8

ACO-based Multi-objective Scheduling of Identical Parallel Batch Processing Machines in Semiconductor Manufacturing

Li Li, Pan Gu, Fei Qiao, Ying Wu and Qidi Wu
Tongji University
China

1. Introduction

The batch processing machines (BPMs) have the ability to process more than one job together (called a batch). So the scheduling problem of the BPMs concerns not only the priorities of the jobs obtaining the processing service of a BPM, but the number of the jobs processed together on them. According to diverse classified criteria (such as the number of the BPMs and the job families), the scheduling problem of the BPMs can be further divided into several styles, e.g., a single BPM scheduling problem (SBPM), identical parallel BPMs scheduling problem (PBPM) , non-identical PBPM, the BPMs scheduling problem with compatible job families and the BPMs scheduling problem with incompatible job families.

In this paper, we address the BPMs scheduling problem in a semiconductor wafer fabrication facility (fab), in where there are many BPMs, such as diffusion machines, oxidation machines and dry strip machines. The jobs processed on those machines cannot be batched together unless they use the same recipe of those BPMs. As a result, the scheduling problem of those BPMs is abstracted as identical PBPM with incompatible job families. In a fab, because most of upstream and downstream machines of the BPMs are non-BPMs, jobs must be batched or split regularly during their fabrication processes. Therefore, a good scheduling solution of those BPMs is essential to efficiently utilize their capacity and satisfy the requirements of their downstream machines to balance the fab-wide workload and achieve better fab-wide operational performance.

In recent years, there have been many studies of the BPMs scheduling problem. Mathirajan and Sivakumar (Mathirajan & Sivakumar, 2006) have reviewed 98 articles published between 1986 and 2004 on this topic, and the research has considerably evolved since 2004. For example, to minimize the makespan or average flow time of the jobs, Chien and Chen (Chien & Chen, 2007) developed a genetic algorithm (GA) for batch sequencing combined with a novel timetabling algorithm to handle waiting time constraints, frequency-based setups, limited machine availability and a rolling horizon-based scheduling mechanism for scheduling of furnaces for semiconductor fabrication. Chou et al. (Chou et al., 2006) presented a hybrid GA for SBPM with arbitrary job release times. To meet due date requirements from customers, Gupta and Sivakumar (Gupta & Sivakumar, 2007) presented a dynamic scheduling method for SBPM with a look-ahead batching strategy to control the delivery performance between earliness and tardiness measures. Erramilli and Mason (Erramilli & Mason, 2006) proposed a

mixed integer program and a simulated annealing (SA)-based heuristic method to solve SBPM to minimize the total weighted tardiness (TWT). To be applicable to a real production environment, some research work has also considered upcoming jobs. Solomon et al.(Solomon et al., 2002) presented a dispatching policy for the BPMs that incorporated knowledge of future arrivals, the status of critical machines in subsequent processing, and setup times into batch processing scheduling. Mönch et al. (Mönch et al., 2006) proposed a simple heuristic method based on the Apparent Tardiness Cost (ATC) Dispatching Rule to minimize the TWT on PBPM with incompatible job families and unequal job ready times, in which inductive decision trees and neural networks from machine learning were used to estimate the look-ahead parameter. Liu et al. (Liu et al., 2007) proved that SBPM of minimizing the total tardiness was NP-hard even if the machine capacity was only two jobs. Accordingly, most studies have used heuristic rules (e.g.,(Gupta & Sivakumar, 2007; Solomon et al., 2002; Mönch et al., 2006)) or metaheuristic searching methods (e.g., (Chien & Chen, 2007; Chou et al., 2006; Erramilli & Mason, 2006)). Although heuristic rules can reach a solution quickly, they are myopic algorithms that pursue local optimization without considering global optimization. Consequently, the metaheuristic searching methods (such as GA and SA) have been gradually adopted to obtain global optima.

Ant Colony Optimization (ACO), inspired by the foraging behavior of real ant colonies, is a population-based approach developed by Dorigo in 1992 (Dorigo M, 1992). ACO has been successfully applied to several NP-hard combinatorial optimization problems, such as the Traveling Salesman Problem (TSP), Quadratic Assignment Problem (QAP), Vehicle Routing Problem (VRP), Job-Shop Scheduling Problem (JSP), Flow-Shop Scheduling Problem (FSP), etc.(Dorigo M. & Stützle T., 2004). However, few researchers have applied ACO to solve the BPMs scheduling problem. Only Srinivasa Raghavan and Venkataramana (Srinivasa & Venkataramana, 2006) used an ACO algorithm to solve a static scheduling problem of minimizing the TWT of PBPM with incompatible job families.

In this paper, firstly, we build an identical PBPM model concerned the practical considerations of incompatible jobs, dynamic job arrivals, sequence-dependent setup times and the qual-run requirements of advanced process control (APC). Then, we propose an ACO-based solution to simultaneously minimize the TWT and makespan of the jobs. Finally, the effectiveness of the proposed method is demonstrated by a variety of simulation experiments. The simulation results show that the proposed method produces smaller TWT and makespan than the common Apparent Tardiness Cost-Batched Apparent Tardiness Cost (ATC-BATC) rule, Max Batch Size rule (MBS), a GA and an Ant System algorithm (AS).

The rest of this paper is organized in four sections. In Section 2, we present the problem description and assumptions. Then we outline the ACO-based solution in Section 3. In section 4, we show computational experiments and results. Finally, we give conclusions and future research topics in section 5.

2. Problem Description and Assumptions

2.1 Problem description

With the 3-field notation, the PBPM scheduling problem in this paper can be denoted as

$$M|A_{ij}, Q_i, Batch, incompatible|\ min(\sum_i \sum_j w_{ij} Tij\ +\ max_{i,j}(F_{ij})) \tag{1}$$

where M is the number of the BPMs in PBPM; A_{ij} is the arrival time of job j of family i; Q_i is the qual-run time of family i; w_{ij} and Tij are the weight, the tardiness and the completion time of job j of family i, respectively.

