doi:10.3772/j.issn.1006-6748.2016.01.007

An effective estimation of distribution algorithm for parallel litho machine scheduling with reticle constraints①

Zhou Binghai (周炳海)②, Zhong Zhenyi
(School of Mechanical Engineering, Tongji University, Shanghai 201804, P. R. China)

Abstract

In order to improve the scheduling efficiency of photolithography, bottleneck process of wafer fabrications in the semiconductor industry, an effective estimation of distribution algorithm is proposed for scheduling problems of parallel litho machines with reticle constraints, where multiple reticles are available for each reticle type. First, the scheduling problem domain of parallel litho machines is described with reticle constraints and mathematical programming formulations are put forward with the objective of minimizing total weighted completion time. Second, estimation of distribution algorithm is developed with a decoding scheme specially designed to deal with the reticle constraints. Third, an insert-based local search with the first move strategy is introduced to enhance the local exploitation ability of the algorithm. Finally, simulation experiments and analysis demonstrate the effectiveness of the proposed algorithm.

Key words: semiconductor manufacturing, parallel machine scheduling, auxiliary resource constraints, estimation of distribution algorithm

0 Introduction

The manufacturing of integrated circuits (ICs) on silicon wafers is a complex production process. Photolithography is one of the main production process steps in the wafer fabrications. 35% to 45% of work-in-process (WIP) of a wafer fabrication system typically resides in the photolithography area[1]. As the bottleneck process of fabricating complex wafers with the most expensive equipment, the photolithography almost determines the throughput and the cost of semiconductor manufacturing[2].

A mould used to produce chips is called a reticle, which must be on the litho machine for the duration of a wafer lot's processing. Reticles can be thought of as auxiliary resource constraints in the photolithography process. ICs are built by repeatedly constructing layers with desired properties on the silicon wafer's surface. Every layer of each product can require its own unique reticle and a set of reticles for a single product can cost well over MYM150 K. In this paper, photolithography scheduling motivates our investigation into parallel litho machine scheduling in the presence of reticle constraints to minimize total weighted completion time, thus improving the scheduling efficiency of photolithography and leading to financial gains.

The scheduling problem of the photolithography area has aroused much attention from home and abroad researchers and has become an issue of concern in our country recently. Ref. [1] investigated scheduling rules to assign WIP to litho machines to maximize production volume. Ref. [3] described several heuristics for litho machine and reticle scheduling based on appropriate modifications of the apparent tardiness cost (ATC) dispatching rule. Ref. [4] proposed two heuristics and a tabu search-based post processing algorithm. Ref. [5] investigated the problem of the load balancing among litho machines and presented a novel model. Ref. [6] solved the photolithography production control problem based on process capability indices. Ref. [7] established a multistage mathematical programming based scheduling approach with the objected goal of maximization of throughput, minimization of setup cost and a balancing of machine utilization. Ref. [8] put forward a time window rolling- and GA-based method for the dynamic dispatching problem in photolithography area. Ref. [9] introduced discrete event simulation (DES) and mathematical programming techniques into semiconductor manufacturing.

① Supported by the National High Technology Research and Development Programme of China (No. 2009AA043000) and the National Natural Science Foundation of China (No. 61273035, 71471135).

② To whom correspondence should be addressed. E-mail: bhzhou@tongji.edu.cn

Received on Jan. 28, 2015

This DES model allows a detailed problem description. Ref. [10] studied non-identical parallel machines with time constraints in semiconductor manufacturing and developed simple heuristics. Ref. [11] solved a litho machine scheduling formulation by using the branch-and-cut method.

A review of the available literature indicates that there is a dearth of research that explicitly considers auxiliary resource constraints in photolithography. Furthermore, in previous research, the general approach for managing reticles is that the number of reticles available is assumed to be infinite or that the number of reticles available is assumed to be finite while only one unique reticle is available for each type. However, in fact, most of wafer fabs have multiple reticles available for each type of reticle so as to provide creation of more effective production schedules. In this study, each reticle type has multiple reticles available and jobs are investigated with non-zero ready times—both of these characteristics are inherent in semiconductor wafer fabs.

