Int J Adv Manuf Technol (2005) 26: 565–571
DOI 10.1007/s00170-003-2030-2

ORIGINAL ARTICLE

Muh-Cherng Wu · Yai Hsiung · Hsi-Mei Hsu

A tool planning approach considering cycle time constraints and demand uncertainty

Received: 1 September 2003 / Accepted: 10 November 2003 / Published online: 1 December 2004

Abstract The tool planning problem is to determine how many tools should be allocated to each tool group to meet some objectives. Recent studies aim to solve the problem for the cases of uncertain demand. Yet, most of them do not involve cycle time constraints. Cycle time, a key performance index in particular in semiconductor foundry, should not be ignored. The uncertain demand is modeled as a collection of scenarios. Each scenario, with an occurrence probability, represents the aggregate demand volume under a given product mix ratio. A genetic algorithm embedded with a queuing analysis is developed to solve the problem. Experiments indicate that the proposed solution outperforms that obtained by considering only a particular scenario.

Keywords Capacity planning · Cycle time · Demand uncertainty · Genetic algorithm · Tool planning

Notation

Indices

i	Product type ($1 \leqslant i \leqslant n$)
j	Tool group ($1 \leqslant j \leqslant m$)
k	Scenario ($1 \leqslant k \leqslant l$)

Parameters

d_k	The aggregate demand quantity under scenario k
r_k	The probability of occurrence for scenario k
PX_0	The given product mix ratio, $PX_0 = (b_1 : \ldots : b_i : \ldots : b_n)$, where b_i denotes the ratio of product i, $\sum_{i=1}^{n} b_i = 1$
X^c	The existing toolset of the fab, $X^c = \left(x_1^c, \ldots, x_j^c, \ldots, x_m^c\right)^T$, where x_j^c represents the number of tools in tool group j
C	Tool cost vector, $C = \left(c_1, \ldots, c_j, \ldots, c_m\right)$, where c_j is the cost per tool for tool group j
s_k	Maximum allowable stock-out in scenario k
B	The budget constraint for tool procurement
CT_0	The target mean cycle time
p_{ik}	The price of product i per unit in scenario k
v_{ik}	The variable cost of product i per unit in scenario k
e_{ik}	The stock-out cost of product i per unit in scenario k
$\bar{p}_k$	The weighted price per wafer for scenario k under the product mix PX_0, $\bar{p}_k = \sum_{i=1}^{n} b_i \cdot p_{ik}$
$\bar{v}_k$	The weighted variable cost per wafer for scenario k under the product mix PX_0, $\bar{v}_k = \sum_{i=1}^{n} b_i \cdot v_{ik}$
$\bar{e}_k$	The weighted stock-out cost per wafer for scenario k under the product mix PX_0, $\bar{e}_k = \sum_{i=1}^{n} b_i \cdot e_{ik}$
T	Total number of periods used in calculating tool depreciation

Variables

X^f	The final toolset, $X^f = \left(x_1^f, \ldots, x_j^f, \ldots, x_m^f\right)^T$, where x_j^f represents the number of tools in tool group j
X^n	The new toolset to be procured, $X^n = X^f - X^c = \left(x_1^n, \ldots, x_j^n, \ldots, x_m^n\right)^T$, where x_j^n represents the number of new tools in tool group j
$n_k\left(X^f\right)$	The amount of wafer produced by toolset X^f in scenario k

M.-C. Wu (✉) · Y. Hsiung · H.-M. Hsu
Department of Industrial Engineering and Management,
National Chiao Tung University,
Hsin-Chu, Taiwan, R.O.C.
E-mail: mcwu@cc.nctu.edu.tw
Tel.: +886-35-731913
Fax: +886-35-720610

1 Introduction

Semiconductor wafer fabrication is a capital-intensive industry. A typical semiconductor wafer fabrication facility (fab) includes several hundred tools classified into dozens of tool groups; each tool in a tool group is functionally identical. The investment in a fab normally exceeds one billion US dollars, and equipment accounts for about 80% of the investment. The tool planning problem, determining how many tools should be allocated to each tool group to achieve some objectives, is thus very important. In this paper, a tool allocation plan in the tool planning problem is called a toolset.

Compared to other production systems, the performance evaluation of a toolset for wafer fabrication is relatively more complicated. Semiconductor wafers undergo hundreds of manufacturing operations. A collection of 25 wafers, known as a lot, moves among tool groups with the reentrant feature, that is, a lot visits a tool group many times. A highly complex manufacturing flow combined with some stochastic factors such as tool breakdown makes the estimation of the performance a toolset non-trivial.