There are two way to solve a PBPM scheduling problem. One is to distribute the jobs to PBPM first, then batch the jobs and determine the priorities of the batches on each BPM. The other is to batch the jobs first, then distribute the batches to PBPM and sequence the batches on each BPM. Balasubramanian et al.(Balasubramanian et al., 2004) have shown by extensive simulations that the second way achieved better solutions with less computation time. Therefore, we have adopted the second style.

There are two main constraints when forming the batches. First, only jobs belonging to the same family can be processed together. Second, the number of the jobs in a batch cannot exceed the capacity of the PBPM (i.e., maximum batch size constraint). Another important consideration is the trade-off between the waiting time for forming a full batch and the waste of the PBPM capacity.

Distributing and sequencing the batches are the same as in other problems of scheduling parallel machines. The issues to consider are the hot lots, workload balance and the utilization of the PBPM. It is also worthwhile to consider the trade-off between the setup times of scheduling and the qual-run requirements of APC. In real semiconductor manufacturing environments, APC could achieve the best quality result by frequent changeovers between jobs from different families, possibly avoiding the need for qual-runs. However, frequent changeovers cost setup time and cause capacity loss. Instead of achieving the best quality, APC determines a parameter range which represents acceptable quality for every job family. Based on this range, APC provides a threshold value n_i for each job family i. If a machine has been processing no less than n_i jobs (or batches for BPMs) from family $j(j \neq i)$,then before the next time it processes jobs from family i ,a qual-run is required on that machine. In a qual-run, no real job is processed. A blank wafer is processed to obtain the status of the machine so that the operator can properly set the machine parameters to achieve high quality results. Before the result of the qual-run is available, jobs cannot be processed on that machine. Therefore, a trade-off between the time lost for setups and the time lost due to qual-runs is required. However, most related research has not considered the qual-run requirements of APC. We have found only the studies of Cai et al. (Cai et al., 2007) and Patel (Patel N-S, 2004) that incorporated the constraint of process control into scheduling decisions.

2.2 Problem assumptions

The assumptions involved in the PBPM scheduling problem include:

(i) The machines in the PBPM are identical;

(ii) The PBPM scheduling problem is considered with a schedule horizon (e.g., one shift, one day or several days), within which the scheduling plan of the jobs from multiple families is decided;

(iii) The processing time of a batch on one machine is independent of the number of the jobs in the batch;

(iv) Once processing begins on a batch, no job can be removed from or added to the machine until it finishes.

(v) There are sequence-dependent random setup times for changeovers between jobs from different families, and no setup times between jobs from the same family.

3. ACO-Based Solution

3.1 Build a search space

Before we use an ACO algorithm to find a solution, the first task is to build a search space for the ACO algorithm. In this paper, the search space is composed of nodes which are combinations of the batches and the BPMs in the PBPM.

For N_i jobs, there are $C_{N_i}^1 + C_{N_i}^2 + ... + C_{N_i}^B$ (C is the combination operator) batching styles, subject to the constraint of maximum batch size. In the case of many jobs (especially with a number of dynamic arrival jobs), this kind of batching style will result in lower computation efficiency. In this paper, we form the batches using the time window concept (denoted by Δt) proposed by Mönch et al. (Mönch et al., 2005).

$$\Delta t = dt \times Avg(P_{ij}) \tag{2}$$

where P_{ij} is the processing time of job j of family i ; $Avg(P_{ij})$ is the average processing time of the jobs; dt is a distribution parameter of Δt .

At each batching decision point t (t is set as the earliest ready time of the jobs to be batched), the jobs of family i with arrival (ready) time less than the upper boundary of the time window interval $t + \Delta t$ is denoted as $M(j, t, \Delta t) = \{ij | A_{ij} \leq t + \Delta t\}$ .Then, we batch the jobs in $M(j, t, \Delta t)$ subject to the maximum batch size constraint. We repeat the above process until all jobs have been assigned to a batch. The ready time of each batch equals the latest arrival time of the jobs in the batch. Finally, the search space (denoted as S)is built with nodes composed of the batches and the BPMs in the PBPM.

Table 1 shows a simple example of building a search space. We assume that there are 2 machines in the PBPM and their batch size is 2 jobs. There are 2 families of jobs whose processing times are set to 10 min and 15 min, respectively. For each family, there are 3 jobs to be scheduled. Here dt is set to 1. Then the time window Δt can be computed as $\Delta t = 1 \times (10 + 10 + 10 + 10 + 15 + 15 + 15)/6 = 12.5\ min$ .The batches formed are shown in Table 2. These batches and machines constitute the search space $S = \{(l_{11}, m_1), (l_{12}, m_1), ((l_{11}, l_{12}), m_1), (l_{13}, m_1), (l_{21}, m_1), (l_{22}, m_1), (l_{23}, m_1), ((l_{22}, l_{23}), m_1), (l_{11}, m_2), (l_{12}, m_2), ((l_{11}, l_{12}), m_2), (l_{13}, m_2), (l_{21}, m_2), (l_{22}, m_2), (l_{23}, m_2), ((l_{22}, l_{23}), m_2)\}$,whose size is 16 nodes.

$Job(l_{ij})$	l_{11}	l_{12}	l_{13}	l_{21}	l_{22}	l_{23}
$A_{ij}(min)$	0	5	15	8	13	18

Table 1. An example of building a search space

$Batches$	l_{11}	l_{12}	(l_{11}, l_{12})	l_{13}	l_{21}	l_{22}	l_{23}	(l_{22}, l_{23})
$A_{Batch}(min)$	0	5	5	15	8	13	18	18

Table 2. The formed batches for the simple example

3.2 Find a solution with an ACO algorithm

The parameters used in the ACO algorithm to find a solution are defined in Table 3.