What's more, some of those methods adopted in photolithography such as discrete event simulation are expensive and time-consuming to develop and run while the solutions of those simple scheduling rules are not so promising. Therefore, this paper seeks to find an effective algorithm aiming at keeping a balance between time and solution quality. As a relatively new population-based optimization algorithm, estimation of distribution algorithm (EDA) has been successfully developed to solve a variety of optimization problems in academic and engineering fields[12-14]. But there has been no research about EDA on auxiliary resource constraints so far. Besides, the general method to deal with constraint violations is that a penalty function is added to the objective function or that infeasible solutions are simply removed. However, in this article, a specially designed decoding scheme is proposed to deal with the reticle constraints to transform infeasible solutions into feasible solutions. In addition, an insert-based local search with the first move strategy is introduced to exploit the neighborhood of the best individuals. Simulation results indicate that the proposed algorithm can lead to satisfactory results.

1 Problem statements

The problem assumptions and notations of parallel litho machines with reticle constraints are given as follows.

The assumptions are given as follows: (1) There are a total of n independent jobs (wafer lots) and m identical litho machines. (2) Jobs can be assigned to each of the m available litho machines. (3) Each litho machine can only process one job at most once. In addition, once a job starts to be processed on a litho machine, it can't be interrupted until it is finished. (4) Each job has its own processing time, release time and weight. (5) Each job should be processed on a certain layer. (6) One type of reticle can only process one certain layer. (7) The same reticle is not allowed to be used on different litho machines in overlapping time periods. (8) The same reticle is allowed to be used on different litho machines in non-overlapping time periods.

In order to establish a mathematical model, the following notations are given.

J	The set of jobs or production lots;
L	The set of processing layers;
δ_l	The set of jobs which require layer $l \in L$ processing;
n_l	Number of reticles available for processing layer $l \in L$;
	Job $j \in J$ has the following parameters:
p_j	The processing time of job $j \in J$;
r_j	The release time of job $j \in J$;
w_j	The weight (priority) of job $j \in J$;
C_j	The time at which job $j \in J$ finishes its required processing;
m	The total number of litho machines;
	Binary variables:
x_{ij}	If job i immediately precedes job j on the same litho machine, $x_{ij} = 1$; otherwise, $x_{ij} = 0$.
e_{ij}	If job i immediately precedes job j on the same reticle, $e_{ij} = 1$; otherwise, $e_{ij} = 0$. $\forall i \in \delta_l,\ \forall j \in \delta_l,\ \forall l \in L: i \neq j$
	Dummy variable:
*job*0	A job whose processing time, ready time, and weight are set equal to 0 each. Thus, it can indicate both the starting and finishing of job processing on each litho machine and each reticle.

According to the above assumptions and notations, the following mathematical model is given.

An objective of the scheduling problem is to minimize total weighted completion time (TWCT):

$$\min TWCT = \min \sum_{j:\, j \in J} w_j C_j$$

Eqs(1) and (2) indicate that jobs are assigned to each of the m available litho machines:

$$\sum_{j \in J:\, j \neq 0} x_{oj} \leqslant m \tag{1}$$

$$\sum_{j \in J:\, j \neq 0} x_{j0} \leqslant m \qquad (2)$$

Eqs(3) and (4) indicate litho machine assignment and job sequencing for jobs on the same litho machine:

$$\sum_{j \in J:\, j \neq i} x_{ji} = 1 \quad \forall i \in J: i \neq 0 \qquad (3)$$

$$\sum_{j \in J:\, j \neq i} x_{ij} = 1 \quad \forall i \in J: i \neq 0 \qquad (4)$$

Eqs(5) and (6) indicate that jobs require layer $l \in L$ processing are assigned to each of the n_l available reticles:

$$\sum_{j \in \delta_l:\, j \neq 0} e_{oj} \leqslant n_l \quad \forall l \in L \qquad (5)$$

$$\sum_{j \in \delta_l:\, j \neq 0} e_{j0} \leqslant n_l \quad \forall l \in L \qquad (6)$$

