Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 33 (2007) 476–483
Expert Systems with Applications
www.elsevier.com/locate/eswa

A short-term capacity trading method for semiconductor fabs with partnership

Muh-Cherng Wu [1], Wen-Jen Chang [*]

Department of Industrial Engineering and Management, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan

Abstract

This paper presents a capacity trading method for two semiconductor fabs that have established a capacity-sharing partnership. A fab that is predicted to have insufficient capacity at some workstations in a short-term period (e.g. one week) could purchase tool capacity from its partner fab. The population of such a capacity-trading portfolio may be quite huge. The proposed method involves three modules. We first use discrete-event simulation to identify the trading population. Secondly, some randomly sampled trading portfolios with their performance measured by simulation are used to develop a neural network, which can efficiently evaluate the performance of a trading portfolio. Thirdly, a genetic algorithm (GA) embedded with the developed neural network is used to find a near-optimal trading portfolio from the huge trading population. Experiment results indicate that the proposed trading method outperforms two other benchmarked methods in terms of number of completed operations, number of wafer outs, and mean cycle time.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Semiconductor fab; Capacity planning; Capacity trading; Neural network; Genetic algorithm

1. Introduction

Semiconductor manufacturing is a capital intensive industry. An up-to-date semiconductor fab may cost a few billion dollars to build, and an individual tool may charge over several tens of million dollars. Due to tremendously high equipment cost, how to effectively manage tool capacity is very important to semiconductor fabs. Such decisions associated with tool capacity are generally called *capacity planning* problems (Christie & Wu, 2002). The objective of capacity planning is to well prepare and allocate tool capacity in order to maximize a semiconductor fabs' profit. In terms of planning horizon, capacity planning problems could be classified into three levels: strategic, tactical, and operational levels.

At strategic level, capacity planning means the investment decisions on procuring new tools. Based on demand forecast, the decision (also called tool planning) is to optimally determine the type and number of tools needed for new or existing fabs. The tool planning decision is distinct in requiring large amount of expenditure and long-lead time (3–9 months) in tool procurement. The tool planning problem, a *long-term* decision, usually covers 1–2 years in the planning horizon, and the annual tool expenditure of a semiconductor company may be over billions of dollars. Much literature on tool planning has been published (Wu, Erkoc, & Karabuk, 2005a). Some addressed scenarios with demand uncertainty and developed mix-integer programming models (Barahona, Bermon, Gunluk, & Hood, 2001; Çatay, Erengüç, & Vakharia, 2003; Hood, Bermon, & Barahona, 2003; Swaminathan, 2000, 2002); some others were concerned with scenarios with constraints imposed by target cycle time and developed solution methods by the application of queuing network (Bard, Srinivasan, & Tirupati, 1999; Connors, Feigin, & Yao, 1996; Iwata, Taji, & Tamura, 2003; Wu, Hsiung, & Hsu, 2005b) or by

[*] Corresponding author. Tel.: +886 3 5927700x2954; fax: +886 3 5926848.
E-mail addresses: mcwu@cc.nctu.edu.tw (M.-C. Wu), wjchang6@ms39.hinet.net (W.-J. Chang).
[1] Tel.: +886 3 5731913; fax: +886 3 5720610.

0957-4174/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2006.05.012

simulation models (Chen & Chen, 1996; Grewal, Bruska, Wulf, & Robinson, 1998; Mollaghsemi & Evans, 1994) or by simulation and queuing techniques (Chou, Wu, Kao, & Hsieh, 2001; Hopp, Spearman, Chayet, Donohue, & Gel, 2002).

At tactical level, capacity planning refers to the decisions of product-mix and production planning. The product-mix decision is to find an optimal product mix to maximize the profit of a fab, equipped with a given tool portfolio. The production planning decision is to determine the release schedule of each fab given a corporate demand output schedule (Chen, Chen, Lin, & Rau, 2005). These two decisions can be regarded as *medium-term* decisions because they usually cover several months in the planning horizon. Various techniques for solving the semiconductor product-mix and production planning problems have been developed. These techniques involve the use of multi-objective programming (Chung, Lee, & Pearn, 2003, 2005), linear programming (Bermon & Hood, 1999), mix-integer programming (Chou & Hong, 2000).

A semiconductor fab constantly faces some unexpected events such as machine breakdown, urgent orders, and hold lots. As known, the exact time when an unexpected event will occur cannot be predicted. Therefore, in the decisions of tool planning, product-mix planning and production planning, the *stochastic* effects of unexpected events are treated in a *deterministic* manner. For example, the effect of machine breakdown is modeled by giving each tool a static availability, even though the tool availability in fact changes stochastically and dynamically. Due to the stochastic behavior of unexpected events, the tool capacity of a fab that is available to deal with the *planned* demand in an upcoming short-term period (e.g. one week) may become excessive or insufficient for some workstations. We call this phenomenon a *short-term capacity disequilibrium* problem.