Static models are often used in industry for tool planning, owing to their fast computation and ease of use [1, 2]. Yet, these methods have two drawbacks, namely, inaccuracy of capacity requirement estimation and lack of queuing delay information. Therefore, discrete event simulation and queuing network models are commonly used in tool planning literature for evaluating the performance of a toolset.

Discrete event simulation provides precise modeling, yet it requires lengthy computation. Queuing network models, analytical techniques equipped with less precise modeling, can rapidly give performance estimates, but the estimates may be less accurate than that obtained by simulation.

Using simulation to estimate performance, the literature on tool planning has published various methods for seeking candidate toolsets. Grewal et al. [3] present a marginal allocation procedure that updates the candidate toolset by adding one tool at a time to the tool group according to certain criteria. Mollaghasemi and Evans [4] develop an interactive method to locate a desired toolset for meeting multiple objectives. Chen and Chen [5] present an experimental design approach, combined with a response surface methodology, to identify a satisfactory toolset. Methods based on simulation are not suitable for performing an extensive search in locating the desired toolset because they have a lengthy run-time requirement.

A great number of studies have developed queuing models for estimating the performance of a toolset; some of them are listed in [6–10]. Based on these queuing models, various methods for seeking candidate toolsets have been proposed. Yoneda et al. [11] present a simulated annealing approach. Bretthauer [12] develops a branch and bound algorithm. Connors et al. [10] propose a marginal allocation algorithm that adds one tool at a time to the tool group, which tends to reduce the mean cycle time most effectively. Hopp et al. [13] proposed a greedy heuristic method for searching a near-optimal toolset. Bard et al. [14] compare the solution quality of several heuristic methods for tool planning. Based on a queuing model, Chou [15] proposes a qualitative reasoning method and establishes a solution architecture for tool planning. Chou and You [1] develop equi-throughput curves, and use a marginal allocation procedure to identify a toolset that meets the planner's criteria. Chou and Wu [16] propose a utility function, which integrates two criteria (cycle time and throughput) as a single criterion in evaluating a toolset.

The aforementioned literature on tool planning assumes that the future demand of product is certain. Yet, some other studies assume that the future demand is uncertain. Swaminathan [17] addresses a tool planning problem under demand uncertainty in a single period. The uncertain demand is modeled as a collection of scenarios. Each scenario, with an occurrence probability, represents the demand for each type of product. Swaminathan [17] formulates the tool planning problem by a mixed integer program. The fab throughput is only constrained by available machine hours. Without applying queuing or simulation techniques, his approach cannot yield some important performance measures of a fab such as the mean cycle time. In his further study, the uncertain demand is extended to multiple periods [18]. Barahona et al. [19] and Hood et al. [20] enhance the tool planning model by modifying the objective function and including some more constraints such as the stock-out volume.

In a semiconductor fab, the cycle time is the period from bare silicon wafer start to wafer out. Mean cycle time is a key performance index of semiconductor fabs. The life cycles of some semiconductor products are quite short; prices of such products may even drop to half in six months. A shorter manufacturing cycle time will reduce the time-to-market and increase profit. In tool planning studies, including the target mean cycle time in fab models is thus very important. This point has been addressed in the context of certain demand [1, 3, 10], but has been lacking in the context of uncertain demand.

This study presents a model for a tool planning problem, which includes the cycle time constraint in the context of uncertain demand. The uncertain demand is modeled by a collection of scenarios. Each scenario, with an occurrence probability, represents the aggregate demand volume under a given product mix ratio. A genetic algorithm embedded with a queuing analysis is developed for the solution. The remainder of this paper is organized as follows: Sect. 2 formulates the tool planning problem, Sect. 3 explains the genetic algorithm (GA) for locating the near optimum toolset, Sect. 4 describes a method for defining the search space of the proposed GA, Sect. 5 presents numerical experiments, and concluding remarks are given in Sect. 6.

2 Problem formulation

The addressed tool planning problem is to determine how to purchase additional tools for a future period of an existing fab in an environment with the following two characteristics. First, the forecasted demand in the concerned period is uncertain in aggregate volume under a given product mix ratio. Second, the mean cycle time of products should be under a predefined target. The objective of the planning is to maximize the amount of profit, which is caused by the procurement of new tools.