Parameter	*Meaning*
m	The index of the BPMs in PBPM
B	The capacity of the BPMs in PBPM
ij	The index of the jobs, which means job j of family i
Δt	The time window for job batching
dt	The distribution parameter of time window Δt
K	The number of the ants in the artificial ant colony
k	The index of the artificial ants
t_{max}	The maximum number of iterations
t_1	The iteration index
δ	The minimum change of the minimum objective values in two consecutive iterations
L^k_{tabu}	The tabu-list of ant k
L^k_{task}	The task-list of ant k
τ_0	The initial pheromone on each arc
$WT_{ATC-BATC}$	The TWT of the scheduling results obtained by ATC-BATC rule
$F_{ATC-BATC}$	The makespan of the scheduling results obtained by ATC-BATC rule
l	The task-list of ant k
c	A candidate node in L^k_{task}
$\tau_{c_0 l}$	The pheromone on the arc (c_0, l)
c_0	The last node selected by artificial ant k using the same machine as c
q	A probability parameter $(0 \leq q \leq 1)$
α	A parameter denoting the relative importance of the pheromone density and the heuristic factor
$\eta_{c_0 c}$	The heuristic factor if c is selected as the successor task of c_0
P_c	The processing time of c
$U_{c_0 c}$	The setup time for the changeover between c_0 and c
x_c	The qual-run parameter of c
A_c	The arrival time of c
Q_c	The qual-run time of c
F_{c_0}	The processing finish time of c_0
B_c	The batch size of c
W_c	The workload of the machine processing c_0 if c is selected as the successor task of c_0
W_m	The workload of machine m
γ	A parameter to regulate the workload among the BPMs in PBPM
$min_m(F_{t_1-1})$	The earliest finish time of the BPMs in PBPM in iteration $t_1 - 1$
$max_m(F_{t_1-1})$	The latest finish time of the BPMs in PBPM in iteration $t_1 - 1$
ξ	A parameter to reduce the pheromone trail on an arc used by an ant to make it less attractive to the following ants
OV_{t_1}	The minimum objective value of the solutions in iteration t_1
OV_{t_1-1}	The minimum objective value of the solutions in iteration $t_1 - 1$
x, y	Two different nodes in the search space

$\tau_{xy}(t_1)$	The pheromone on arc (x,y) in iteration t_1
$\tau_{xy}(t_1+1)$	The pheromone on arc (x,y) in iteration t_1+1
$\Delta\tau_{xy}^{bs}$	The new pheromone deposition related to the best-so-far solution
ρ	The pheromone evaporation parameter
T^{bs}	The best-so-far solution during the search process

Table 3. The list of parameters used in the ACO algorithm

The detailed flowchart of the ACO algorithm is shown in Figure 1.