Eqs(7) and (8) indicate reticle assignment and job sequencing for jobs on the same reticle:

$$\sum_{j \in \delta_l:\, j \neq i} e_{ji} = 1 \quad \forall i \in \delta_l: i \neq 0, \quad \forall l \in L \qquad (7)$$

$$\sum_{j \in \delta_l:\, j \neq i} e_{ij} = 1 \quad \forall i \in \delta_l: i \neq 0, \quad \forall l \in L \qquad (8)$$

Eq. (9) shows how job i's completion time C_i is determined:

$$\begin{aligned} C_i - C_j + (M - \min_{k \in J:\, k \neq 0} \{r_k + p_k\} + p_j) x_{ij} \\ \leqslant M - \min_{k \in J:\, k \neq 0} \{r_k + p_k\} \\ \forall i \in J, \forall j \in J: j \neq 0,\ i \neq j \end{aligned} \qquad (9)$$

In Eq. (9), $M \geqslant \max_{j \in J} \{r_j\} + \sum_{j \in J} p_j$ (i. e. big M)

Eq. (10) ensures that if two jobs require the same reticle, one of the jobs should complete its processing before the other job starts its processing

$$\begin{aligned} C_i - C_j + (M - \min_{k \in J:\, k \neq 0} \{r_k + p_k\} + p_j) e_{ij} \\ \leqslant M - \min_{k \in J:\, k \neq 0} \{r_k + p_k\} \\ \forall i \in \delta_l,\ \forall j \in \delta_l,\ \forall l \in L:\ i \neq j \end{aligned} \qquad (10)$$

2 Proposed estimation of distribution algorithm

As a relatively new paradigm in the field of evolutionary computation, estimation of distribution algorithm employs explicit probability distributions in optimization[15].

2.1 Encoding schemes

Every individual of the population denotes a solution, which is represented by a sequence of all the job numbers as $\pi = \{\pi_1, \pi_2, \cdots, \pi_i, \cdots, \pi_n\}$ to determine the schedule order of all the jobs where $\pi_i \in J$ is the ith job in π. For example, a solution $\pi = \{1, 2, 5, 4, 3, 0\}$ implies that job 1 is scheduled first, and next are job 2, job 5, job 4 and job 3 in sequence. Job 0 is the last job to be scheduled.

2.2 Decoding schemes

To decode a sequence is to arrange the machines for all the jobs and determine the processing order in each machine. In this paper, a schedule is feasible only if the auxiliary resource constraints are not violated.

The following variables and definitions are used for the method.

Let t_A^S and t_A^E represent the beginning and ending time of time period A respectively. Based on this, the definition about intersection is given:

If the time period $A[t_A^S, t_A^E]$ and $B[t_B^S, t_B^E]$ satisfy $t_A^S < t_B^E$ and $t_A^E > t_B^S$, then A and B have intersection $I[\max(t_A^S, t_B^S), \min(t_A^E, t_B^E)]$ and denote it as $A \cap B = I$.

s: the first machine completes its required workload.

r'_s: the time at which s becomes available.

$J' = \{\pi_j \mid 1 \leqslant j < i\} \cap \delta_l$: the set of jobs that appear before position i and require layer l processing as π_i does.

$I = [t_{\pi_i}^S, t_{\pi_i}^E]$: the time period that π_i occupies the reticle which can process layer l.

$I_{\pi_j} = [t_{\pi_j}^S, t_{\pi_j}^E]$: the time period that $\pi_j \in J'$ occupies the reticle which can process layer l.

$N(I_{\pi_j})$: the number of reticles occupied during I_{π_j}.

Based on the above variables, definitions and sequence π, the decoding heuristic is described as follows.

Step 1: For job $\pi_i \in J$ in the ith position in sequence π, assign the machine s to it. Compute parameter a as $a = \max\{r'_s, r_{\pi_i}\}$.

Step 2: Initialize $I = [a, a + p_{\pi_i}]$, $I_{\pi_j} = [C_{\pi_j} - p_{\pi_j}, C_{\pi_j}]$ and $N(I_{\pi_j}) = 0$. According to the above definition of the intersection, for each $\pi_j \in J'$, if $I_{\pi_j} \cap I \neq \phi$, update $I_{\pi_j} = I_{\pi_j} \cap I$, $N(I_{\pi_j}) = N(I_{\pi_j}) + 1$.