The environment of interest is frequently faced by a typical semiconductor foundry, which is a completely make-to-order fab and has a large number of customers. Cycle time is a very important performance index for semiconductor foundries, due to the short life cycle of electronic products. Lengthy cycle time may seriously cause the loss of customer orders. Therefore, the tool planning decisions in semiconductor foundries should include the constraints of cycle time. A semiconductor foundry has to periodically (e.g., every year) survey the future demand of their customers to review the need for tool procurement. Customers' feedbacks can be summarized to yield the following information: a certain product mix ratio and several probabilistic aggregate demand volumes. The product mix ratio denotes the demand ratios among each type of product. For example, the product mix ratio for three types of products may be (A : B : C) = (1 : 2 : 3). A probabilistic demand volume means a volume associated with an occurrence probability. For example, for the given product mix ratio, the aggregate demand volume may involve the following three scenarios: 40 K wafers/month with 0.5 probability, 35 K wafers/month with 0.3 probability, and 30 K wafers/month with 0.2 probability.

The following notations are used to formulate the problem.

The tool planning problem for the multiple probabilistic demand scenarios (briefly termed the MDTP problem) can be formulated as follows:

MDTP: Maximize $F(X^f) - F(X^c)$ s.t.:

$$n_k(X^f) \leqslant Q(X^f) \quad \forall k \tag{1}$$

$$n_k(X^f) \leqslant d_k \quad \forall k \tag{2}$$

$$d_k - n_k(X^f) \leqslant s_k \quad \forall k \tag{3}$$

$$CX^n \leqslant B \tag{4}$$

$$x_j^f, x_j^n, n_k\left(X^f\right) \in Z^+ . \tag{5}$$

In the above formulation, $F(\bullet)$ and $Q(\bullet)$ are both functions of a toolset. As shown in Eq. 6, $F(X)$ denotes the expected profit of a particular toolset X across all demand scenarios, where the first term represents the expected contribution margin, the second term is the expected stock-out cost, and the third term is the depreciation cost of tools over the concerned period:

$$F(X) = \sum_{k=1}^{l} \left[r_k \cdot n_k(X) \cdot (\bar{p}_k - \bar{v}_k)\right] - \sum_{k=1}^{l} \left[r_k \cdot (d_k - n_k(X)) \cdot \bar{e}_k\right] - \frac{C \cdot X}{T} \tag{6}$$

$$Q(X) = f_{qb}\,(X;\, PX_0, CT_0) \ . \tag{7}$$

As shown in Eq. 7, $Q(X)$ denotes the maximum throughput of a particular toolset X under two constraints, i.e., the product mix ratio is PX_0 and the mean cycle time should be less than or equal to a predefined target CT_0. The function f_{qb} is a binary search method embedded within a queuing model. The queuing model is for computing the mean cycle time for a given throughput. The throughput should be decreased or increased if its associated mean cycle time is larger or smaller than CT_0. The throughput updating procedure terminates when the mean cycle time is very close to CT_0. Details of the function f_{qb} can be found in the Appendix.

The objective function is to maximize the profit increased by the procurement of new tools, where $F(X^f)$ represents the expected profit of the final toolset X^f, and $F(X^c)$ is the expected profit of the current toolset X^c across all scenarios. Constraints Eq. 1 denote that the production volume in each demand scenario should be less than the maximum throughput. Constraints Eq. 2 indicate that the production volume should be less than the demand in each scenario. Constraints Eq. 3 specify the upper bonds for stock out. Equation 4 is the budget constraint and Eq. 5 specifies the integer requirements for variables.

By consolidating the aforementioned constraints, the model can be concisely presented as follows:

MDTP: Maximize $F(X^f) - F(X^c)$

$$n_k(X^f) = Min\left(Q(X^f), d_k\right) \quad \forall k \tag{8}$$

$$Q(X^f) \geqslant (d_k - s_k) \quad \forall k \tag{9}$$

$$CX^n \leqslant B \tag{10}$$

$$x_j^f, x_j^n, n_k\left(X^f\right) \in Z^+ . \tag{11}$$

Equation 8 can be easily derived from constraints Eqs. 1 and 2. Referring to Eq. 6, $F(X^f)$ is a monotonically increasing function with respect to $n_k(X^f)$. That is, the higher the production volume, the larger the profit. Since the upper bound of $n_k(X^f)$ is $Min\left(Q(X^f), d_k\right)$, we thus can conclude that $n_k(X^f) = Min\left(Q(X^f), d_k\right)$, as indicated in Eq. 8, for maximizing the objective function.

