OR Spectrum 29:391–419 (2007)
DOI 10.1007/s00291-006-0062-3

REGULAR ARTICLE

Chen-Fu Chien · Chien-Hung Chen

A novel timetabling algorithm for a furnace process for semiconductor fabrication with constrained waiting and frequency-based setups

Published online: 3 November 2006

Abstract This study aims to solve the scheduling problem arising from oxide–nitride–oxide (ONO) stacked film fabrication in semiconductor manufacturing. This problem is characterized by waiting time constraints, frequency-based setups, and capacity preoccupation. To the best of our knowledge, none of the existing studies has addressed constrained waiting time and frequency-based setups at the same time. To fill this gap, this study develops a genetic algorithm for batch sequencing combined with a novel timetabling algorithm. For validation, we conducted several experiments based on empirical data. As a benchmark for small-sized problem instances, a mixed-integer linear programming model was used. The results show that the proposed algorithm optimally solves most cases of the ONO scheduling problem in real settings and significantly outperforms dispatching rule-based heuristics.

Keywords Scheduling · Genetic Algorithm · Frequency-based setup · Waiting time constraint · Semiconductor manufacturing

1 Introduction

Semiconductor fabrication facilities (fabs) are the most capital-intensive and complex manufacturing plants today in which similar equipment and processes are used to produce integrated circuits including microprocessors, memories, digital signal processor, and application-specific logic devices. Semiconductor companies are facing global competition in selling interchangeable products or providing

C.-F. Chien (✉) · C.-H. Chen
Department of Industrial Engineering and Engineering Management,
National Tsing Hua University, Hsinchu, 30013, Taiwan
E-mail: cfchien@mx.nthu.edu.tw

foundry service with similar technologies to various customers worldwide (Leachman and Hodges 1996). Semiconductor manufacturing process primarily consists of four phases: wafer fabrication, wafer probe, assembly and packing, and final test (Uzsoy et al. 1992). Wafer fabrication involves the most complex and lengthy process including oxidation/deposition/metallization, lithography, etching, ion implantation, photo-resist strip, cleaning, inspection, and measurement. In particular, the oxidation, deposition, and metallization are processed in the furnace area. Although the lithography area is generally the bottleneck owing to its extremely expensive equipment, the furnace equipment takes about 20% of the total cost in a semiconductor factory, and the processing time spent in the furnace area is longer than that in other areas (Lin and Huang 1998). Therefore, effective and efficient production scheduling and management in the furnace area is important to improve tool productivity and maintain the competitive advantage of operation efficiency. Yet little research has been done to address the scheduling problem arising from oxide–nitride–oxide (ONO) stacked film fabrication.

Focusing on real settings, this study formulates the ONO stacked film fabrication process as an ONO scheduling problem. The fabrication process in the furnace area can be described as a typical job shop scheduling problem (JSSP) in an unrelated parallel machines environment. The furnace is considered as the machine, and the batch aggregated from several lots of wafer is defined as a job. Indeed, the batch is referred to as a parallel batch as a lot becomes available when the whole batch to which it belongs has been processed. In other words, the batch availability (Potts and Kovalyov 2000) is considered in this study. Each batch requires a sequence of operations subject to linear precedence constraints. In addition, to increase the diversity and flexibility of the product mix in practice, batches processed by different routes are often produced simultaneously while the resources are limited. Hence, in contrast to the flowshop problem, each batch has its own routing sequence in this study. Finally, furnaces are often set in parallel as a machine group (workcenter) to maintain the robustness given the dynamic manufacturing environment. As most furnaces are multifunctional and originally designed for specific purposes, the machines within a machine group are unrelated parallel (Piersma and Dijk 1996).

Furthermore, the following technological and managerial characteristics complicate this ONO scheduling problem.

1. *Waiting time constraint*. Wafers for the interpoly ONO stacked film fabrication are processed in different furnaces consecutively. However, the surface of poly-silicon oxide and nitride layers are extremely unstable. Thus, there are waiting time constraints in which the nitride (or top-oxide) layer should be deposited within a strictly defined time interval after depositing the poly-silicon oxide (or nitride) layer. If the waiting time constraint is violated, the wafers will be reworked to strip off the oxide (or nitride) deposition or even be scraped in the worst case. Hence, waiting time constraints are critical for ONO scheduling for both product quality and tool productivity concerns.
2. *Frequency-based setup*. Wafer fabrication is highly particle-sensitive, yet particles continuously accumulate in the furnace chambers during manufacturing. If the particle density (number of particle in a predefined volume) is accumulated over a threshold, the wafers will also need to be reworked or even be scraped. Hence, after conducting a certain number of runs, the furnace

should be purged to ensure the particle density within the furnace chamber below the tolerance.

3. *Limited machine availability.* Machines will not be always available, as scheduled or unpredicted machine downtimes are unavoidable.
4. *Rolling horizon-based scheduling mechanism*. In practice, the new batches arriving during the scheduling period (e.g., 6 or 12 h) that can be defined based on the operator shift are accumulated and then scheduled after all the in-process batches are completed. The existing practice is simple for scheduling calculation because no batch is still in process as scheduling the new batches. However, because the capacity would be wasted during waiting, resource utilization is low. Thus, a rolling horizon-based scheduling mechanism is developed in which the unprocessed scheduled batches along with the newly coming batches that arrive during the scheduling period are all considered in each scheduling calculation. As for those in-process batches, this approach will reserve their processing operations and the corresponding successive operations to prevent over waiting.

Scheduling problems can be represented in the form of $n/m/A/B$, which consists of four parameters. The first parameter, n, represents the size of the job set. The second parameter, m, represents the number of machines in the system with $m>1$ for a multiple-machine system. The third parameter, A, represents the system information including job characteristics, system configuration, machine layout. The fourth parameter, B, represents the performance criterion.

This study aims to solve the ONO scheduling problem, characterized by waiting time constraints, frequency-based setup, and capacity preoccupation. Capacity preoccupation means that the necessary capacities are reserved for machine unavailability and successive operations of in-process batches. Based on the parallel machine scheduling problem representation schema (Cheng and Sin 1990) and a modification of the no-wait job shop scheduling problem (NWJSSP) $n/m/G$/no-wait/$C_{\max}$ (Macchiaroli et al. 1999), the present problem can be denoted as $n/m/G$/con-wait, SetupFrq$_m$, NC$_{win}$/$C_{\max}$, where the limited machine availabilities (Schmidt 2000) are also represented. In addition, other problem characteristics that no preemption is allowed and the machines are unrelated parallel are not explicitly denoted in this schema. Furthermore, we conducted a number of experiments based on the empirical data from a wafer fab for validation, and the results showed the practical viability of this approach.

The rest of this paper is organized as follows. Literature review section reviews related studies contributing to the fundamentals of this approach. On one hand, the section on Mixed-integer linear programming formulation develops a mixed-integer linear programming (MILP) model to provide optimal solutions as a benchmark. On the other hand, the section on Batch sequencing for ONO scheduling describes the genetic algorithm for batch sequencing and the Batch timetabling for ONO scheduling section elaborates the construction of the proposed timetabling algorithm for chromosome evaluation. The Experiments section analyzes the experimental results to estimate the validity of this approach based on real data collected from a wafer fab in Taiwan. The Discussion section discusses research findings including the relations between problem characteristics and the proposed algorithm. The Conclusion section concludes this study with discussions on future research directions.

2 Literature review

Waiting time issues have long been important problem characteristics in scheduling. Because these issues are theoretically complicated, many studies have been done on the NWJSSP. In particular, Hall and Sriskandarajah (1996) conducted a thorough survey on no-wait shop scheduling problems. Mascis and Pacciarelli (2002) used an alternative graph formulation to solve the NWJSSP. Brizuela et al. (2001) proposed a GA-based procedure for NWJSSP. Raaymakers and Hoogeveen (2000) adopted simulated annealing (SA) to obtain optimal or near-optimal solutions of NWJSSP of practical problem size. Macchiaroli et al. (1999) constructed a scheduling procedure that decomposed the no-wait scheduling problem into a sequencing and a timetabling module. Schuster and Framinan (2003) also proposed an approximate procedure for both flow shop and job shop scheduling problems. In addition, Chauvet et al. (2001) proposed the concept of time windows for solving NWJSSP in which each workcenter contains one or several identical machines.

Setup time is another important issue. As it may be complicated enough to handle waiting time constraints, many studies considering setups focused only on flow shop scheduling problems (FSSP) that were subject to sequence-independent setup time (Aldowaisan 2001; Aldowaisan and Allahverdi 1998; Allahverdi and Aldowaisan 2000; Gupta et al. 1997; Lin and Cheng 2001; Pranzo 2004; Sidney et al. 2000). Although Allahverdi and Aldowaisan (2001) extended their approach to deal with sequence-dependent setup times, it still concentrated on the flow shop problem. Considering complex problem natures such as the unrelated parallel machine environment, inseparable sequence-dependent setup time, dynamic job arrival, non-preemption, multiple-resource requirements, general precedence constraints, and job recirculation, the authors Chien and Chen (2006) proposed an optimization-based schedule generator that is composed of a GA-based scheduling strategy and a colored timed Petri net (CTPN)-based generalized schedule generator for solving the generalized scheduling problems arising from semiconductor manufacturing. The separation of the problem structure and problem configuration in the proposed schedule generator contributes to the structural independence, making it robust and convenient in analysis and problem solving in real settings.

Little research has been done to deal with the practical characteristics involved in the present ONO scheduling problem at the same time. Comparing to the existing studies, the completion of an operation is not necessary to be immediately followed by its successive operation in the present problem. Although most studies considered a single machine or identical parallel machine environment, unrelated parallel machines considered in this study are more realistic. To the best of our knowledge, none of the existing studies has addressed both the waiting time constraint and frequency-based setup. Furthermore, the combination of frequency-based setup and capacity preoccupation in this study contributes to another complicated problem concerning the solution searchability in the proposed timetabling algorithm, which has seldom been discussed.

Hence, an approach based on genetic algorithm (GA) (Gen and Cheng 1997) is developed to obtain near-optimal solutions. This approach adapted the decomposition structure (Macchiaroli et al. 1999) to decompose the present problem into sequencing and timetabling problem. Then, GA is employed as local search

heuristic to construct the sequencing module. However, it is a critical issue to encode a solution of the problem into a chromosome so as to ensure that a chromosome will correspond to a feasible solution. Different chromosome representations (e.g., Bean 1994; Bierwirth et al. 1996; Cheng et al. 1996) have been proposed in GA for solving scheduling problems with permutation nature. In particular, random key representation (Bean 1994) that encodes a solution with random numbers was developed to address optimization problems including multiple machine scheduling, resource allocation, and the quadratic assignment problem, while maintaining the feasibility from parent to offspring.