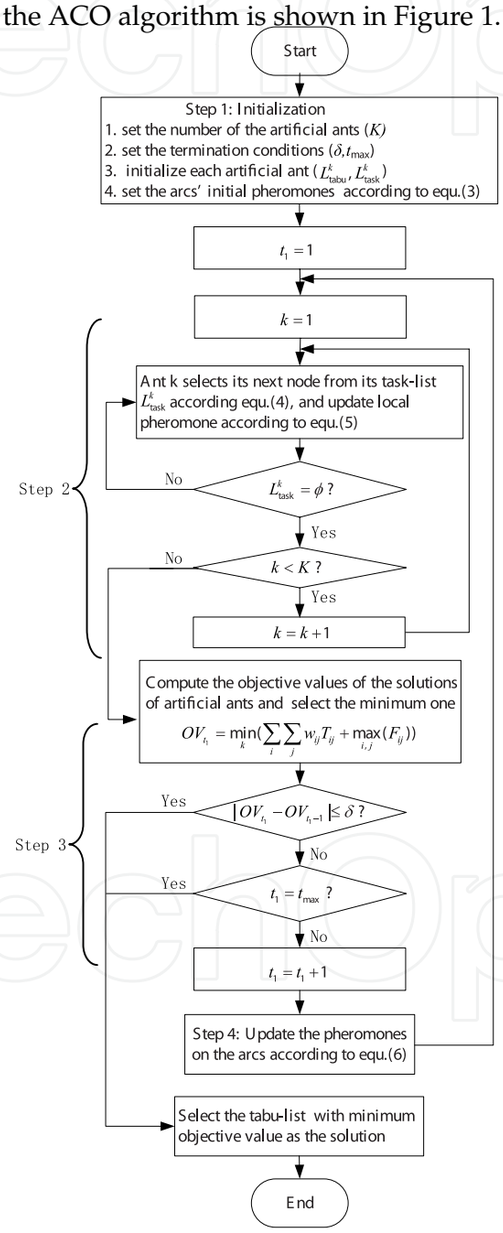


Fig. 1. The flowchart of the ACO algorithm

Step 1: Initialization. There are four main tasks in the initialization stage: to determine the number of the ants in the artificial ant colony, to set the termination conditions for the search, to initialize each artificial ant, and to set the initial pheromones on the arcs.
a) The number of ants in the artificial ant colony
Here we set the number of ants in the artificial ant colony to the number of nodes in the search space. For example, the number of artificial ants for solving the problem in Table 1 can be set to 16.
b) The termination conditions
Here we set two kinds of termination conditions. One is the maximum number of iterations (denoted by t_{max}). The other is the minimum change of the minimum objective values in two consecutive iterations (denoted by δ).
c) Initialization of each artificial ant
First, we build a tabu-list and a task-list for each artificial ant (indexed with k), denoted by L^k_{tabu} and L^k_{task} , whose initial values are set to ϕ and S ,respectively. Then, we distribute the start points (i.e., the nodes in the search space) randomly to each artificial ant. The node distributed to ant k is added to L^k_{tabu} , and deleted from L^k_{task} . To guarantee that each job is processed only once, the nodes with the same job as the distributed node are also deleted from L^k_{task} .Take the problem in Table 1 as an example, and assume that the node (l_{11}, m_1) is assigned to ant 1 .Then the tabu-list and task-list of ant 1 become $\{(l_{11}, m_1)\}$ and $\{(l_{12}, m_1), (l_{13}, m_1), (l_{21}, m_1), (l_{22}, m_1), (l_{23}, m_1), ((l_{22}, l_{23}), m_1), (l_{12}, m_2), (l_{13}, m_2), (l_{21}, m_2), (l_{23}, m_2), ((l_{22}, l_{23}), m_2)\}$,respectively.
d) Initialization of the pheromone on each arc
The initial pheromone on each arc is set with the scheduling results obtained by the ATC-BATC rule, which also guarantees that the scheduling results achieved by the proposed ACO algorithm are no worse than those of the ATC-BATC rule.

$$\tau_0 = 1 / (K \times (WT_{ATC-BATC} + F_{ATC-BATC})) \tag{3}$$

Step 2: Each artificial ant searches for its solution. Artificial ant k selects its next node l from its task-list L^k_{task} according to the so-called pseudorandom proportional rule, given by

$$l = \begin{cases} arg\ max_{c \in L^k_{task}} \{\frac{\alpha\tau_{c_0c} + (1-\alpha)\eta_{c_0c}}{\Sigma_c \alpha\tau_{c_0c} + (1-\alpha)\eta_{c_0c}}\}, & q \le q_0 \\ max_c(rand(0,1) \times \frac{\alpha\tau_{c_0c} + (1-\alpha)\eta_{c_0c}}{\Sigma_c \alpha\tau_{c_0c} + (1-\alpha)\eta_{c_0c}}, & otherwise \end{cases} \tag{4}$$

$$\eta_{c_0c} = (1 - \frac{P_c + U_{c_0c} + x_c Q_c + max((A_c - F_{c_0}),0)}{max_c(P_c) + max_c(U_{c_0c}) + max_c(Q_c) + max((max_c(A_c) - F_{c_0}),0)}) + \frac{B_c}{B} + \gamma \Delta W_c$$

$$\Delta W_c = \begin{cases} \frac{W_c}{max_m(W_m)}, & W_c \le max_m(W_m) \\ \frac{W_c}{max_m(W_m)} - 1, & W_c > max_m(W_m) \end{cases}$$

$$\gamma = \frac{\Sigma_i \Sigma_j P_{ij}/M - min_m(F_{t_1-1})}{max_m(F_{t_1-1}) - \Sigma_i \Sigma_j P_{ij}/M}$$

$$x_c = \begin{cases} 1, & \text{if no less than } n_i \text{batches from family} j (j \ne c) \text{have been processed} \\ -1, & \text{if} (n_i - 1) \text{batches from family} j (j \ne c) \text{have been processed} \\ 0, & otherwise \end{cases}$$

Obviously, ant k selects the node with the highest attractiveness indicated by the learned pheromone trails and the heuristic information with probability q_0 , while with probability $1-q_0$ it performs a biased exploration of the arcs. The heuristic factor η_{c_0c} simultaneously takes into consideration on c 's occupation time (including possible qual-run time, processing time, setup time and waiting time), the capacity utilization rate and the relative workload of the machine.

The selected node is added to L^k_{tabu} and deleted from L^k_{task} .Meanwhile, the nodes with the same job as the selected node are also deleted from L^k_{task} .Then, the local pheromone trail is updated

$$\tau_{c_0l} \;=\; (1-\xi)\tau_{c_0l} \;+\; \xi\tau_0 \tag{5}$$

The parameter ξ allows artificial ants to increase their exploration of new arcs, and in practice avoids a stagnation behavior (i.e., the ants do not converge on a common path).

The process is repeated until L^k_{task} is empty. Obviously, the tabu-list L^k_{tabu} is the solution obtained by ant k 's search process.

Step 3: Determine whether the termination conditions are satisfied. First, we compute the objective values of the solutions obtained by the artificial ants. Then we select the minimum to compare with that of the last iteration. If the difference between these two consecutive minimum objective values is no more than a small positive value (denoted by δ), we stop the search process. The tabu-list with the minimum objective value is taken as the solution. Otherwise, we determine whether t_{max} has been reached. If the answer is yes, we select the tabu-list with the minimum objective value as the solution. Otherwise, we go to step 4.

Step 4: Pheromone updating. We update the pheromone values on the arcs with the best-so-far solution according to equation (6), and then repeat steps 2 and 3.

$$\tau_{xy}(t_1+1) \;=\; (1-\rho)\tau_{xy}(t_1) \;+\; \rho\Delta\tau^{bs}_{xy}, \forall(x,y)\in T^{bs} \tag{6}$$

$$\Delta\tau^{bs}_{xy} = 1/min_k(\Sigma_i\Sigma_j w_{ij}T_{ij} + max_{i,j}(F_{ij})),\;\; 0<\rho<1$$

The pheromone trail update, both evaporation and new pheromone deposition, only applies to the arcs of the best-so-far solution, not to all the arcs. In this way, the computational complexity of the pheromone update at each iteration is reduced from $O(K^2)$ to $O(K)$ .The deposited pheromone is discounted by a factor ρ , which results in the new pheromone trail being a weighted average between the old pheromone value and the newly deposited pheromone.

4. Computational experiments and results

4.1 The scheduling methods to be compared

We compared the performances of the proposed ACO algorithm with those of ATC-BATC, MBS, a GA (refer to (Balasubramanian et al., 2004)) and an Ant System algorithm (refer to (Li et al., 2008)). As a common BPM scheduling rule, ATC-BATC has good performance on static BPM scheduling problems. To adapt it to dynamic jobs arrival, we made some small modifications to the rule. The decision flow of the modified ATC-BATC rule is as follows.

