Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 980984, 15 pages
http://dx.doi.org/10.1155/2013/980984

Research Article

A Novel Fuzzy-Neural Slack-Diversifying Rule Based on Soft Computing Applications for Job Dispatching in a Wafer Fabrication Factory

Toly Chen and Richard Romanowski

Department of Industrial Engineering and Systems Management Feng Chia University, Taiwan No. 100, Wenhwa Rd., Seatwen, Taichung 40724, Taiwan

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

Received 7 December 2012; Revised 7 March 2013; Accepted 9 March 2013

Academic Editor: Yang Xu

This study proposes a slack-diversifying fuzzy-neural rule to improve job dispatching in a wafer fabrication factory. Several soft computing techniques, including fuzzy classification and artificial neural network prediction, have been applied in the proposed methodology. A highly effective fuzzy-neural approach is applied to estimate the remaining cycle time of a job. This research presents empirical evidence of the relationship between the estimation accuracy and the scheduling performance. Because dynamic maximization of the standard deviation of schedule slack has been shown to improve performance, this work applies such maximization to a slack-diversifying fuzzy-neural rule derived from a two-factor tailored nonlinear fluctuation smoothing rule for mean cycle time (2f-TNFSMCT). The effectiveness of the proposed rule was checked with a simulated case, which provided evidence of the rule's effectiveness. The findings in this research point to several directions that can be exploited in the future.

1. Introduction

Wafer fabrication is one of the most important steps in semiconductor manufacturing [1]. Wafer fabrication factories exist all over the world; many are in Taiwan. The production operations of a wafer fabrication factory are very expensive, and the factory must be fully utilized to stay in business. If the factory is to remain in operation, its capacity must not substantially exceed the demand. Factory managers must plan the use of the existing capacity to shorten the cycle time. To maximize the product turnover rate is an important goal. In this regard, scheduling is undoubtedly a very useful tool. Kim et al. [2] simultaneously considered three issues: release control, mask scheduling, and batch scheduling. However, Chen et al. [3–5] noted that job dispatching is very difficult task in a wafer fabrication factory. Traditionally, a scheduling problem is formulated as a mathematical programming problem. The optimal solution of the mathematical programming problem gives the optimal scheduling of the manufacturing system. However, the mathematical programming problem of scheduling a wafer fabrication factory is large and effectively intractable. In practice, many wafer fabrication factories suffer from lengthy cycle times and cannot speed up their deliveries to their customers.

Semiconductor manufacturing can be divided into four stages: wafer fabrication, wafer probing, packaging, and final testing. The most important and most time-consuming stage is wafer fabrication. This study investigates job dispatching for this stage.

An effective scheduling and dispatching algorithm is an urgent necessity for wafer fabrication. This field includes many different methods, including dispatching rules, heuristics, data mining-based approaches [6, 7], agent technologies [6, 8–10], and simulation. The prevalent methods for practical applications include 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), the fluctuation smoothing rule for cycle time variation (FSVCT), FIFO+, SRPT+, and

Table 1: The differences between the proposed methodology and Wang et al.'s method.

Method	Rule base	Degree of freedom	Exclude extreme values	Considering the range of SK_{ju}
Wang et al.'s method	1f-TNFSVCT	1	No	No
The proposed methodology	2f-TNFSMCT	2	Yes	Yes

SRPT++), all of which have received a lot of attention over the last few years [6–8]. For details on traditional dispatching rules, please refer to Lu et al. [11].

Recently, Chen proposed the one-factor tailored nonlinear fluctuation smoothing rule for mean cycle time (1f-TNFSMCT). This rule contains an adjustable parameter that allows it to be customized for a target wafer fabrication factory. Chen et al. [12] proposed the two-factor tailored nonlinear fluctuation smoothing rule for mean cycle time (2f-TNFSMCT), which outperformed four existing rules in scheduling a wafer fabrication factory.

Magnifying the difference in the slack seems to improve scheduling performance, especially with respect to the average cycle time. To exploit this advantage, Wang et al. [13] derived the slack-diversifying nonlinear fluctuation smoothing rule by diversifying the slack in the 1f-TNFSVCT rule. To extend this advantage and to enhance the scheduling of wafer fabrication factories, a slack-diversifying fuzzy-neural rule is proposed in this study. The objective function includes the average and standard deviation of the cycle time, and therefore the problem can be denoted by J/r_j, $d_j/\overline{CT_j}$, and σ_{CTj}.

The proposed methodology applies several soft computing techniques, including fuzzy classification and artificial neural network prediction. First, the remaining cycle time of a job needs to be estimated by the slack-diversifying fuzzy-neural rule. In this research, an innovative and highly effective fuzzy-neural method estimates the remaining cycle time of a job. The fuzzy-neural approach is based on the fuzzy c-means and back propagation network (FCM-BPN) approach [14]. According to Chen and Wang [4], improvements to the accuracy of remaining cycle time estimation can significantly improve the scheduling performance of a fluctuation smoothing rule. In the original study, Chen and Wang used a time-consuming and not very accurate gradient search algorithm to train the BPN. In this study, we use the Levenberg-Marquardt algorithm [15] to achieve the same purpose; it is more efficient and more accurate than the algorithm in Chen and Wang's study. In addition, we also found some empirical evidence regarding the relationship between the estimation accuracy and the scheduling performance.

The slack-diversifying nonlinear fluctuation smoothing rule is modified from 2f-TNFSMCT [12]; the new rule maximizes the difference in the slack, as measured by the standard deviation of the slack. Slack is a fuzzy concept, and in this study it is defined in a way that is conducive to scheduling performance. The factor values for achieving this must be determined, and that calculation turns out to be a complex optimization problem. We applied a polynomial fitting technique to convert it into a more tractable form for which several optimal solutions can be found. After screening some values out of the specified range, we used the remaining values to construct an optimized 2f-TNFSMCT rule. It is possible that some jobs have very large or small slacks, which will distort the optimization results. For this reason, such jobs are excluded. Further, the values of parameters influence the range of the slack. For a fair comparison, the range of the slack should be considered to determine the optimal parameter values. The differences between the proposed methodology and Wang et al.'s method are summarized in Table 1.

This paper is arranged as follows. Section 2 reviews the existing approaches to scheduling a wafer fabrication factory. Section 3 provides the details of the proposed methodology. In Section 4, a simulated case is used to validate the effectiveness of the slack-diversifying fuzzy-neural rule. The performance levels of some of the existing rules in this field are also tested with the simulated data. Finally, we draw our conclusions in Section 5 and discuss some worthwhile topics for future work.

2. Related Work

Some earlier work in this field is relevant. Mönch et al. [16] classified the scheduling problems in a semiconductor manufacturing factory into six categories: batch scheduling problems, parallel machine scheduling problems, job shop scheduling problems, scheduling problems with auxiliary resources, multiple orders per job scheduling problems, and scheduling problems related to cluster tools. In Yao et al. [17], a decentralised multiobjective scheduling methodology was presented for semiconductor manufacturing, in which global objectives were decentralised into local ones of work stations. Lee et al. [18] adopted the Petri nets to accurately model the semiconductor manufacturing activities. Through representing the token movements in a Petri net with the well-established scheduling model for batch chemical processes, the optimal schedule of the given semiconductor process could be determined accordingly. Yugma et al. [19] proposed an efficient heuristic algorithm based on iterative sampling and simulated annealing for solving a complex batching and scheduling problem in a diffusion area of a semiconductor plant. Altendorfer et al. [20] proposed the work in parallel queue (WIPQ) rule to maximize throughput with a low level of work in process (WIP). Zhang et al. [21] proposed the dynamic bottleneck detection (DBD) approach that classifies workstations into several categories and then applies different dispatching rules to these categories. They used three dispatching rules including FIFO, the shortest processing time until the next bottleneck (SPNB), and CR. In view of the uncertainty in the classification of workstations, Chen [22] proposed the fuzzy DBD approach.