Equation 9 is derived from Eq. 3 by employing $n_k(X^f) = Min\left(Q(X^f), d_k\right)$. Equation 3 can be easily transformed as $n_k(X^f) \geqslant (d_k - s_k)$, or equivalently, $Min\left(Q(X^f), d_k\right) \geqslant (d_k - s_k)$, which implies that $Q(X^f) \geqslant (d_k - s_k)$ and $d_k \geqslant (d_k - s_k)$. $d_k \geqslant (d_k - s_k)$ is always true and can be removed from the constraint list. The equivalence of Eqs. 3 and 9 is thus proved.

The MDTP problem formulated above is a complex nonlinear integer programming model, where $Q(X^f)$ and $n_k(X^f)$ are related to X^f by a set of nonlinear functions [10]. Moreover, the search space of the problem is quite huge. A typical wafer fab includes about 100 tool groups. Given a particular toolset X, the decision of whether or not to add one tool to each tool group j ($j = 1, \ldots, 100$) will yield a search space that comprises 2^{100} (or 1.27×10^{30}) toolsets. Due to the complex nonlinear functions and the enormous solution space, it seems impractical to solve the problem analytically. A genetic algorithm is therefore proposed to solve the problem.

3 Genetic algorithm

A genetic algorithm is proposed to locate efficiently a near-optimal solution in the enormous space. The genetic algorithm (GA) technique was first proposed in the early 1970s [21] and has been widely applied to various areas [22]. Various applications have shown that GAs are powerful techniques for effectively and efficiently solving large scale space-search problems.

A GA is an iterative procedure that maintains a constant-sized population $P(t)$ of candidate solutions (also known as chromosomes). During each iteration step t, called a generation, new chromosomes are created by invoking some genetic operators.

Each existing and newly generated chromosome is evaluated to determine its fitness value, which denotes how good the solution is. Based on these evaluations, a set of chromosomes are screened out by a selection procedure to form the new population $P(t+1)$. The procedure is iteratively performed until the termination conditions are met.

Appropriate methods for representing a chromosome, genetic operators, a fitness function, a selection strategy for forming new population, and termination conditions must all be defined in designing a genetic algorithm [22]. Each of these aspects of the proposed GA is presented below.

3.1 Representation of a chromosome

A chromosome, the final toolset, is expressed by a string of m positive integers, $X^f = \left[x_1^f, \ldots, x_j^f, \ldots, x_m^f\right]$. Each position of a chromosome is called a gene. Let N_p be the total number of chromosomes in the population $P(t)$. The initial population $P(0)$ is created by randomly generating N_p chromosomes. In generating a chromosome, the value of each gene x_j^f is randomly chosen from the interval $\left[LB\left(x_j^f\right), UB\left(x_j^f\right)\right]$, where $LB\left(x_j^f\right)$ and $UB\left(x_j^f\right)$ denote the lower and upper bounds of x_j^f, respectively. The method for determining $LB\left(x_j^f\right)$ and $UB\left(x_j^f\right)$ is presented in Sect. 4.

3.2 Fitness function

The fitness function in a GA is defined to evaluate the quality of a chromosome. In the proposed GA, the fitness function, as shown in Eq. 12, is based on the objective function of the MDTP problem, with embellishments for including the required constraints as penalties [23].

$$
\begin{aligned}
Fitness &= \left[F\left(X^f\right) - F\left(X^c\right)\right] - Y_k \sum_{k=1}^{l} \left(1 - \frac{Q\left(X^f\right)}{(d_k - s_k)}\right) \\
&\quad - Z \cdot \left(\frac{C \cdot X^n}{B} - 1\right) \\
Y_k &= \begin{cases} 0, & \text{if } Q\left(X^f\right) \geqslant (d_k - s_k); \\ M_k, & \text{otherwise} \end{cases} \\
Z &= \begin{cases} 0, & \text{if } C \cdot X^n \leqslant B; \\ H, & \text{otherwise} \end{cases}
\end{aligned}
\tag{12}
$$

where M_k and H denote large positive numbers.

The first term denotes the objective function, where $F\left(X^f\right)$ is computed by taking $n_k\left(X^f\right) = Min\left(Q\left(X^f\right), d_k\right)$. The second term is a penalty reflecting constraint Eq. 9, while the third term is another penalty reflecting constraint Eq. 10. The penalty terms lead to a small fitness value if the solution violates the two constraints. A toolset with a small fitness value is less likely to survive during the evolution of the population and tends to finally be excluded from the population.

The penalty design can seemingly be removed from the fitness function by excluding all the violation chromosomes (due to over stock-out or over budget) from each population. However, so doing might also exclude "good genes" from the population. For a violation chromosome, particular segments of its genes may exactly match a part of the optimum solution. Possibly carrying good genes, violation chromosomes shall not be forcibly excluded from each population.