Macchiaroli et al. (1999) proposed a polynomial-time timetabling policy (TP) to determine the processing timing given a priority sequence of all jobs. Schuster and Framinan (2003) also proposed a simple timetabling algorithm to find a feasible set of operation starting times. In addition, Brizuela et al. (2001) summarized that the primary step of the decoding procedure is to find all matches among machine idle time intervals and the processing time of the current job so that the no-wait condition is not violated, yet, without further describing the matching methods. Nevertheless, none of the above approaches can be used directly to solve the ONO scheduling problem with consideration of the realistic constraints. A novel timetabling algorithm is developed as a decoding procedure in GA for handling the crucial problem nature including the waiting time tolerance, frequency-based setup, limited machine availability, and the rolling horizon-based scheduling mechanism. In addition, an MILP model that will be detailed in the following section was also developed as a benchmark to validate the solution quality of the proposed algorithm for small-sized test problems.

3 Mixed-integer linear programming formulation

The terminology and notation used in this study are summarized as follows:

$\mathbf{J}$	Set of all jobs; indexed by j
J_j	Job j
$\mathbf{O}_j$	Set of all operations of J_j; indexed by o.
T_{jo}	oth operation of J_j
$\mathbf{M}$	Set of all machines; indexed by m
$\mathbf{M}_{jo}$	Set of machines on which T_{jo} can be processed
M_m	Machine m
m_{jo}	Index of machine on which T_{jo} is processed
$\mathbf{P}$	Set of all positions; indexed by p
O_j^{first}	The first operation of J_j
O_j^{last}	The last operation of J_j
R_j^J	Release time of J_j
R_m^M	Release time of M_m
P_{jom}	Processing time of T_{jo} on M_m
Q_{jo_1}	Waiting time allowable after completing the process of T_{jo}
Pred_{jo}	Immediate predecessor of T_{jo}
Succ_{jo}	Immediate successor of T_{jo}
Z	A positive large number

$C_{\max}$	Makespan
B_{jo}	Process starting time of T_{jo}
C_{jo}	Process completion time of T_{jo}
BP_{mp}	Process starting time of pth task on M_m
CP_{mp}	Process completion time of pth task on M_m
X_{jomp}	=1 if T_{jo} is processed on M_m as pth task; 0 otherwise
J^{Chrom}	Permutation sequence; indexed by h
J_h^{Chrom}	The hth job in the permutation sequence
$\mathbf{W}_m$	Set of time windows of M_m; indexed by w
α_{wm}	Start time of window w on M_m
β_{wm}	Completion time of window w on M_m
α_{jom}	Earliest possible start time (EPST) of T_{jo} on M_m
β_{jom}	Latest possible completion time (LPCT) of T_{jo} on M_m
α^*_{jo}	EPST of T_j
β^*_{jo}	LPCT of T_{jo}
r_{jo}	Release time of T_{jo}
d_{jo}	Due time of T_{jo}
S_m	Setup time/purge time of M_m
$\mathrm{TAfterS}_m$	Completion time of last setup on M_m
$\mathrm{RAfterS}_m$	Number of completed process after last setup on M_m
$\mathrm{SetupFrq}_m$	Setup frequency of M_m

The ONO scheduling problem is basically characterized as a JSSP with unrelated parallel machines. In addition, both the waiting time constraints and frequency-based setups are considered in this model. Despite the exponentially growing computational time, the ONO scheduling problem can be optimally solved within reasonable CPU time if the problem size is small. The consideration of capacity preoccupation in the MILP model can be modeled as constraints with the corresponding parameters B_{jo} and C_{jo} being determined a priori.

In particular, the proposed MILP model is formulated as follows:

Objective function:

$$\text{Minimize } C_{\max} \tag{0}$$

Subject to:

$$C_{\max} \geq C_{jo}, \forall j \in \boldsymbol{J}, o = O_j^{\text{last}}. \tag{1}$$

$$\sum_{m \in \mathbf{M}_{jo}} \sum_{p \in \mathbf{P}} X_{jomp} = 1, \forall j \in \boldsymbol{J}, o \in \boldsymbol{O}_j. \tag{2}$$

$$R_j^J \leq B_{jo}, \forall j \in \boldsymbol{J}, o = \boldsymbol{O}_j^{\text{first}}. \tag{3}$$

$$R_m^M \le B_{jo} + Z \cdot \left(1 - \sum_{p \in \mathbf{P}} X_{jomp}\right),\ \forall j \in \mathbf{J},\ o = O_j^{\text{first}},\ m \in \mathbf{M}_{jo}. \tag{4}$$

$$B_{jo} + \sum_{m \in \mathbf{M}_{jo}} P_{jom} \left(\sum_{p \in \mathbf{P}} X_{jomp}\right) = \boldsymbol{C}_{jo},\ \forall \boldsymbol{j} \in \mathbf{J},\ o \in \boldsymbol{O}_j. \tag{5}$$

$$\boldsymbol{C}_{jo_1} \le \boldsymbol{B}_{jo_2}, \forall j \in \mathbf{J}, o_1, o_2 \in \boldsymbol{O}_j, o_1 \to o_2. \tag{6}$$

$$\boldsymbol{B}_{jo_2} - C_{jo_1} \le Q_{jo_1}, \forall j \in \mathbf{J}, o_1, o_2 \in \boldsymbol{O}_j, o_1 \to o_2. \tag{7}$$

$$BP_{mp} - Z \cdot \left(1 - X_{jomp}\right) \le B_{jo} \le BP_{mp} + Z \cdot \left(1 - X_{jomp}\right), \forall m \in \mathbf{M}, p \in \mathbf{P} \tag{8-1}$$

$$CP_{mp} - Z \cdot \left(1 - X_{jomp}\right) \le C_{jo} \le CP_{mp} + Z \cdot \left(1 - X_{jomp}\right), \forall m \in \mathbf{M}, p \in \mathbf{P} \tag{8-2}$$

$$CP_{mp} \le BP_{m(p+1)}, \forall m \in \mathbf{M}, p \in \mathbf{P} \tag{9-1}$$

$$CP_{mp} + S_m \le BP_{m(p+1)}, \forall m \in \mathbf{M}, p \in \text{ multiple of } \text{SetupFrq}_m \tag{9-2}$$

$$\sum_{j \in \mathbf{J}} \sum_{o \in \left(\mathbf{O}_j\right) \cap \left(m \in \mathbf{M}_{jo}\right)} X_{jomp} \le 1, \forall m \in \mathbf{M}, p \in \mathbf{P} \tag{10}$$

$$\sum_{j \in \mathbf{J}} \sum_{o \in \left(\mathbf{O}_j\right) \cap \left(m \in \mathbf{M}_{jo}\right)} X_{jomp} \ge \sum_{j \in \mathbf{J}} \sum_{o \in \left(\mathbf{O}_j\right) \cap \left(m \in \mathbf{M}_{jo}\right)} X_{jom(p+1)}, \forall m \in \mathbf{M}, p \in \mathbf{P} \tag{11}$$

Constraint 1 defines the makespan. Constraint 2 specifies that each operation must be assigned to exactly one machine to complete its process. Constraint 3 prevents a job from being processed until it is released. Constraint 4 prevents starting a process on a given machine until the machine is released. Constraint 5

models the unrelated parallel machines by defining a time interval, i.e., the machine-dependent processing time between process starting and process completion time of a given operation. Equation 6 represents the precedence constraints. Constraint 7 represents the waiting time conditions. The no-wait manufacturing condition means $Q_{jo_1} = 0$. Constraint 8-1 constructs the relationship between the operation-based and the position-based process starting time. If T_{jo} is pth-processed on M_m, then B_{jo} and BP_{mp} will be forced to be the same. The same explanation applies to constraint 8-2. Constraint 9-1 restricts the processing sequence so that a machine only processes one job at one time. Constraint 9-2 guarantees sufficient time between consecutive jobs so that a machine can be purged if necessary. Constraint 10 restricts no more than one job being assigned to the pth position of M_m. Finally, constraint 11 ensures that the position is counted from the smallest number, and thus, position shifting is not allowed.

The number of positions used in this model should be predetermined, finite, and sufficient to complete all operations. Any heuristics that can provide a feasible solution can be applied to estimate the upper bound of the number of positions. Nevertheless, the number of positions derived from the proposed algorithm has two advantages. First, the number of positions is certainly large enough because it is the upper bound of an optimal solution. Second, as the makespan would be a near-optimal solution after the evolution of GA, its deviation from the optimal solution and the use of dummy variables will be limited. Thus, unnecessary computation time can be saved.

4 Batch sequencing for ONO scheduling

This study solves the ONO scheduling problem with the constructed batch sequencing module based on GA and the timetabling algorithm as its decoding procedure. GA is employed for the sequencing module that speculates new search points by exploiting the historic information. It is a guided stochastic search technique searching for the best solution within a coded solution space. Briefly, the searching mechanism performs the following steps iteratively. Firstly, a set of initial solutions is encoded as a set of chromosomes called the population. Secondly, the genetic operators including crossover and mutation are applied to the population to generate offsprings from parents. An evaluation and scaling function are then used to assign a fitness value to each chromosome. Finally, a selection mechanism is employed to improve the solution quality. These iterative steps are terminated until the maximum number of generations is reached.

4.1 Chromosome representation

In general, it is not a good choice to use the whole original solution of a given problem as the chromosome representation, as the solution structure of real problems is too complex to be a natural representation suitable for the GA (Cheng et al. 1996). As GA is employed only to find batch sequences that do not have the issue of infeasibility, permutation representation is thus adapted to represent batch processing sequences. The length of each chromosome equals the number of

batches to be scheduled. Indeed, the permutation representation is an indirect encoding schema that is similar to the job-based representation (Cheng et al. 1996), and thus, the actual schedule can be derived by the proposed timetabling algorithm.