Considering the current conditions in a wafer fabrication factory, Hsieh et al. [7] chose one approach from FSMCT, FSVCT, largest deviation first (LDF), one step ahead (OSA), or FIFO. Chen [23] 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. This was followed by Chen [24], 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 adjustable parameters that allow them to be customized for a target wafer fabrication factory. Dabbas and Fowler [25] and Dabbas et al. [26] combined some dispatching rules into a single rule by forming their linear combination with relative weights. However, that research lacked a systematic procedure to determine the weights of those rules. In a multiple-objective study, Chen and Wang [27] 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 enhance the performance of customizable rules. For this reason, Chen et al. [14] extended 1f-biNFS to a bi-objective fluctuation smoothing rule with four adjustable factors (4f-biNFS). 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., [14, 23, 24, 27]) performed extensive simulations. This is not only time-consuming, but it also fails to consider a sufficient number of factor combinations.

Chen [28] 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 [29] attempted to relate the scheduling performance to the factor values using a back propagation network (BPN). If that had worked, then the factor values contributing to the optimal scheduling performance could have been found. However, the explanatory ability of the BPN was not sufficient.

At the same time, Chen [28] 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 [27] 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.

3. Methodology

The variables and parameters that will be used in the proposed methodology are defined as follows:

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

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

(3) SK_{ju}: the slack of job j at step u;

(4) λ: the mean release rate;

(5) x_{jp}: inputs to the three-layer BPN of job j, $p = 1 \sim P$;

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

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

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

(9) θ_l^h: the threshold on hidden-layer node l;

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

The proposed methodology includes the following six steps.

Step 1. Normalize the collected data [30].

Step 2. Use FCM to classify jobs. The required inputs for this step are job attributes. To determine the optimal number of categories, we use the S test [31]. The output of this step is the category of each job.

Step 3. Use the BPN approach to estimate the remaining 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 attributes of a job, while the output is the estimated remaining cycle time of the job.

Step 4. Derive the 2f-TNFSMCT rule.

Step 5. Diversify the slack in the 2f-TNFSMCT rule.

Step 6. Incorporate the estimated remaining cycle time into the new rule.

The flowchart of the proposed methodology is shown in Figure 1.

Table 2 is used to compare the proposed methodology with the existing methods.

The remaining cycle time of a job being produced in a wafer fabrication factory is the time still needed to complete the job (see Figure 2). If the job has just been released into the wafer fabrication factory, then the remaining cycle time of the job is its cycle time [32–38]. Tai et al. [32] provided a statistical approach to calculate the cycle time for multilayer semiconductor final testing involving the sum of multiple Weibull-distributed waiting times. The remaining cycle time is an important attribute (or performance measure) for the WIP in the wafer fabrication factory. We need to estimate the remaining cycle time for each job because the remaining cycle time is an important input to the scheduling rule. Past studies (e.g., [12]) have shown that the accuracy of remaining cycle time estimation can be improved by job classification. Soft computing methods (e.g., [4, 12, 37, 38]) have received much attention in this regard.

3.1. Step 1: Normalize the Collected Data. First, in order to facilitate the subsequent calculations and problem solving, all collected data $\mathbf{x_j} = [x_{jp}]$ are normalized into [0.1 0.9] [38] as follows:

$$z_{jp} = N_L + \frac{x_{jp} - \min\left(x_{jp}\right)}{\max\left(x_{jp}\right) - \min\left(x_{jp}\right)} \cdot \left(N_U - N_L\right), \quad (1)$$

Table 2: Comparison of the proposed methodology and some existing methods.

Method	Static or dynamic	Number of objectives	Number of adjustable parameters	Optimized
1f-TNFSMCT	Static	1	1	No
1f-TNFSVCT	Static	1	1	No
1f-biNFS	Static	2	1	No
2f-TNFSMCT	Static	1	2	No
4f-biNFS	Static	2	4	No
Dynamic 1f-biNFS	Dynamic	2	1	No
The proposed methodology	Dynamic	2	2	Yes

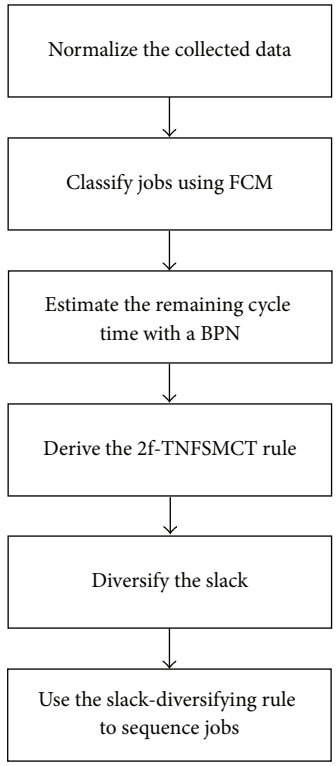

Figure 1: The flowchart of the proposed methodology.

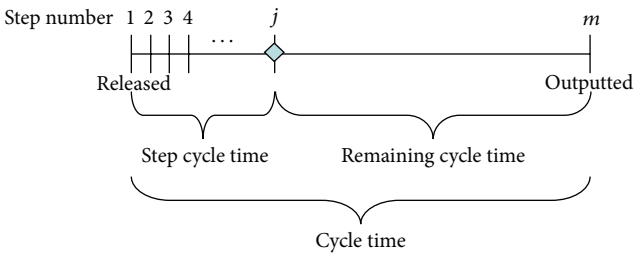

Figure 2: The concept of the remaining cycle time.

where z_{jp} is the normalized value of x_{jp} and N_L and N_U indicate the lower and upper bounds of the range of the normalized value, respectively. $\min(x_{jp})$ and $\max(x_{jp})$ are the minimum and maximum of x_j, respectively. The formula can be written as

$$x_{jp} = \frac{z_{jp} - N_L}{N_U - N_L} \cdot \left(\max\left(x_{jp}\right) - \min\left(x_{jp}\right)\right) + \min\left(x_{jp}\right), \tag{2}$$

if the unnormalized value is to be obtained. Then, we place the (normalized) attributes of a job in vector $\mathbf{z_j} = [z_{jp}]$. To illustrate the proposed methodology, a real case containing the data of 35 jobs was used. For each job, twelve attributes were collected from the reports and databases of a production management information system (PROMIS). After a backward elimination by regression analysis, six attributes that were the most influential to the job cycle time were chosen (see Table 3).

The results of partial normalization are shown in Table 4.

3.2. Step 2: Classify Jobs Using FCM. In the proposed methodology, jobs are classified into K categories using FCM. If a crisp clustering methods were applied, then it would be possible that some clusters would have very few examples. By contrast, in a fuzzy clustering method, an example belongs to multiple clusters to different degrees, which provides a solution to this problem. Similarly, in probability theory the naïve Bayes method provides the probability that a given item belongs to each class. However, the application of FCM can consider subjective issues in job classification.

FCM classifies jobs by minimizing the following objective function:

$$\text{Min} \sum_{k=1}^{K} \sum_{j=1}^{n} \mu_{j(k)}^{m} e_{j(k)}^{2}, \tag{3}$$

where K is the required number of categories; n is the number of jobs; $\mu_{j(k)}$ indicates the degree of membership with which 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) (for some number of iterations) calculate the centroid of each category as

$$\overline{z}_{(k)} = \left\{\overline{z}_{(k)p}\right\}, \quad k = 1 \sim K,$$

$$\overline{z}_{(k)p} = \frac{\sum_{j=1}^{n} \mu_{j(k)}^{m} z_{jp}}{\sum_{j=1}^{n} \mu_{j(k)}^{m}}, \quad k = 1 \sim K, p = 1 \sim P,$$

$$\mu_{j(k)} = \frac{1}{\sum_{g=1}^{K} \left(e_{j(k)}/e_{j(g)}\right)^{2/(m-1)}}, \quad j = 1 \sim n, k = 1 \sim K,$$

$$e_{j(k)} = \sqrt{\sum_{p=1}^{P} \left(z_{jp} - \overline{z}_{(k)p}\right)^2}, \quad j = 1 \sim n, k = 1 \sim K, \tag{4}$$

where $\overline{z}_{(k)}$ is the centroid of category k. $\mu_{j(k)}^{(t)}$ is the degree of membership that job j holds in 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 (3) as follows:

$$\max_k \max_j \left|\mu_{j(k)}^{(t)} - \mu_{j(k)}^{(t-1)}\right| < d, \tag{5}$$

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 [31] can be applied to determine the optimal number of categories K as follows:

$$\begin{aligned} &\min \quad S \\ &\text{subject to} \quad J_m = \sum_{k=1}^{K} \sum_{j=1}^{n} \mu_{j(k)}^m e_{j(k)}^2, \\ &\qquad e_{\min}^2 = \min_{k1 \neq k2} \left(\sum_{p=1}^{P} \left(\overline{z}_{(k1)p} - \overline{z}_{(k2)p}\right)^2 \right), \\ &\qquad S = \frac{J_m}{n \times e_{\min}^2}, \\ &\qquad K \in Z^+. \end{aligned} \tag{6}$$

The Fuzzy Logic Toolbox of MATLAB is used to implement the FCM approach. The FCM program code for the illustrative example is shown in Algorithm 1. The S value can also be obtained using this program.

