084	2274.2	0.01	1474	2289.0	0.66	34
80	2035.1	1731	2056.4	2055.0	2038.8	2043.7	2038.8	2035.1	0.00	2722	2049.6	0.71	47	2035.1	0.00	1174	2035.3	0.01	1857	2035.1	0.00	1309	2049.2	0.69	33
70	1805.3	1659	1827.3	1818.1	1809.8	1813.3	1809.8	1805.3	0.00	2053	1819.6	0.80	43	1805.3	0.00	1139	1805.4	0.01	1648	1805.3	0.00	1158	1819.7	0.80	33
60	1539.1	1352	1561.2	1547.4	1544.4	1547.7	1544.4	1539.1	0.00	1767	1553.1	0.91	39	1539.1	0.00	1098	1539.3	0.02	1393	1539.3	0.02	978	1553.1	0.91	32
50	1329.6	1130	1351.3	1338.9	1335.0	1337.7	1335.0	1329.6	0.00	1726	1344.5	1.12	36	1329.6	0.00	1074	1329.7	0.00	1205	1329.8	0.02	844	1343.6	1.05	32
40	1040.5	864	1061.9	1053.1	1050.9	1050.0	1050.0	1040.5	0.00	1109	1055.0	1.39	32	1040.5	0.00	1034	1040.7	0.02	954	1040.8	0.03	660	1054.7	1.36	31
30	834.9	708	855.2	844.0	841.5	843.5	841.5	834.9	0.00	1262	848.8	1.67	28	834.9	0.00	998	834.9	0.00	744	835.0	0.02	521	8485	1.63	31
25	675.1	601	703.1	688.1	684.3	685.3	684.3	675.1	0.01	1315	695.5	3.03	26	675.1	0.00	967	675.2	0.03	610	675.5	0.07	426	693.8	2.78	30
20	588.5	529	639.0	616.3	614.9	625.6	614.9	588.9	0.07	1997	624.5	6.12	24	588.5	0.01	954	589.2	0.12	543	590.0	0.25	374	623.0	5.86	30
15	484.2	392	727.8	517.1	517.2	525.1	517.1	487.1	0.59	2126	504.0	4.08	21	484.6	0.07	931	491.9	1.59	366	492.8	1.76	253	515.5	6.45	30

Table 6
Makespan comparison at different binomial yield scenarios for job 100.

Jobs	100																								
	GA_Tabu		OHR					GA			SA			Tabu			MMAX			PACO			PSO		
Yield (%)	$C_{GA\text{-}tabu}$ (min)	$t_{GA\text{-}tabu}$ (s)	Random (min)	SPT (min)	LPT (min)	NEH (min)	C_B (min)	C_{GA} (min)	γ_{GA} (%)	t_{GA} (s)	C_{SA} (min)	γ_{SA} (%)	t_{SA} (s)	C_{tabu} (min)	γ_{tabu} (%)	t_{tabu} (s)	$C_{\max}$ (min)	$\gamma_{\max}$ (%)	$t_{\max}$ (s)	C_{paco} (min)	γ_{paco} (%)	t_{paco} (s)	C_{PSO} (min)	γ_{PSO} (%)	T_{pso} (s)
90	2826.8	3402	2853.9	2862.2	2832.4	2837.3	2832.4	2826.8	0.00	4002	2845.0	0.64	31	2826.8	0.00	2328	2826.8	0.00	4053	2827.0	0.01	2851	2844.4	0.63	58
80	2537.5	3348	2564.6	2563.0	2540.8	2547.1	2540.8	2537.5	0.00	3250	2555.5	0.71	28	2537.5	0.00	2261	2537.5	0.00	3636	2537.7	0.01	2546	2554.8	0.68	58
70	2257.0	2734	2282.2	2273.1	2260.5	2267.2	2260.5	2257.0	0.00	3459	2274.6	0.78	25	2257.0	0.00	2193	2257.1	0.01	3229	2257.1	0.01	2253	2274.6	0.78	57
60	1936.4	2294	1961.2	1946.5	1941.8	1946.1	1941.8	1936.4	0.00	2389	1954.0	0.91	22	1936.4	0.00	2117	1936.6	0.01	2769	1936.7	0.02	1920	1954.0	0.91	57
50	1582.9	1868	1608.7	1593.5	1589.2	1593.5	1589.2	1582.9	0.00	2454	1600.4	1.10	18	1582.9	0.00	2041	1583.0	0.01	2257	1583.1	0.01	1568	1599.9	1.07	56
40	1331.6	1711	1357.6	1342.0	1338.1	1341.3	1338.1	1331.6	0.00	2183	1350.1	1.39	16	1331.6	0.00	1988	1331.7	0.01	1886	1331.7	0.00	1308	1349.4	1.33	55
30	1012.6	1231	1039.1	1023.4	1017.4	1022.7	1017.4	1012.6	0.00	1780	1030.1	1.74	12	1012.6	0.00	1900	1012.7	0.02	1405	1012.8	0.02	984	1030.2	1.74	55
25	899.8	1153	929.1	911.0	908.9	911.0	908.9	899.8	0.00	1760	921.1	2.36	11	899.8	0.00	1870	900.0	0.03	1258	900.0	0.03	882	919.3	2.17	55
20	736.0	961	792.8	759.7	759.3	770.0	759.3	736.5	0.06	2952	783.2	6.42	9	736.0	0.00	1825	736.3	0.04	1055	739.3	0.45	728	778.9	5.83	54
15	610.6	681	711.5	648.5	646.9	652.7	646.9	616.4	0.94	3151	634.3	3.88	6	612.1	0.25	1783	619.2	1.40	740	624.1	2.20	511	648.9	6.27	54

is better than the benchmark, while a negative one denotes the GA-Tabu is worse.

From Tables 2–6, we could see that all γ_x ranges from 0% to 10%. This indicates that the GA-Tabu outperforms all the benchmark algorithms, in terms of solution quality. Notice that this merit appears more impressive in low-yield scenarios than in high-yield scenarios. The GA-Tabu is relatively computationally extensive. However, compared to the TS and the GA (the 2nd and 3rd best ones in terms of solution quality), the GA-Tabu is faster computationally.

The experiment results indicate that the proposed GA-Tabu has its merit—in particular in a low-yield scenario. With in-line steppers as the bottleneck of a fab, even a 1% increase in the in-line stepper throughput would have a substantial positive impact on gross margins.

6. Concluding remarks

This study examines a scheduling problem for a semiconductor in-line stepper, with makespan as the performance criterion. We propose a meta-heuristic algorithm, called GA-Tabu, to solve the problem. Seven other scheduling algorithms are compared with the GA-Tabu by numerical experiments. Experiment results indicated that the GA-Tabu outperforms all the benchmarks in terms of solution quality. This merit is in particular more impressive in low-yield scenarios.

One extension to this research is the scheduling of two or more in-line steppers. Such an extension would involve one more decision-making—how to allocate jobs to each in-line stepper. Another extension is the configuration design for an in-line stepper, for example, determining the optimum number of ports.

References

Armour, G., & Buffa, E. (1963). A heuristic algorithm and simulation approach to the relative location of facilities. *Management Science, 9*, 294–309.

Bac, F. Q., & Perov, V. L. (1993). New evolutionary genetic algorithms for NP-complete combinatorial optimization problems. *Biological Cybernetics, 69*, 229–234.

Chern, C. C., & Liu, Y. L. (2003). Family-based scheduling rules of a sequence-dependant wafer fabrication system. *IEEE Transactions on Semiconductor Manufacturing, 16*(1), 15–25.

Chiang, T. C., & Fu, L. C. (2008). Using a family of critical ratio-based approaches to minimize the number of tardy jobs in the job shop with sequence dependent setup times. *European Journal of Operational Research*. doi:10.1016/j.ejor.2007.12.042.

Chung, S. H., & Hsieh, M. H. (2008). Long-term tool elimination planning for a wafer fab. *Computers and Industrial Engineering, 54*, 589–601.

Croce, F. D., Tadei, R., & Volta, G. (1995). A genetic algorithm for the job shop problem. *Computers and Operations Research, 22*(1), 15–24.