First, at each batching decision point t , compute the index of job j of family i which belongs to $M(j,t,\Delta t)$.

$$I_{ij}(t_0) \;=\; \frac{w_{ij}}{P_{ij}} exp(\,-\,\frac{(d_{ij}-P_{ij}-t_0+(A_{ij}-t_0)^+)^+}{k_1\overline{P}}) \tag{7}$$

where w_{ij} , P_{ij} , d_{ij} and A_{ij} are the weight, the processing time, the due date and the arrival time of job j of family i ,respectively; t_0 is the scheduling decision point; k_1 is the look-ahead parameter, which usually ranges from 0.1 to 5; and $\overline{P}$ is the average processing time of the jobs. Then form batches by selecting jobs in a non-increasing order of I_{ij} ,subject to the maximum batch size constraint. Repeat the above process until the batching is complete. Second, make a batch sequence in non-increasing order of I_{bi} according to equation (8) and distribute the batches to BPMs of the PBPM in turn

$$I_{bi}(t_0) = \Sigma_{j=1} n_{bi} I_{ij}(t_0) \times min(\frac{n_{bi}}{B}, 1) \tag{8}$$

where n_{bi} is the number of the jobs in a batch of family i .

4.2 The problem cases for the simulations

We used the dry strip operations in a real wafer fab to demonstrate the proposed ACO algorithm. There are 3 identical machines for the dry strip operations in this wafer fab. Each machine has a capacity of 3 jobs and can process 4 different recipes. The threshold value for the qual-run requirements of each recipe is 3 jobs. The processing times for $recipe_1$, $recipe_2$, $recipe_3$ and $recipe_4$ are random variables from the uniform distributions $Uniform(90,100), Uniform(90,100), Uniform(70,80) and Uniform(90,100)$, respectively. The setup times are from the distribution $Uniform(10,20)$ .The qual-run times for $recipe_1$, $recipe_2$, $recipe_3$ and $recipe_4$ conform to the distributions $Uniform(30,40), Uniform(30,40), Uniform(20,30)$, and $Uniform(30,40)$, respectively. The time unit is minutes.

4.3 Determining the distribution parameter dt for time window Δt

Although Mönch et al.(Mönch et al., 2005) presented the concept of the time window, they did not specify how to determine its value. We ran a number of simulations on random problem cases (shown in Table 4) to determine the best value for the distribution parameter dt of the time window Δt . ATC-BATC was used for the simulations, with its look-ahead parameter k_1 set to 0.5 (according to extensive simulations). Values of dt in the range from 0 to 2.0 at intervals of 0.25 were tested. The simulation results are shown in Table 5 and Figure 2. From the simulation results, we can reach the following conclusions.

Problem parameter	*Value used*	*Number of values*
Number of jobs	20	1
Arrival times of jobs	$Uniform(-r\Sigma_i\Sigma_j P_{ij}/(BM), r\Sigma_i\Sigma_j P_{ij}/(BM))$ $r = 0.25, 0.50, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0$	8
Due dates of jobs	$A_{ij} + P_{ij} + Uniform(0, Avg(P_{ij}))$	1
Time window Δt	$dt \cdot Avg(P_{ij})$ $dt = 0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2.0$	9
Weight per job	$Uniform(0,1)$	1
	Total parameter combinations	72
	Number of problems per combination	5
	Total problems	360

Table 4. Problem cases for determining the time window Δt

	$r = 0.25$	$r = 0.5$	$r = 0.75$	$r = 1.0$	$r = 1.25$	$r = 1.5$	$r = 1.75$	$r = 2.0$
$dt = 0.25$	169.40	242.45	239.39	335.41	257.45	253.24	312.09	345.01
$dt = 0.50$	168.44	226.08	182.97	250.39	257.18	253.24	262.68	345.01
$dt = 0.75$	168.44	178.25	182.97	250.39	250.27	251.35	277.44	342.01
$dt = 1.00$	168.44	178.25	182.97	230.49	205.27	233.97	257.13	330.12
$dt = 1.25$	168.44	178.25	182.34	230.49	206.65	233.97	257.13	330.12
$dt = 1.50$	168.44	178.25	182.34	230.49	206.65	233.97	261.68	332.98
$dt = 1.75$	168.44	178.25	182.34	230.20	218.49	233.97	261.68	333.52
$dt = 2.00$	168.44	178.25	182.34	230.20	218.49	233.97	262.90	333.52

Table 5. The average objective values with variables dt and r

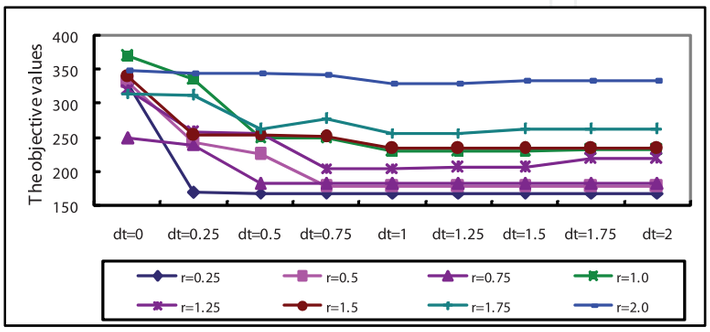


Fig. 2. The objective values with variables dt and r

i) The parameter dt interacts strongly with the arrival time distribution parameter r ; the best choice is $dt = 1$ for most cases. ii) Larger dt is not better than smaller dt when r is large. For example, when r was set to 1.25, 1.75 or 2.0, the objective values with $dt = 1.5$, $dt = 1.75$ or $dt = 2.0$ were more than with $dt = 1$.