At operational level, capacity planning refers to the decision of solving the short-term capacity disequilibrium problem. A few studies proposed to solve the problem by leveraging the tool capacity of multiple fabs that belong to the *same company*. Taking these fabs as a *single big* fab and their operational control is *centralized*, these studies aimed to develop *real-time inter-fab* dispatching rules for lots (Deboo, 2000; Toba, Izumi, Hatada, & Chikushima, 2005). However, when the operational control systems of these fabs are not centralized, such *real-time* inter-fab dispatching rules may not be effectively implemented due to lack of real time shop-floor information. In practice, two fabs, not being equipped with a centralized operational control system, may still leverage their tool capacity by a weekly trading agreement. That is, based on a prediction of tool utilization, a fab with insufficient capacity at some tools would purchase tool capacity from its partner fab. The short-term *off-line* capacity trading decision is important but has been rarely investigated in literature.

This study aims to develop an effective method for making the short-term off-line capacity trading decision to maximize the total number of completed operations of the two trading fabs. The proposed method involves three modules: (1) identifying a population of capacity trading portfolios, (2) evaluating the performance of a trading portfolio by the use of neural network together with discrete-event simulation techniques, (3) finding a near-optimal trading portfolio from the population by the use of a genetic algorithm (GA).

The remainder of the paper is organized as follows. Section 2 describes how to identify the population of capacity trading portfolios. Section 3 presents the method for evaluating the performance of a trading portfolio. Section 4 discusses the GA technique for finding a near-optimal trading portfolio. Section 5 shows a numerical example and concluding remarks are placed in Section 6.

2. Population of capacity trading portfolios

There are two factors that determine the population of all possible capacity trading portfolios. The first factor refers to the number of workstations that would participate in capacity trading—herein called *tradable workstations*. The second factor refers to the upper bound of trading volume for each tradable workstation. This section begins with the operational assumptions for the wafer fabs of interest, and presents the methods for identifying *tradable workstations* and the population of capacity trading portfolios.

2.1. Operational assumptions of wafer fabs

In the decision problem, we assume that two semiconductor fabs have established a capacity-trading agreement and trade capacity weekly. The two fabs have *n*-pairs of *common workstations* by which capacity trading can be implemented. In each pair, the two workstations, physically located in different fabs, are functionally identical and therefore can trade capacity. Some operational assumptions of the two fabs are described below.

(1) Each operation is only processed by a particular type of workstation.
(2) For an operation that can be processed by a pair of common workstation, its processing times at the two fabs are the same.
(3) The weekly tool capacity bought by a fab is uniformly used; that is, the capacity used per day is a constant.
(4) A workstation with bought-in capacity is given a WIP threshold. If the WIP profile is higher than the threshold, wafer lots waiting before the workstation are continuously sent to the other fab until the daily amount of bought-in capacity has been used up.
(5) Both fabs use the uniform policy in releasing lots. That is, given a product mix, the number of lots released to each fab per day is a constant.
(6) The lot dispatching policy at each workstation in each fab is based on the First-In-First-Out rule.

2.2. Identification of tradable workstations

A pair of common workstations is *tradable* if the tool utilization of one workstation in the upcoming week is lower than L_B and that of the other is higher than U_B, where L_B and U_B are two predefined thresholds used to define the intended extent of capacity difference in forming a trading pair. For a trading pair, the low-utilization workstation could sell tool capacity to the high-utilization one, which hereafter are respectively called a *seller workstation* and a *buyer workstation*.

We develop a discrete-event simulation program to estimate the tool utilization of the upcoming week. The simulation program is distinct in twofold.

First, the simulation program is a *deterministic* model, in which the breakdown of machine is modeled in an "average" manner. That is, the tool breakdown is modeled by enlarging the processing time of an operation—through dividing the original processing time by the average tool availability. The reason for using a deterministic simulation model for estimating the tool utilization of the upcoming week is based on the findings of Kim, Shim, Choi, and Hwang (2003) on real-time scheduling. They claimed that in making a short-term scheduling decision the performance of using deterministic simulation is superior to that of using a single run of stochastic simulation. This reflects that a fab's short-term behavior had better be predicted by using deterministic simulation. Using multiple runs of stochastic simulations may improve accuracy; however, it needs lengthy computation time.

Second, the simulation program must capture in detail the fab status at the decision point, which includes the profiles of WIP, hold lots, breakdown of machines, and expected times for the recoveries of down-machines. Such information is important to the capacity trading decision because the time horizon for capacity trading is short—only one week. The ignorance of the initial fab status may seriously affect the performance of a capacity trading portfolio.

2.3. Characterizing the population of capacity trading portfolio

For a pair of tradable workstations, the possible amount of tradable capacity is determined by two factors: basic trading unit and upper bound of trading volume. We define the basic trading unit of capacity (u) as follows: $u = c \times \max(P_{ij} | i \in \mathrm{TW})$ where TW is a set consisting of all tradable workstations, P_{ij} is the processing time of operation j processed at workstation i, and $c \geqslant 1$ is a predefined integer.

For a pair of tradable workstations i, its maximum number of trading units can be defined as: $B_i = \frac{1}{2u}(\rho_{\mathrm{S}i} - \rho_{\mathrm{B}i}) \times Q_i \times T$, where $\rho_{\mathrm{S}i}$ is the before-trading tool utilization of the seller workstation and $\rho_{\mathrm{B}i}$ represents that of the buyer workstation, Q_i is the number of tools in the seller; and T is the time horizon of the trading decision (i.e. one week).

The population of trading portfolio can be defined as follows: $P = \{[b_1, b_2, \ldots, b_m],\ b_i \in Z,\ 0 \leqslant b_i \leqslant B_i\}$, where m denotes the number of tradable workstations, and the number of trading portfolios is $N(P) = \prod_{i=1}^{m}(b_i + 1)$. The value of $N(P)$ can be quite huge. Assume that the maximum tradable units of each workstation are q units. Then, the scenario has q^m trading portfolios; that is, if $q = 20$ and $m = 4$, there are totally 160,000 (20^4) capacity trading portfolios.

3. Performance evaluation of trading portfolios

For each trading portfolio, we can use the aforementioned deterministic simulation program to estimate the performance of each fab. The performance refers to the total number of completed operations of the fab in the upcoming week and is briefly called *move number* hereafter. The sum of move numbers of the two fabs is regarded as the aggregated performance of the trading portfolio.

If it takes one minute to simulate a trading portfolio, an exhaustive evaluation of a typical trading population (with 20^4 trading portfolios) would take about 111 days. The required computation time is undoubtedly too long to be accepted in practice. To reduce the computation time for the performance evaluation of a trading portfolio, we attempt to construct a neural network to effectively and efficiently emulate the simulation of a fab.

3.1. Development of neural network

The basic idea of modeling the simulation of a fab by a neural network is as follows. First, we sample n trading portfolios from population P and evaluate the performance of each portfolio in the fab through simulation. After simulation, the behavior of the fab in the upcoming week can be characterized by n sets of input/output vectors. An input vector refers to a trading portfolio and an output vector refers to the move number of the fab after capacity trading. Second, the n sets of input/output data are used to construct a neutral network for representing the simulation of the fab. The detailed procedure for constructing such a neural network is presented below.

Assume that fabs A and B have m tradable workstations. Each tradable workstation i has at most b_i trading alternatives. The trading population then has $N(P) = \prod_{i=1}^{m}(b_i + 1)$ trading portfolios and can be expressed as $P = \{X_k | 1 \leqslant k \leqslant N(P)\}$, where $X_k = (x_{1k}, x_{2k}, \ldots, x_{mk})$ represents the kth trading portfolio and x_{ik} refers to the number of trading units for workstation i. Herein, $x_{ik} > 0$ denotes that fab A buys in capacity from fab B, and vice versa. Define $Y_k = -X_k$, then Y_k represents the capacity that fab B purchases from fab A in the kth trading portfolio.

From trading population P, n trading portfolios are randomly sampled and put in a set $\tilde{P} = \{X_j\}$. For each trading portfolio in $\tilde{P}$, a simulation program is executed for calculating the move number of each fab in the upcoming week

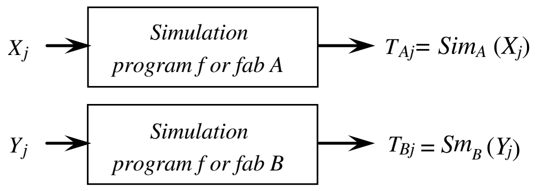


Fig. 1. The input/output relationship of the simulation program.

after capacity trading. As shown in Fig. 1, a functional relationship $T_{Aj} = \text{Sim}_A(X_j)$ is used to represent the simulation program executed for fab A, where T_{Aj} represents the move number of fab A under trading portfolio X_j. The n sampled trading portfolios by simulation will generate n pairs of input/output vectors for fab A, denoted by $\text{Data\_Set\_}A = \{(X_j, T_{Ai}) | X_j \in \tilde{P}\}$. Similarly, a functional relationship $T_{Bj} = \text{Sim}_B(Y_j)$ is used to represent the simulation program executed for fab B, and n pairs of input/output data $\text{Data\_Set\_}B = \{(Y_j, T_{\text{B}i}) | X_j \in \tilde{P}\}$ can be likewise generated. Notice that the two simulation programs Sim_A and Sim_B are both based on a deterministic simulation model because the short-term (one week) performance of a fab is concerned.

Based on Data_Set_A, we construct a neural network Net_A for representing the simulation program Sim_A. Out of the n pairs of input/output vectors in Data_Set_A, n_1 pairs are randomly sampled and used to build a back-propagation neural network Net_A and the other $(n - n_1)$ pairs is used to test the effectiveness of Net_A. The n pairs of data is called training data of Net_A. This neural network model is represented as $\tilde{T}_{Ak} = \text{Net}_A(X_k)$, where $\tilde{T}_{Ak}$ represents the performance of fab A under trading portfolio X_k, calculated by the trained neural network. Likewise, Data_Set_B can be used to construct a neural network $\tilde{T}_{Bk} = \text{Net}_B(Y_k)$ for modeling the simulation program Sim_B.

3.2. Neural network algorithm

We construct Net_A and Net_B by using the back-propagation neural network (BPN) technique (Fausett, 1994), which has been widely used and justified to be effective in various applications (Vellido, Lisboa, & Vaughan, 1999). The architecture of a BPN involves three layers of neurons (Fig. 2); the first layer represents the input, the last layer is the output, and the hidden layer models the transformation mechanism from input to output. Each neuron in a layer and that in its subsequent layer is connected by a link on which a weight (a real number) is to be found.

The development of a BPN, also called training, is to determine the weight on each link so that the BPN can well model the input/output mapping of the simulation program of concern. For example, given an input X_j in Data_Set_A, the output $\tilde{T}_{Aj}$ computed by a well-trained BPN would be fairly close to the target output T_{Aj}. A well-trained BPN, Net_A, can therefore be used to model the simulation program Sim_A.

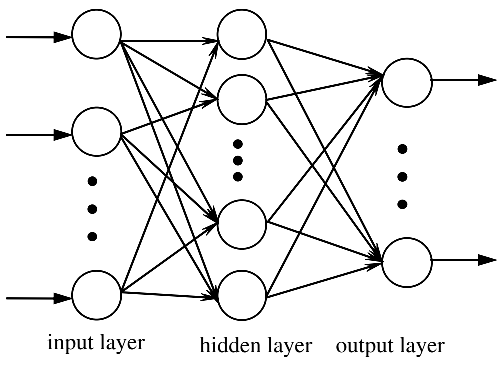


Fig. 2. Architecture of back-propagation neural network with one hidden layer.

The detailed algorithm for training a BPN can be referred to (Fausett, 1994). The algorithm is essentially based on the *gradient descent* technique, which changes the network weights iteratively by the formula:

$$w_{ijk}(n+1) = w_{ijk}(n) + \eta \cdot w_{ijk}(n) + \alpha \cdot w_{ijk}(n-1)$$

where i denotes a neuron in layer k, j denotes a neuron in the preceding layer $(k-1)$, and w_{ijk} represents the weight between the two neurons. The constant η (which lies in the range 0–1) is called the learning rate, which determines the speed of convergence and α (also in the range 0–1) is called the momentum constant.

The accuracy of the model is evaluated in terms of the root-mean-square error (RMSE), the prediction of RMSE_x for fab x is calculated by obtaining the square-root error between the neural network's predicted value and the actual target value and is given by

$$\text{RMSE}_x = \sqrt{\frac{1}{n-1}\sum_{i-1}^{n}(T_{xi} - \tilde{T}_{xi})^2}$$

where n is the number of training/testing data, T_{xi} is the move number of fab x computed by the simulation program Sim_x, and $\tilde{T}_{xi}$ is the move number of fab x computed by the neural network Net_x. The training error is the RMSE of the data used for the network training, and the prediction error is the RMSE of the data reserved for network testing.

Network structure and training issues, such as the number of layers, number of neurons, the learning rate and the momentum constant are determined during the network development process. These values are selected such that after training, the outputs of the established neural network best match the experiment data.

4. Finding a near-optimal trading portfolio

Using the developed neural network to evaluate the performance of a trading portfolio indeed saves computation. Yet, applying an exhaustive search embedded with the neural network technique to find an optimal trading portfolio may still need a lengthy computation time because the solution space is quite huge. We therefore developed a genetic algorithm (GA) to efficiently obtain a near-optimal solution. The reason for using GA is that the technique has been widely applied and shown satisfactory results in various space-searching problems.

In the proposed GA, a trading portfolio $X_j = (x_{1j}, x_{2j}, \ldots, x_{mj})$ is called a chromosome where x_{ij} is called a gene. The performance of a trading portfolio, represented by $F(X_j)$, is called the *fitness* of the chromosome X_j. The higher the value of $F(X_j)$, the higher quality is chromosome X_j.

The procedure of the GA is briefly described as follows. An initial GA population $G(t = 0)$ is generated by randomly sampling N chromosomes from the trading portfolio population P. Three *genetic operators* (reproduction, crossover and mutation) are used to manipulate the chromosomes in $G(t)$ to form the next generation population $G(t + 1)$. The population is iteratively updated until a terminating condition is reached.

Reproduction is a process by which a chromosome with higher fitness value has higher probability of being reproduced so that a larger number of the chromosome copies may be included in the new population. We use the elitist roulette wheel selection method in the reproduction process (Mitchell, 1998). After reproduction, the survived chromosomes are stored in a "mating pool" and await mutation and crossover operations. The crossover operation randomly takes two chromosomes and interchanges part of their genetic values to produce two new chromosomes based on a crossover rate (C_r). The mutation operation is implemented by randomly changing a genetic value based on a specific mutation rate (M_r). The entire process of three genetic operators is intended to create "more fit" chromosomes in the next population. The successive updating of $G(t)$ is terminated when *the best solution* in $G(t)$ keeps unchanged for N_b generations or when maximum number of iterations (N_f) is reached (i.e. $t = N_f$).

5. Numerical example

We use a numerical example to justify the effectiveness of the proposed method for making the capacity trading decision. The example scenario includes two fabs (Fab_A and Fab_B) that attempt to trade capacity on a weekly basis. Table 1 shows the tool numbers and the products of the two fabs, in which the 4P1M product is a memory product while the other 1PXM products are logic products. Notice that the tools required for producing memory products are significantly different from that for producing logic products. The tool portfolios of the two fabs are thus essentially different and may be able to supplement each other in tool capacity. The routing information of each example product is provided by a semiconductor company in Taiwan.

We use eM-plant (Tecnomatix Technologies Ltd., 2001) to establish three simulation programs. Each of these three is a variant of a particular simulation program, with distinction in their simulation settings.

The first simulation program (called Sim_1), executed in a single-replicate and stochastic simulation model, was used to create and update the decision scenario before capacity trading. The output of the scenario-creation program provides the WIP profiles and the machine up-down status of the two fabs, which reflects the *initial status* of the two fabs before trading and is the input of the second simulation program.

The second simulation program (called Sim_2), executed in a deterministic simulation model, aimed to generate the training data set for establishing the neural network. We firstly used Sim_2 to identify tradable workstations to define the solution space. In the testing example, four types of workstations (E10, E27, E31, E55) are found to be tradable (Table 2). Fab_A would buy-in tool capacity for workstations E10 and E55, while Fab_B would buy-in tool capacity for E27 and E31. Using 20 h as the basic unit for capacity trading, the solution space involves 103,488 trading portfolios. Secondly, we randomly select 2000 sets of trading portfolios and obtain each of their performances by Sim_2. The simulation run for measuring the performance of each trading portfolio can be executed on a different computer. Therefore we use 20 personal computers (Pentium IV, 2.0 GHz and 256-MB memory) to execute the 2000 simulation runs in parallel; and it takes about 1.35 h to finish all the simulation runs. The 2000 set of trading portfolios and their performance were used to establish the neural network. The neural network combined with the GA algorithm can be used to determine a near-optimal trading portfolio. The parameters of neural network and GA are shown in Table 3.

Table 1
Tools and products for Fab_A and Fab_B

FAB	Number of workstations	Total number of machines	Product	Total processing time (h)	Total number of operations
Fab_A	60	270	4P1M	400	358
			1P7M	440	412
Fab_B	60	198	1P3M	318	276
			1P8M	480	446

Table 2
Capacity utility estimation of buyer workstations in each fab and maximum buy-in volume at the first trading decision scenario

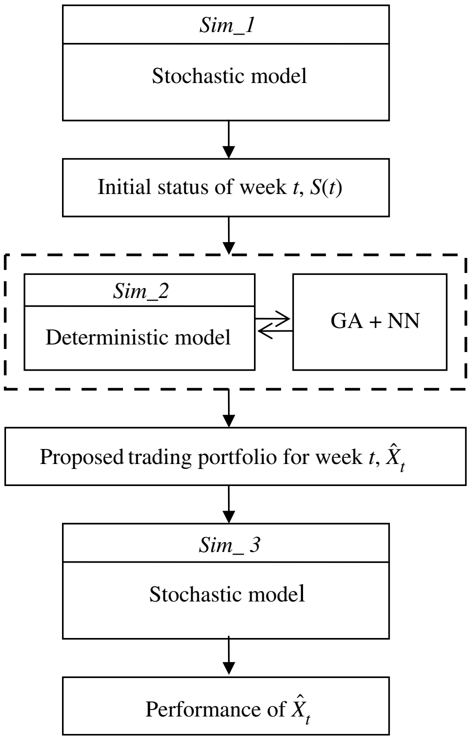
Tradable workstation	E10	E55	E27	E31
Estimate of utilization in upcoming week at Fab_A (%)	100	99.4	42.4	32.4
Estimate of utilization in upcoming week at Fab_B (%)	30.0	49.8	96.6	95.2
Maximum amount of tradable tool capacity (hours)	220	320	280	840

Table 3
Parameters of neural network and genetic algorithm

	Item	Value	Item	Value
Neural network	Number of input nodes	4	Number of training data	1500
	Number of hidden nodes	7	Number of testing data	500
	Number of output nodes	1	Maximum iterations	50,000
	Learning rate	0.10	Momentum	0.80
Genetic algorithm	Population size	100	Maximum iteration, N_f	20,000
	Cross over rate	0.80	Maximum generation, N_b	1000
	Mutation rate	0.05		

Table 4
The proposed buy-in decision and the performance of no-trading and proposed method

	Buyer workstation	Buy-in capacity	Move number with no-trading	Move number with proposed method
Fab_A	E10	160	72,071	74,598
	E55	220		
Fab_B	E27	180	50,126	51,665
	E31	720		


Sim_1
Stochastic model
Initial status of week t, $S(t)$
Sim_2
Deterministic model
GA + NN
Proposed trading portfolio for week t, $\hat{X}_t$
Sim_ 3
Stochastic model
Performance of $\hat{X}_t$
Fig. 3. The input/output relationship of the three simulation programs.

The third simulation program (called Sim_3), executed in a stochastic simulation model with 20 replicates and the simulation time horizon is one week, was used to justify the effectiveness of the trading portfolio suggested by the proposed algorithm. The output of Sim_3 includes the move number of each fab in the upcoming week after capacity trading. The sum of move numbers of the two fabs is taken as the performance of the trading portfolio. The proposed buy-in decision and its performance are shown in Table 4.

In summary, the three simulation programs are integrated into a procedure to create the proposed trading decision for a particular week and justify its effectiveness. The input/output relationships in the procedure are shown in Fig. 3. Program Sim_1 was used to create $S(t)$, the *decision scenario* or the *initial status* of week t. Program Sim_2 was used to generate data for establishing a neural network, by which we can use the GA to yield $\hat{X}_t$, the proposed trading decision for week t. Program Sim_3 was used to evaluate the performance of $\hat{X}_t$. With $\hat{X}_t$ and $S(t)$ as input, program

Table 5
Comparison of the weekly effectiveness of proposed trading portfolio with no-trading and max-trading methods

Period	Proposed trading portfolio			Without any trading			Maximum trading portfolio		
	Fab_A	Fab_B	Aggregate	Fab_A	Fab_B	Aggregate	Fab_A	Fab_B	Aggregate
1	74,598	51,665	126,263	72,071	50,126	122,197	73,756	51,183	124,939
	(103.5%)	(103.1%)	(103.3%)	(100.0%)	(100.0%)	(100.0%)	(102.3%)	(102.1%)	(102.2%)
2	73,981	50,545	124,526	71,675	49,404	121,079	74,071	49,124	123,195
	(103.2%)	(102.3%)	(102.9%)	(100.0%)	(100.0%)	(100.0%)	(103.3%)	(99.4%)	(101.7%)
3	74,797	51,957	126,754	72,463	50,254	122,717	73,026	50,265	123,291
	(103.2%)	(103.4%)	(103.3%)	(100.0%)	(100.0%)	(100.0%)	(100.8%)	(100.0%)	(100.5%)
4	72,833	50,693	123,463	70,664	49,678	120,342	71,426	51,036	122,462
	(103.1%)	(102.1%)	(102.7%)	(100.0%)	(100.0%)	(100.0%)	(101.1%)	(100.7%)	(101.8%)
5	73,978	49,944	123,922	71,722	48,369	120,091	72,738	48,927	121,665
	(103.2%)	(103.3%)	(103.2%)	(100.0%)	(100.0%)	(100.0%)	(101.4%)	(101.1%)	(101.3%)
6	74,852	49,896	124,748	70,849	49,369	120,218	73,580	49,808	123,388
	(105.7%)	(101.1%)	(103.8%)	(100.0%)	(100.0%)	(100.0%)	(103.8%)	(100.9%)	(102.6%)
7	74,065	51,147	125,212	72,918	48,975	121,893	74,213	48,755	122,968
	(101.6%)	(104.4%)	(102.7%)	(100.0%)	(100.0%)	(100.0%)	(101.7%)	(99.5%)	(100.9%)
8	73,998	51,268	125,266	71,394	49,863	121,257	73,986	49,861	123,847
	(103.7%)	(102.8%)	(103.3%)	(100.0%)	(100.0%)	(100.0%)	(103.6%)	(100.0%)	(102.1%)

The move numbers are shown as a percentage of that of without any trading method in parenthesis.

Table 6
The results after performing eight consecutive weeks of proposed trading portfolio and without any trading and maximum trading methods

	With proposed trading portfolio				Without any trading				Maximum trading			
	Fab_A		Fab_B		Fab_A		Fab_B		Fab_A		Fab_B	
	4P1M	1P7M	1P3M	1P8M	4P1M	1P7M	1P3M	1P8M	4P1M	1P7M	1P3M	1P8M
Throughput	495	335	447	262	476	332	431	252	484	337	435	253
Mean CT	582.7	625.2	419.2	649.2	614.0	648.7	441.6	672.3	603.3	641.3	431.4	655.6

Sim_1 was use to obtain $S(t+1)$, the decision scenario of week $t+1$. The procedure can be repeatedly performed to obtain the results of consecutively implementing capacity trading for multiple weeks.

We justify the effectiveness of the proposed method by performing capacity trading for eight consecutive weeks. The proposed method is compared with two other methods. The first one is without any trading. The second one, called the *maximum trading method*, requests each buyer workstation buys-in its maximum amount of trading units (B_i). Table 5 shows that in each of the eight weeks the proposed trading method outperforms the other two methods (about 2.7–3.8% higher) in terms of the aggregate move number. This implies that the proposed trading method can effectively increase the aggregated number of completed operations for the two fabs.

One may wonder such an increase in *completed operations* could lead to an increase in *completed products*. Table 6 indicates that the proposed method outperforms the other trading methods in terms of throughput and mean cycle time. Herein, throughput refers to the total number of completed products during the eight trading weeks. This implies that the proposed capacity trading method could also effectively increase the aggregated fab throughput.

6. Concluding remarks

This paper develops a short-term *off-line* capacity trading method for two semiconductor fabs. The method involves three major techniques. The first technique—discrete-event simulation is essentially used to evaluate the performance for a set of randomly sampled trading portfolios. The second technique—neural network is used to emulate the function of the simulation technique; that is, evaluating the performance of a trading portfolio at a much faster speed. The third technique—genetic algorithm is used to find a near-optimum trading portfolio in an efficient manner.

Two other trading methods (without-trading and maximum-trading) are compared with the proposed one. Experiment results indicate that the proposed capacity trading method outperforms the two other methods in each of the three performance indices: aggregated number of completed operations, aggregated throughput, and mean cycle time.

The implementation of the proposed method may take a significant amount of computation time in collecting the simulation data for establishing the neural network. One possible extension to this study is developing methods for reducing the computation time for establishing the neural network.

References

Barahona, F., Bermon, S., Gunluk, O., & Hood, S. (2001). Robust capacity planning in semiconductor manufacturing. IBM report RC22196.

Bard, J. F., Srinivasan, K., & Tirupati, D. (1999). An optimization approach to capacity expansion in semiconductor manufacturing facilities. *International Journal of Production Research, 37*(15), 3359–3382.

Bermon, S., & Hood, S. J. (1999). Capacity optimization planning system. *Interfaces, 29*(5), 31–50.

Çatay, B., Erengüç, Ş. S., & Vakharia, A. J. (2003). Tool capacity planning in semiconductor manufacturing. *Computers & Operations Research, 30*, 1349–1366.

Chen, J. C., Chen, C. W., Lin, C. J., & Rau, H. (2005). Capacity planning with capability for multiple semiconductor manufacturing fabs. *Computers & Industrial Engineering, 48*, 709–732.