3.3 Crossover and mutation operators

The proposed GA defines two genetic operators, known as crossover and mutation, to create new chromosomes.

The crossover operator is designed to create $N_p \times P_{cr}$ new chromosomes in each generation, where P_{cr} is a predefined crossover probability. This operator is applied by first randomly choosing $N_p \times P_{cr}$ chromosomes from $P(t)$ and randomly grouping them into $(N_p \times P_{cr})/2$ pairs. For each pair of chromosomes, a position in a chromosome (called the crossover point) is randomly chosen, and the segments to the right of the crossover point are exchanged. Let the pair of chromosomes for crossover be $X_1 = [5, 7, 9, 1 : \underline{2, 3, 6}]$ and $X_2 = [1, 4, 5, 6 : \underline{9, 8, 1}]$. Suppose that the crossover point (:) has been chosen as indicated. The resulting two new chromosomes would be $Y_1 = [5, 7, 9, 1 : \underline{9, 8, 1}]$ and $Y_2 = [1, 4, 5, 6 : \underline{2, 3, 6}]$.

The mutation operator is designed to create $N_p \times P_{mu}$ new chromosomes from $P(t)$, where P_{mu} is a predefined probability of mutation. This operator is applied by first randomly selecting $N_p \times P_{mu}$ chromosomes from $P(t)$. For each chosen chromosome, a gene x_j^f is randomly selected. Then, x_j^f is replaced by an integer randomly chosen from the interval $\left[LB\left(x_j^f\right), UB\left(x_j^f\right)\right]$.

3.4 Selection strategy

The chromosomes in population $P(t)$, together with the new chromosomes created by crossover and mutation, are put in a pool. Let S represent the pool in which the number of chromosomes is $h = N_p \cdot (1 + P_{cr} + P_{mu})$; N_p chromosomes are to be selected from S to the population $P(t+1)$. The selection strategy used in this paper is termed the rank-space method [24], which is presented below.

```
Step 1: Sort in descending order the chromosomes in $S$ according to their fitness values. Let $X_1, X_2, \ldots, X_h$ be the
        sorted result. Such a ranking of $X_i$, termed quality-ranking, is represented by $R_q(X_i)$.
Step 2: Move the best quality-ranking chromosome from $S$ to $P(t+1)$.
        $S = S - \{X_1\}$;
        $P(t+1) \leftarrow X_1$;
        $Y_1 = X_1$;
        /* rename the chromosome selected for $P(t+1)$ */
        $N = 1$; /* count the chromosome number in $P(t+1)$ */
Step 3: For each chromosome $X_i$ in $S$, compute the diversity index $D(X_i)$:
        $D(X_i) = \sum_{k=1}^{N} \frac{1}{|X_i - Y_k|}$;
        /* $Y_k$ is a chromosome in $P(t+1)$ */
Step 4: Sort in ascending order the chromosomes in $S$ according to $D(X_i)$. Such a ranking of $X_i$, termed diversity-ranking, is represented by $R_d(X_i)$.
Step 5: Compute the sum of quality-ranking and diversity-ranking of $X_i$ in $S$:
        $T(X_i) = R_q(X_i) + R_d(X_i)$ .
Step 6: Sort in ascending order the chromosomes in $S$ according to $T(X_i)$. Such a ranking of $X_i$, termed combined-ranking, is represented by $R_c(X_i)$.
Step 7: For each chromosome in $S$, compute the probability of putting $X_i$ in $P(t+1)$:
        $r = R_c(X_i)$;
        $Prob(X_i) = p \cdot (1-p)^{r-1}$; /* $p$ is a predefined probability, typically set to 0.667 */
Step 8: Generate a random number and determine which chromosome in $S$ is selected. Let $X_m$ be the selected chromosome. Move $X_m$ from $S$ to $P(t+1)$.
        $S = S - \{X_m\}$;
        $P(t+1) \leftarrow X_m$;
        $Y_m = X_m$;
        /* rename the chromosome selected for $P(t+1)$ */
        $N = N + 1$;
        /* count the chromosome number in $P(t+1)$ */
Step 9: Termination check,
        if $N < N_p$ then go to Step 3,
        else stop.
```

3.5 Terminating conditions

Population $P(t)$ is iteratively updated until the following termination conditions are met. The GA stops when a particular chromosome keeps the best solution for over N_G generations or when $t = N_f$.

4 Determination of searching space

Scale of searching space is critical to the efficiency of a GA. In our model, the value of each gene x_j^f is chosen from the interval $\left[LB\left(x_j^f\right), UB\left(x_j^f\right)\right]$. The determination of $LB\left(x_j^f\right)$ and $UB\left(x_j^f\right)$ is described below.