In addition, we simulated the same problem cases with the proposed ACO algorithm. The parameters q_0 , α , ρ , δ , ξ and t_{max} were set to 0.5, 0.5, 0.1, 0.001, 0.1 and 100, respectively. The average improvements of ACO with different values of r compared with ATC-BATC are shown in Figure 3. The improvement increased with r , with an inflection in the curve at $r = 1$. When r was from 0.25 to 1 or from 1.25 to 2, the larger was r , the better were the improvements by ACO. Furthermore, the improvements for r above 1 were better than those for r from 0.25 to 1. In the following simulations, we only consider r values from 0.25 to 1.

4.4 Determining the values of the parameters in the ACO algorithm

a) Determine the probability parameter q_0 Tuning the parameter q_0 allows adjusting the degree of exploration and the choice of whether to concentrate the search around the best-so-far solution or to explore other solutions. We determined the probability parameter q_0 of the proposed ACO algorithm with the same problem cases shown in Table 4 with the time window Δt 's distribution parameter dt set to 1. The parameters α , ρ , δ , ξ and t_{max} were set to 0.5, 0.1, 0.001, 0.1 and 100, respectively. The simulation results are shown in Figure 4. From these results, we can conclude that $q_0 = 0.2$ is the best selection for most cases.

b) Determine the pheromone importance parameter α We determined the pheromone importance parameter α with the same problem cases shown in Table 4 with $dt = 1$ and $q_0 = 0.2$.

The parameters , ρ , δ , ξ and t_{max} were set to 0.1, 0.001, 0.1 and 100, respectively. The simulation results are shown in Figure 5. In all cases, $\alpha = 0.7$ was the best choice.

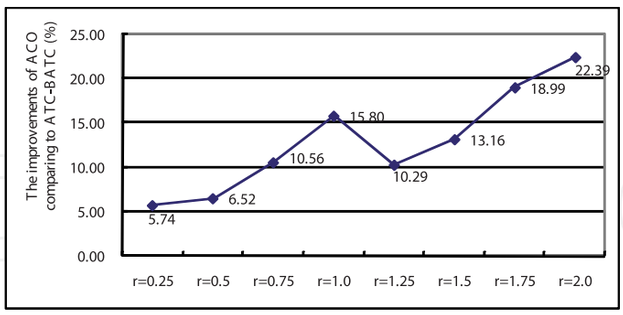


Fig. 3. Analysis of r 's impacts on the ACO algorithm's performance

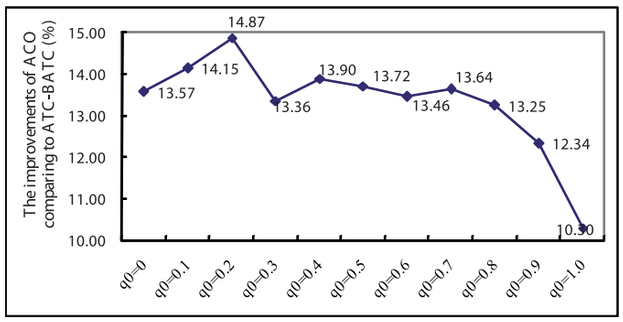


Fig. 4. The simulation results for determining the probability parameter q_0

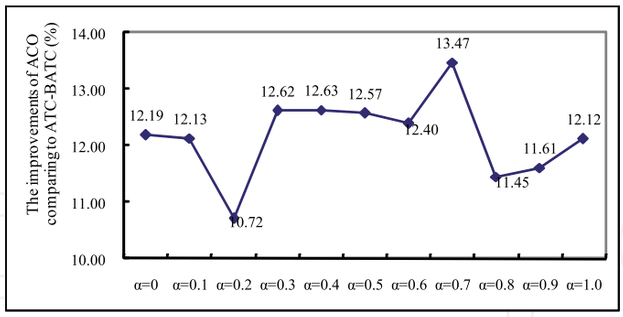


Fig. 5. The simulation results for determining the probability parameter α

4.5 Comparison between ACO, ATC-BATC, MBS, GA and AS

With the above simulation results, the parameters of the ACO algorithm were set as Table 6. The parameters of the GA and the AS were set according to (Balasubramanian et al., 2004) and (Li et al., 2008), respectively.

The problem cases for comparing between ACO, ATC-BATC, MBS, GA and AS are shown in Table 7. In the simulations, we considered the impacts of the number and the arrival time

Parameter	*Value*
α	0.7
q_0	0.2
ρ	0.1
δ	0.001
ξ	0.1
t_{max}	100

Table 6. The parameters of the ACO algorithm

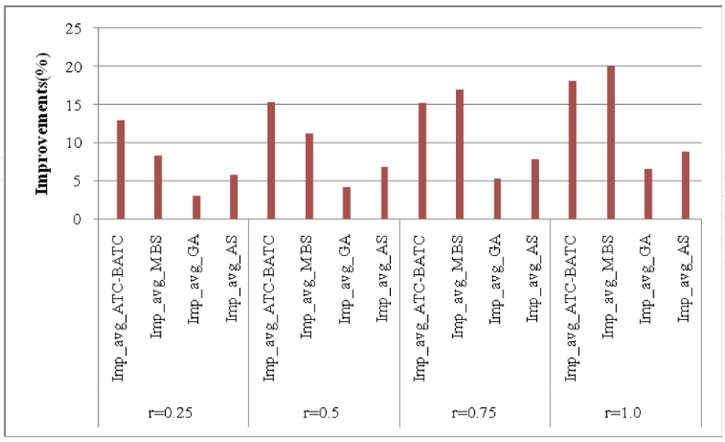
distribution of the jobs on the ACO algorithm's performance. The number of jobs was gradually increased by multiplying the number of machines and the number of recipes on each machine. The average improvements on the TWT and makespan of ACO are shown in Figure 6. From the simulation results, we can make the following conclusions.