Chen, L. H., & Chen, Y. H. (1996). A design procedure for a robust job shop manufacturing system under a constraint using computer simulation experiments. *Computers & Industrial Engineering, 30*(1), 1–12.

Chou, Y. C., & Hong, I. H. (2000). A methodology for product mix planning in semiconductor foundry manufacturing. *IEEE Transactions on Semiconductor Manufacturing, 13*(3), 278–285.

Chou, Y. C., Wu, C. S., Kao, C. E., & Hsieh, S. H. (2001). Integration of capacity planning techniques for tool portfolio planning in semiconductor manufacturing. *International Journal of Industrial Engineering – Theory Applications and Practice, 8*(4), 279–288.

Christie, R. M. E., & Wu, S. D. (2002). Semiconductor capacity planning: stochastic model and computational studies. *IIE Transaction, 34*, 131–143.

Chung, S. H., Lee, A. H. I., & Pearn, W. L. (2003). Product mix optimization for semiconductor manufacturing based on AHP and ANP Analysis. *The International Journal of Advanced Manufacturing Technology, 25*(11–12), 1144–1156.

Chung, S. H., Lee, A. H. I., & Pearn, W. L. (2005). Analytic network process (ANP) approach for product mix planning in semiconductor fabricator. *International Journal of Production Economics, 96*, 15–36.

Connors, D. P., Feigin, G. E., & Yao, D. (1996). A queueing network model for semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing, 9*(3), 412–427.

Deboo, S. (2000). Block cross processing: an innovative approach to constrain management. In *The ninth international symposium on semiconductor manufacturing*, pp. 225–228.

Fausett, L. (1994). *Fundamental of neural networks: Architectures, algorithms, and applications*. Englewood Cliffs, New Jersey: Prentice Hall.

Grewal, N. S., Bruska, A. C., Wulf, T. M., & Robinson, J. K. (1998). Integrating targeted cycle-time reduction into the capital planning process. In *Proceedings of the 1998 winter simulation conference*, Washington, DC, pp. 1005–1010.

Hood, S. J., Bermon, S., & Barahona, F. (2003). Capacity planning under demand uncertainty for semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing, 16*(2), 273–280.

Hopp, W. J., Spearman, M. L., Chayet, S., Donohue, K. L., & Gel, E. S. (2002). Using an optimized queuing network model to support wafer fab design. *IIE Transactions, 34*(2), 119–130.

Iwata, Y., Taji, K., & Tamura, H. (2003). Multi-objective capacity planning for agile semiconductor manufacturing. *Production Planning & Control, 14*(3), 244–254.

Kim, Y. D., Shim, S. O., Choi, B., & Hwang, H. (2003). Simplification methods for accelerating simulation-based real-time scheduling in a semiconductor wafer fabrication facility. *IEEE Transactions on Semiconductor Manufacturing, 16*(2), 290–298.

Mitchell, M. (1998). *An introduction to genetic algorithms*. Cambridge, MA: MIT Press.

Mollaghsemi, M., & Evans, G. W. (1994). Multicriteria design of manufacturing systems through simulation optimization. *IEEE Transactions on Systems, Man, and Cybernetics, 24*(8), 1407–1411.

Swaminathan, J. M. (2000). Tool capacity planning for semiconductor fabrication facilities under demand uncertainty. *European Journal of Operational Research, 120*, 545–558.

Swaminathan, J. M. (2002). Tool procurement planning for wafer fabrication facilities: a scenario-based approach. *IIE Transaction, 34*, 145–155.

Tecnomatix Technologies Ltd. (2001). *EM-PLANT objects manual*. Germany: Tecnomatix Software Company.

Toba, H., Izumi, H., Hatada, H., & Chikushima, T. (2005). Dynamic load balancing among multiple fabrication lines through estimation of minimum inter-operation time. *IEEE Transactions on Semiconductor Manufacturing, 18*(1), 202–213.

Vellido, A., Lisboa, P. J. G., & Vaughan, J. (1999). Neural networks in business: a survey of applications (1992–1998). *Expert Systems with Applications, 17*, 51–70.

Wu, S. D., Erkoc, M., & Karabuk, S. (2005a). Managing capacity in the high-tech industry: a review of literature. *The Engineering Economist, 50*(2), 125–158.

Wu, M. C., Hsiung, Y. I., & Hsu, H. M. (2005b). A tool planning approach considering cycle time constraints and demand uncertainty. *International Journal of Advanced Manufacturing Technology, 26*(5), 565–572.