Equation 9 suggests that n_L, the lower bound of throughput, should be $n^L = Max_{k=1}^{l}(d_k - s_k)$. The lower bound throughput for product i thus equals $n^L \cdot b_i$. The least required number of tool group j, denoted by x_j^L, can be computed by

$$x_j^L = \left\lceil \frac{\sum_{i=1}^{n} t_{ij} \cdot n^L \cdot b_i}{a_j} \right\rceil ,$$

where t_{ij} is the standard process time of product i on tool j, a_j is the available time for tool j during the time horizon and $\lceil \cdot \rceil$ denotes taking the integer ceiling. Further considering the existing tools x_j^c, the lower bound for x_j^f is given by:

$$LB\left(x_j^f\right) = Max\left(x_j^L, x_j^c\right) \quad \text{for } j = 1, \ldots, m \ .$$

Constraint Eq. 8 gives an upper bound of the throughput for each scenario. Since $n_k\left(X^f\right) \leqslant d_k$ for every scenario k, the throughput of the fab is thus bounded by $n^U = Max_{k=1}^{l} d_k$. The upper bound of tool group j, denoted by x_j^U, can be accordingly computed as:

$$x_j^U = \left\lceil \sum_{i=1}^{n} \frac{t_{ij} \cdot n^U \cdot b_i}{a_j \cdot \beta} \right\rceil ,$$

where $0 < \beta < 1$ is a heuristically defined parameter for considering the effects of cycle time constraints. The upper bound for x_j^f can therefore be defined as

$$UB\left(x_j^f\right) = Max\left(x_j^U, x_j^c\right) \quad \text{for } j = 1, \ldots, m \ .$$

5 Numerical examples

The performance of the proposed method is evaluated by solving a hypothetical MDTP problem. Four cases are tested to examine our model. The four cases are characterized by the different probability combination for demand scenarios. The following assumptions are made about the context of the testing examples.

The semiconductor fab produces four product types: A, B, C, and D. The planning horizon of tool procurement for the fab is half a year. The forecasted product mix ratio for the planning horizon is $PX_0 = (0.3, 0.2, 0.3, 0.2)$. Each product requires 400 to 500 operations on 101 tool groups. The target mean cycle time is $CT_0 = 4PT_0$, where PT_0 is the mean processing time of all product types. The budget for tool procurement is 400 million dollars.

The existing toolset can produce 180 K wafers for PX_0 in the planning horizon. Table 1 shows the aggregate demand volume of three possible scenarios, forecasted by the marketing department. Scenario 1 represents a moderate demand; scenario 2 denotes a high demand; and scenario 3 implies a fair demand. Table 2 shows the probability of each scenario in four tested cases, where r_i denotes the probability of scenario i.

Table 3 shows the profit caused by MDTP versus the profit incurred by planning only for a particular scenario. Let X_1^n represent the new toolset planned for Case 1, where $(r_1, r_2, r_3) = (0.5, 0.3, 0.2)$; X_0^n represents the new toolset planned for a case with $(r_1, r_2, r_3) = (1, 0, 0)$. In row 2 of Table 3, which indicates the occurrence of Case 1, the expected profit by purchasing

Table 1. The aggregate demand volumes for three scenarios

	Scenario 1	Scenario 2	Scenario 3
Demand d_k	210 K	238 K	189 K

Table 2. The occurrence probability of each scenario in four tested cases

	(r_1, r_2, r_3)
Case 1	(0.5, 0.3, 0.2)
Case 2	(0.3, 0.5, 0.2)
Case 3	(0.3, 0.2, 0.5)
Case 4	(0.34, 0.33, 0.33)

Table 3. Comparison of profits by MDTP and single scenario planning (unit: $ 1000)

Case	(r_1, r_2, r_3)	MDTP	(1, 0, 0)	(0, 1, 0)	(0, 0, 1)
Case 1	(0.5, 0.3, 0.2)	24 065	22 678	22 363	7913
Case 2	(0.3, 0.5, 0.2)	35 945	25 453	35 102	8846
Case 3	(0.3, 0.2, 0.5)	12 163	10 771	6179	5281
Case 4	(0.34, 0.33, 0.33)	21 217	18 867	20 021	7446

toolset X_1^n (column 3) is higher than that by purchasing toolset X_0^n (column 4). Likewise, for each tested case (row), the solution obtained by MDTP (column 3) always outperforms the three other solutions obtained by considering only one scenario (column 4–6). The percentage difference in profit ranges from 2.4% to 306% in the tested cases, where $306\% = (35\,945 - 8846)/(8846)$. This implies that the toolset obtained by MDTP performs well across all scenarios, and thus reduces the risk of demand uncertainty.

The proposed GA method was coded in C++ language and performed on a Pentium IV computer. In the GA, the crossover rate was set to $P_{cr} = 0.6$, the mutation rate to $P_{mu} = 0.01$, and the population size to $N_p = 50$. The search terminates when a particular toolset maintains the best solution for over $N_G = 500$ generations. In this example, the search space comprises around 2.22×10^{37} toolsets and the number of toolsets visited by the proposed GA is about 1.2×10^5. The computation time for the GA to obtain the final solution is about 8 h, which seems acceptable for a long-term decision problem such as tool planning.

6 Concluding remarks

This study presents a methodology for solving the tool planning problem in an environment, which involves multiple and probabilistic demand volumes for a given product mix ratio with a target mean cycle time. The previous literature on the tool planning issue for multiple uncertain demand scenarios has not addressed cycle time – a key performance index in semiconductor manufacturing. The formulated MDTP problem does reflect a business environment in the real world, in which semiconductor fabs face market demand variation and short time-to-market.

The proposed methodology is a GA-based approach. Candidate toolsets are iteratively generated and evaluated until the optimum solution is obtained. A binary search procedure combined with a queuing model is proposed to compute the throughput of a toolset, which is subsequently used in evaluating the performance of the toolset. For a typical semiconductor fab, identifying the optimum solution using the GA takes about 8 h. This computational time is acceptable because tool planning in semiconductor manufacturing is a long-term and large-scale problem.

Experimental results show that the proposed solution to the MDTP problem is better than the solution obtained for a particular demand scenario. The percentage difference in profit ranges from 2.4% to 306% in the tested cases. Finally, future research may address the tool planning problem under uncertain demand volumes and varying product mix ratios.

Acknowledgement This research was financially sponsored by the National Science Council, Taiwan, R.O.C., under research grant NSC92-2213-E009-106.

References

1. Chou Y-C, You R-C (2001) A resource portfolio planning methodology for semiconductor wafer manufacturing. Int J Adv Manuf Technol 18:12–19
2. Witte JD (1996) Using static modeling techniques in semiconductor manufacturing. IEEE/SEMI Advanced Semiconductor Manufacturing Conference, Cambridge, MA, pp 31–35
3. Grewal NS, Bruska AC, Wulf TM, Robinson JK (1998) Integrating targeted cycle-time reduction into the capital planning process. Proceedings of the 1998 Winter Simulation Conference, Washington, D.C., pp 1005–1010
4. Mollaghsemi M, Evans GW (1994) Multicriteria design of manufacturing systems through simulation optimization. IEEE Trans Syst Man Cybern 24(8):1407–1411
5. Chen LH, Chen YH (1996) A design procedure for a robust job shop manufacturing system under a constraint using computer simulation experiments. Comput Ind Eng 30(1):1–12
6. Whitt W (1983) The queuing network analyzer. Bell Syst Tech J 62(8):2779–2815
7. Suri R, Hildebrant RR (1984) Modeling flexible manufacturing systems using mean-value analysis. J Manuf Syst 3(1):27–37
8. Buzacott JA, Yao DD (1986) Flexible manufacturing systems: a review of analytical models. Manage Sci 3:890–905
9. Kouvelis P, Tirupati D (1991) Approximate performance modeling and decision making for manufacturing systems: a queueing network optimization framework. J Intell Manuf 2:107–134
10. Connors DP, Feigin GE, Yao D (1996) A queueing network model for semiconductor manufacturing. IEEE Trans Semicond Manuf 9(3):412–427
11. Yoneda K, Wada I, Haruki K (1992) Job shop configuration with queueing networks and simulated annealing. Proceedings of the IEEE International Conference on Systems Engineering, Kobe, Japan, 1719 Sept 1992, pp 407–410
12. Bretthauer KM (1996) Capacity planning in manufacturing and computer networks. Eur J Oper Res 91:386–394
13. Hopp WJ (2002) Using an optimized queueing network model to support wafer design. IIE Trans 34:119–130
14. Bard JF (1999) An optimization approach to capacity expansion in semiconductor manufacturing facilities. Int J Prod Res 37:3359–3382
15. Chou Y-C (1999) Configuration design of complex integrated manufacturing systems. Int J Adv Manuf Technol 15:907–913
16. Chou Y-C, Wu C-S (2002) Economic analysis and optimization of tool portfolio in semiconductor manufacturing. IEEE Trans Semicond Manuf 15(4):447–453
17. Swaminathan JM (2000) Tool capacity planning for semiconductor fabrication facilities under demand uncertainty. Eur Oper Res 120:545–558
18. Swaminathan JM (2002) Tool procurement planning for wafer fabrication facilities: a scenario-based approach. IIE Trans 34:145–155