Step 3: For each $\pi_j \in J'$, define $K = \{\pi_k \mid j < k < i\} \cap \delta_l$ as the set of jobs that appear before position i and after position j and require layer l processing as π_i does. Then, for each $\pi_j \in J'$, do Step 4.

Step 4: For each $\pi_k \in K$, if $I_{\pi_j} \cap I_{\pi_k} \neq \phi$, update $I_{\pi_j} = I_{\pi_j} \cap I_{\pi_k}$, $N(I_{\pi_j}) = N(I_{\pi_j}) + 1$.

Step 5: If $\max_{\pi_j \in J'} \{N(I_{\pi_j})\} > n_l$, then $\pi_j^* = \arg\max_{\pi_j \in J'} \{N(I_{\pi_j})\}$ and go to Step 6. Otherwise, go to Step 7.

Step 6: Define set $Q = \{\pi_j^*\}$. For each $\pi_j \in$

J'/π_j^*, let $I_{\pi_j} = [C_{\pi_j} - p_{\pi_j}, C_{\pi_j}]$, if $I_{\pi_j^*} \cap I_{\pi_j} \neq \phi$, then $Q = Q \cup \{\pi_j\}$. And go to Step 8.

Step 7: $C_{\pi_i} = a + p_{\pi_i}$, $J = J/\pi_i$ and go to step 9.

Step 8: Compute $a = \min_{\pi_j \in \mathbf{Q}}\{C_{\pi_j}\}$ and go to step 2.

Step 9: If $J \neq \phi$, go to step 1. Otherwise, stop.

2.3 Population initialization

In order to guarantee the diversity of the initial population, initial random population of *popsize* individuals are used which are distributed uniformly.

2.4 Probability model

Different from the Genetic algorithm (GA) that produces offspring through crossover and mutation operators, EDA does it by sampling according to a probability model. So, the probability model has a great effect on the performance of EDA. In this paper, the probability model is designed as a probability matrix $\boldsymbol{P}$. The element $p_{ij}(gen)$ of the probability matrix $\boldsymbol{P}$ represents the probability that job j appears before or in position i of the solution sequence at generation *gen*. The value of p_{ij} refers to the importance of a job when deciding the scheduling order.

The initial population with *popsize* individuals determines the superior sub-population that consists of the best SP solutions, where $SP = \eta \times popsize$, and η is a parameter representing the proportion of the superior sub-population in the whole population. Then the probability matrix P is initialized according to

$$p_{ij}(0) = \frac{1}{i \times SP} \cdot \sum_{s=1}^{SP} I_{ij}^s, \quad \forall i, j \tag{11}$$

where I_{ij}^s is the following indicator function of the sth individual in the superior subpopulation.

$$I_{ij}^s = \left\{\begin{matrix} 1, \text{ if job } j \text{ appears before or in position } i \\ 0, \text{ else} \end{matrix}\right\}$$

In each generation of EDA, the new individuals are generated via sampling the solution space according to the probability matrix $\boldsymbol{P}$. For every position i, job j is selected with a probability p_{ij}. If job j has already appeared, it means that job j has been scheduled. Then, the whole jth column of probability matrix $\boldsymbol{P}$ will be set as zero and all the elements of $\boldsymbol{P}$ will be normalized to maintain that each row sums up to 1. In such a way, an individual is constructed until all the jobs appear in the sequence. In EDA, a population with *popsize* individuals is generated.

2.5 Updating mechanism

A new population with *popsize* individuals determines the superior sub - population that consists of the best SP solutions. And then probability matrix $\boldsymbol{P}$ is updated according to

$$p_{ij}(gen + 1) = (1 - \alpha)p_{ij}(gen) + \frac{\alpha}{i \times SP} \cdot \sum_{s=1}^{SP} I_{ij}^s, \ \forall i, j \tag{12}$$

where $\alpha \in (0,1)$ is the learning rate of $\boldsymbol{P}$.

The updating process can be regarded as a kind of increased learning, where the second term on the right hand side of the equation represents learning information from the superior sub-population.