4.2 Genetic operators

During the evolution process, cycle crossover and displacement mutation were adopted as the genetic operators to generate offsprings from parents to prevent chromosome illegality and infeasibility. Following Gen and Cheng (1997), the procedures of cycle crossover and displacement mutation are summarized as follows:

4.2.1 Cycle crossover

Step A-1. Find the cycle that is defined by the corresponding positions of genes between parents.
Step A-2. Copy the genes in the cycle to a child with the corresponding positions of one parent.
Step A-3. Determine the remaining genes for the child by deleting those genes that are already in the cycle from the other parent.
Step A-4. Fill up the child with the remaining genes, as illustrated in Fig. 1a.

4.2.2 Displacement mutation

Step B-1. Select a subtour at random and insert it into a random position as illustrated in Fig. 1b.

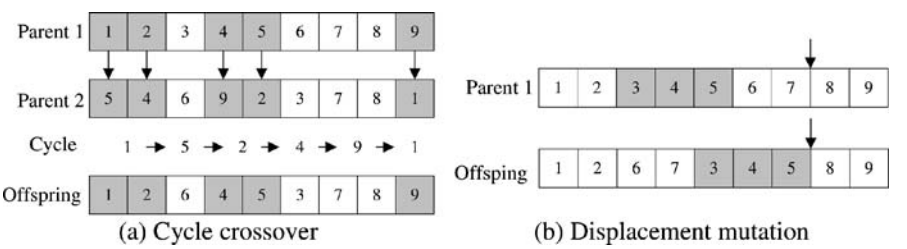


Fig. 1 Genetic operators. **a** Cycle crossover, **b** Displacement mutation

4.3 Decoding procedure and evaluation function

As permutation representation is an indirect encoding scheme, a timetabling algorithm that essentially matches the available time windows and operation processing times subject to waiting time constraints and setup requirements is developed. Thus, the batch processing sequence can be transformed into a recognizable and meaningful schedule. After decoding, the fitness value that is defined as the makespan of the schedule with a penalty function incorporated to ensure the feasibility of capacity preoccupation can be computed. The timetabling algorithm and evaluation function will be thoroughly elaborated in the section on Batch timetabling for ONO scheduling.

4.4 Scaling and selection mechanism

The roulette wheel approach was adopted to create the new population. Instead of using the original fitness value, the selection probability is defined as follows (Reeves 1995):

$$p_k = \frac{2k}{\mathrm{Pop}(\mathrm{Pop}+1)}$$

where k refers to the kth chromosome in descending order of the fitness value, Pop denotes the population size, and p_k denotes the selection probability of the kth chromosome. This selection probability can prevent premature convergence and maintain the distinguishability between the chromosomes in the later stage of the evolution process.

5 Batch timetabling for ONO scheduling

5.1 Timetabling algorithm

With the above GA, the batch processing sequence can be derived. However, transforming such sequence based on an indirect encoding scheme into a direct and meaningful schedule is far from trivial. While meeting the waiting time constraints and the requirement of sufficient time between operations for purgation, a non-delay schedule may not be optimal or even feasible. For example, a machine may still have to wait even if a job is waiting to be processed. As there is a predefined waiting tolerance between an operation and its successor, determining the process starting time of each batch is not enough. The proposed timetabling algorithm had to determine the process starting time for each of the operations.

The proposed timetabling algorithm is designed to handle waiting time constraints, frequency-based setups, limited machine availability and a rolling horizon-based scheduling mechanism. The basic idea of this algorithm is as follows. Given a batch (by using Step 1 to get the next unscheduled batch from the batch processing sequence) all the involved operations are considered sequentially according to the predefined precedence (by using Step 2.2, Step 2.3, Step 3.4, and Step 3.5 to keep tracking of the current operation). Then, for each operation, the

locally optimal match between available capacity time windows and operation processing time can be found such that this match is feasible (by using Step 2, Step 3, and Step 3.2). The local optimality is defined as finding the earliest possible start time (EPST) and/or the latest possible completion time (LPCT) of this operation. The feasibility is defined as not violating the constraints such as the waiting time constraint. Furthermore, a batch is defined as scheduled after all of its operations are scheduled according to local optimality and feasibility (by using Step 2.1, Step 3.1, and Step 3.3 to temporarily set start time and completion time and then using Step 4 to occupy the resources). After a batch is scheduled, it cannot be moved by the following batches. The next batch from the processing sequence is then selected to repeat the above procedure until all the batches are scheduled.

There are three possible situations as the EPST is found:

(1) Waiting time constraints are satisfied.
(2) Waiting time constraints will be satisfied after postponing the upstream operation(s) by a nonnegative period of time.
(3) Waiting time constraints will be satisfied after postponing the upstream operation(s) and the current operation itself.

The above postponement might be propagated to the first operation. In addition, a feasible schedule always exists as long as the postponement is large enough. The key idea of the proposed timetabling algorithm is designing the way of finding the feasible schedule while minimizing the postponement, and thus, minimizing the makespan. The steps are elaborated as follows:

Given a permutation sequence J^{Chrom}. Index h is 0 initially.

Step 1 [Next Batch]: $h=h+1$, $j=J_h^{Chrom}$, $jo=\{jk \mid k=O_j^{\text{first}}\}$, goto Step 2.

Step 2 [Find EPST]: $\alpha_{jom} = \min\left\{t^s \middle| \left[t^s, t^s + P_{jom}\right] \in [\alpha_{wm}, \beta_{wm}] \text{and } t^s \geq \max\left\{\text{TAfterS}_m, \text{r}_{\text{jo}}\right\}, w = 1, 2, \cdots |\mathbf{W}_m|\right\}$, for all $m \in \mathbf{M}_{jo}$ $\alpha_{jo}^* \min_{m\in\mathbf{M}_{jo}}\left\{\alpha_{jom}\right\}$, $m^* = {}_{m\in\mathbf{M}_{jo}}\left\{m \middle| \alpha_{jom} = \alpha_{jo}^*\right\}$, goto Step 2.1.

Step 2.1 [Set Start Time and Completion Time]: $B_{jo} = \alpha_{jo}^*$, $C_{jo} = \alpha_{jo}^* + P_{jom^*}$, $m_{jo}=m^*$, goto Check A.

Step 2.2 [Previous Operation]: $d_{\text{Pred}_{jo}} = B_{jo}$, $jo=\text{Pred}_{jo}$, goto Step 3.

Step 2.3 [Next Operation]: $r_{\text{Succ}_{jo}} = C_{jo}$, $jo=\text{Succ}_{jo}$, goto Step 2.

Step 3 [Find LPCT]: $\beta_{jom} = \left\{ \begin{array}{l} \max \\ Z, o.w. \end{array} \left\{t^s \middle| \left[t^s - P_{jom}, t^s\right] \in [\alpha_{wm}, \beta_{wm}] \text{ and } \text{TAfterS}_m + P_{jom} \leq t^s \leq d_{jo}, w = 1, 2, \cdots |\mathbf{W}_m|, \text{if found}\right\} \right\}$, for all $m \in \mathbf{M}_{jo}$

$$\beta_{jo}^* = \left\{ \begin{array}{l} Z, \text{ if } \beta_{jom} = Z, \text{ for all } m \in \mathbf{M}_{jo} \\ \max\limits_{m\in\mathbf{M}_{jo}} \left\{\beta_{jom} \middle| \beta_{jom} \neq Z\right\}, \ o.w. \end{array} \right., \quad m^* = {}_{m\in\mathbf{M}_{jo}}\left\{m \middle| \beta_{jom} = \beta_{jo}^*\right\},$$

goto Check F.

Step 3.1 [Set Start Time and Completion Time]: $B_{jo} = \beta^*_{jo} - P_{jom^*}$, $C_{jo} = \beta^*_{jo}$, $m_{jo}=m^*$, goto Check D.

Step 3.2 [Find EPST]: $\alpha_{jom} = \min\left\{t^s \middle| \left[t^s, t^s + P_{jom}\right] \in [\alpha_{wm}, \beta_{wm}]\right\}$ and $t^s \geq \max\left\{\text{TAfterS}_m, \left\{\begin{array}{c}\alpha_{wm}, d_{jo} \in [\alpha_{wm}, \beta_{wm}] \\ d_{jo},\ o.w.\end{array}\right\}, w = 1, 2, \cdots |\mathbf{W}_m|\right\}$ for all m$\in \mathbf{M}_{jo}$ $\alpha^*_{jo} = \min_{m\in\mathbf{M}_{jo}}\{\alpha_{jom}\}$, $m^* = {}_{m\in\mathbf{M}_{jo}}\left\{m \middle| \alpha_{jom} = \alpha^*_{jo}\right\}$, $m^* = {}_{m\in\mathbf{M}_{jo}}\{m|\alpha_{jom} = \alpha^*_{jo}\}$, goto Step 3.3.

Step 3.3 [Set Start Time and Completion Time]: $B_{jo} = \alpha^*_{jo}$, $C_{jo} = \alpha^*_{jo} + P_{jom^*}$, $m_{jo}=m^*$, goto Check D

Step 3.4 [Previous Operation]: $d_{\text{Pred}_{jo}} = B_{jo}$, jo=Pred$_{jo}$, goto Step 3.

Step 3.5 [Next Operation]: $r_{\text{Succ}_{jo}} = C_{jo}$, jo=Succ$_{jo}$, goto Step 2.

Step 4 [Resource Occupation] for all $o\in\mathbf{O}_j$: $m=m_{jo}$, use (B_{jo},C_{jo}) to update $\mathbf{W}_m$, goto Step 5.

Step 5 [Setup Check] for all $o\in\mathbf{O}_j$: $m=m_{jo}$, increase RAfterS$_m$ by 1. If RAfterS$_m$= SetupFrq$_m$, set TAfterS$_m$ as $\alpha_{|\mathbf{W}_m|m} + S_m$, use $\left(\alpha_{|\mathbf{W}_m|m}, \text{TAfterS}_m\right)$ to update $\mathbf{W}_m$, reset RAfterS$_m$ as 0. goto Check E.

Step 6 [Fitness Calculation] fitness value $= C_{\max} + \sum_{j\in\mathbf{J}, o=O_j^{\text{last}}} w_j C_{jo}$, where w_j denotes the priority (or weight) of batch j.

Check A if $\left(\text{Pred}_{jo} = \phi \text{ or } \alpha^*_{\text{jo}} - C_{\text{Pred}_{jo}} \leq Q_{\text{Pred}_{jo}}\right)$ goto Check C, else goto Step 2.2

Check B if $(\text{Succ}_{jo} = \phi \text{ or } B_{\text{Succ}_{jo}} - \beta^*_{jo} \leq Q_{jo})$ goto Step 3.1, else goto Step 3.2.