Problemparameter	*Valueused*	*Numberofvalues*
Numberofjobs	20,32,44,56,68,80	6
Arrivaltimesofjobs	$Uniform(-r\Sigma_i\Sigma_j P_{ij}/(BM), r\Sigma_i\Sigma_j P_{ij}/(BM))$ $r = 0.25, 0.50, 0.75, 1.0$	4
Duedatesofjobs	$A_{ij} + P_{ij} + Uniform(0, Avg(P_{ij}))$	1
TimewindowΔt	$dt \cdot Avg(P_{ij}), dt = 1$ $dt = 0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2.0$	1
Weightperjob	$Uniform(0, 1)$	1
	Totalparametercombinations	24
	Numberofproblemspercombination	10
	Totalproblems	240

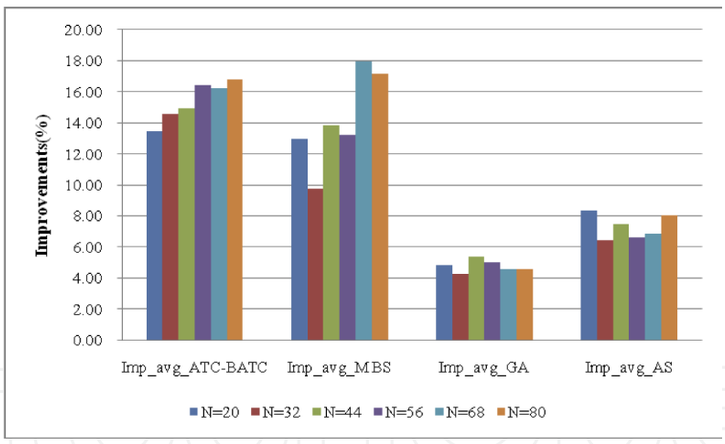
Table 7. The problem cases for comparing ACO and ATC-BATC

i) The value of the arrival time distribution parameter r had an important impact on the ACO algorithm's average improvements on the TWT and makespan. Larger r , i.e., the job arrivals were spread over a larger time range, resulted in better improvements on the TWT and makespan. In addition, the performance of MBS increasingly deteriorated with larger r (shown in Figure 6(a)).
ii) Comparing to the heuristic rules (ATC-BATC and MBS), the number of jobs affected the ACO algorithm's average improvements on the average of the TWT and makespan. The more jobs, the better the average improvements, independent on r 's value. However, comparing to the GA and AS, the impact of change in the number of jobs on the improvements of ACO on the average of the TWT and makespan fluctuated (shown in Figure 6(b)).
To further discuss the impacts of the batch size and the number of the recipes on the BPMs, and the number of the BPMs on the performance of the proposed method, it is assumed that the range of the number of the machines is from 3 to 5, the range of the batch size of a BPM is from 3 to 5, and the range of the number of the recipes of a BPM is from 4 to 6. Other conditions are the same as Table 7. The simulation results are shown in Figure 7. Obviously, the number and the capacity of the machines and the number of the recipes play an important role on the average improvements on the TWT and makespan of the ACO algorithm. Less machines, the bigger capacity and more recipes, the more average improvements on the TWT

and makespan are. In addition, the performance of MBS increasingly improved with more recipes and bigger capacity.



(a) The improvements by ACO with variable arrival time distribution of jobs * *Imp_avg_ATC_BATC* the average improvements on the TWT and makespan of ACO compared to ATC-BATC; the *Imp_avg_GA* : the average improvements on the TWT and makespan of ACO compared to GA; *Imp_avg_AS* : the average improvements on the TWT and makespan of ACO compared to AS



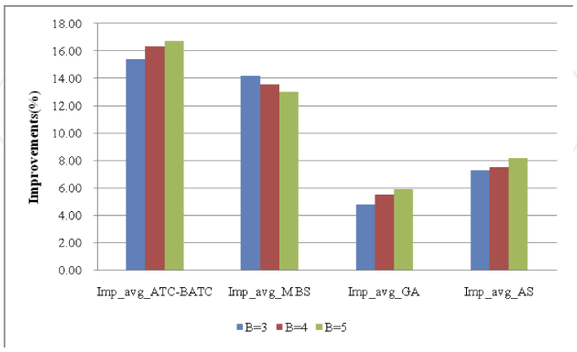
(b) The improvements by ACO with variable number of the jobs

Fig. 6. Simulation results for comparison between ACO and ATC-BATC, MBS, GA and AS

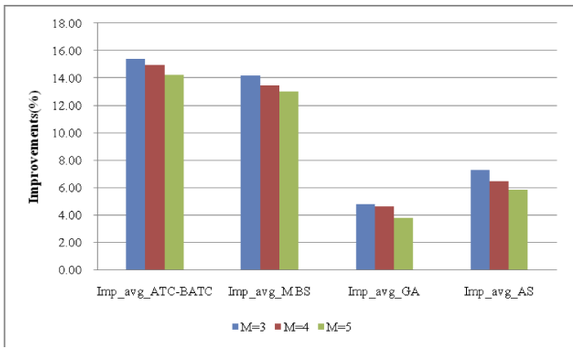
5. Conclusions

Batch processing machines play important roles in semiconductor wafer fabrication facilities. In this paper, we modeled the batch processing operations in a real wafer fab as an identical PBPM problem considering the practical complications of incompatible job families, dynamic job arrivals, sequence-dependent setup times and qual-run requirements of APC, and proposed an ACO algorithm to solve the problem with smaller TWT and makespan than

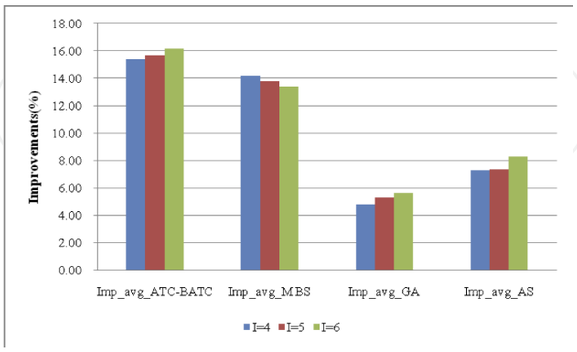
ATC-BATC, MBS, GA and AS. The main contributions of the paper are to create a method applicable in a production environment, to propose a better value for the time window Δt from simulations, and to apply the ACO algorithm to obtain the solutions. Our next step is to integrate the ACO algorithm with the advanced planning and scheduling software of the real wafer fab.