2.6 Insert-based local search with the first move strategy

As EDA pays more attention to global exploration while its exploitation capability is relatively limited, an effective EDA should balance the exploration and the exploitation abilities. In order to enhance the local exploitation ability of the algorithm, an insert-based local search with the first move strategy is introduced.

$\pi_{local}^0(gen)$ represents the best individual found till the *gen*th generation. The details are described as follows:

Step 1: Choose u and v randomly $(u \neq v)$, $\pi^A = Insert(\pi_{local}^0(gen), u, v)$. If $f(\pi^A) < f(\pi_{local}^0(gen))$, then $\pi_{local}^0(gen) = \pi^A$ and stop;

Step 2: $iter = 1$;

Step 2.1: Choose u and v randomly $(u \neq v)$, $\pi^{A\_1} = Insert(\pi^A, u, v)$;

Step 2.2: If $f(\pi^{A\_1}) < f(\pi_{local}^0(gen))$, then $\pi_{local}^0(gen) = \pi^{A\_1}$ and stop;

Step 2.3: If $f(\pi^{A\_1}) < f(\pi^A)$, then $\pi^A = \pi^{A\_1}$;

Step 2.4: $iter = iter + 1$;

Step 2.5: If $iter \leqslant (n \cdot (n\text{-}1))$, then go to step 2.1; otherwise, stop.

2.7 Procedure of proposed EDA

Step 1: Initialize the population according to 2.3 and initialize the probability matrix $\boldsymbol{P}$ according to 2.4.

Step 2: Sample the probability model to generate new population according to 2.4 and select superior sub-population.

Step 3: Update the probability matrix $\boldsymbol{P}$ according to 2.5.

Step 4: Perform the local search according to 2.6.

Step 5: If the termination criterion is not met, go to step 2; otherwise, stop.

3 Simulation and analysis

In this section, in order to evaluate the computational results of EDA and EDA with the local search, the two algorithms are compared with GA and the particle swarm optimization (PSO). The performance measure employed in our numerical study is the average value of TWCT. All the algorithms are run on a 1.86GHz portable computer with 896MB of RAM running Windows XP professional. The codes are written in C ++ language.

3.1 Datasets

Since there are no standard test data in the open literature, the test problems are randomly generated on the basis of the following factors:

1. number of jobs or wafer lots (n),
2. number of machines (m),
3. number of reticles available for processing layer l (n_l).

For the set of test instances, the level settings for each factor are: 3 levels for n, 2 for m, and 2 for n_l. For instance, a problem denoted as n10m2a represents $n = 10$, $m = 2$ and $n_l = [R_l/3] + 1$, while a problem denoted as n10m2b represents $n = 10$, $m = 2$ and $n_l = [R_l/5] + 1$. And for each parameter combination, 10 instances will be generated randomly according to the parameter settings in Table1. This results in a total of 120 test problems.

Table 1 Parameters for the test data

n	10, 20, 50
m	$[n/10] \times 2, [n/10] \times 3$
number of the set of processing layers L	$[n/4] + 1$
n_l	$[R_l/3] + 1$, $[R_l/5] + 1$, R_l: number of jobs requiring layer l processing
Processing time of job j, p_j	U[45,75]
Weight of job j, w_j	U[1,20]
Release time of job j, r_j	50% are generated from $U[1, 360]$, while the remaining 50% have $r_j = 0$

3.2 Parameters setting

EDA contains several key parameters: *popsize* (the population size), η (the parameter associated with the superior sub-population), α (the learning rate). To investigate the influence of these parameters on the performance of EDA, the Taguchi method of design of experiment is implemented by using a moderate-scale problem n20m4a.

For each parameter combination, EDA with local search operator is run 20 times independently and the average response variable (ARV) value is the average value of TWCT obtained by the proposed EDA. According to the number of parameters and the number of factor levels, the orthogonal array $L_{25}(5^3)$ is chosen. That is, the total number of treatment is 25, the number of parameters is 3, and the number of factor levels is 5. Different combinations of these parameter values are listed in Table 2. The orthogonal array and the obtained ARV values are listed in Table 3.