Check C if (Succ$_{jo}=\phi$) goto Step 4, else goto Step 2.3.

Check D if Pred$_{jo}=\phi$ goto Step 3.5, else goto Step 3.4.

Check E if $h=|\mathcal{J}^{Chrom}|$ then goto Step 6, else goto Step 1.

Check F if $\left(\beta^*_{jo} \neq Z\right)$ goto Check B, else goto Step 3.2.

The fitness value is defined as the makespan of schedule. The smaller the makespan is, the better the schedule will be. However, scheduling with capacity preoccupation can lead to the waste of capacity time windows if frequency-based setup is considered. In other words, capacity preoccupation may cause that the proposed timetabling algorithm cannot find good solutions. An approach is, thus, developed to recover the solution searchability by transforming the capacity preoccupation into a normal product batch with extremely high priority and incorporating a penalty function into the fitness value to ensure the feasibility of original preoccupation, given that the priority of a normal product batch is 0. Therefore, the feasibility constraint of a transformed high-priority product batch is relaxed into the fitness value in a Lagrangean fashion. More details about capacity preoccupation and its transformation will be provided in the sections on Capacity preoccupation and Solution searchability.

The primary steps in this algorithm are how to find the EPST after release time (Step 2), find the LPCT before due time (Step 3), and find the EPST such that the current due time is violated (Step 3.2). All the other steps are used to explicitly describe how to keep tracking of the current batch, the current operation, and the corresponding process starting and completion time. While Steps 2 and Step 3 are straightforward, Step 3.2 requires comprehensive explanation on how the lower bound of t^s is determined. It would be straightforward for t^s to be set as the maximum of TAfterSm and d_{jo} in Step 3.2, as d_{jo} intuitively separates the whole space into two parts. As illustrated in the following example, however, we can either simply find the EPST after due time, as shown in Fig. 2b or allow the processing start before due time, as shown in Fig. 2c. As shown in Fig. 2, Fig. 2c provides a better solution. In addition, Fig. 3 shows the equivalence when d_{jo} does not lie within a capacity time window.

5.2 Capacity preoccupation

In real settings, machines cannot all be expected to be constantly available during the scheduling period. Scheduled or unpredicted machine downtimes are usually unavoidable in the unstable and changing manufacturing environment. In addition, batches being processed cannot be interrupted, and their successive operations should also be guaranteed to prevent over-waiting. Hence, this study considered capacity preoccupation for modeling the limited machine availability and rolling horizon-based scheduling mechanism. Before scheduling all the normal product batches, capacity preoccupation is conducted as the first step by using specified time intervals preoccupied by machine downtimes or all the successors of currently processed operations to update the machines' capacity time windows.

5.2.1 Step 0[preoccupation]

for all m∈**M**:
if (R_m^M>0), use (0, R_m^M) to update $\mathbf{W}_m$
, increase RAfterS$_m$ by 1
, if RAfterS$_m$=SetupFrq$_m$, set TAfterS$_m$ as $\alpha_{|\mathbf{W}_m|m} + S_m$

, use ($\alpha_{|\mathbf{W}_m|m}$, TAfterS$_m$) to update $\mathbf{W}_m$

, reset RAfterS$_m$ as 0.
for all preoccupied capacity time windows (η_{im}, λ_{im}), i=number of preoccupation:
use (η_{im}, λ_{im}) to update $\mathbf{W}_m$
, increase RAfterS$_m$ by 1
, if RAfterS$_m$=SetupFrq$_m$, set TAfterS$_m$ as $\alpha_{|\mathbf{W}_m|m} + S_m$

, use $\alpha_{|\mathbf{W}_m|m}$, TAfterS$_m$ to update $\mathbf{W}_m$

, reset RAfterS$_m$ as 0.

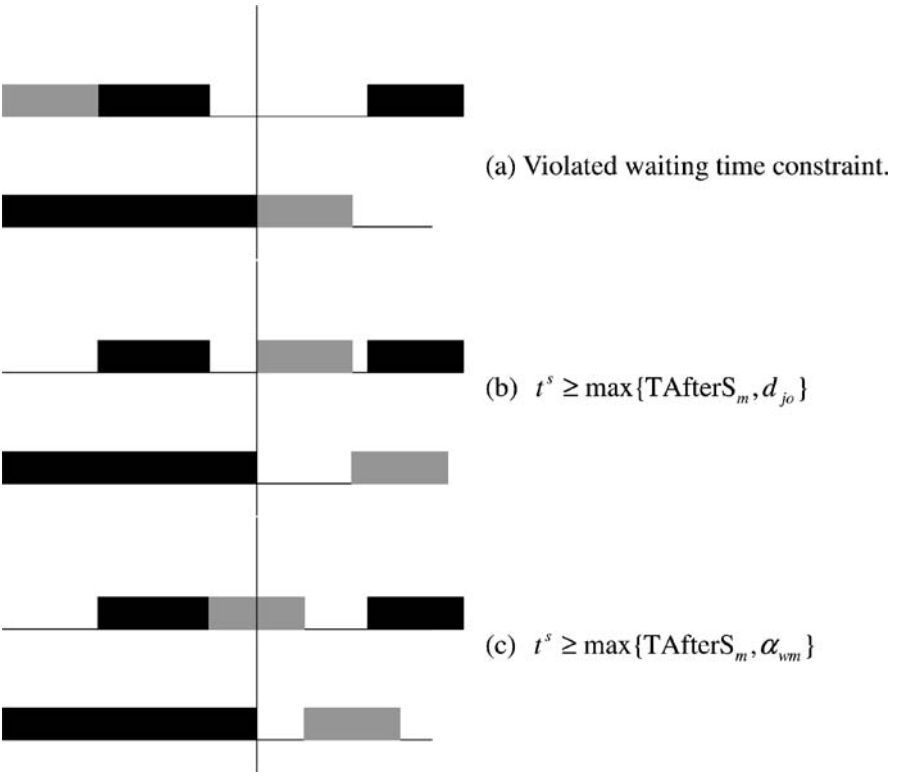


Fig. 2 Illustration of Step 3.2 when $d_{jo} \in [\alpha_{wm}, \beta_{wm}]$. **a** Violated waiting time constraint. **b** $t^s \geq \max\{\text{TAfterS}_m, d_{jo}\}$. **c** $t^s \geq \max\{\text{TAfterS}_m, \alpha_{wm}\}$

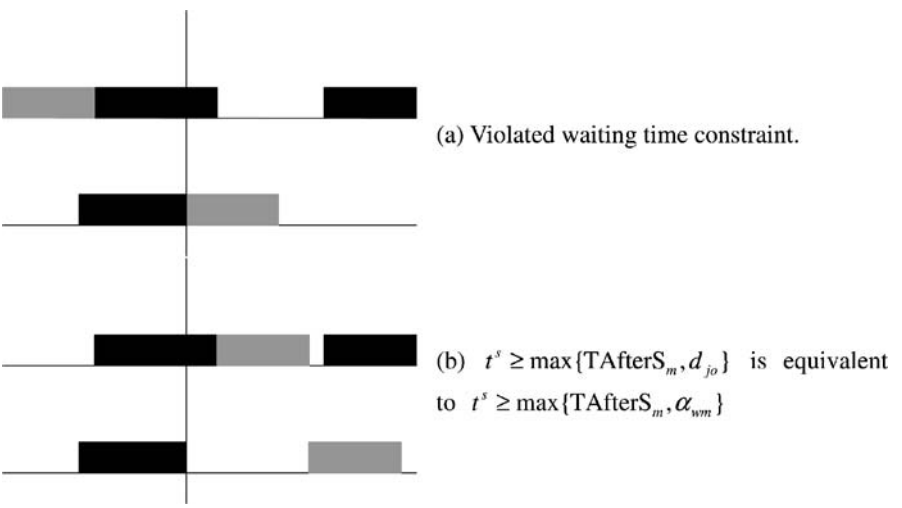


Fig. 3 Illustration of Step 3.2 when $d_{jo} \notin [\alpha_{wm}, \beta_{wm}]$. **a** Violated waiting time constraint. **b** $t^s \geq \max\{\text{TAfterS}_m, d_{jo}\}$ is equivalent to $t^s \geq \max\{\text{TAfterS}_m, \alpha_{wm}\}$

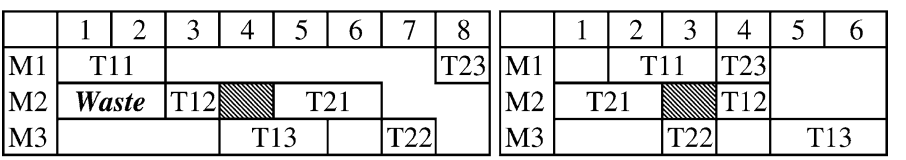
5.3 Solution searchability

In the proposed timetabling algorithm, it can be observed that the searching of EPST and LPCT is strictly constrained by the completion time of the last setup, denoted as TAfterS*m*. Clearly, many capacity time windows that are still available before the completion of the last setup would be wasted, or "masked," by TAfterS*m*. However, capacity masking is unavoidable if both the waiting time constraint and frequency-based setup are considered simultaneously. As the setup is frequency-based, the number of operations processed between two consecutive setups is strictly limited. If a setup is scheduled, then no extra operations before it are allowed. If a capacity time window available before a scheduled setup is allowed to become occupied, the original scheduled setup will become infeasible because the setup frequency is violated. Therefore, while searching for the EPST and LPCT, only those capacity time windows available after the TAfterS*m* will be considered.

Nevertheless, as elaborated in the sections on Timetabling algorithm and Capacity preoccupation, TAfterS*m* can be updated by both scheduling normal product batches and capacity preoccupation. Although some solutions may become bad owing to the restrictions of TAfterS*m*, we can still derive better or even the best solutions by changing the batch processing sequence in the GA evolution process. Figure 4 illustrates the difference between two batch processing sequences, in which the grey-shaded area denotes the setup. As the capacity waste is reduced in Fig. 4b, the solution in Fig. 4a is gradually eliminated from the population.