The results of the S test are summarized in Table 5. In this case, the optimal number of job categories was 3. The threshold for the convergence of membership (d) was set to 0.01. A common practice is to set a threshold of membership μ_L to determine whether a job belongs to each category. For example, the classification results for the assumption that $\mu_L = 0.5$ are shown in Table 6. With each decrease in the threshold, each category will contain more jobs. For example, the classification results for the assumption that $\mu_L = 0.1$ are shown in Table 7. This can solve the problem of an insufficient number of examples. In addition, a job may be classified under more than one category, which makes FCM different from crisp classification methods. The disadvantage of this approach is that some jobs do not belong to any category, that is, the outliers. Another way is to assign a job to the category for which it has the highest degree of membership. The disadvantage of this approach is that some categories will have very few jobs.

The application of FCM has the following problems:

(1) how to determine the membership threshold μ_L. In theory, setting μ_L to $1/K$ ensures that each job can be classified under some category with absolute certainty. Conversely, if μ_L is set to a large value, it is possible that some jobs cannot be classified under any category with absolute certainty;

Table 3: An illustrative example.

Job number	u	x_{j1}	x_{j2}	x_{j3}	x_{j4}	x_{j5}	x_{j6}	RCTE$_{ju}$
1	197	96%	22	9	35	63	98	1181
2	192	97%	26	9	35	63	257	1194
3	188	96%	14	8	36	65	99	1260
4	202	97%	11	9	32	61	131	1240
5	187	97%	26	8	37	66	247	1180
6	184	97%	19	9	39	68	191	1227
7	197	96%	20	9	38	63	219	1236
8	184	96%	25	9	40	66	219	1215
9	178	97%	33	9	41	67	219	1228
10	212	96%	31	9	38	66	54	1266
11	177	96%	36	9	39	67	54	1285
12	186	96%	20	9	37	65	54	1272
13	199	96%	19	9	36	66	54	1310
14	195	96%	11	9	37	66	49	1265
15	223	97%	12	8	34	63	201	1308
16	223	96%	32	8	36	62	103	1331
17	206	95%	30	9	33	59	53	1294
18	212	95%	18	9	35	64	53	1314
19	168	97%	24	9	34	60	248	1321
20	200	97%	20	9	34	60	248	1353
21	140	95%	34	9	35	60	82	1226
22	156	96%	10	8	36	60	98	1301
23	206	96%	19	9	38	67	67	1280
24	152	96%	23	9	40	69	67	1286
25	187	95%	24	8	39	68	67	1252
26	185	97%	32	9	39	64	223	1214
27	137	97%	28	9	39	66	176	1251
28	151	98%	19	9	40	67	462	1222
29	148	97%	9	9	39	66	168	1187
30	173	96%	20	9	39	68	141	1205
31	161	96%	16	9	39	69	95	1120
32	108	97%	19	9	40	68	179	1133
33	115	96%	35	9	41	71	209	1130
34	100	96%	22	9	38	67	237	1113
35	97	99%	19	9	41	67	684	1107

```
A = [0.300 0.485 0.900 0.367 0.367 0.162; . . .; 0.900 0.396 0.900 0.900 0.633 0.900]
K = 4
[center, U, obj_fun] = fcm(A, K);
Jm = min(obj_fun)
e2_min = 9999;
for i = 1: K
  for j = i + 1:K
    e2_sum = 0;
    for p = 1: 3
      e2_sum = e2_sum + (center(i, p) − center(j, p))^ 2;
    end
    if e2_sum < e2_min
      e2_min = e2_sum;
    end
  end
end
e2_min
S = min(Jm)/(35∗e2_min)
```

Algorithm 1: The FCM program code in MATLAB.

(2) whether to consider the degree of membership in the remaining cycle time estimation. If two jobs belong to the same category but have different degrees of membership, how can one guarantee that the BPN of this category has the same predictive power for the two jobs? To address this issue, the degree of membership can be considered when estimating the remaining cycle time with the BPN.

3.3. Step 3: Estimate the Remaining Cycle Time with a BPN. After clustering, for each category, a three-layer BPN is built to estimate the remaining cycle times of jobs in the category. Some of the jobs in the category are input as "training examples" to the BPN to determine the parameter values. The configuration of the three-layer BPN is as follows. First, the inputs are the P parameters associated with the jth job. These parameters have to be normalized before they are fed into the three-layer BPN. Subsequently, there is only a single hidden layer with neurons that are twice as many as the input layer. Finally, the output from the three-layer BPN is the (normalized) remaining cycle time estimate (RCTE_j) of the example. The activation function used in each layer is the Sigmoid function,

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{7}$$

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, summed, and transferred to the hidden layer. Then activated signals are output from the hidden layer as

$$h_l = \frac{1}{1 + e^{-n_l^h}}, \tag{8}$$

where

$$\begin{aligned} n_l^h &= I_l^h - \theta_l^h, \\ I_l^h &= \sum_{p=1}^{P} w_{pl}^h x_p, \end{aligned} \tag{9}$$

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

$$o = \frac{1}{1 + e^{-n^o}}, \tag{10}$$

where

$$\begin{aligned} n^o &= I^o - \theta^o, \\ I^o &= \sum_{l=1}^{L} w_l^o h_l. \end{aligned} \tag{11}$$

Some algorithms are applicable for training a BPN in the backward phase, such as gradient descent algorithms, conjugate gradient algorithms, the Levenberg-Marquardt algorithm, and others. 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 in the following.

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

$$\mathrm{SSE}(\boldsymbol{\beta}) = \sum_{j=1}^{n} \left(N\left(\mathrm{RCTE}_{ju}\right) - f\left(\mathbf{x_j}, \boldsymbol{\beta}\right)\right)^2. \tag{12}$$

Table 4: The results of partial normalization.

Job number	z_{j1}	z_{j2}	z_{j3}	z_{j4}	z_{j5}	z_{j6}
1	0.300	0.485	0.900	0.367	0.367	0.162
2	0.500	0.604	0.900	0.367	0.367	0.362
3	0.300	0.248	0.100	0.456	0.500	0.163
4	0.500	0.159	0.900	0.100	0.233	0.203
5	0.500	0.604	0.100	0.544	0.567	0.349
6	0.500	0.396	0.900	0.722	0.700	0.279
7	0.300	0.426	0.900	0.633	0.367	0.314
8	0.300	0.574	0.900	0.811	0.567	0.314
9	0.500	0.811	0.900	0.900	0.633	0.314
10	0.300	0.752	0.900	0.633	0.567	0.106
11	0.300	0.900	0.900	0.722	0.633	0.106
12	0.300	0.426	0.900	0.544	0.500	0.106
13	0.300	0.396	0.900	0.456	0.567	0.106
14	0.300	0.159	0.900	0.544	0.567	0.100
15	0.500	0.189	0.100	0.278	0.367	0.291
16	0.300	0.781	0.100	0.456	0.300	0.168
17	0.100	0.722	0.900	0.189	0.100	0.105
18	0.100	0.367	0.900	0.367	0.433	0.105
19	0.500	0.544	0.900	0.278	0.167	0.351
20	0.500	0.426	0.900	0.278	0.167	0.351
21	0.100	0.841	0.900	0.367	0.167	0.142
22	0.300	0.130	0.100	0.456	0.167	0.162
23	0.300	0.396	0.900	0.633	0.633	0.123
24	0.300	0.515	0.900	0.811	0.767	0.123
25	0.100	0.544	0.100	0.722	0.700	0.123
26	0.500	0.781	0.900	0.722	0.433	0.319
27	0.500	0.663	0.900	0.722	0.567	0.260
28	0.700	0.396	0.900	0.811	0.633	0.620
29	0.500	0.100	0.900	0.722	0.567	0.250
30	0.300	0.426	0.900	0.722	0.700	0.216
31	0.300	0.307	0.900	0.722	0.767	0.158
32	0.500	0.396	0.900	0.811	0.700	0.264
33	0.300	0.870	0.900	0.900	0.900	0.302
34	0.300	0.485	0.900	0.633	0.633	0.337
35	0.900	0.396	0.900	0.900	0.633	0.900

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

$$f\left(\mathbf{x_j}, \boldsymbol{\beta} + \boldsymbol{\delta}\right) \approx f\left(\mathbf{x_j}, \boldsymbol{\beta}\right) + \mathbf{J_j}\boldsymbol{\delta}, \tag{13}$$

Table 5: The results of the S test.

Number of categories (K)	J_m	$e^2_{\min}$	S
2	0.0739	0.0027	0.7946
3	0.0444	0.0028	0.4586
4	0.0316	0.0014	0.6298
5	0.0247	0.0014	0.4882
6	0.0205	0.0009	0.6336
7	0.0175	0.0005	1.1102
8	0.0154	0.0003	1.2577

Table 6: The classifying results ($\mu_L = 0.5$).

Category	Jobs
1	6, 8, 9, 10, 11, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34
2	3, 5, 15, 16, 22, 25
3	1, 2, 4, 7, 12, 13, 17, 18, 19, 20, 21

Table 7: The classifying results ($\mu_L = 0.1$).

Category	Jobs
1	2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35
2	3, 4, 5, 11, 14, 15, 16, 17, 20, 21, 22, 25, 28, 29, 33, 35
3	1, 2, 4, 5, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 33, 34, 35

Table 8: Training algorithm performance (convergence criterion: MSE $< 10^{-4}$).

Algorithm	Number of epochs	MAE	MAPE	RMSE
Gradient descent	10000	79	6.7%	98
Levenberg-Marquardt	<100	47	4.1%	70

where

$$\mathbf{J_j} = \frac{\partial f\left(\mathbf{x_j}, \boldsymbol{\beta}\right)}{\partial \boldsymbol{\beta}} \tag{14}$$

is the gradient vector of f with respect to $\boldsymbol{\beta}$. Substituting (13) into (12),

$$\mathrm{SSE}\left(\boldsymbol{\beta} + \boldsymbol{\delta}\right) \approx \sum_{j=1}^{n}\left(N\left(\mathrm{RCTE}_{ju}\right) - f\left(\mathbf{x_j}, \boldsymbol{\beta}\right) - \mathbf{J_j}\boldsymbol{\delta}\right)^2. \tag{15}$$

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

$$\left(\mathbf{J^T J}\right)\boldsymbol{\delta} = \mathbf{J^T}\left(N\left(\mathrm{RCTE}_{ju}\right) - f\left(\mathbf{x_j}, \boldsymbol{\beta}\right)\right), \tag{16}$$

where $\mathbf{J}$ is the Jacobian matrix containing the first derivative of network error with respect to the weights and biases. Equation (16) includes a set of linear equations that can be solved for $\boldsymbol{\delta}$.

Finally, the three-layer BPN can be applied to estimate the remaining cycle time of a job. When a new job is released into the factory, the parameters associated with the new job are

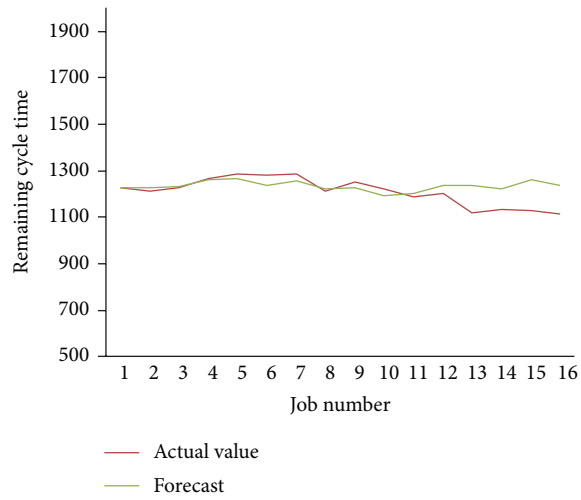

Figure 3: The estimation results by FCM-BPN.

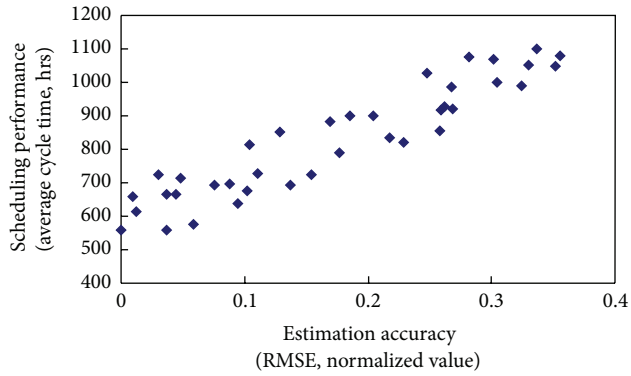

Figure 4: The relationship between the scheduling performance and the estimation accuracy.

recorded. Then the new job is classified into a category, and the three-layer BPN of the category can be applied to estimate the remaining cycle time of the new job.

Consider category 1 of the previous example. When $\mu_L = 0.5$, there are 16 jobs in this category. These jobs are split into two parts: the training data (the first 12 jobs) and the testing data (the remaining jobs). A three-layer BPN estimates the remaining cycle time of jobs in this category according to their six attributes with the following settings:

single hidden layer,

the number of neurons in the hidden layer: 12,

convergence criterion: mean squared error (MSE) $< 10^{-4}$.

The Neural Network Toolbox of MATLAB is used to implement the BPN approach. The BPN program code for the illustrative example is shown in Algorithm 2. The estimation results are shown in Figure 3.

The following indexes evaluate the estimation accuracy of both training and testing data:

mean absolute error (MAE) = 43 (hrs),

mean absolute percentage error (MAPE) = 3.7%,

RMSE = 62 (hrs).

In contrast, if the remaining cycle times of these jobs are predicted in association with those of the other unclassified jobs, then the estimation performance will be

mean absolute error (MAE) = 47 (hrs),

mean absolute percentage error (MAPE) = 4.1%,

root mean squared error (RMSE) = 70 (hrs)

which is much poorer. Table 8 compares the performance levels of the gradient descent algorithm and the Levenberg-Marquardt algorithm.

How does the accuracy of remaining cycle time estimation affect the performance of the dispatching rule? To answer this question, we add a noise factor to the remaining cycle time before feeding it to the dispatching rule as follows:

$$\mathrm{RCTE}_{ju} + \varepsilon_{ju}, \quad \varepsilon_{ju} \in N(0 \sim 200, 0 \sim 50), \tag{17}$$

where ε_{ju} follows the normal distribution. After a simulation study using FSMCT, the estimated remaining cycle time was compared with the actual value to evaluate the estimation accuracy. The relationship between the scheduling performance and the estimation accuracy is shown in Figure 4. Forty (40) replications of a simulation experiment were run with FSMCT. The scheduling performance and the estimation accuracy were evaluated with the average cycle time and RMSE, respectively. Obviously, the more accurately the remaining cycle time can be estimated, the better the schedule will perform. Therefore, the efforts made in this paper to improve the accuracy of remaining cycle time estimation make sense.

3.4. The Slack-Diversifying Fuzzy-Neural Rule. In traditional fluctuation smoothing (FS) there are two different formulation methods with two separate strengths [11]. One method is aimed at minimizing the mean cycle time with FSMCT as follows:

$$\mathrm{SK}_{ju}(\mathrm{FSMCT}) = \frac{j}{\lambda} - \mathrm{RCTE}_{ju}. \tag{18}$$

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

$$\mathrm{SK}_{ju}(\mathrm{FSVCT}) = R_j - \mathrm{RCTE}_{ju}. \tag{19}$$

Jobs with the smallest slack values are given high priority. These two rules and their variants have been proven to be very effective in shortening the cycle times of wafer fabrication factories [11, 14, 24, 27]. In the traditional FSMCT rule, RCTE_{ju} might be much greater than j/λ. As a result, the slack of a job is determined solely by RCTE_{ju}. To tackle this

```
tn_input = [0.500 0.300 0.500...; 0.396 0.574 0.811...; 0.900 0.900 0.900...; 0.722
 0.811 0.900...; 0.700 0.567 0.633...; 0.279 0.314 0.314...]
tn_target = [0.341 0.383 0.598···0.100]
net = newff([0 1; 0 1; 0 1; 0 1; 0 1; 0 1],[12, 1],{"logsig", "logsig"}, "trainlm");
net = init(net);
net.trainParam.show = 10;
net.trainParam.lr = 0.1;
net.trainParam.epochs = 1000;
net.trainParam.goal = 1e − 4;
[net, tr] = train(net, tn_input, tn_target);
tn_output = sim(net, tn_input)
te_input = [0.300 0.500 0.300...; 0.307 0.396 0.870...; 0.900 0.900 0.900...;
0.722 0.811 0.900...; 0.767 0.700 0.900...; 0.158 0.264 0.302...]
te_output = sim(net, te_input)
```

Algorithm 2: The BPN code in MATLAB (category 1).

problem, both terms in the FSMCT rule are normalized as follows:

$$\begin{aligned}\mathrm{Nor}\left(\mathrm{RCTE}_{ju}\right) &= \frac{\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}{\left(\max\left(\mathrm{RCTE}_{ju}\right) - \min\left(\mathrm{RCTE}_{ju}\right)\right)},\\ \mathrm{Nor}\left(\frac{j}{\lambda}\right) &= \frac{\left(j/\lambda - \min\left(j/\lambda\right)\right)}{\left(\max\left(j/\lambda\right) - \min\left(j/\lambda\right)\right)}\\ &= \frac{\left(j/\lambda - 1/\lambda\right)}{\left(n/\lambda - 1/\lambda\right)}\\ &= \frac{\left(j-1\right)}{\left(n-1\right)}.\end{aligned} \tag{20}$$

After normalization, both terms now range from 0 to 1. Subsequently, to improve the responsiveness of the FSMCT rule, the division operator is applied instead of the traditional subtraction operator as follows:

$$\text{nonlinear FSMCT rule: } \mathrm{SK}_{ju} = \frac{\mathrm{Nor}\left(j/\lambda\right)}{\mathrm{Nor}\left(\mathrm{RCTE}_{ju}\right)}. \tag{21}$$

However, this rule cannot be tailored to the wafer fabrication factory that is to be scheduled. To address this problem, the transition from a traditional FSMCT rule to its nonlinear form is analyzed as follows. The nonlinear form can be rewritten as

$$\begin{aligned}&\mathrm{SK}_{ju}\\ &= \frac{\mathrm{Nor}\left(j/\lambda\right)}{\mathrm{Nor}\left(\mathrm{RCTE}_{ju}\right)}\\ &= \frac{\left(j-1\right)/\left(n-1\right)}{\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)/\left(\max\left(\mathrm{RCTE}_{ju}\right) - \min\left(\mathrm{RCTE}_{ju}\right)\right)}\\ &= \frac{\beta}{\gamma}\cdot\frac{j-1}{\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)}\\ &= \frac{\beta\lambda}{\gamma}\cdot\frac{\left(j/\lambda\right) - \left(1/\lambda\right)}{\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)}\\ &= \frac{\beta\lambda}{\gamma\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}\\ &\quad\cdot\left(\frac{j}{\lambda} - \mathrm{RCTE}_{ju} + \mathrm{RCTE}_{ju} - \frac{1}{\lambda}\right)\\ &= \left(\frac{\gamma\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}{\beta\lambda}\right)^{-1}\\ &\quad\cdot\left(\frac{j}{\lambda} - \mathrm{RCTE}_{ju} + \left(\mathrm{RCTE}_{ju} - \min\left(\frac{j}{\lambda}\right)\right)\cdot 1\right),\end{aligned} \tag{22}$$

where $\beta = \max(\mathrm{RCTE}_{ju}) - \min(\mathrm{RCTE}_{ju})$ and $\gamma = n - 1$. Conversely, the linear form can also be rewritten as

$$\begin{aligned}\mathrm{SK}_{ju} &= R_j - \mathrm{RCTE}_{ju}\\ &= \left(\frac{\gamma\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}{\beta\lambda}\right)^{-0}\\ &\quad\cdot\left(\frac{j}{\lambda} - \mathrm{RCTE}_{ju} + \left(\mathrm{RCTE}_{ju} - \min\left(\frac{j}{\lambda}\right)\right)\cdot 0\right).\end{aligned} \tag{23}$$

These two formulas can be generalized into the following form:

$$\begin{aligned}\mathrm{SK}_{ju} &= \left(\frac{\gamma\left(\mathrm{RCTE}_{ju} - \min\left(\mathrm{RCTE}_{ju}\right)\right)}{\beta\lambda}\right)^{-\xi}\\ &\quad\cdot\left(\frac{j}{\lambda} - \mathrm{RCTE}_{ju} + \left(\mathrm{RCTE}_{ju} - \min\left(\frac{j}{\lambda}\right)\right)\cdot\zeta\right).\end{aligned} \tag{24}$$

Wang et al. derived [13] the slack-diversifying nonlinear fluctuation smoothing rule by diversifying the slack in the 1f-TNFSVCT rule. In this study, we diversify the slack in the 2f-TNFSMCT rule as follows:

$$\begin{aligned}\mathrm{SK}_{ju} &= \left(\frac{\beta\lambda}{\gamma\left(\mathrm{RCTE}_{ju}-\min\left(\mathrm{RCTE}_{ju}\right)\right)}\right)^{\xi} \\ &\quad\cdot\left(\frac{j}{\lambda}-\mathrm{RCTE}_{ju}+\zeta\left(\mathrm{RCTE}_{ju}-\frac{1}{\lambda}\right)\right) \\ &= a_{ju}^{\xi}\left(b_{ju}+c_{ju}\zeta\right) \\ &= b_{ju}a_{ju}^{\xi}+c_{ju}\zeta a_{ju}^{\xi},\end{aligned} \tag{25}$$

where

$$\begin{aligned}a_{ju} &= \frac{\beta\lambda}{\gamma\left(\mathrm{RCTE}_{ju}-\min\left(\mathrm{RCTE}_{ju}\right)\right)}, \\ b_{ju} &= \frac{j}{\lambda}-\mathrm{RCTE}_{ju}, \\ c_{ju} &= \mathrm{RCTE}_{ju}-\frac{1}{\lambda}.\end{aligned} \tag{26}$$

There are many possible models that can form the combination of ξ and ζ. For example,

$$(\text{Linear model})\ \xi=\zeta, \tag{27}$$

$$(\text{Nonlinear model})\ \xi=\zeta^{k},\quad k\ \geq\ 0, \tag{28}$$

$$(\text{Logarithmic model})\ \xi=\frac{\ln(1+\zeta)}{\ln 2}. \tag{29}$$

However, (25) is difficult to deal with. For this reason, the following polynomial fitting techniques are used to convert it into a more tractable form:

$$x^{\xi}\cong(1.15-0.11\xi)+(-0.07+0.57\xi)\,x. \tag{30}$$

The mean absolute percentage error (RMSE) of (30) is less than 9% when $x\leq 10$. RMSE will not be a serious problem, since it is the ξ value associated with the maximum $\sigma_{\mathrm{SK}_{ju}}$ that is to be found, not the SK_{ju} value. These polynomial fitting techniques are especially effective when x exceeds 1.

Applying (30) to (25) gives

$$\begin{aligned}\mathrm{SK}_{ju} &\cong b_{ju}\left(1.15-0.11\xi+(-0.07+0.57\xi)\,a_{ju}\right) \\ &\quad+c_{ju}\zeta\left(1.15-0.11\xi+(-0.07+0.57\xi)\,a_{ju}\right) \\ &= \left(1.15b_{ju}-0.07a_{ju}b_{ju}\right) \\ &\quad+\left(-0.11b_{ju}+0.57a_{ju}b_{ju}\right)\xi \\ &\quad+\left(1.15c_{ju}-0.07a_{ju}c_{ju}\right)\zeta \\ &\quad+\left(-0.11c_{ju}+0.57a_{ju}c_{ju}\right)\xi\zeta \\ &= d_{ju}+e_{ju}\xi+f_{ju}\zeta+g_{ju}\xi\zeta,\end{aligned} \tag{31}$$

where

$$\begin{aligned}d_{ju} &= 1.15b_{ju}-0.07a_{ju}b_{ju}, \\ e_{ju} &= -0.11b_{ju}+0.57a_{ju}b_{ju}, \\ f_{ju} &= 1.15c_{ju}-0.07a_{ju}c_{ju}, \\ g_{ju} &= -0.11c_{ju}+0.57a_{ju}c_{ju}.\end{aligned} \tag{32}$$

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

$$\begin{aligned}\sigma_{\mathrm{SK}_{ju}} &= \sqrt{\frac{\sum_{j=1}^{n}\left(\mathrm{SK}_{ju}-\overline{\mathrm{SK}_{ju}}\right)^{2}}{n-1}} \\ &= \sqrt{\frac{1}{(n-1)}}\sqrt{\sum_{j=1}^{n}\mathrm{SK}_{ju}^{2}-\frac{1}{n}\left(\sum_{j=1}^{n}\mathrm{SK}_{ju}\right)^{2}}\end{aligned} \tag{33}$$

which is equivalent to maximizing the following term:

$$\begin{aligned}&\sum_{j=1}^{n}\mathrm{SK}_{ij}^{2}-\frac{1}{n}\left(\sum_{j=1}^{n}\mathrm{SK}_{ij}\right)^{2} \\ &\quad=\sum_{j=1}^{n}\left(d_{ju}+e_{ju}\xi+f_{ju}\zeta+g_{ju}\xi\zeta\right)^{2} \\ &\qquad-\frac{1}{n}\left(\sum_{i=1}^{n}\left(d_{ju}+e_{ju}\xi+f_{ju}\zeta+g_{ju}\xi\zeta\right)\right)^{2} \\ &\quad=w_{1}+w_{2}\zeta+w_{3}\xi+w_{4}\zeta^{2}+w_{5}\xi^{2} \\ &\qquad+w_{6}\zeta\xi+w_{7}\zeta\xi^{2}+w_{8}\zeta^{2}\xi.\end{aligned} \tag{34}$$

By taking the derivatives of (34) with respect to ζ and ξ and setting them equal to zero, we can obtain

$$w_{2}+2w_{4}\zeta+w_{6}\xi+w_{7}\xi^{2}+2w_{8}\zeta\xi=0, \tag{35}$$

$$w_{3}+2w_{5}\xi+w_{6}\zeta+2w_{7}\zeta\xi+w_{8}\zeta^{2}=0. \tag{36}$$

In addition, ζ and ξ must follow some model, for example, (27), (28), or (29). For example, if the nonlinear model in (28) is satisfied ($k=2$), then according to (35),

$$w_{2}+2w_{4}\zeta+w_{6}\zeta^{2}+2w_{8}\zeta^{3}+w_{7}\zeta^{4}=0;\quad \xi=\zeta^{2}, \tag{37}$$

while according to (36)

$$w_{3}+w_{6}\zeta+\left(2w_{5}+w_{8}\right)\zeta^{2}+2w_{7}\zeta^{3}=0;\quad \xi=\zeta^{2}. \tag{38}$$

The general solutions are too long to be presented here. Further, to guarantee a maximum, the second-order derivative must be negative as follows:

$$\begin{aligned}2w_{4}+2w_{8}\xi &\leq 0, \\ 2w_{5}+2w_{7}\zeta &\leq 0.\end{aligned} \tag{39}$$

Table 9: A demonstrative example.

j	R_j	RCTE_{ju}
1	158	54
2	154	109
3	161	118
4	131	52
5	162	198
6	248	25
7	220	11
8	87	106
9	22	31
10	178	118
11	229	95
12	148	105
13	8	8
14	228	55
15	242	132
16	99	51
17	4	3
18	246	154
19	191	157
20	131	148

In addition, it is possible that some jobs have very large or very small slacks, either of which will distort the calculation of $\sigma_{\mathrm{SK}_{ju}}$. For this reason, such jobs are excluded. Further, the values of ξ and ζ influence the range of SK_{ju}. For a fair comparison, $\sigma_{\mathrm{SK}_{ju}}$ should be divided by the range of SK_{ju} to determine the optimal values of ξ and ζ.

An example is given in Table 9 to illustrate the procedure mentioned above, in which $\alpha = 244$, $\beta = 195$, and $\lambda = 1.18$. Assume that the nonlinear model with $k = 2$ is used for the relationship between ξ and ζ. The optimal solution is $\xi^* = 0.53$ and $\zeta^* = 0.28$ with the maximum $\sigma_{\mathrm{SK}_{ju}}$ equal to 9.21. In Figure 5 the results of the proposed methodology are compared with those of Wang et al.'s method. Obviously, the proposed methodology performed better at diversifying the slacks of jobs, which reduces the risk of misscheduling and promotes scheduling performance.

4. Numerical Simulation and Results

To evaluate the effectiveness of the slack-diversifying fuzzy-neural rule for job dispatching in a wafer fabrication factory, simulated data were used to avoid disturbing the regular operations of the wafer fabrication factory. Simulation is a widely used technology that can assess the effectiveness of a scheduling policy, especially when the proposed policy and the current practice are very different. The actual production environment is dedicated to make real products and is not available for algorithm testing. Therefore, a real wafer fabrication factory located in Taichung Scientific Park of Taiwan with a monthly capacity of about 25,000 wafers was simulated. That factory's real time scheduling systems input information very rapidly into its production management information systems (PROMIS). 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 produces 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. Each job released into the fabrication factory is assigned one of three priorities, that is, "normal," "hot," and "super hot." Jobs with the highest priorities will be processed first. The large scale of operations with reentrant process flows makes 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 factory managers seek better dispatching rules to replace first-in first-out (FIFO) and EDD, in order to shorten the average cycle times and ensure on time deliveries to customers. One hundred replications of the simulation were successively run. The time required for each simulation replication was about 30 minutes using a PC with Intel Dual CPU E2200 2.2 GHz and 1.99G RAM. A horizon of twenty-four months was simulated.

To assess the effectiveness of the proposed methodology and to make comparisons with some existing approaches—FIFO, EDD, shortest remaining processing time (SRPT), CR, FSVCT, FSMCT, and a nonlinear fluctuation smoothing rule (NFS) [4]—all of these methods were applied to schedule the simulated wafer fabrication factory with the data of 1000 jobs; the collected data were then separated by product types and priorities. That is about the amount of work that can be achieved with 100% of the monthly capacity. Some cases produced insufficient data, so the present paper does not discuss those cases.

The due date of each job was determined as follows. The FCM-BPN approach was applied to estimate the cycle time; 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 as follows:

$$\mathrm{DD}_j = \mathrm{CTE}_j + 72. \tag{40}$$

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. With SRPT, the remaining processing time of each job was calculated. Then, jobs were sequenced first by their priorities, then by their remaining processing times. With CR, jobs were sequenced first by their priorities, then by their critical ratios as follows:

$$\mathrm{CR}_{ju} = \frac{\left(\mathrm{DD}_j - t\right)}{\mathrm{RPT}_j}. \tag{41}$$

FSVCT and FSMCT consisted of two stages. The first stage scheduled jobs by FIFO; the remaining cycle times of all jobs were recorded and averaged at each step. The second

Table 10: The performance of various approaches in the average cycle time.

Avg. cycle time (hrs)	A (normal)	A (hot)	A (super hot)	B (normal)	B (hot)
FIFO	1256	401	320	1278	457
EDD	1087	346	306	1433	478
SRPT	966	350	309	1737	483
CR	1143	356	301	1497	470
FSMCT	1401	405	320	1408	430
FSVCT	1046	385	317	1745	519
NFS	1456	407	321	1452	421
The proposed methodology	1134	295	278	1208	399

Table 11: The performance of various approaches in cycle time standard deviation.

Cycle time std. dev. (hrs)	A (normal)	A (hot)	A (super hot)	B (normal)	B (hot)
FIFO	56	24	23	87	40
EDD	130	25	23	50	39
SRPT	246	32	23	106	30
CR	68	30	19	58	37
FSMCT	42	44	23	35	28
FSVCT	319	35	28	222	55
biNFS	87	49	19	44	47
The proposed methodology	89	26	15	55	20

stage applied FSVCT or FSMCT policy to schedule the jobs based on the average remaining cycle times obtained earlier. In other words, jobs were sequenced on each machine first by their priorities, and then by their slack values, which were determined by (18) and (19). In NFS, the slack of a job was defined as follows:

$$\mathrm{SK}_{ju} = \left(\frac{R_j - \min R_j}{\max R_j - \min R_j}\right)^{\xi} \cdot \frac{((j-1)/(n-1))^{1-\xi}}{\left(\left(\mathrm{RCTE}_{ju} - \min \mathrm{RCTE}_{ju}\right) / \left(\max \mathrm{RCTE}_{ju} - \min \mathrm{RCTE}_{ju}\right)\right)}, \quad (42)$$

where $\xi \in [0\ \ 1]$. After a preliminary experiment, ξ was set to 0.8. In the proposed methodology, the nonlinear model with $k = 2$ was used.

Subsequently, the average cycle time and cycle time standard deviation of all cases were calculated to assess the scheduling performance. Comparisons for the average cycle time used FSMCT policy; comparisons for cycle time standard deviation used FSVCT. The results are summarized in Tables 10 and 11. Both products A and B are dynamic random access memory products. The main difference between the two products is their storage capacities.

(1) For the average cycle time, the proposed methodology outperformed the baseline approach, FSMCT policy. The average advantage was about 17%.

(2) In addition, the proposed methodology surpassed the FSVCT policy in reducing cycle time standard deviation. The most obvious advantage was 74%.

(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 sometimes gave 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; this makes SRPT ineffective. SRPT is similar to FSMCT. Both try to make all jobs equally early or late.

(4) The performance of EDD was satisfactory for product types with short cycle times. If the cycle time is long, it is more likely to deviate from the prescribed internal due date, which makes EDD ineffective. That becomes 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 Wilcoxon signed-rank test [39], a commonly used nonparametric statistical hypothesis test for comparisons of two related samples or repeated measurements on a single sample, was used in this study to assess whether population means differed or not. The results are summarized in Table 12. The null hypothesis H_a was rejected at $\alpha = 0.025$, which showed that the slack-diversifying fluctuation-smoothing rule was statistically superior to four existing approaches to reduce the average cycle time. With regard to cycle time standard deviation, the advantage of the slack-diversifying fluctuation-smoothing rule over SRPT and FSVCT was significant.

(6) The slack-diversifying 2f-TNFSMCT rule was also compared with the traditional one without slack diversification. Taking product type A with normal

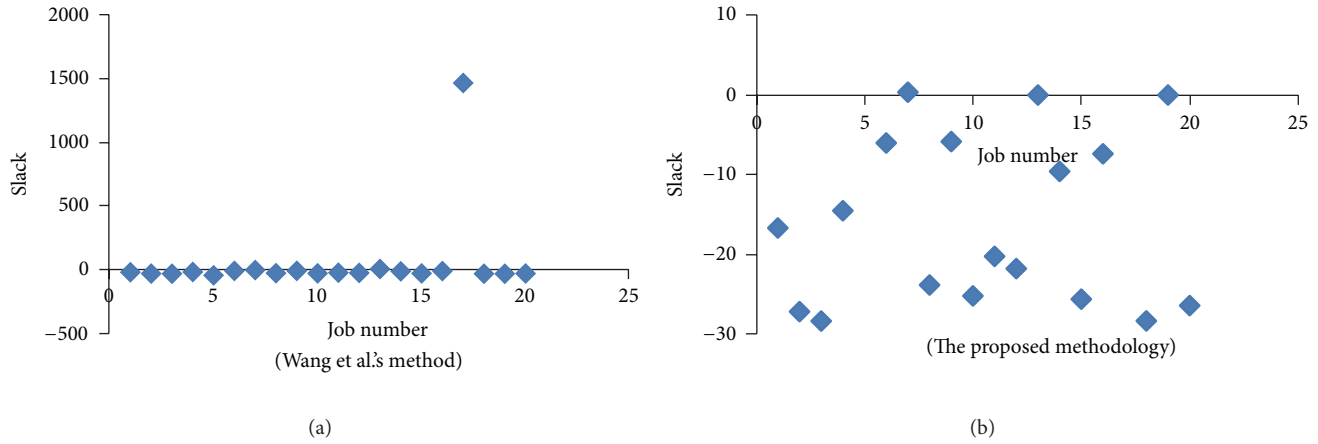

Figure 5: Comparison of the results by the two methods.

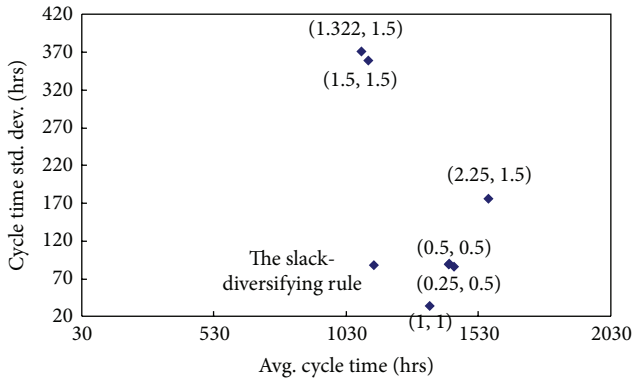

Figure 6: Comparing the slack-diversifying rule with traditional rules without slack diversification.

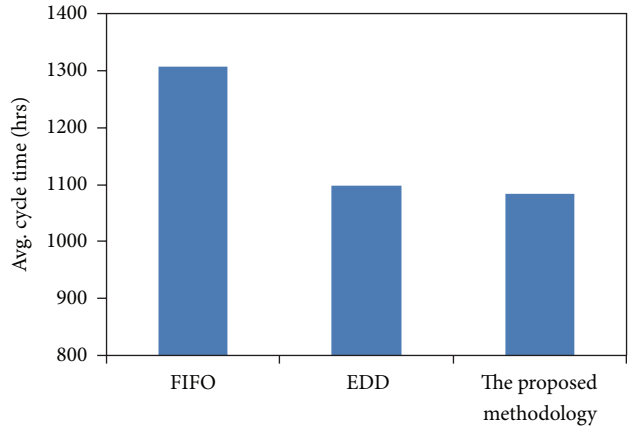

Figure 7: The performances of various rules in the average cycle time.

priority as an example, the comparison results are shown in Figure 6. Obviously, the slack-diversifying rules dominated most of the traditional rules without slack diversification. According to these results, the treatments carried out in this study did indeed improve on the performance levels of the traditional policies.

(7) The proposed methodology can be decomposed into two parts. In the first part, the remaining cycle time of a new job is estimated using FCM-BPN, based on the historical data retrieved from PROMIS. In the experiment, it took less than 2 minutes to run all MATLAB programs and generate the remaining cycle time estimate. In contrast, Chen and Wang's method used the gradient search algorithm, and it took about 15 minutes to train the BPN. In the second part, the slacks of all jobs are calculated according to the slack-diversifying 2f-TNFSMCT rule. That can be done almost instantly.

The slack-diversifying 2f-TNFSMCT rule has been proven effective in real world situations because it has been applied to schedule a wafer fabrication factory. However, it has not been possible to experiment with a large assortment of scheduling policies in that wafer fabrication factory. In the past, FIFO and EDD have been applied to this wafer fabrication factory, and their performance statistics have been analyzed. For these reasons, the performance level of

Table 12: Results of the Wilcoxon signed-rank test.

	H_{a0} (the average cycle time)	H_{b0} (cycle time standard deviation)
FIFO	2.02**	0.40
EDD	1.48	1.21
SRPT	0.94	2.02**
CR	2.02**	0.54
FSMCT	2.02**	−0.54
FSVCT	1.48	2.02**
biNFS	2.02**	0.94

*$P < 0.05$.
**$P < 0.025$.
***$P < 0.01$.

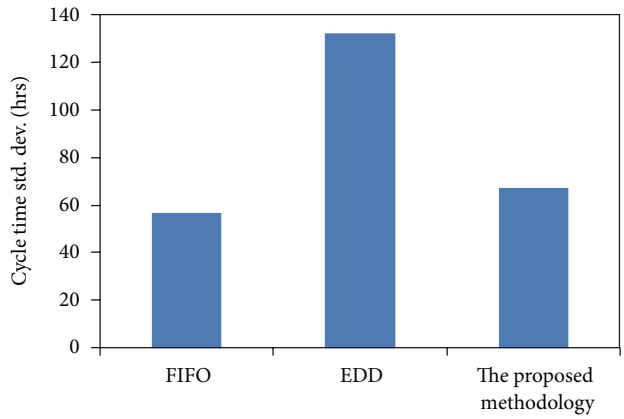

Figure 8: The performances of various rules in cycle time standard deviation.

the slack-diversifying 2f-TNFSMCT rule was compared with those of FIFO and EDD. For example, consider the case in which the majority of the factory's capacity is occupied. Figures 7 and 8 compare the performance levels of the three approaches. With regard to the average cycle time, the slack-diversifying 2f-TNFSMCT rule surpassed the two old rules. The performance of the slack-diversifying 2f-TNFSMCT rule was also comparable to that of FIFO for reducing cycle time standard deviation. Both results supported the practicability of the proposed slack-diversifying 2f-TNFSMCT rule.

5. Conclusions and Directions for Future Research

To optimize job dispatching in a wafer fabrication factory is a challenging but important task. This study attempts to innovate by a slack-diversifying fuzzy neural rule that optimizes the average cycle time and cycle time standard deviation for job dispatching in a wafer fabrication factory.

The proposed methodology applies soft computing techniques like fuzzy classification and artificial neural network prediction. An effective fuzzy-neural approach is applied to estimate the remaining cycle time of a job, which is empirically shown to ameliorate scheduling performance. This slack-diversifying fuzzy-neural rule is a modification of the 2f-TNFSMCT rule.

After a simulation study, we observed the following phenomena:

(1) through improving the accuracy of remaining cycle time estimation, the performance of a scheduling rule can indeed be strengthened;

(2) optimizing the adjustable factors in the 2f-TNFSMCT rule appears to be an appropriate tool to enhance the scheduling performance of the rule.

However, a complete assessment of the effectiveness and efficiency of the proposed methodology requires actual application in a real-world wafer fabrication factory. Future studies can optimize other rules in the same way, taking into account the great uncertainty inherent in wafer fabrication systems [40] and possibly using a more effective data mining approach [41].

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] Y.-D. Kim, D.-H. Lee, J.-U. Kim, and H.-K. Roh, "A simulation study on lot release control, mask scheduling, and batch scheduling in semiconductor wafer fabrication facilities," *Journal of Manufacturing Systems*, vol. 17, no. 2, pp. 107–117, 1998.

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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*, vol. 7, pp. 280–285, October 2002.

[10] 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.

[11] 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.

[12] T. Chen, Y.-C. Wang, and Y.-C. Lin, "A fuzzy-neural system for scheduling a wafer fabrication factory," *International Journal of Innovative Computing, Information and Control*, vol. 6, no. 2, pp. 687–700, 2010.

[13] 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 and Computer Integrated Manufacturing*, vol. 29, no. 3, pp. 41–47, 2013.

[14] 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.

[15] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, Berlin, Germany, 2nd edition, 2006.

[16] L. Mönch, J. W. Fowler, S. Dauzère-Pérès, S. J. Mason, and O. Rose, "A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations," *Journal of Scheduling*, vol. 14, no. 6, pp. 583–599, 2011.

[17] S. Yao, Z. Jiang, N. Li, N. Geng, and X. Liu, "A decentralised multi-objective scheduling methodology for semiconductor manufacturing," *International Journal of Production Research*, vol. 49, no. 24, pp. 7227–7252, 2011.

[18] Y.-H. Lee, C.-T. Chang, D. S.-H. Wong, and S.-S. Jang, "Petri-net based scheduling strategy for semiconductor manufacturing processes," *Chemical Engineering Research and Design*, vol. 89, no. 3, pp. 291–300, 2011.

[19] C. Yugma, S. Dauzère-Pérès, C. Artigues, A. Derreumaux, and O. Sibille, "A batching and scheduling algorithm for the diffusion area in semiconductor manufacturing," *International Journal of Production Research*, vol. 50, no. 8, pp. 2118–2132, 2012.

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

[21] 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.

[22] T. Chen, "A fuzzy-neural DBD approach for job scheduling in a wafer fabrication factory," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 6, pp. 4024–4044, 2012.

[23] 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," *Journal of Engineering Manufacture* , vol. 223, no. 8, pp. 1033–1043, 2009.

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

[25] R. M. Dabbas and J. W. Fowler, "A new scheduling approach using combined dispatching criteria in wafer fabs," *IEEE Transactions on Semiconductor Manufacturing*, vol. 16, no. 3, pp. 501–510, 2003.

[26] R. M. Dabbas, H.-N. Chen, J. W. Fowler, and D. Shunk, "A combined dispatching criteria approach to scheduling semiconductor manufacturing systems," *Computers and Industrial Engineering*, vol. 39, no. 3-4, pp. 307–324, 2001.

[27] 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.

[28] T. Chen, "Dynamic fuzzy-neural fluctuation smoothing rule for jobs scheduling in a wafer fabrication factory," *Journal of Systems and Control Engineering*, vol. 223, no. 8, pp. 1081–1094, 2009.

[29] 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.

[30] 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.

[31] 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.

[32] Y. T. Tai, W. L. Pearn, and J. H. Lee, "Cycle time estimation for semiconductor final testing processes with Weibull-distributed waiting time," *International Journal of Production Research*, vol. 50, no. 2, pp. 581–592, 2012.

[33] T. Chen, "A hybrid SOM-BPN approach to lot output time prediction in a wafer fab," *Neural Processing Letters*, vol. 24, no. 3, pp. 271–288, 2006.

[34] T. Chen, "An intelligent hybrid system for wafer lot output time prediction," *Advanced Engineering Informatics*, vol. 21, no. 1, pp. 55–65, 2007.

[35] T. Chen, "An intelligent mechanism for lot output time prediction and achievability evaluation in a wafer fab," *Computers and Industrial Engineering*, vol. 54, no. 1, pp. 77–94, 2008.

[36] T. Chen, "A SOM-FBPN-ensemble approach with error feedback to adjust classification for wafer-lot completion time prediction," *International Journal of Advanced Manufacturing Technology*, vol. 37, no. 7-8, pp. 782–792, 2008.

[37] 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.

[38] 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.

[39] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[40] X. Tian and C. M. Li, "Guaranteed cost control of linear uncertain time-delay switched singular systems based on an LMI approach," *International Journal of Modelling, Identification and Control*, vol. 17, no. 2, pp. 109–115, 2012.

[41] H. Meng, X. L. Wu, X. Lv, and Q. Han, "Improved observer-based reliable guaranteed cost control for neutral systems," *International Journal of Modelling, Identification and Control*, vol. 10, no. 3-4, pp. 250–255, 2010.