Table 2 Parameter levels

Parameters	Factor levels				
	1	2	3	4	5
popsize	30	40	50	60	70
η	0.1	0.2	0.3	0.4	0.5
α	0.1	0.3	0.5	0.7	0.9

Table 3 Orthogonal table and ARV values

Experiment Number	Factor			*ARV*
	popsize	η	α	
1	1	1	1	30565.35
2	1	2	2	30567.8
3	1	3	3	30556.55
4	1	4	4	30561.15
5	1	5	5	30586
6	2	1	2	30565.9
7	2	2	3	30572.55
8	2	3	4	30544.55
9	2	4	5	30564.1
10	2	5	1	30548.4
11	3	1	3	30547.85
12	3	2	4	30598.75
13	3	3	5	30580.05
14	3	4	1	30570.15
15	3	5	2	30557.6
16	4	1	4	30547.75
17	4	2	5	30570.9
18	4	3	1	30560.9
19	4	4	2	30564.75
20	4	5	3	30572.45
21	5	1	5	30571.9
22	5	2	1	30564.15
23	5	3	2	30560.6
24	5	4	3	30566.9
25	5	5	4	30559.7

According to the orthogonal table and the ARV values, the response values of each parameter listed in Table 4 can be obtained. Then, according to the response values, the trend of each factor level is illustrated in Fig. 1.

Table 4 Response table

level	*popsize*	η	α
1	30567.37	30559.75	30561.79
2	30559.1	30574.83	30563.33
3	30570.88	30560.53	30563.26
4	30563.35	30565.41	30562.38
5	30564.65	30564.83	30574.59

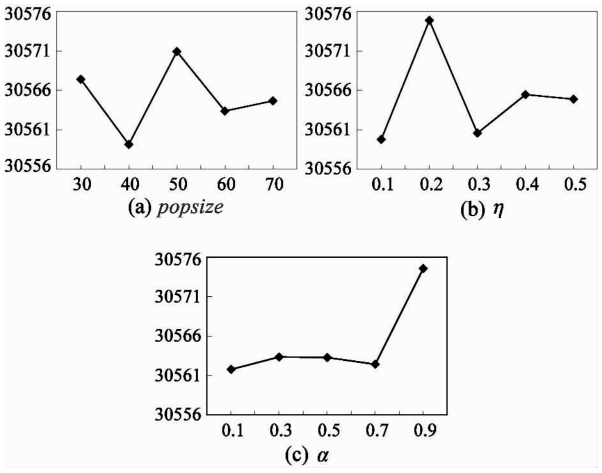

Fig. 1 Factor level trend of the parameters

According to the above analysis, a good choice of parameter combination is suggested as *popsize* = 40, η = 0.1, α = 0.1.

3.3 Computational results

The average TWCT value and CPU time of 10 instances under 12 problems obtained by the four algorithms after 500 generations are listed in Table 5, where EDA represents the general EDA algorithm and EDA + LS stands for EDA with the local search operator. Both GA and PSO are of general procedure with the decoding scheme the same as the one proposed in this paper. The parameters for GA and PSO are set as follows: (1) For GA, *popsize* = 60, crossover probability P_c = 0.6, mutation probability P_m = 0.25, where the parameters are also set by the Taguchi method of design of experiment using the problem n20m4a. (2) For PSO, *popsize* = 70, learning factors c_1 = c_2 = 0.2, inertia weight w is initially set as 0.9 and then linearly decreased to 0.4 according to the number of iterations, where *popsize* is set by experiment and the other parameters are set according to general conditions[16].