(a) The improvements by ACO with variable capacity of a BPM



(b) The improvements by ACO with variable number of the BPMs



(c) The improvements by ACO with variable number of the recipes of a BPM

Fig. 7. The impacts of the batch size, the number of the recipes on the BPMs and the number of the BPMs

6. Acknowledgement

This work was supported in part by National Natural Science Foundation of China (No.50905129,No.70531020), Program for Young Excellent Talents in Tongji University (No.2006KJ006), the Grant from the Ph.D. Programs Foundation of Ministry of Education of China (No. 20070247007), and Shanghai Innovation Program (No. 09DZ1120600).

7. References

Mathirajan,M.& Sivakumar,A.I., (2006). A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. *International Journal of Advanced Manufacturing Technology*, Vol.29, No.9-10, July 2006,990-1001,ISSN 0268-3768

Chien C-F. & Chen C-H, (2007). A novel timetabling algorithm for a furnace process for semiconductor fabrication with constrained waiting and frequency-based setups. *OR Spectrum*, Vol.29, No.3, July 2007,391-419,ISSN 0171-6468

Chou F-D.; Chang P-C & Wang H-M (2006). A hybrid genetic algorithm to minimize makespan for the single batch machine dynamic scheduling problem. *International Journal of Advanced Manufacturing Technology*, Vol.31, No.3-4, Nov 2006,350-359,ISSN 0268-3768

Gupta A.K. & Sivakumar A.I. (2007). Controlling delivery performance in semiconductor manufacturing using look ahead batching. *International Journal of Production Research*, Vol.45, No.3, Feb 2007, 591-613(23), ISSN 0020-7543

Erramilli V. & Mason S.J. (2006). Multiple orders per job compatible batch scheduling. *IEEE Transactions on Electronics Packaging Manufacturing*, Vol.29, No.4, Oct 2006, 285-296, ISSN 1521-334X

Solomon M., Fowler J.W., Pfund M.et al.(2002). The inclusion of future arrivals and down-stream setups into wafer fabrication batch processing desicions. *Journal of Electionics Manufacturing*, Vol.11, No.2,2002,149-159, ISSN 0960-3131

Mönch L.; Zimmermann J. & Otto p (2006). Machine learning techniques for scheduling jobs with incompatible famlies and unequal ready times on parallel batch machines. *Engineering Applications of Artificil Intelligence*, Vol.19, No.3, Apr 2006, 235-245, ISSN 0952-1976

Liu L-L, Ng C-T & Cheng T-C-E (2007). Scheduling jobs with agreeable processing times and due dates on a single batch processing machine. *Theoretical Computer Science*, Vol.374, No.1-3, April 2007, 159-169, ISSN 0304-3975

Dorigo M. (1992) *Optimization, Learning and Natural Algorithms*, Ph.D. Thesis, Politecnico di Milano, Italy.

Dorigo M. & Stützle T (2004). *Ant colony optimization*, Cambridge,MIT Press.

Srinivasa Raghavan N.R. & Venkataramana M. (2006). Scheduling parallel batch processors with incompatible job families using ant colony optimization, *Proceeding of the 2006 IEEE International Conference on Automation Science and Engineering*, Oct 2006,pp.507-512, ISSN 1545-5955, Shanghai.

Balasubramanian H, Mönch L & Fowler J et al. (2004). Genetic algorithm based scheduling of parallel batch machines with incompatible job families to minimize total weighted tardiness. *International Journal of Production Research*, Vol.42, No.8,April 2004, 1621-1638, ISSN 1366-588X

Cai Y.W, Kutanoglu E, Hesenbein J et al.(2007). Single-machine scheduling problem with advanced process control constraints,*AEC/APC Symopsium XIX*, Indian Wells, Calif. USA, pp. 507-512.

Patel N-S (2004).Lot allocation and process control in semiconductor manufacturing - a dynamic game approach. *Proceeding of 43rd IEEE Conference on Decision and Control*,Vol.4, pp.4243-4248,,ISSN 0005-1179 ,Atlantis,Paradise Island, Nassau, Bahamas, Dec 2004

Mönch L, Balasubramanian H, Fowler J-W et al. (2005). Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. *Computers and Operations Research* , Vol.32, No.11, Nov 2005, 2731-2750, ISSN 0305-0548

Li Li, Qiao F & Wu Q.D. (2008).ACO-Based Scheduling of Parallel Batch Processing Machines with Incompatible Job Families to Minimize Total Weighted Tardiness, *Ant Colony Optimization and Swarm Intelligence*, Vol.5217,219-226, ISBN 978-3-540-87526-0,Sept 2008,Springer, Berlin

Future Manufacturing Systems

Edited by Tauseef Aized

ISBN 978-953-307-128-2

Hard cover, 268 pages

Publisher Sciyo

Published online 17, August, 2010

Published in print edition August, 2010

This book is a collection of articles aimed at finding new ways of manufacturing systems developments. The articles included in this volume comprise of current and new directions of manufacturing systems which I believe can lead to the development of more comprehensive and efficient future manufacturing systems. People from diverse background like academia, industry, research and others can take advantage of this volume and can shape future directions of manufacturing systems.

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen