Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2012, Article ID 471973, 9 pages
doi:10.1155/2012/471973

Research Article

A Nonlinear Programming and Artificial Neural Network Approach for Optimizing the Performance of a Job Dispatching Rule in a Wafer Fabrication Factory

Toly Chen

Department of Industrial Engineering and Systems Management, Feng Chia University, No. 100 Wenhwa Road, Seatwen, Taichung 407, Taiwan

Correspondence should be addressed to Toly Chen, tcchen@fcu.edu.tw

Received 21 May 2012; Accepted 18 July 2012

Academic Editor: Yi-Chi Wang

A nonlinear programming and artificial neural network approach is presented in this study to optimize the performance of a job dispatching rule in a wafer fabrication factory. The proposed methodology fuses two existing rules and constructs a nonlinear programming model to choose the best values of parameters in the two rules by dynamically maximizing the standard deviation of the slack, which has been shown to benefit scheduling performance by several studies. In addition, a more effective approach is also applied to estimate the remaining cycle time of a job, which is empirically shown to be conducive to the scheduling performance. The efficacy of the proposed methodology was validated with a simulated case; evidence was found to support its effectiveness. We also suggested several directions in which it can be exploited in the future.

1. Introduction

This study attempts to optimize the performance of a job dispatching rule in a wafer fabrication factory. The production equation required by a wafer fabrication factory is very expensive and must be fully utilized. For this purpose, to ensure that the capacity does not substantially exceed the demand is a perquisite. Subsequently, how to plan the use of the existing capacity to shorten the cycle time and maximize the turnover rate is an important goal. In this regard, scheduling is undoubtedly a very useful tool.

However, some studies [1–4] noted that job dispatching is very difficult task in a semiconductor manufacturing factory. Theoretically, it is an NP-hard problem. In practice, many semiconductor manufacturing factories suffer from lengthy cycle times and are not able to improve on their delivery promises to their customers.

Semiconductor manufacturing can be divided into four stages: wafer fabrication, wafer probing, packaging, and final testing. The most important stage is wafer fabrication. It is also the most time-consuming one. In this study, we investigated the job dispatching for this stage. This field includes many different methods, including dispatching rules, heuristics, data-mining-based approaches [5, 6], agent technologies [5, 7–9], and simulation. Among them, dispatching rules (e.g., first-in first out (FIFO), earliest due date (EDD), least slack (LS), shortest processing time (SPT), shortest remaining processing time (SRPT), critical ratio (CR), the fluctuation smoothing rule for the mean cycle time (FSMCT), and the fluctuation smoothing rule for cycle time variation (FSVCT), FIFO+, SRPT+, and SRPT++) all have received a lot of attention over the last few years [5–7] and are the most prevalent methods used in practical applications. For details on the traditional dispatching rules, please refer to Lu et al. [10].

Some advances in this field are as follows. Altendorfer et al. [11] proposed the work in parallel queue (WIPQ) rule targeting maximizing throughput at a low level of work in process (WIP). Zhang et al. [12] proposed the dynamic bottleneck detection (DBD) approach by classifying workstations into several categories and then applied different dispatching rules to these categories. They used three dispatching rules including FIFO, the shortest processing time until the next bottleneck (SPNB), and CR. Based on the current conditions in the wafer fabrication factory, Hsieh et al. [6] chose one approach from FSMCT, FSVCT, largest

deviation first (LDF), one step ahead (OSA), or FIFO. Chen [13] modified FSMCT and proposed the nonlinear FSMCT (NFSMCT) rule, in which he smoothed the fluctuation in the estimated remaining cycle time and balanced it with that of the release time or the mean release rate. To diversify the slack, he applied the "division" operator instead. This was followed by Chen [14], in which he proposed the one-factor-tailored NFSMCT (1f-TNFSMCT) rule and the one-factor-tailored nonlinear FSVCT (1f-TNFSVCT) rule. Both rules contain an adjustable parameter to allow them to be customized for a target wafer fabrication factory. Chen [15] used more parameters and proposed 2f-TNFSMCT and 2f-TNFSVCT.

In a multiple-objective study, Chen and Wang [16] proposed a biobjective nonlinear fluctuation smoothing rule with an adjustable factor (1f-biNFS) to optimize both the average cycle time and the cycle time variation at the same time. More degrees of freedom seem to be helpful in the performance of customizable rules. For this reason, Chen et al. [17] extended 1f-biNFS to a biobjective fluctuation smoothing rule with four adjustable factors (4f-biNFS). For a summary of these rules please refer to Table 1. One drawback of these rules is that only static factors are used, and they must be determined in advance. To this end, most studies (e.g., [13–17]) performed extensive simulations. This is not only time-consuming but it also fails to consider enough possible combinations of these factors. Chen [18] established a mechanism that was able to adjust the values of the factor in 1f-biNFS dynamically (dynamic 1f-biNFS). However, even though satisfactory results were obtained in his experiment, there was no theoretical basis supporting the proposed mechanism. Chen [19] attempted to relate the scheduling performance to the factor values using a back propagation network (BPN). If that would have worked, then the factor values contributing to the optimal scheduling performance could have been found. However, the explanatory ability of the BPN was not good enough.

At the same time, Chen [18] stated that a nonlinear fluctuation smoothing rule uses the divisor operator instead of the subtraction operator, which diversifies the slack and makes the nonlinear fluctuation smoothing rule more responsive to changes in the parameters. Chen and Wang [16] proved that the effects of the parameters are balanced better in a nonlinear fluctuation smoothing rule than in a traditional one if the variation in the parameters is large. In addition, there will be fewer ties since the slack values are very different. Further, magnifying the difference in the slack seems to improve the scheduling performance, especially with respect to the average cycle time [20]. For these reasons, a slack-diversifying fuzzy-neural rule is used in chen et al. [20] for job dispatching in a wafer fabrication factory, in order to further improve the performance of job dispatching in a wafer fabrication factory. The slack-diversifying nonlinear fluctuation smoothing rule is modified from 1f-TNFSVCT by maximizing the difference in the slack measured with the standard deviation of the slack.

This study adopts several treatments to further improve Wang et al.'s approach.

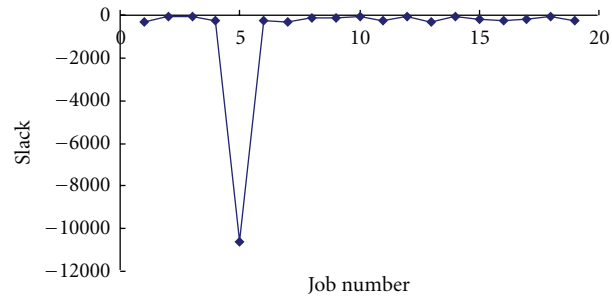

Figure 1: The extreme cases.

(1) In nonlinear fluctuation smoothing rules, it is common that some jobs have very large or small slack values, that is the extreme case (see Figure 1), which usually distorts the results of calculating the standard deviation of slacks. In this study, the extreme cases are excluded before calculation.

(2) Two objectives, the average cycle time and cycle time standard deviation, are considered at the same time by fusing the results from 2f-TNFSMCT and 2f-TNFSVCT.

(3) A nonlinear programming problem is solved to find the optimal values of parameters in 2f-TNFSMCT and 2f-TNFSVCT.

(4) On the other hand, the remaining cycle time of a job needs to be estimated in 2f-TNFSMCT and 2f-TNFSVCT. For this reason, we also propose a more effective fuzzy-neural approach to estimate the remaining cycle time of a job. The fuzzy-neural approach is a modification of the fuzzy c-means and back propagation network (FCM-BPN) approach [17] by incorporating in the concept of principal component analysis (PCA). According to Chen and Wang [3], with more accurate remaining cycle time estimation, the scheduling performance of a fluctuation smoothing rule can be significantly improved. In the original study, Chen and Wang used a gradient search algorithm for training the BPN, which is time-consuming and not very accurate. In this study, we use the Levenberg-Marquardt algorithm to achieve the same purpose, which is more efficient than that in Chen and Wang's study and can produce more accurate forecasts.

The differences between the proposed methodology and the previous methods are summarized in Table 1.

The remainder of this paper is arranged as follows. Section 2 provides the details of the proposed methodology. In Section 3, a simulated case is used to validate the effectiveness of the nonlinear programming and artificial neural network approach. The performances of some existing approaches in this field are also examined using the simulated data. Finally, we draw our conclusions in Section 4 and provide some worthwhile topics for future work.

Table 1: The differences between the proposed methodology and the previous methods.

Rule	Number of objectives	Objectives	Number of adjustable parameters	Optimized?	How to derive the rule?
NFSMCT	1	Average cycle time	1	No	(i) Generalizing FSMCT
1f-TNFSVCT	1	Cycle time standard variation	1	No	(i) Generalizing FSVCT (ii) Adding adjustable parameters
1f-TNFSMCT	1	Average cycle time	1	No	(i) Generalizing FSMCT (ii) Adding adjustable parameters
2f-TNFSVCT	1	Cycle time standard deviation	2	No	(i) Generalizing FSVCT (ii) Adding adjustable parameters
4f-biNFS	2	Average cycle time, cycle time standard deviation	2	Yes	(i) Fusing FSVCT and FSMCT (ii) Adding adjustable parameters
The proposed methodology	2	Average cycle time, cycle time standard deviation	2	Yes	(i) Fusing 2f-TFSMCT and 2f-TNFSVCT (ii) Nonlinear programming

2. Methodology

The variables and parameters that will be used in the proposed methodology are defined in the following.

(1) R_j: the release time of job j; $j = 1 \sim n$.

(2) BQ_j: the total queue length before bottlenecks at R_j.

(3) CR_{ju}: the critical ratio of job j at step u.

(4) CT_j: the cycle time of job j.

(5) CTE_j: the estimated cycle time of job j.

(6) D_j: the average delay of the three most recently completed jobs at R_j.

(7) DD_j: the due date of job j.

(8) FQ_j: the total queue length in the whole factory at R_j.

(9) Q_j: the queue length on the processing route of job j at R_j.

(10) RCTE_{ju}: the estimated remaining cycle time of job j from step u.

(11) RPT_{ju}: the remaining processing time of job j from step u.

(12) SCT_{ju}: the step cycle time of job j until step u.

(13) SK_{ju}: the slack of job j at step u.

(14) U_j: the average factory utilization before job j is released. If the utilization of the factory is reported on a daily basis, then U_j is the utilization of the day before job j is released.

(15) WIP_j: the factory work in progress (WIP) at R_j.

(16) λ: mean release rate.

(17) x_p: inputs to the three-layer BPN, $p = 1 \sim 6$.

(18) h_l: the output from hidden-layer node l, $l = 1 \sim L$.

(19) w_l^o: the connection weight between hidden-layer node l and the output node.

(20) w_{pl}^h: the connection weight between input node p and hidden-layer node l, $p = 1 \sim 6$; $l = 1 \sim L$.

(21) θ_l^h: the threshold on hidden-layer node l.

(22) θ^o: the threshold on the output node.

The proposed methodology includes the following seven steps.

Step 1. Replacing parameters using PCA.

Step 2. Use FCM to classify jobs. The required inputs for this step are the new variables determined by PCA. To determine the optimal number of categories, we use the S-test. The output of this step is the category of each job.

Step 3. Use the BPN approach to estimate the cycle time of each job. Jobs of different categories will be sent to different three-layer BPNs. The inputs to the three-layer BPN include the new variables of a job, while the output is the estimated cycle time of the job.

Step 4. Derive the remaining cycle time of each job from the estimated cycle time.

Step 5. Incorporate the estimated remaining cycle time into the new rule that is composed of two subrules–2f-TNFSMCT and 2f-TNFSVCT.

Step 6. Find out the optimal value of parameters in the new rule by solving a nonlinear programming problem.

The remaining cycle time of a job being produced in a wafer fabrication factory is the time still needed to complete the job. If the job is just released into the wafer fabrication factory, then the remaining cycle time of the job is its cycle time. The remaining cycle time is an important input for the scheduling rule. Past studies (e.g., [21–24]) have shown that the accuracy of the remaining cycle time forecasting can be improved by job classification. Soft computing methods (e.g., [3, 20, 25, 26]) have received much attention in this field.

2.1. PCA Analysis. First, PCA is used to replace the inputs to the FCM-BPN. The combination of PCA and FCM has

been shown to be a more effective classifier than FCM alone. Although there are more advanced applications of PCA, in this study PCA is used to enhance the efficiency of training the FCM-BPN. PCA consists of the four following steps:

(1) Raw data standardization: to eliminate the difference between the dimensions and the impact of large numerical difference in the original variables$\{U_j, Q_j, \mathrm{BQ}_j, \mathrm{FQ}_j, \mathrm{WIP}_j, D_j\}$, the original variables are standardized:

$$x_{ij}^* = \frac{x_{ij} - \overline{x}_j}{\sigma_j},$$
$$\overline{x}_j = \frac{\sum_{i=1}^n x_{ij}}{n}, \tag{1}$$
$$\sigma_j = \sqrt{\frac{\sum_{i=1}^n \left(x_{ij} - \overline{x}_j\right)^2}{n-1}},$$

where $\overline{x}_j$ and σ_j indicate the mean and standard deviation of variable j, respectively,

(2) Establishment of the correlation matrix R:

$$R = \frac{1}{n-1} X^{*T} X^*, \tag{2}$$

where X^* is the standardized data matrix. The eigenvalues and eigenvectors of R are calculated and represented as $\lambda_1 \sim \lambda_m$ and $u_1 \sim u_m$, respectively; $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$.

(3) Determination of the number of principal components: the variance contribution rate is calculated as:

$$\eta_i = \frac{\lambda_i}{\sum_{i=1}^m \lambda_i} \cdot 100\%, \tag{3}$$

and the accumulated variance contribution rate is

$$\eta_\Sigma(q) = \sum_{i=1}^q \eta_i. \tag{4}$$

Choose the smallest q value such that $\eta_\Sigma(q) \geq 85\%\sim 90\%$.

(4) Formation of the following matrixes:

$$U_{m\times q} = \left[u_1, u_2, ..., u_q\right],$$
$$Z_{n\times q} = X^*_{n\times m} U_{m\times q}. \tag{5}$$

After PCA, examples are then classified using FCM.

2.2. The FCM Approach. In the proposed methodology, jobs are classified into K categories using FCM. If a crisp clustering method is applied, then it is possible that some clusters will have very few examples. In contrast, an example belongs to multiple clusters to different degrees in FCM, which provides a solution to this problem.

FCM classifies jobs by minimizing the following objective function:

$$\mathrm{Min} \sum_{k=1}^K \sum_{j=1}^n \mu_{j(k)}^m e_{j(k)}^2, \tag{6}$$

where K is the required number of categories; n is the number of jobs; $\mu_{j(k)}$ indicates the membership that job j belongs to category k; $e_{j(k)}$ measures the distance from job j to the centroid of category k; $m \in [1, \infty)$ is a parameter to adjust the fuzziness and is usually set to 2. The procedure of FCM is as follows.

(1) Produce a preliminary clustering result.

(2) (Iterations) Calculate the centroid of each category as

$$\overline{x}_{(k)} = \left\{\overline{x}_{(k)p}\right\}; \quad p = 1 \sim q,$$
$$\overline{x}_{(k)p} = \frac{\sum_{j=1}^n \mu_{j(k)}^m x_{jp}}{\sum_{j=1}^n \mu_{j(k)}^m},$$
$$\mu_{j(k)} = \frac{1}{\sum_{q=1}^K \left(e_{j(k)}/e_{j(q)}\right)^{2/(m-1)}}, \tag{7}$$
$$e_{j(k)} = \sqrt{\sum_{\text{all } p} \left(x_{jp} - \overline{x}_{(k)p}\right)^2},$$

where $\overline{x}_{(k)}$ is the centroid of category k. $\mu_{j(k)}^{(t)}$ is the membership that job i belongs to category k after the tth iteration.

(3) Remeasure the distance from each job to the centroid of each category, and then recalculate the corresponding membership.

(4) Stop if the following condition is met. Otherwise, return to step (2):

$$\max_k \max_j \left| \mu_{j(k)}^{(t)} - \mu_{j(k)}^{(t-1)} \right| < d, \tag{8}$$

where d is a real number representing the threshold for the convergence of membership.

Finally, the separate distance test (S-test) proposed by Xie and Beni [24] can be applied to determine the optimal number of categories K:

$$\mathrm{Min}\ S \tag{9}$$

subject to

$$J_m = \sum_{k=1}^K \sum_{j=1}^n \mu_{j(k)}^m e_{j(k)}^2,$$
$$e_{\min}^2 = \min_{k1 \neq k2} \left(\sum_{\text{all } p} \left(\overline{x}_{(k1)p} - \overline{x}_{(k2)p}\right)^2 \right), \tag{10}$$
$$S = \frac{J_m}{n \times e_{\min}^2},$$
$$K \in Z^+.$$

The K value minimizing S determines the optimal number of categories.

2.3. The BPN Approach. After clustering, a portion of the jobs in each category is input as the "training examples" to the three-layer BPN to determine the parameter values. The configuration of the three-layer BPN is set up as follows. First, inputs are the six parameters associated with the jth example/job including the q new variables. These parameters have to be normalized before feeding into the three-layer BPN. Subsequently, there is only a single hidden layer with neurons that are twice that in the input layer. Finally, the output from the three-layer BPN is the (normalized) estimated cycle time (CTE_j) of the example. The activation function used in each layer is Log Sigmoid function:

$$f(x) = \frac{1}{(1 + e^{-x})}. \tag{11}$$

The procedure for determining the parameter values is now described. Two phases are involved at the training stage. At first, in the forward phase, inputs are multiplied with weights, summated, and transferred to the hidden layer. Then activated signals are outputted from the hidden layer as

$$h_l = \frac{1}{1 + e^{-n_l^h}}, \tag{12}$$

where

$$\begin{aligned} n_l^h &= I_l^h - \theta_l^h \\ I_l^h &= \sum_{p=1}^{q} w_{pl}^h \cdot x_{jp}. \end{aligned} \tag{13}$$

h_l's are also transferred to the output layer with the same procedure. Finally, the output of the BPN is generated as

$$o_j = \frac{1}{1 + e^{-n^o}}, \tag{14}$$

where

$$\begin{aligned} n^o &= I^o - \theta^o, \\ I^o &= \sum_{l=1}^{L} w_l^o \cdot h_l. \end{aligned} \tag{15}$$

Subsequently, in the backward phase, some algorithms are applicable for training a BPN, such as the gradient descent algorithms, the conjugate gradient algorithms, and the Levenberg-Marquardt algorithm. In this study, the Levenberg-Marquardt algorithm is applied. The Levenberg-Marquardt algorithm was designed for training with second-order speed without having to compute the Hessian matrix. It uses approximation and updates the network parameters in a Newton-like way, as described below.

The network parameters are placed in vector $\boldsymbol{\beta} = [w_{11}^h, \ldots, w_{qL}^h, \theta_1^h, \ldots, \theta_L^h, w_1^o, \ldots, w_L^o, \theta^o]$. The network output o_j can be represented with $f(\mathbf{x}_j, \boldsymbol{\beta})$. The objective function of the BPN is to minimize the root mean-squared error (RMSE) or equivalently the sum of squared error (SSE):

$$\mathrm{SSE}(\boldsymbol{\beta}) = \sum_{j=1}^{n} \left(N\left(\mathrm{CT}_j\right) - f\left(\mathbf{x}_j, \boldsymbol{\beta}\right)\right)^2. \tag{16}$$

The Levenberg-Marquardt algorithm is an iterative procedure. In the beginning, the user should specify the initial values of the network parameters $\boldsymbol{\beta}$. Let $\boldsymbol{\beta}^{\mathbf{T}} = (1, 1, \ldots, 1)$ is a common practice. In each step, the parameter vector $\boldsymbol{\beta}$ is replaced by a new estimate $\boldsymbol{\beta} + \boldsymbol{\delta}$, where $\boldsymbol{\delta} = [\Delta w_{11}^h, \ldots, \Delta w_{qL}^h, \Delta\theta_1^h, \ldots, \Delta\theta_L^h, \Delta w_1^o, \ldots, \Delta w_L^o, \Delta\theta^o]$. The network output becomes $f(\mathbf{x}_j, \boldsymbol{\beta}+\boldsymbol{\delta})$ that is approximated by its linearization as

$$f\left(\mathbf{x}_j, \boldsymbol{\beta} + \boldsymbol{\delta}\right) \approx f\left(\mathbf{x}_j, \boldsymbol{\beta}\right) + \mathbf{J}_j\boldsymbol{\delta}, \tag{17}$$

where

$$\mathbf{J}_j = \frac{\partial f\left(\mathbf{x}_j, \boldsymbol{\beta}\right)}{\partial \boldsymbol{\beta}} \tag{18}$$

is the gradient vector of f with respect to $\boldsymbol{\beta}$. Substituting (17) into (16),

$$\mathrm{SSE}(\boldsymbol{\beta} + \boldsymbol{\delta}) \approx \sum_{j=1}^{n} \left(N\left(CT_j\right) - f\left(\mathbf{x}_j, \boldsymbol{\beta}\right) - \mathbf{J}_j\boldsymbol{\delta}\right)^2. \tag{19}$$

When the network reaches the optimal solution, the gradient of SSE with respect to $\boldsymbol{\delta}$ will be zero. Taking the derivative of $\mathrm{SSE}(\boldsymbol{\beta} + \boldsymbol{\delta})$ with respect to $\boldsymbol{\delta}$ and setting the result to zero gives

$$\left(\mathbf{J}^{\mathbf{T}}\mathbf{J}\right)\boldsymbol{\delta} = \mathbf{J}^{\mathbf{T}}\left(N\left(CT_j\right) - f\left(\mathbf{x}_j, \boldsymbol{\beta}\right)\right), \tag{20}$$

where $\mathbf{J}$ is the Jacobian matrix containing the first derivative of network error with respect to the weights and biases. Equation (20) includes a set of linear equations that can be solved for $\boldsymbol{\delta}$.

Finally, the BPN can be applied to estimate the cycle time of a job, and then the remaining cycle time of the job can be derived as

$$\mathrm{RCTE}_{ju} = \mathrm{CTE}_j - \mathrm{SCT}_{ju}, \tag{21}$$

2.4. The New Rule. In traditional fluctuation smoothing (FS) rules there are two different formulation methods, depending on the scheduling purpose [22]. ne method is aimed at minimizing the average cycle time with FSMCT:

$$\mathrm{SKM}_{ju} = \frac{j}{\lambda} - \mathrm{RCTE}_{ju}. \tag{22}$$

The other method is aimed at minimizing the variance of cycle time with FSVCT:

$$\mathrm{SKV}_{ju} = R_j - \mathrm{RCTE}_{ju}. \tag{23}$$

Jobs with the smallest slack values (SKM_{ju} or SKV_{ju}) will be given higher priority. These two rules and their variants have

been proven to be very effective in shortening the cycle time in wafer fabrication factories [10, 14–17].

Chen [15] normalized the parameters and used the division operator instead and derived the 2f-TNFSVCT rule:

$$\mathrm{SKM}_{ju} = \left(\frac{\beta}{\alpha\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}\right)^{\xi} \cdot \left(R_j - \mathrm{RCTE}_{ju} + \zeta\left(\mathrm{RCTE}_{ju} - \min\left(R_j\right)\right)\right), \tag{24}$$

and the 2f-TNFSMCT rule:

$$\mathrm{SKV}_{ju} = \left(\frac{\lambda\beta}{(n-1)\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}\right)^{\xi} \cdot \left(\frac{j}{\lambda} - \mathrm{RCTE}_{ju} + \zeta\left(\mathrm{RCTE}_{ju} - \frac{1}{\lambda}\right)\right), \tag{25}$$

where

$$\alpha = \max\left(R_j\right) - \min\left(R_j\right), \quad \beta = \max\left(\mathrm{RCTE}_{ju}\right) - \min\left(\mathrm{RCTE}_{ju}\right), \tag{26}$$

$0 \le \xi, \zeta \le 1$. There are many possible models to form the combination of ξ and ζ. For example,

$$\text{(Linear model) } \xi = \zeta, \quad \text{(Nonlinear model) } \xi = \zeta^k, \quad k \ge 0, \quad \text{(Logarithmic model) } \xi = \frac{\ln(1+\zeta)}{\ln 2}. \tag{27}$$

The new rule is composed of two rules. The first rule is derived by diversifying the slack in the 2f-TNFSVCT rule, aimed at minimizing the variation of cycle time [22]. To diversify the slack, the standard deviation of the slack is to be maximized as follows:

$$\sigma_{\mathrm{SKM}_{ju}} = \sqrt{\frac{\sum_{j=1}^{N}\left(\mathrm{SKM}_{ju} - \overline{\mathrm{SKM}_{ju}}\right)^2}{N-1}}. \tag{28}$$

However, in nonlinear fluctuation smoothing rules, it is common that two of the jobs will have very large or small slack values, that is, the extreme cases, which distort the sequencing results. For this reason, such jobs are put in a set EC that will be excluded from calculating the standard deviation:

$$\overline{\mathrm{SKM}'_{ju}} = \frac{\sum_{j=1\ j\notin \mathrm{EC}}^{N} \mathrm{SKM}_{ju}}{N-2}, \quad \sigma'_{\mathrm{SKM}_{ju}} = \sqrt{\frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N}\left(\mathrm{SKM}_{ju} - \overline{\mathrm{SKM}'_{ju}}\right)^2}{N-3}}. \tag{29}$$

The second rule is derived by diversifying the slack in the 2f-TNFSMCT rule, aimed at minimizing the mean cycle time:

$$\mathrm{SKV}_{ju} = \left(\frac{\lambda\beta}{(n-1)\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}\right)^{\xi} \cdot \left(\frac{j}{\lambda} - \mathrm{RCTE}_{ju} + \zeta\left(\mathrm{RCTE}_{ju} - \frac{1}{\lambda}\right)\right). \tag{30}$$

To diversify the slackness, the standard deviation of the slack is to be maximized:

$$\overline{\mathrm{SKV}'_{ju}} = \frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N} \mathrm{SKV}_{ju}}{N-2}, \tag{31}$$

$$\sigma'_{\mathrm{SKV}_{ju}} = \sqrt{\frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N}\left(\mathrm{SKV}_{ju} - \overline{\mathrm{SKV}'_{ju}}\right)^2}{N-3}}. \tag{32}$$

To generate a biobjective rule, the two rules need to be combined into a single one, for which the following nonlinear programming model is to be optimized:

$$\text{Max } Z = \omega_1 \sigma'_{\mathrm{SKM}_{ju}} + (1-\omega_1)\sigma'_{\mathrm{SKV}_{ju}} \tag{33}$$

s.t.

$$\overline{\mathrm{SKM}'_{ju}} = \frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N} \mathrm{SKM}_{ju}}{N-2}, \quad \sigma'_{\mathrm{SKM}_{ju}} = \sqrt{\frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N}\left(\mathrm{SKM}_{ju} - \overline{\mathrm{SKM}'_{ju}}\right)^2}{N-3}},$$

$$\overline{\mathrm{SKV}'_{ju}} = \frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N} \mathrm{SKV}_{ju}}{N-2}, \quad \sigma'_{\mathrm{SKV}_{ju}} = \sqrt{\frac{\sum_{j=1,\ j\notin \mathrm{EC}}^{N}\left(SKV_{ju} - \overline{SKV'_{ju}}\right)^2}{N-3}},$$

$$\mathrm{SKM}_{ju} = \left(\frac{\beta}{\alpha\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}\right)^{\xi} \cdot \left(R_j - \mathrm{RCTE}_{ju} + \zeta\left(\mathrm{RCTE}_{ju} - \min\left(R_j\right)\right)\right),$$

$$\mathrm{SKV}_{ju} = \left(\frac{\lambda\beta}{(n-1)\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}\right)^{\xi} \cdot \left(\frac{j}{\lambda} - \mathrm{RCTE}_{ju} + \zeta\left(\mathrm{RCTE}_{ju} - \frac{1}{\lambda}\right)\right),$$

$$0 \le \xi, \qquad \zeta \le 1 \tag{34}$$

which is an NP problem.

3. A Simulation Study

To evaluate the effectiveness of the proposed methodology, simulated data were used to avoid disturbing the regular

operations of the wafer fabrication factory. Simulation is a widely used technology to assess the effectiveness of a scheduling policy, especially when the proposed policy and the current practice are very different. This investigation is not possible to implement in the actual production environment. The real-time scheduling systems will input information very rapidly into the production management information systems (PROMIS). To this end, a real wafer fabrication factory located in Taichung Scientific Park of Taiwan with a monthly capacity of about 25,000 wafers was simulated. The simulation program has been validated and verified by comparing the actual cycle times with the simulated values and by analyzing the trace report, respectively. The wafer fabrication factory is producing more than 10 types of memory products and has more than 500 workstations for performing single-wafer or batch operations using 58 nm~110 nm technologies. Jobs released into the fabrication factory are assigned three types of priorities, that is, "normal," "hot," and "super hot." Jobs with the highest priorities will be processed first. Such a large scale accompanied with reentrant process flows make job dispatching in the wafer fabrication factory a very tough task. Currently, the longest average cycle time exceeds three months with a variation of more than 300 hours. The wafer fabrication factory is therefore seeking better dispatching rules to replace first-in first-out (FIFO) and EDD, in order to shorten the average cycle times and ensure the on-time delivery to its customers. One hundred replications of the simulation are successively run. The time required for each simulation replication is about 30 minute using a PC with Intel Dual CPU E2200 2.2 GHz and 1.99G RAM. A horizon of twenty-four months is simulated.

To assess the effectiveness of the proposed methodology and to make comparison with some existing approaches–FIFO, EDD, SRPT, CR, FSVCT, FSMCT, Justice [27], NFS [16], 2f-TNFSMCT, and 2f-TNFSVCT all of these methods were applied to schedule the simulated wafer fabrication factory to collect the data of 1000 jobs, and then we separated the collected data by their product types and priorities. That is about the amount of work that can be achieved with 100% of the monthly capacity. In some cases, there was too little data, so they were not discussed.

To determine the due date of a job, the PCA-FCM-BPN approach was applied to estimate the cycle time, for which the Levenberg-Marquardt algorithm rather than the gradient descent algorithm was applied to speed up the network convergence. Then, we added a constant allowance of three days to the estimated cycle time, that is, $\kappa = 72$, to determine the internal due date.

Jobs with the highest priorities are usually processed first. In FIFO, jobs were sequenced on each machine first by their priorities, then by their arrival times at the machine. In EDD, jobs were sequenced first by their priorities, then by their due dates. In CR, jobs were sequenced first by their priorities, then by their critical ratios. In the proposed methodology, the nonlinear model with $k = 2$ is used. In Justice, jobs were sequenced on each machine first by their priorities, then according to the job speed matrix (Table 2).

Table 2: The job speed matrix.

		Machine's bottleneck status		
		Hungry	Proper	Crowded
Work progress status	Behind	Rapid	Rapid	Normal
	Just in time	Rapid	Normal	Suspended
	Advanced	Normal	Normal	Suspended

Subsequently, the average cycle time and cycle time standard deviation of all cases were calculated to assess the scheduling performance. With respect to the average cycle time, the FIFO policy was used as the basis for comparison, while FSVCT was compared in evaluating cycle time standard deviation. The results are summarized in Tables 3 and 4.

According to the experimental results, the following points can be made:

(1) For the average cycle time, the proposed methodology outperformed the baseline approach, the FIFO policy. The average advantage was about 16%.

(2) In addition, the proposed methodology surpassed the FSVCT policy in reducing cycle time standard deviation. The most obvious advantage was 59%.

(3) As expected, SRPT performed well in reducing the average cycle times, especially for product types with short cycle times (e.g., product A), but might give an exceedingly bad performance with respect to cycle time standard deviation. If the cycle time is long, the remaining cycle time will be much longer than the remaining processing time, which leads to the ineffectiveness of SRPT. SRPT is similar to FSMCT. Both try to make all jobs equally early or late.

(4) The performance of EDD was also satisfactory for product types with short cycle time. If the cycle time is long, it is more likely to deviate from the prescribed internal due date, which leads to the ineffectiveness of EDD. That becomes more serious if the percentage of the product type is high in the product mix (e.g., product type A). CR has similar problems.

(5) The proposed rule was also compared with the traditional one without slack diversification. Taking product type A with normal priority as an example, the comparison results are shown in Figure 2. Obviously, the proposed rule dominated most of the traditional rules without slack diversification. According to these results, slack diversification did indeed improve the performances of the fluctuation smoothing policies.

4. Conclusions and Directions for Future Research

For capital-intensive industries like wafer fabrication, efficient use of expensive equipment is very important. To this end, job dispatching is a challenging but important task. However, for such a complex production system, to optimize the scheduling performance is a tough task.

Table 3: The performances of various approaches in the average cycle time.

Avg. cycle time (hrs)	A (normal)	A (hot)	A (super hot)	B (normal)	B (hot)
FIFO	1254	400	317	1278	426
EDD	1094	345	305	1433	438
SRPT	948	350	308	1737	457
CR	1148	355	300	1497	440
FSMCT	1313	347	293	1851	470
FSVCT	1014	382	315	1672	475
NFS	1456	407	321	1452	421
Justice	1126	378	322	1576	489
2f-TNFSMCT	1369	379	306	1361	399
2f-TNFSVCT	1465	416	318	1551	500
The proposed methodology	1076	289	269	1132	388

Table 4: The performances of various approaches in cycle time standard deviation.

Cycle time standard deviation (hrs)	A (normal)	A (hot)	A (super hot)	B (normal)	B (hot)
FIFO	55	24	25	87	51
EDD	129	25	22	50	63
SRPT	248	31	22	106	53
CR	69	29	18	58	53
FSMCT	419	33	16	129	104
FSVCT	280	37	27	201	77
NFS	87	49	19	44	47
Justice	120	26	20	69	32
2f-TNFSMCT	75	37	17	47	19
2f-TNFSVCT	38	38	29	33	24
The proposed methodology	86	26	15	54	21

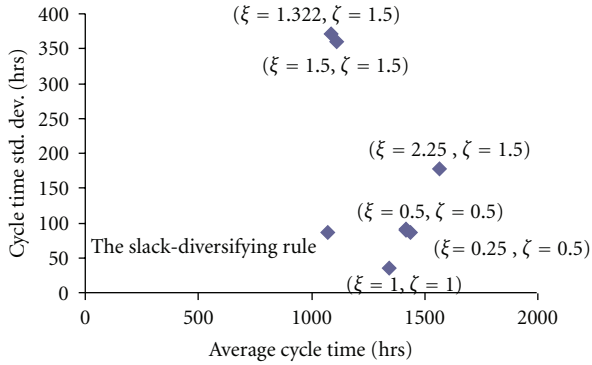

Figure 2: Comparing the slack-diversifying rule with traditional rules without slack diversification.

As an innovative attempt, this study presents a nonlinear programming and artificial neural network approach to optimize the performance of a slack-diversifying dispatching rule in a wafer fabrication factory, to optimize the average cycle time, and to optimize cycle time standard deviation.

The proposed methodology merges two existing rules—2f-TNFSMCT and 2f-TNFSVCT, and constructs a nonlinear programming model to choose the best values of parameters in the two rules. A more effective approach is also applied to estimate the remaining cycle time of a job, which is empirically shown to be conducive to the scheduling performance. To further enhance the accuracy of the remaining cycle time estimation, other dynamic parameters must be considered. In addition, some advanced methods for the cycle time estimation, such as data mining methods [28], can be applied as well.

After a simulation study, we observed the following phenomena.

(1) Through improving the accuracy of estimating the remaining cycle time, the performance of a scheduling rule can indeed be strengthened.

(2) Optimizing the adjustable factors in the two rules appears as an appropriate tool to enhance the scheduling performance of the rule.

(3) Slack diversification is indeed conducive to the performance of a fluctuation smoothing rule.

However, to further assess the effectiveness and efficiency of the proposed methodology, the only way is to apply it to an actual wafer fabrication factory. In addition, other rules can be optimized in the same way in future studies.

Acknowledgment

This work was supported by the National Science Council of Taiwan.

References

[1] C. N. Wang and C. H. Wang, "A simulated model for cycle time reduction by acquiring optimal lot size in semiconductor manufacturing," *International Journal of Advanced Manufacturing Technology*, vol. 34, no. 9-10, pp. 1008–1015, 2007.

[2] T. Chen and Y. C. Lin, "A fuzzy-neural fluctuation smoothing rule for scheduling jobs with various priorities in a miconductor manufacturing factory," *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems*, vol. 17, no. 3, pp. 397–417, 2009.

[3] T. Chen and Y. C. Wang, "A nonlinear scheduling rule incorporating fuzzy-neural remaining cycle time estimator for scheduling a semiconductor manufacturing factory-a simulation study," *International Journal of Advanced Manufacturing Technology*, vol. 45, no. 1-2, pp. 110–121, 2009.

[4] T. Chen, "Optimized fuzzy-neuro system for scheduling wafer fabrication," *Journal of Scientific and Industrial Research*, vol. 68, no. 8, pp. 680–685, 2009.

[5] D. A. Koonce and S. C. Tsai, "Using data mining to find patterns in genetic algorithm solutions to a job shop schedule," *Computers and Industrial Engineering*, vol. 38, no. 3, pp. 361–374, 2000.

[6] B. W. Hsieh, C. H. Chen, and S. C. Chang, "Scheduling semiconductor wafer fabrication by using ordinal optimization-based simulation," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 599–608, 2001.

[7] H. J. Yoon and W. Shen, "A multiagent-based decision-making system for semiconductor wafer fabrication with hard temporal constraints," *IEEE Transactions on Semiconductor Manufacturing*, vol. 21, no. 1, pp. 83–91, 2008.

[8] Y. Harrath, B. Chebel-Morello, and N. Zerhouni, "A genetic algorithm and data mining based meta-heuristic for job shop scheduling problem," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 280–285, October 2002.

[9] K. Sourirajan and R. Uzsoy, "Hybrid decomposition heuristics for solving large-scale scheduling problems in semiconductor wafer fabrication," *Journal of Scheduling*, vol. 10, no. 1, pp. 41–65, 2007.

[10] S. C. H. Lu, D. Ramaswamy, and P. R. Kumar, "Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants," *IEEE Transactions on Semiconductor Manufacturing*, vol. 7, no. 3, pp. 374–388, 1994.

[11] K. Altendorfer, B. Kabelka, and W. Stöcher, "A new dispatching rule for optimizing machine utilization at a semiconductor test field," in *Proceedings of the IEEE/SEMI Advanced Semiconductor Manufacturing Conference (ASMC '07)*, pp. 188–193, June 2007.

[12] H. Zhang, Z. Jiang, and C. Guo, "Simulation-based optimization of dispatching rules for semiconductor wafer fabrication system scheduling by the response surface methodology," *International Journal of Advanced Manufacturing Technology*, vol. 41, no. 1-2, pp. 110–121, 2009.

[13] T. Chen, "Fuzzy-neural-network-based fluctuation smoothing rule for reducing the cycle times of jobs with various priorities in a wafer fabrication plant: a simulation study," *Proceedings of the Institution of Mechanical Engineers Part B, Journal of Engineering Manufacture*, vol. 223, no. 8, pp. 1033–1043, 2009.

[14] T. Chen, "A tailored non-linear fluctuation smoothing rule for semiconductor manufacturing factory scheduling," *Proceedings of the Institution of Mechanical Engineers Part I, Journal of Systems and Control Engineering*, vol. 223, no. 2, pp. 149–160, 2009.

[15] T. Chen, "Intelligent scheduling approaches for a wafer fabrication factory," *Journal of Intelligent Manufacturing*, vol. 23, no. 3, pp. 897–911, 2012.

[16] T. Chen and Y. C. Wang, "A bi-criteria nonlinear fluctuation smoothing rule incorporating the SOM-FBPN remaining cycle time estimator for scheduling a wafer fab—a simulation study," *International Journal of Advanced Manufacturing Technology*, vol. 49, no. 5–8, pp. 709–721, 2010.

[17] T. Chen, Y.-C. Wang, and Y.-C. Lin, "A bi-criteria four-factor fluctuation smoothing rule for scheduling jobs in a wafer fabrication factory," *International Journal of Innovative Computing, Information and Control*, vol. 6, pp. 4289–4303, 2010.

[18] T. Chen, "Dynamic fuzzy-neural fluctuation smoothing rule for jobs scheduling in a wafer fabrication factory," *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, vol. 223, no. 8, pp. 1081–1094, 2009.

[19] T. Chen, "An optimized tailored nonlinear fluctuation smoothing rule for scheduling a semiconductor manufacturing factory," *Computers and Industrial Engineering*, vol. 58, no. 2, pp. 317–325, 2010.

[20] T. Chen, Y. C. Wang, and H. C. Wu, "A fuzzy-neural approach for remaining cycle time estimation in a semiconductor manufacturing factory—a simulation study," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 8, pp. 2125–2139, 2009.

[21] Y. C. Wang, T. Chen, and C. W. Lin, "A slack-diversifying nonlinear fluctuation smoothing rule for job dispatching in a wafer fabrication factory," *Robotics & Computer Integrated Manufacturing*. In press.

[22] T. Chen and T. Wang, "Enhancing scheduling performance for a wafer fabrication factory: the bi-objective slack-diversifying nonlinear fluctuation-smoothing rule," *Computational Intelligence and Neuroscience*. In press.

[23] T. Chen and M. Huang, "A fuzzy-neural slack-diversifying NFS rule for job dispatching in a wafer fabrication factory," *ICIC Express Letters*, vol. 6, no. 9, pp. 2243–2248, 2012.

[24] X. L. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 8, pp. 841–847, 1991.

[25] T. Chen and Y. C. Lin, "A fuzzy back propagation network ensemble with example classification for lot output time prediction in a wafer fab," *Applied Soft Computing Journal*, vol. 9, no. 2, pp. 658–666, 2009.

[26] T. Chen, Y. C. Wang, and H. R. Tsai, "Lot cycle time prediction in a ramping-up semiconductor manufacturing factory with a SOM-FBPN-ensemble approach with multiple buckets and partial normalization," *International Journal of Advanced Manufacturing Technology*, vol. 42, no. 11-12, pp. 1206–1216, 2009.

[27] T. Nakata, K. Matsui, Y. Miyake, and K. Nishioka, "Dynamic bottleneck control in wide variety production factory," *IEEE Transactions on Semiconductor Manufacturing*, vol. 12, no. 3, pp. 273–280, 1999.

[28] T. Chen, "Job cycle time estimation in a wafer fabrication factory with a bi-directional classifying fuzzy-neural approach," *International Journal of Advanced Manufacturing Technology*, vol. 56, pp. 1007–1018, 2011.