Dabbas, R. M., & Fowler, J. W. (2003). A new scheduling approach using combined dispatching criteria in wafer fabs. *IEEE Transactions on Semiconductor Manufacturing, 16*, 3.

Duwayri, Z., Mollaghasemi, M., Nazzal, D., & Rabadi, G. (2006). Scheduling setup changes at bottleneck workstations in semiconductor manufacturing. *Production Planning and Control, 17*(7), 717–727.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Boston: Addison-Wesley.

Holland, J. H. (1975). *Adaptation in neural and artificial systems*. Ann Arbor, MI: Univ. Michigan Press.

Liaoa, C. J., Tseng, C. T., & Luarnb, P. (2007). A discrete version of particle swarm optimization for flowshop scheduling problems. *Computers and Operations Research, 34*, 3099–3111.

Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs* (3rd ed.). Berlin, Heidelberg, New York: Springer.

Morrison, J. R., & Martin, D. P. (2007). Performance evaluation of photolithography cluster tools. *OR Spectrum, 33*, 375–389.

Nawaz, M., Enscore, J. E. E., & Ham, I. (1983). A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem. *OMEGA, The International Journal of Management Science, 11*(1), 91–95.

Nearchou, A. C. (2004). The effect of various operators on the genetic search for large scheduling problems. *International Journal of Production Economics, 88*, 191–203.

Osman, I. H., & Potts, C. N. (1989). Simulated annealing for permutation flow-shop scheduling. *OMEGA, The International Journal of Management Science, 17*(6), 551–557.

Quirk, M. (2001). *Semiconductor manufacturing technology*. Prentice Hall.

Rajendran, C., & Ziegler, H. (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research, 155*, 426–438.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computers and Operations Research, 22*(1), 5–13.

Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research, 169*, 781–800.

Sha, D. Y., Hsu, S. Y., Che, Z. H., & Chen, C. H. (2006). A dispatching rule for photolithography scheduling with an on-line rework strategy. *Computers and Industrial Engineering, 50*, 233–247.

Wang, L., & Zheng, D. Z. (2003). An effective hybrid heuristic for flowshop scheduling. *International Journal of Advanced Manufacturing Technology, 21*(1), 38–44.

Widmer, M., & Hertz, A. (1989). A new heuristic method for the flow shop sequencing problem. *European Journal of Operational Research, 41*, 186–193.

Wu, M. C., & Chang, W. J. (2007). A short-term capacity trading method for semiconductor fabs with partnership. *Expert Systems with Applications, 33*, 476–483.

Wu, M. C., & Chang, W. J. (2008). A multiple criteria decision for trading capacity between two semiconductor fabs. *Expert Systems with Applications, 35*, 938–945.

Wu, M. C., & Chiou, C. W. (2009). Scheduling semiconductor in-line steppers in new product/process introduction scenarios. *International Journal of Production Research*. doi:10.1080/00207540802577920.

Wu, M. C., Chiou, S. J., & Chen, C. F. (2007). Dispatching for make-to-order wafer fabs with machine-dedication and mask set-up characteristics. *International Journal of Production Research*, 1–17.

Wu, M. C., Huang, Y. L., Chang, Y. C., & Yang, K. F. (2006). Dispatching in semiconductor fabs with machine-dedication features. *International Journal of Advanced Manufacturing Technology, 28*, 978–984.

Wu, M. C., Jiang, J. H., & Chang, W. J. (2008). Scheduling a hybrid MTO/MTS semiconductor fab with machine-dedication features. *International Journal Production Economics, 112*, 416–426.

Xiao, H. (2001). *Introduction to semiconductor manufacturing technology*. Prentice Hall.

Ying, K. C., & Lin, S. W. (2009). Raising the hit rate for wafer fabrication by a simple constructive heuristic. *Expert Systems with Applications*, *36*(2P2), 2894–2900.

Zhang, L., Wang, L., & Zheng, D. Z. (2006). A adaptive genetic algorithm with multiple operators for flowshop scheduling. *International Journal of Advanced Manufacturing Technology, 27*, 580–587.