As can be seen from Table 5, although GA and PSO appear more efficient in terms of CPU time because it requires a linear time to create new individuals for GA and PSO while this task requires $O(N^2)$ time for EDA, EDA is better than GA and PSO in terms of solution quality. Besides, the solution quality of GA and PSO can be hardly improved even if they use the

Table 5 The computational results

Problem	EDA		EDA + LS		GA		PSO	
	TWCT	CPU(s)	TWCT	CPU(s)	TWCT	CPU(s)	TWCT	CPU(s)
n10m2a	18923.0	0.30	18921.0	0.39	18967.7	0.23	18958.3	0.23
n10m2b	16644.0	0.31	16616.9	0.39	16804.7	0.23	16710.6	0.23
n10m3a	15861.9	0.31	15784.7	0.41	15884.1	0.25	15873.7	0.24
n10m3b	17369.1	0.31	17281.8	0.42	17298.2	0.26	17419.3	0.24
n20m4a	33062.2	1.22	32497.1	2.78	34675.1	0.52	34350.2	0.54
n20m4b	35681.0	1.22	34734.4	2.79	37833.3	0.54	36131.2	0.54
n20m6a	32775.5	1.23	32138.8	2.99	33363.4	0.57	33206.8	0.60
n20m6b	35792.4	1.25	34941.7	3.14	36163.9	0.60	36288.4	0.61
n50m10a	108477.0	13.37	95771.6	55.38	112651.4	2.40	110139.7	2.60
n50m10b	108489.3	13.36	95277.6	56.42	112424.0	2.43	112579.2	2.63
n50m15a	97688.7	13.48	87015.0	61.72	96507.1	2.73	100258.5	2.84
n50m15b	98634.6	13.55	88442.3	66.29	98239.8	2.84	100551.6	2.95
average	51616.6	4.99	47451.9	21.09	52567.7	1.13	52705.6	1.19

same CPU time as EDA does because their convergence curves become flat after 500 generations as presented in Fig. 2 and Fig. 3. Furthermore, the performance of EDA can be greatly improved by adding the local search to it. Obviously, EDA + LS performs the best for all the problems and its CPU time is still acceptable.

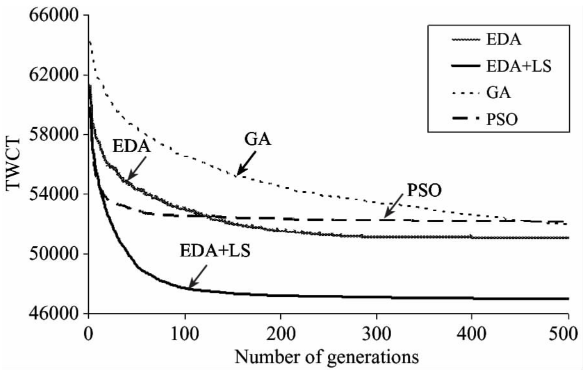

Fig. 2 Convergence curves in solving $n_l = [R_l/3] + 1$

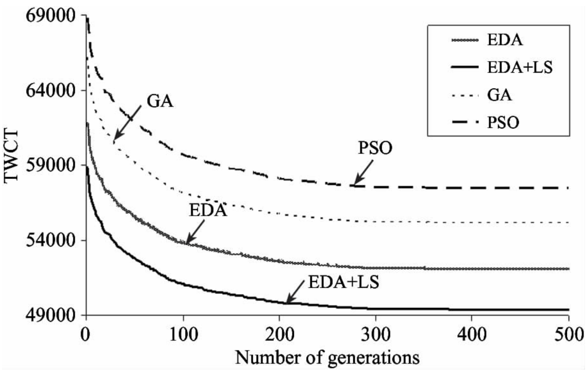

Fig. 3 Convergence curves in solving $n_l = [R_l/5] + 1$

Fig. 2 and Fig. 3 depict the convergence curves obtained by the four algorithms for the 60 problems with $n_l = [R_l/3] + 1$ and $n_l = [R_l/5] + 1$ respectively. The vertical axis shows the average TWCT value of 60 instances for each algorithm. It can be seen from Fig. 2 that although PSO evolves faster than the other algorithms at first, it turns out with a premature convergence. Fig. 3 shows that the four algorithms are nearly of the same convergence speed, but they end with obvious difference in solution quality due to different solutions at the first generation.

It's apparent from both Fig. 2 and Fig. 3 that EDA outperforms GA and PSO in terms of solution quality. Furthermore, EDA + LS can provide even higher quality solutions than EDA. What's more, the superiority of EDA and EDA + LS to GA and PSO in solution quality is more significant in Fig. 3 compared with that in Fig. 2, which indicates that EDA and EDA + LS may perform much better than GA and PSO when the auxiliary resources are more tightly constrained as the number of reticles of each type affects the availability of the auxiliary resources. All in all, the results show that the proposed EDA is effective, especially in solving scarce auxiliary resources problems.