19. Barahona F, Bermon S, Gunluk O, Hood S (2001) Robust Capacity Planning in Semiconductor Manufacturing. IBM report RC22196. Available online in PostScript format form http://www.optimization-online.org/DB_HTML/2001/10/379.html
20. Hood SJ, Bermon S, Barahona F (2003) Capacity Planning Under Demand Uncertainty for Semiconductor Manufacturing. IEEE Transactions on Semiconductor Manufacturing 16(2):273–280
21. Bethke AD (1981) Genetic algorithm as function optimizers. Dissertation, Dept of Computer and Communication Sciences, University of Michigan
22. Gen M, Cheng R (2000) Genetic algorithms and engineering optimization. Wiley, New York, pp 1–2
23. Levitin G (2000) Multistate series-parallel system expansion-scheduling subject to availability constraints. IEEE Trans Reliab 49(1):71–79
24. Winston PH (1992) Artificial intelligence. Addison-Wesley, Boston pp 520–527

Appendix: Throughput estimation

In this appendix we present an iterative scheme for estimating the capacity (throughput) of a toolset under a target mean cycle time. The use of queuing models for estimating the performance of a toolset a in fab has been widely published. Typical examples can be found in [6, 10].

The input and output relationships of such a queuing model can be formulated as Eq. 13. This equation shows that given a product mix (PX_0) and a wafer release rate (R_0), a queuing model f_q can quickly compute the mean cycle time CT_i for a toolset X_i. The subscript q denotes that the function f_q is a queuing model:

$$CT_i = f_q(X_i;\, PX_0, R_0)\,. \tag{13}$$

Notice that in the function f_q, PX_0, R_0 and X_i denote three independent variables. The symbol ";" is used to clarify which variables are to be varied and which are to be fixed in a particular application. The variables to the right-hand side of ";" are fixed, and that to the left-hand side is varied. Equation 13 therefore denotes that for a particular scenario (PX_0, R_0), the mean cycle time CT_i for a toolset X_i can be computed by the queuing model f_q.

In steady state, the wafer release R_0 rate may be considered as being equal to the throughput Q_0 (the output rate). Equation 13 can therefore be reformulated as Eq. 14:

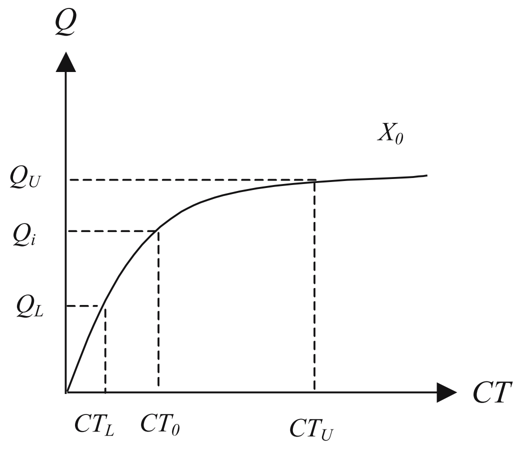

Fig. 1. The relationship between throughput and cycle

$$CT_i = f_q(X_i;\, PX_0, Q_0)\,. \tag{14}$$

The queuing model f_q can also be used to compute the mean cycle times for various throughputs, in a particular scenario (PX_0, X_0), as formulated in Eq. 15. By evaluating the queuing model [10], the relationship between mean cycle time and throughput is as shown in Fig. 1;

$$CT_i = f_q(Q_i;\, PX_0, X_0)\,. \tag{15}$$

With reference to Fig. 1, increasing Q_i in general will increase CT_i. Given this monotonically increasing feature, a binary search procedure based on Eq. 15 can be used to find a Q_i, at which $CT_i \cong CT_0$ in the scenario (PX_0, X_0).

A function f_{qb} representing the binary search procedure is formulated in Eq. 16, where the subscripts qb denote the combination of a queuing model and the binary search procedure. This equation denotes that in a scenario (PX_0, CT_0), the function f_{qb} can determine the maximum throughput $Q(X)$ for a toolset X:

$$Q(X) = f_{qb}(X;\, PX_0, CT_0)\,. \tag{16}$$