However, such capacity waste is invulnerable to sequence changes if TAfterS*m* is updated due to capacity preoccupation, as shown in Fig. 5. As the booked capacity is occupied before considering other normal product batches, the available capacity time window is also masked as the given initial conditions for sequence evaluation. In other words, the capacity preoccupation and chromosome permutation are virtually independent. Therefore, trying to prevent capacity waste owing to capacity preoccupation by using different batch processing sequences is futile. As shown in Fig. 5, regardless of the batch processing sequence, capacity waste on M2 is still unaffected by the sequence change.

Indeed, the illustrative problem in Fig. 5 has other solutions with shorter makespan and less capacity waste. However, capacity preoccupation prevents the proposed timetabling algorithm from finding better solutions. An approach to compensate such shortage and recover the solution searchability is proposed to transform the capacity preoccupation into a normal product batch with an extremely high priority. The release time of this transformed batch equals the start time of original preoccupation. Capacity preoccupation can, thus be scheduled along with all the other normal product batches, making the batch processing



(a) chromosome [1, 2] (b) chromosome [2, 1]

Fig. 4 Capacity waste from batch sequence. **a** Chromosome [1, 2]; (**b**) chromosome [2, 1]

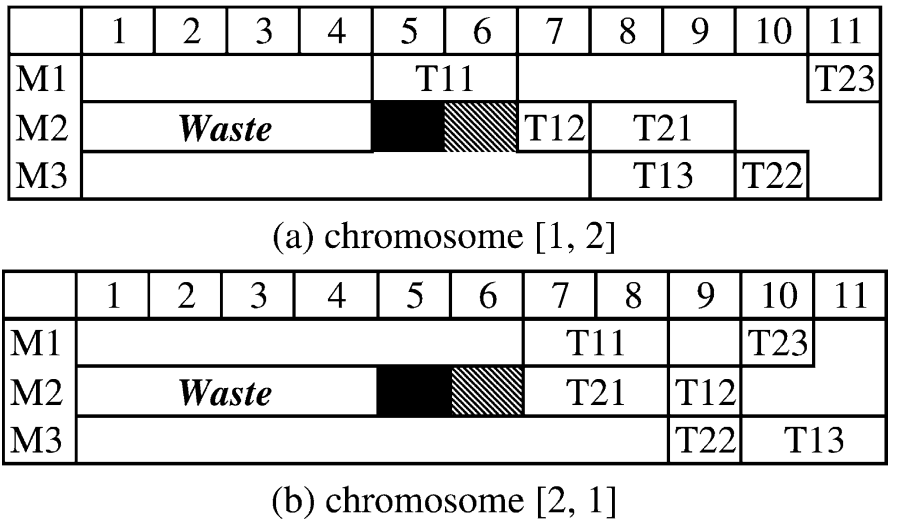


Fig. 5 Capacity waste from capacity preoccupation. **a** chromosome [1, 2]; **b** chromosome [2, 1]

sequence useful in reducing the capacity waste. Then, by incorporating the objective value defined in Timetabling algorithm, transformed batches that do not occupy their predefined time intervals would contribute to a poor chromosome in terms of fitness value. Batch processing sequencing, thus, becomes useful in eliminating poor solutions in the GA evolution process.

Figure 6 illustrates the transformation of capacity preoccupation resulting from the limited machine availability, where "*X*" represents the transformed high-priority product batch. Compared to Fig. 5, Figs. 6a and 5c show a shorter makespan and less capacity waste and "*X*" can still occupy its predefined time interval. Indeed, Fig. 6a,c show the optimal solutions. Although the solutions in Fig. 6b and d are not good enough and will be eliminated eventually, they also show the reduced makespan and capacity waste. However, Fig. 6d shows an infeasible solution because the transformed high-priority product batch fails to occupy its predefined time interval. Hence, the solution in Fig. 7d incurs high penalty and will, thus, hardly be selected in the next generation.

Furthermore, capacity preoccupation resulting from the rolling horizon-based scheduling mechanism can also be transformed into a high-priority product batch, with additional information to indicate the operation immediately succeeding the currently processed operation. Figure 7 illustrates this case, in which batch 1 is released at $t=0$ and batch 2 is released at $t=1$. In Fig. 7b, the first operation of batch 1 cannot be interrupted, and thus, the successive operations should preoccupy the capacities to prevent over-waiting. On the contrary, Fig. 7c considers batch 1 as a high-priority product batch with additional information indicating the second operation of this batch. The machine downtime illustrated in Fig. 6 indicates scheduled events such as engineering experiments, R & D testing, or preventive maintenance. As for the unpredicted machine breakdown, it is equivalent to irregular rescheduling when machine malfunctions occur.

Notably, the second part of Step 0 in the timetabling algorithm will be unnecessary if all the preoccupied capacity time windows are transformed into

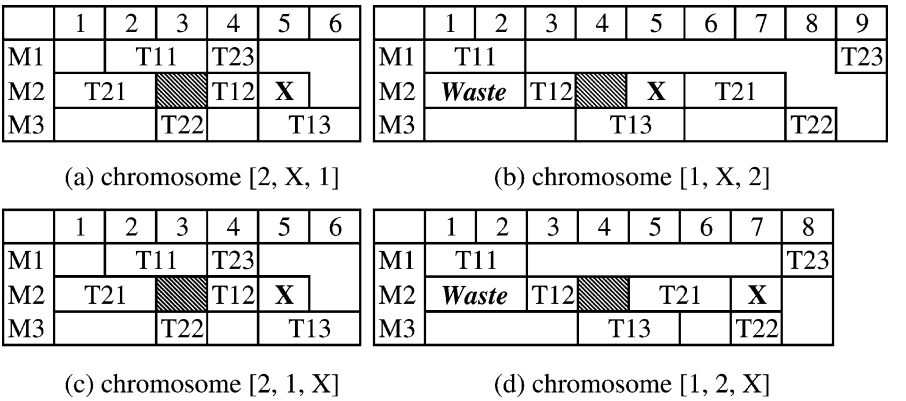


Fig. 6 Transformation of capacity preoccupation. **a** Chromosome [2, X, 1]; **b** chromosome [1, X, 2]; **c** chromosome [2, 1, X]; **d** chromosome [1, 2, X]

high-priority product batches. However, the transformation usually incorporates awkward data preprocessing and takes significant effort for data maintenance, as it generates temporary data. Therefore, if the setup is negligible or the setup frequency is large enough such that no setup will be expected during the scheduling horizon, we can simply use the capacity preoccupation for easy data preprocessing and maintenance.

5.4 Numerical illustration of timetabling algorithm

The timetabling algorithm is illustrated with a numerical example. Considering a simple no-wait scheduling problem with four machines and two batches, the

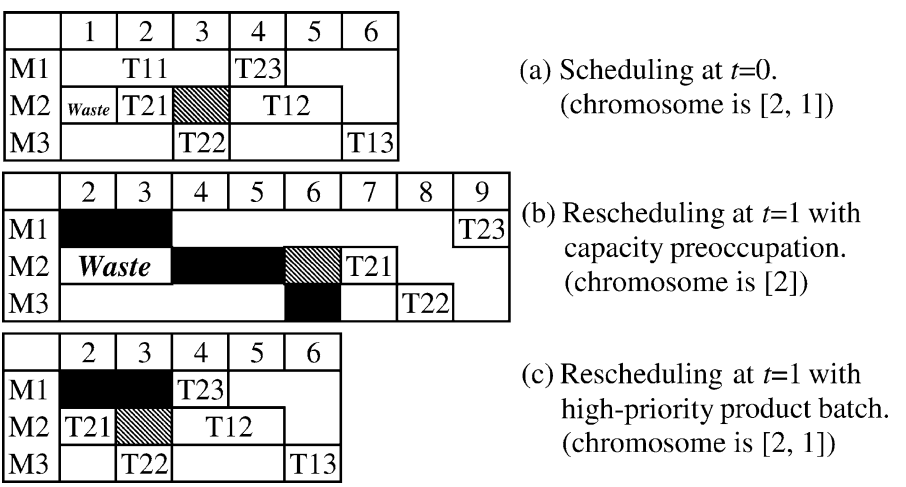


Fig. 7 Rolling horizon-based scheduling. **a** Scheduling at $t=0$ (chromosome is [2, 1]); **b** Rescheduling at $t=1$ with capacity preoccupation (chromosome is [2]); **c** Rescheduling at $t=1$ with high-priority product batch (chromosome is [2, 1])

illustrative routing sequence and processing time are configured as listed in Table 1.

Assume that there is no preoccupied capacity and all the machines are available at the beginning. Thus, the initial condition of $\mathbf{W}_m$ is $\mathbf{W}_1=\mathbf{W}_2=\mathbf{W}_3=\mathbf{W}_4=\{(0,\infty)\}$. In addition, the setup frequency is large enough to ignore the setup check. Given that $J^{Chrom}=[1, 2]$, the graphical illustration is shown in Fig. 8. As shown in Fig. 8a, all the operations of Batch 1 are sequentially scheduled simply by using Step 2 to calculate the EPST. In Fig. 8b, the waiting time constraint is violated when Oper3 of Batch 2 is scheduled. Figure 8c shows how Oper2 is postponed to meet the waiting time constraint. The waiting time violation is now propagated to Oper1. Figure 8d shows how Steps 3 and 3.2 determine the current due time should be violated. In Fig. 8e,f, Oper1 and Oper2 are further postponed for feasibility. The subsequent procedures in Fig. 8g,h complete this illustration.

Figure 9 shows the solution if $J^{Chrom}=[2, 1]$. Obviously, [2, 1] is better than [1, 2] if the makespan is considered as the evaluation criterion.

6 Experiments

6.1 Experimental setting

For the validation of the different approaches, we conducted experiments using randomly generated test problems. Empirical data including routing and machine information and the processing times were collected from a semiconductor fab in Hsinchu Science Park in Taiwan. There are totally 12 routes, 35 operations, and 14 machines. For confidentiality issues, all the specific terms about routes, operations, and machines are replaced by general terms without loss of generality. The values of processing times, waiting times, setup times, and setup frequency are summarized in Table 2.

Table 2 shows the problem characteristic called machine overlapping or tool overlapping, which results from the unrelated parallel machines and job recirculation in semiconductor manufacturing. Tool overlapping means that a machine can process different kinds of tasks and different machine sets by which different tasks can be processed having some machines in common. Although rarely discussed in existing studies, tool overlapping is common and important in realistic semiconductor manufacturing. Tool overlapping has the advantage to deal with the diversity of product mix and provides the flexibility to increase tool productivity under the constraints of limited resources. As in semiconductor

Table 1 Illustrative routing sequence and illustrative processing time

	Oper1	Oper2	Oper3	Oper4
Illustrative routing sequence				
Batch 1	M1	M2	M3	M4
Batch 2	M2	M4	M3	M1
Illustrative processing time				
Batch 1	4	2	4	2
Batch 2	1	2	1	4

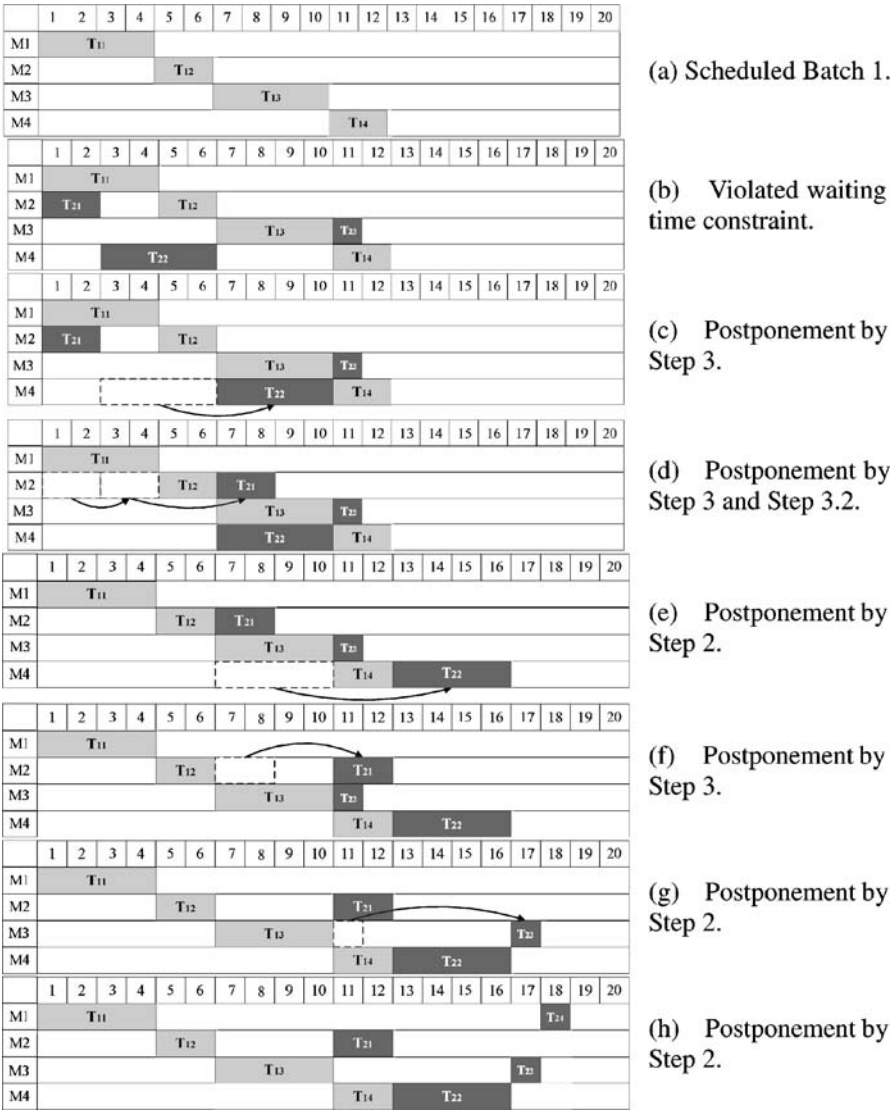


Fig. 8 Illustration of decoding [1, 2]. **a** Scheduled Batch 1. **b** Violated waiting time constraint. **c** Postponement by Step 3. **d** Postponement by Step 3 and Step 3.2. **e** Postponement by Step 2. **f** Postponement by Step 3. **g** Postponement by Step 2. **h** Postponement by Step 2

manufacturing, most machines are multifunctional and extremely expensive, it is tool productive which permits exploiting the capabilities of machines instead of dedicating them to specific purposes.

In particular, there are two experimental factors considered in the numerical experiments, i.e., waiting time constraints and frequency-based setups. Each experimental factor was set at two different levels. Hence, there are four experimental sets. Table 3 summarizes the settings of four experimental sets with set name and detailed description. Each experimental set considers four problem sets with different batch sizes, i.e., 8, 20, 40, and 60, respectively. Empirical data

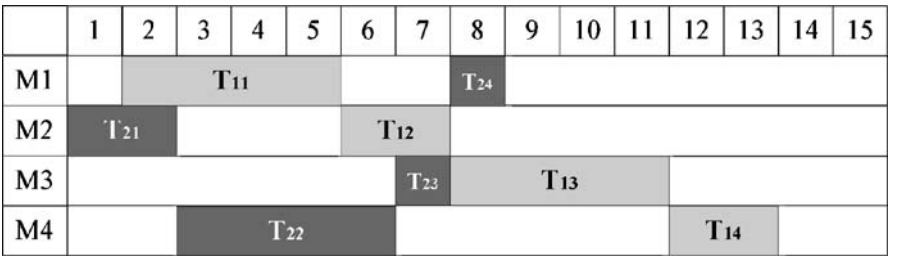


Fig. 9 Illustration of decoding [2, 1]

show that approximately 20 batches were processed in a 1-day schedule. Hence, three different batch sizes are set as 20, 40, and 60. In addition, the other problem set contains only eight batches so that it is small enough to enable the problem being solved with guaranteed optimality within reasonable computation time by the developed MILP model. Waiting time was set at one level as 0 so that the no-wait situation was assumed and the other level was set at the real value. The levels of frequency-based setups were set as low frequency, that is, the actual frequency purged in the fab and high frequency that has more frequent setups than practice for experimental purpose. The setting of high-frequency setup is related to the number of batches. Besides, the setup frequencies were randomly drawn from a discrete uniform distribution (Aldowaisan and Allahverdi 1998; Allahverdi and Aldowaisan 2000) for the smallest problem set with eight batches, the frequencies were randomly assigned from a discrete uniform distribution (Aldowaisan and Allahverdi 1998; Bean 1994) in the other three problem sets.

Furthermore, each problem set has 30 independent instances that are based on the product mix randomly generated from 12 real routes with the same fundamental information of the manufacturing environment in all instances. Finally, each instance was run 30 times (i.e., 30 replications) with independent random number seeds for eliminating initialization bias. The GA solutions were compared with those of the dispatching rule-based heuristics (DRBH) and the optimal solutions from the proposed MILP model as well, if available.

The experiments were conducted on a Windows 2000 Sever with AMD 1.7 GHz CPU and 512 Mb RAM. The MILP problems were solved using ILOG OPL Studio 3.6. In addition, a system prototype was developed for implementing the proposed GA-based approximation algorithm for the ONO scheduling problem. This system was implemented in C++ with Borland C++ Builder 5.0 as the integrated development environment. All the problem instances were stored in the Microsoft Access 2000 format and were connected to the system prototype using ADO. This prototype was employed to obtain the solutions of GA and DRBH.

The GA parameter settings are as follows: the crossover rate is 0.7; the mutation rate is 0.3; the population size is 50, and the number of generation is 100. These parameters are determined, for the purpose of striking a balance between solution quality and computation time, after several test runs were made for alternative parameter combinations.

Table 2 Production table

		M01	M02	M03	M04	M05	M06	M07	M08	M09	M10	M11	M12	M13	M14	WT
R01	Op01		7													4
R01	Op02					6										–
R02	Op01								5	5	5					4
R02	Op02					5	5									4
R02	Op03											5	5			–
R03	Op01								6							4
R03	Op02					5	5									4
R03	Op03											7	7			–
R04	Op01								6	6						4
R04	Op02					4										4
R04	Op03											5				–
R05	Op01								8	8	8					8
R05	Op02													6	6	–
R06	Op01	5	5	5	5											4
R06	Op02					6	6	6								4
R06	Op03	5	5	5	5											–
R07	Op01	5	5	5	5											6
R07	Op02					5	5	5								6
R07	Op03	5	5	5	5											–
R08	Op01	5	5	5	5											8
R08	Op02					5	5	5								8
R08	Op03	6	6	6	6											–
R09	Op01								6		6					4
R09	Op02					5	5									4
R09	Op03											5	5			–
R10	Op01								7	7	7					4
R10	Op02					5	5									4
R10	Op03											6	6			–
R11	Op01								6	6	6					4
R11	Op02					5	5									4
R11	Op03											6	6			–
R12	Op01					7	7									4
R12	Op02		5													4
R12	Op03					5	5									4
R12	Op04												4			–
SF		5	5	5	5	5	5	5	20	20	20	20	20	10	10	
ST		2	2	2	2	3	3	3	6	6	6	5	5	2	2	

6.2 Dispatching rule-based heuristics

DRBH were employed for two purposes. Firstly, it provides a basis for the comparison of solution quality with that of GA. Secondly, it provides the initial chromosomes for GA evolution. The dispatching rules including shortest processing time first (SPT), longest processing time first (LPT), smallest number of operations first (SNO), largest number of operations first (LNO), and highest

Table 3 Experimental sets

Set name	Description	Settings	
		Waiting time	Setups
High-Con	High setup frequency and constrained waiting time	Real values	Eight-batch: [2, 3] Others: [2, 5]
High-No	High setup frequency and no waiting time	Zero	Eight-batch: [2, 3] Others: [2, 5]
Low-Con	Low setup frequency and constrained waiting time	Real values	Real values
Low-No	Low setup frequency and no waiting time	Zero	Real values

machine criticality first (HMC) applied in the experiments. As the processing time may not be identical on parallel machines, the average processing time was adopted when applying the SPT and LPT rule. The SNO and LNO rules were employed to count the total number of operations of each batch.