4 Conclusion

In this paper, an estimation of distribution algorithm is developed to solve the scheduling problem of parallel litho machines with reticle constraints by considering multiple reticles available for each reticle type. An effective decoding scheme is designed for the auxiliary resource constraints and an insert-based local search with the first move strategy is introduced which has been proved to be very useful. Comparisons to GA and PSO demonstrate the effectiveness of the proposed EDA in solving the scheduling problem in photolithography, especially when the auxiliary resources are constrained more tightly.

References

[1] Lee Y H, Park J, Kim S. Experimental study on input and bottleneck scheduling for a semiconductor fabrication line. *IIE transactions*, 2002, 34(2): 179-190

[2] Wang S, Zhang P, Qin W. Composite dispatching rule design for photolithography area scheduling in wafer manufacturing system with multiple objectives. *Applied Mechanics and Materials*, 2013, 252: 418-426

[3] de Diaz S L M, Fowler J W, Pfund M E, et al. Evaluating the impacts of reticle requirements in semiconductor wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 2005, 18(4): 622-632

[4] Cakici E, Mason S J. Parallel machine scheduling subject to auxiliary resource constraints. *Production Planning and Control*, 2007, 18(3): 217-225

[5] Shr A M D, Liu A, Chen P P. Load balancing among photolithography machines in the semiconductor manufacturing system. *Journal of Information Science Engineering*, 2008, 24(2): 379-391

[6] Pearn W L, Kang H Y, Lee A H I, et al. Photolithography control in wafer fabrication based on process capability indices with multiple characteristics. *IEEE Transactions on Semiconductor Manufacturing*, 2009, 22(3): 351-356

[7] Klemmt A, Lange J, Weigert G, et al. A multistage mathematical programming based scheduling approach for the photolithography area in semiconductor manufacturing. In: Proceedings of the 2010 Winter Simulation Conference, Baltimore, USA, 2010. 2474-2485

[8] Yang F C, Kuo C N. A time window rolling-and GA-based method for the dynamic dispatching problem in photolithography area. In: Proceedings of the 40th IEEE International Conference on Computers and Industrial Engineering, Awaji Island, Japan, 2010. 1-6

[9] Klemmt A, Weigert G. An optimization approach for par-

allel machine problems with dedication constraints: Combining simulation and capacity planning. In: Proceedings of the 2011 Winter Simulation Conference, Phoenix, USA, 2011. 1986-1998

[10] Obeid A, Dauzère-Pérès S, Yugma C. Scheduling job families on non-identical parallel machines with time constraints. In: Proceedings of the 2011 Winter Simulation Conference, Phoenix, USA, 2011. 1994-2005

[11] Yan B, Chen H Y, Luh P B, et al. Litho machine scheduling with convex hull analyses. *IEEE Transactions on Automation Science and Engineering*, 2013, 10(4): 928-937

[12] Zhang Y, Li X. Estimation of distribution algorithm for permutation flow shops with total flowtime minimization. *Computers & Industrial Engineering*, 2011, 60(4): 706-718

[13] Wang L, Wang S, Xu Y, et al. A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 2012, 62(4): 917-926

[14] Wang S, Wang L, Liu M, et al. An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem. *International Journal of Production Economics*, 2013, 145(1): 387-396

[15] Larranaga P, Lozano J A. Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Netherlands: Springer, 2002

[16] Liu Z X, Liang H. Parameter setting and experimental analysis of the random number in particle swarm optimization algorithm. *Control Theory & Applications*, 2010, 27(11): 1489-1496

Zhou Binghai, was born in 1965. He received Ph. D and M. S. degrees respectively from School of Mechanical Engineering, Shanghai Jiaotong University in 2001 and 1992. He is a professor, a Ph. D supervisor in School of Mechanical Engineering, Tongji University. He is the author of more than 140 scientific papers. His current research interests are scheduling, modeling, simulation and control for manufacturing systems, integrated manufacturing technology.