The calculation of the HMC indices is more complicated. Firstly, the criticalities of all machines are calculated to identify the potential bottleneck as follows:

$$\mathrm{CRTY}_m = \sum_{j \in \mathbf{J}} \sum_{o \in (\mathbf{O}_j) \cap (m \in \mathbf{M}_{jo})} \frac{P_{jom}}{|\mathbf{M}_{jo}|}, \ \forall m \in \mathbf{M} \tag{12}$$

Secondly, the operation's criticality is defined as the minimum criticality of machines on which the operation can be processed:

$$\mathrm{CRTY}_{jo} = \min_{m \in \mathbf{M}_{jo}} \{\mathrm{CRTY}_m\}, \ \forall j \in \mathbf{J}, o \in \mathbf{O}_j \tag{13}$$

Finally, the criticalities of operations belonging to a batch are summed to derive HMC indices as follows.

$$\mathrm{HCM}_j = \sum_{o \in \mathbf{O}_j} \mathrm{CRTY}_{jo}, \forall j \in \mathbf{J}. \tag{14}$$

The batch sequence was determined by sorting all the batches according to the dispatching rule used. For example, all the batches were sorted nondecreasingly according to their average processing times if the SPT rule was applied. The batches would be sorted nonincreasingly according to their machine criticalities when HMC rule was used. Then, the batch sequence was transformed into a direct schedule using the timetabling algorithm proposed in the section on Batch timetabling for ONO scheduling.

6.3 Experimental results

Considering the eight-batch problem set, the optimal solutions of the proposed MILP can be derived by OPL within 12 h in some instances, as the problem size is small enough. Comparing 30 instances of eight-batch problems, Table 4 shows that GA can solve most of the instances optimally. In the High-Con problem set, 14 out of 28 instances are solved by the proposed GA optimally; in High-No, 21 out of 27 instances are solved optimally; in Low-Con, 24 out of 30 instances are solved optimally; in Low-No, 19 out of 28 instances are solved optimally. Furthermore, comparing with the optimal solutions, GA outperforms DRBH in all instances. The percentages of deviations of MILP and DRBH from the GA-based solutions were also compared as shown in Fig. 10.

Although waiting time tolerance adds more flexibility for scheduling, it makes the problem mathematically more complex than the pure no-wait problem. The reason is that the timetabling algorithm decides the EPST (or LPCT), if waiting time is allowed, in an extreme manner. In other words, the EPST (or LPCT) is always decided such that it is adjacent to the boundary of time window or forms a no-wait solution. However, if the timetabling algorithm works in this way, the optimal solutions may be implicitly excluded from the searching space. In particular, Table 4 shows that GA obtains a significantly lower number of optimal solutions for High-Con compared to MILP. The High-Con problem set is essentially harder to solve by GA, owing to the strict limitations on the selection of available time windows from frequent setups and the intrinsic deficiency of proposed timetabling algorithm.

In addition, the proposed GA outperforms, or at least, is not worse than DRBH in all instances for 20-batch, 40-batch, and 60-batch problems, respectively. Figure 10 shows the percentages of improvement from DRBH to GA that are particularly significant for large problems, i.e., from around 10% to almost 20%. Also, Fig. 10 shows that the benefit of GA is significant in the problem set with high setup frequency. As described in the section on Solution searchability, capacity masking is unavoidable if both waiting time constraints and frequency-based setups are considered. Figure 10 shows that DRBH are more vulnerable to the effect of capacity masking. The High-Con experimental set shows the largest improvements, i.e., 10.48% in 20 batches, 16.03% in 40 batches, and 19.24% in 60 batches, respectively. Indeed, the High-Con set is the most difficult combination, and thus, the solution quality gap between DRBH and GA is the largest. This further confirms the superiority of GA over DRBH.

Table 5 shows the effectiveness of each of the dispatching rules for different batch sizes and experimental settings. The degree of effectiveness of a dispatching rule is defined as the number of instances where the best solution is derived by that rule. If more than one dispatching rule, say N^*, provides the best solution, the

Table 4 Number of optimal solutions obtained by different algorithms

Eight-batch problems	High-Con	High-No	Low-Con	Low-Con
MILP	28	27	30	28
GA-Based	14	21	24	19
DRBH	10	7	6	8

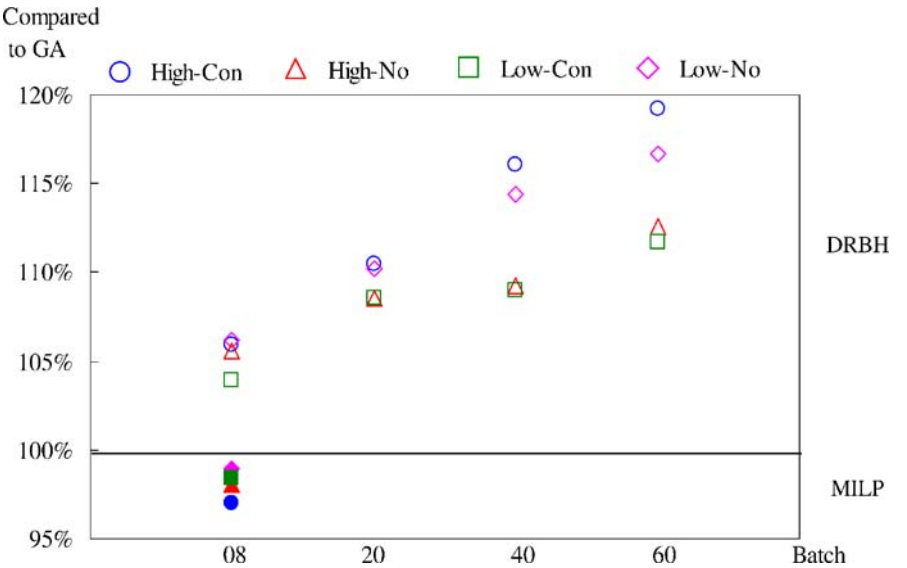


Fig. 10 The comparison of MILP and DRBH to GA-based algorithm

effectiveness degree of each of the N^* rules in that instance is $1/N^*$. As shown in Table 5, HMC is the most effective rule for the ONO scheduling problem in most instances.

While it is usually required for an online scheduling system to solve a problem quickly and correctly, the computation time should also be stable or even predictable so that the decision makers know when and how often they should run this program and get the solution in time. That is, the computation time should not

Table 5 Degree of effectiveness of different dispatching rules

		SPT	LPT	SNO	LNO	HMC
Eight batch	High-Con	0.00	7.83	0.00	7.83	14.33
	High-No	1.00	7.17	0.00	9.17	12.67
	Low-Con	2.66	4.00	2.00	10.16	11.16
	Low-No	1.00	6.25	0.25	10.75	11.75
20 batch	High-Con	0.00	5.50	2.00	6.50	16.00
	High-No	1.00	3.00	2.00	9.50	14.50
	Low-Con	0.00	10.17	1.00	8.17	10.67
	Low-No	0.00	2.83	2.00	12.33	12.83
40 batch	High-Con	0.00	5.33	2.00	4.83	17.83
	High-No	0.00	3.50	1.50	9.50	15.50
	Low-Con	0.00	2.00	1.00	4.00	23.00
	Low-No	0.00	2.50	2.00	4.50	21.00
60 batch	High-Con	1.33	4.00	9.33	9.83	5.50
	High-No	0.50	3.50	5.83	14.83	5.33
	Low-Con	2.00	0.00	6.00	7.50	14.50
	Low-No	1.33	0.50	6.17	4.83	17.17

be significantly affected by unknown or uncontrollable factors such as batch size and product mix.

Table 6 reports the average computation time and the standard deviation provided by OPL, GA-based approximation, and DRBH, respectively, for small problem sets. The computation time for GA is based on running 30 replicates, and the time for DRBH is to run all five rules. Computational times of less than a second are not explicitly shown. It can be seen from Table 6 that OPL spent significantly longer time than GA to derive the solution. While it takes around 1 h for OPL on average to solve a problem instance, it takes only about 2 min for GA. The computation time for OPL increases exponentially with the size of the MILP optimization model. Thus, there is a trade-off between the solution quality and computation time. In practice, it should take less than 1 h to obtain the ONO scheduling solution. Hence, GA is preferred under this circumstances, as it requires much shorter computation time without suffering significant losses of solution quality.

Furthermore, Table 7 shows that the variance of computation time is large for OPL. Thus, it is practically infeasible to implement OPL-embedded algorithms because the time required for solving the problem is uncertain and variable. On the other hand, the computation time of GA is stable considering its small coefficients of variance for different problem sizes. Thus, the proposed GA-based approximation can provide an appropriate alternative with relatively short computation time and near-optimal solutions.

Figure 11 shows that the setting of waiting time affects the computation time of GA. In the timetabling algorithm, Check A and Check B provide tighter inequality constraints under the no-wait setting. Hence, it may require more steps of finding EPST and LPCT to fulfill those inequalities, which results in the significant difference in computation time.

7 Discussion

While none of the existing studies has solved the investigated problem, this study has addressed critical problem characteristics involved in the ONO scheduling problem. Priority rules are probably the most frequently applied heuristics for solving complex scheduling problems because they are easy to implement and they require only little CPU time. The procedure of transforming the indirect priority rules into the direct schedule is often based on generation procedures (Giffler and Thompson 1960). Regardless of whether the procedures are designed for generating an active or non-delay schedule, the existing studies are mostly operation-based. Nevertheless, the timing cannot be calculated operation by

Table 6 Computation time of eight-batch problems (unit: min)

	High-Con			High-No			Low-Con			Low-Con		
	MILP	GA	DRBH	MILP	GA	DRBH	MILP	GA	DRBH	MILP	GA	DRBH
Avg.	62.9	1.8	<1 s	73.4	2.0	<1 s	48.3	1.7	<1 s	75.6	2.0	<1 s
Max	682.7	2.0	<1 s	768.0	2.4	<1 s	503.8	2.0	<1 s	672.1	2.3	<1 s
Min	0.8	1.5	<1 s	0.3	1.6	<1 s	0.2	1.5	<1 s	0.6	1.6	<1 s

Table 7 Coefficient of variance of computation time

	High-Con		High-No		Low-Con		Low-No	
Problems	OPL	GA	OPL	GA	OPL	GA	OPL	GA
Eight-Batch	2.1348	0.0775	2.0495	0.0931	2.2259	0.0744	1.9668	0.0882
20-Batch	N.A.	0.0685	N.A.	0.0740	N.A.	0.0643	N.A.	0.0674
40-Batch		0.0478		0.0517		0.0380		0.0395
60-Batch		0.0445		0.0440		0.0373		0.0440

operation if the problem is subject to waiting time constraints. These can only be handled using job-based timetabling.

Job-based timetabling requires modeling the available capacity time windows before the scheduled operation(s) on the machine. The existing studies concerning timetabling under waiting time constraints were also based on a similar concept. Their difference primarily lies in how to calculate and match the process starting times within capacity time windows. In addition, capacity preoccupation can also be considered in the same procedure.

The most important issue arises from frequency-based setups. While job-based timetabling and frequency-based setups make capacity masking necessary to maintain the feasibility of scheduled setups, capacity masking and capacity preoccupation prohibit the timetabling algorithm from finding superior solutions. This conflict results from the difference in intrinsic priorities between normal product batches and capacity preoccupation. Capacity preoccupation that logically bears the highest priority should be scheduled first to guarantee the required timing.

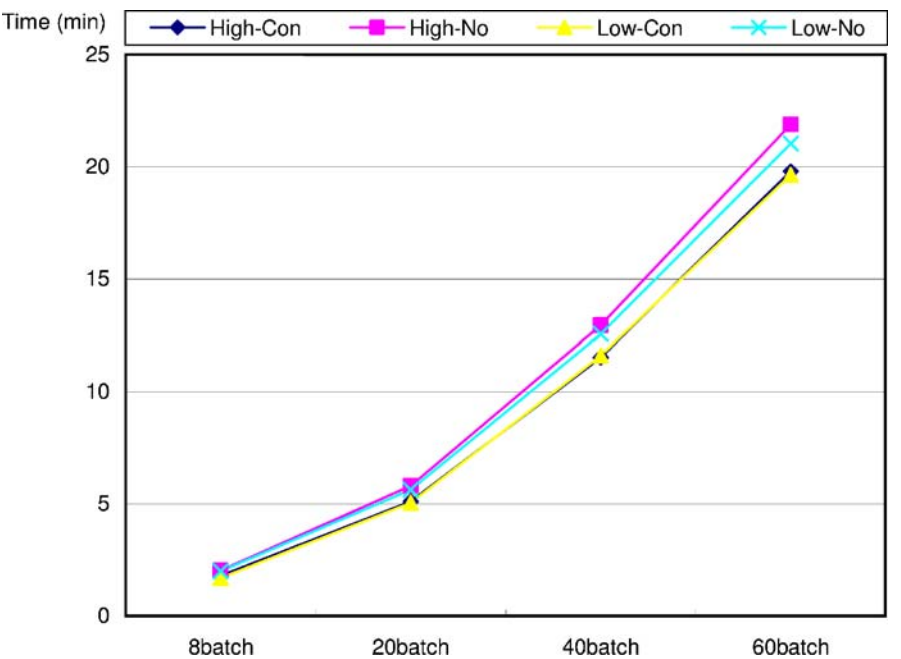


Fig. 11 Computation time of GA

However, if it is better for a setup to be triggered by other product batches to prevent capacity waste, the capacity preoccupation should be downgraded so that it can be considered along with all the other normal product batches, yet still with higher opportunities to seize the capacities at required timing.

This conflict is resolved in our approach by relaxing the constraint in the required preoccupation timing and transforming the preoccupation into a fake product batch. This relaxation approach makes better solutions available, but it also places super-optimal solutions into the searchable region. However, the timetabling algorithm itself cannot overcome the super-optimality. Therefore, in addition to improving the batch processing sequences, the other purpose of local search heuristic is to reflect the super-optimality by penalties incorporated into the fitness function of the GA. This is the reason that the total weighted completion time is also considered as part of the fitness value. During the evolution process, the struggle between optimality and feasibility gradually converges to a steady state, and thus, generates a feasible solution with optimality, or at least, near-optimality.

Indeed, wafer fabrication consists of multiple layers, in which the variability should be reduced to enhance the yield. Although Intel emphasizes to "copy exactly" in their machine configurations, most of the fabs tend to maintain different types of machines at the same time so as to avoid being dominated by specific equipment vendors (Chien and Hsu 2006). Thus, there is a critical need to determine subgroups of similar machines so that backups for unexpected machine breakdowns can be also prioritized for effective scheduling. Thus, Chien et al. (2006) proposed overall tool group efficiency (OGE) indices to enhance tool productivity by tool group.

Finally, local search heuristics are not limited to the genetic algorithm. Other neighborhood search algorithms such as Tabu Search and Simulated Annealing may be employed by embedding into the proposed timetabling algorithm to solve the ONO scheduling problem, yet further investigations are needed.

8 Conclusion

Rapid developments in semiconductor manufacturing technology have made the fabrication process increasingly complicated and delicate. Many manufacturing conditions and constraints such as those considered in the present ONO problem also become critical to ensure the yield. In this study, the ONO scheduling problem shows unique and difficult problem characteristics of waiting time constraints and frequency-based setups required in real settings. Yet, none of the existing studies has addressed this problem. This study fills the gap by explicitly describing the timetabling procedure based on the proposed algorithms. We also proposed an MILP model as a benchmark to estimate the validity of this approach in terms of solution quality. The results of the numerical experiments show that the proposed GA-based approximation algorithm can solve the practical ONO scheduling problem optimally in most cases and significantly outperforms DRBH.

A system prototype was developed for implementing the proposed GA-based approximation algorithm for the ONO scheduling problem. Further research is continuing to develop a decision support system using the proposed scheduling module to assist batch timetabling online. Further studies should be performed to investigate the problem nature of the ONO scheduling problem and its theoretical

implications. Future research can be also done to address the precedence constraint. Other application domains are subject to different precedence constraints such as partial precedence (hybrid shop) in hospital rehabilitation scheduling, tree-shaped precedence in assembly and chemical processes, and network-shaped precedence in project management, while the proposed algorithm is subject to linear precedence constraints. Further research can be done to relax the limitation in the number of predecessors and successors in the timetabling algorithm to integrate these related problems into a generic solution structure if they are subject to the waiting time constraint.

Acknowledgments This research is supported by the National Science Council, Taiwan (NSC 91-2213-E-007-050, NSC 92-2213-E-007-023, and NSC 93-2213-E-007-008) and Macronix International Co., Ltd. Special thanks go to the anonymous referees for their constructive comments and Professor Hans-Otto Guenther and Professor Sheng-Yuan Shen for their kind inputs.

References

Aldowaisan T (2001) A new heuristic and dominance relations for no-wait flowshops with setups. Comput Oper Res 28(6):563–584

Aldowaisan T, Allahverdi A (1998) Total flowtime in no-wait flowshops with separated setup times. Comput Oper Res 25(9):757–765

Allahverdi A, Aldowaisan T (2000) No-wait and separate setup three-machine flowshop with total completion time criterion. Int Trans Oper Res 7(3):245–264

Allahverdi A, Aldowaisan T (2001) Minimizing total completion time in a no-wait flowshop with sequence-dependent additive changeover times. J Oper Res Soc 52:449–462

Bean JC (1994) Genetic algorithms and random keys for sequencing and optimization. ORSA J Comput 6(2):154–160

Bierwirth C, Mattfeld DC, Kopfer H (1996) On permutation representations for scheduling problems. Parallel Problem Solving from Nature-PPSN IV:310–318

Brizuela CA, Zhao Y, Sannomiya N (2001) No-wait and blocking job-shops: challenging problems for GA's. Proceedings of 2001 IEEE International Conference on Systems, Man, and Cybernetics 4:2349–2354

Chauvet F, Proth JM, Wardi Y (2001) Scheduling no-wait production with time windows and flexible processing times. IEEE Trans Robot Autom 17(1):60–69

Cheng TCE, Sin CCS (1990) A state-of-the-art review of parallel-machine scheduling research. Eur J Oper Res 47:271–292

Cheng R, Gen M, Tsujimura Y (1996) A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation. Computers and Industrial Engineering 30(4):983–997

Chien C, Chen C (2006) Using GA and CTPN for modeling the optimization-based schedule generator of a generic production scheduling system. Int J Prod Res 1–27

Chien C, Hsu C (2006) A novel method for determining machine subgroups and backups with an empirical study for semiconductor manufacturing. J Intell Manuf 17:429–440

Chien C, Chen H, Wu J, Hu C (2006) Construct a Group OEE for promoting tool group productivity in semiconductor manufacturing. Int J Prod Res 44:1–16

Gen M, Cheng R (1997) Genetic algorithms and engineering design. Wiley, New York

Giffler B, Thompson G (1960) Algorithms for solving production scheduling problems. Oper Res 8(4):487–503

Gupta JND, Strusevich VA, Zwaneveld CM (1997) Two-stage no-wait scheduling models with setup and removal times separated. Comput Oper Res 24(11):1025–1031

Hall NG, Sriskandarajah C (1996) A survey of machine scheduling problems with blocking and no-wait in process. Oper Res 44(3):510–525

Leachman RC, Hodges DA (1996) Benchmarking semiconductor manufacturing. IEEE Trans Semicond Manuf 9(2):158–169

Lin Bertrand MT, TC Edwin Cheng (2001) Batch scheduling in the no-wait two-machine flowshop to minimize the makespan. Comput Oper Res 28(7):613–624

Lin SY, Huang HP (1998) Modeling and emulation of a furnace in IC fab based on colored-timed Petri net. IEEE Transact Semicond Manuf 11(3):410–420
Macchiaroli R, Mole S, Riemma S (1999) Modeling and optimization of industrial manufacturing processes subject to no-wait constraints. Int J Prod Res 37(11):2585–2607
Mascis A, Pacciarelli D (2002) Job-shop scheduling with blocking and no-wait constraints. Eur J Oper Res 143:498–517
Piersma N, Dijk WV (1996) A local search heuristic for unrelated parallel machine scheduling with efficient neighborhood search. Math Comput Model 24(9):1–19
Potts CN, Kovalyov MY (2000) Scheduling with batching: a review. Eur J Oper Res 120(2):228–249
Pranzo M (2004) Batch scheduling in a two-machine flow shop with limited buffer and sequence independent setup times and removal times. Eur J Oper Res 153(3):581–592
Raaymakers WHM, Hoogeveen JA (2000) Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. Eur J Oper Res 126:131–151
Reeves C (1995) A genetic algorithm for flow shop sequencing. Comput Oper Res 22:5–13
Schmidt G (2000) Scheduling with limited machine availability. Eur J Oper Res 121:1–15
Schuster CJ, Framinan JM (2003) Approximative procedures for no-wait job shop scheduling. Oper Res Lett 31:308–318
Sidney JB, Potts CN, Sriskandarajah C (2000) A heuristic for scheduling two-machine no-wait flow shops with anticipatory setups. Oper Res Lett 26(4):165–173
Uzsoy R, Lee CY, Martin-Vega LA (1992) A review of production planning and scheduling models in the semiconductor industry: Part I. IIE Trans 24(4):47–60