Int J Adv Manuf Technol (2009) 45:110–121
DOI 10.1007/s00170-009-1941-y

ORIGINAL ARTICLE

A nonlinear scheduling rule incorporating fuzzy-neural remaining cycle time estimator for scheduling a semiconductor manufacturing factory—a simulation study

Toly Chen · Yi-Chi Wang

Received: 31 March 2008 / Accepted: 19 January 2009 / Published online: 12 February 2009
© Springer-Verlag London Limited 2009

Abstract A nonlinear scheduling rule incorporating a fuzzy-neural remaining cycle time estimator is proposed in this study to improve scheduling performance in a semiconductor manufacturing factory. The proposed scheduling rule is modified from the well-known fluctuation smoothing rule with three treatments. At first, the look-ahead self-organization map-fuzzy back-propagation network approach in our previous study is used to estimate the remaining cycle time of every job in the semiconductor manufacturing factory. Subsequently, the release time and remaining cycle time of a job are both normalized to balance their importance in the fluctuation smoothing rule. Finally, the normalized release time is divided by the normalized remaining cycle time to obtain the slack. In this way, the proposed scheduling rule becomes a nonlinear one. To evaluate the effectiveness of the proposed methodology, production simulation is used to generate some test data. According to experimental results, the proposed methodology outperformed many existing approaches in reducing both the average cycle times and cycle time standard deviations. The advantage was up to 41% over the basis p-FS policy when the cycle time standard deviations were to be minimized.

Keywords Nonlinear · Scheduling · Fluctuation smoothing · Fuzzy-neural · Remaining cycle time · Simulation · Semiconductor manufacturing

T. Chen (✉) · Y.-C. Wang
Department of Industrial Engineering and Systems Management, Feng Chia University,
No. 100, Wenhwa Rd., Seatwen,
Taichung City, Taiwan
e-mail: tolychen@ms37.hinet.net

1 Introduction

A semiconductor manufacturing factory usually consists of several complicated production systems. According to Gupta and Sivakumar [11], a semiconductor manufacturing factory can be viewed as a highly complex job shop. Typical characteristics of a semiconductor manufacturing factory include: fluctuating demand, jobs with various product types and priorities, unbalanced capacity, jobs' reentrance to machines, hundreds of processing steps, alternative machines with unequal capacity, shifting bottlenecks, etc. Scheduling then becomes a nightmare for the managers of a semiconductor manufacturing factory. Wein [20] demonstrated that a good job release policy led to a significant reduction on the average cycle time, while many semiconductor manufacturing factories (especially foundry factories) had to release the jobs right after the order was received and could not adopt the predetermined job release policy. Besides, many studies, e.g., [19, 17], have shown that applying the general scheduling rules [such as first-in first-out (FIFO), earliest due date (EDD), least slack (LS), shortest processing time (SPT), shortest remaining processing time (SRPT), shortest setup plus processing time (SSPT), cost over time (COVERT), apparent tardiness cost with setups (ATCS), time in system (TIS), work in next queue (WINQ), critical ratio (CR), FIFO+, SRPT+, and SRPT++)] to a semiconductor manufacturing factory did not lead to very satisfactory results. Nevertheless, the research focusing on scheduling a semiconductor manufacturing factory has become a very important issue at present [11]. However, the existing approaches have the following problems:

1. Most scheduling rules in this field consider only a few attributes of the jobs gathered at the same place and

lack an effective way of taking the whole factory into consideration.

2. Most scheduling approaches are deterministic and cannot reflect to the dynamic changes in the semiconductor manufacturing factory. Though there are some scheduling rules incorporating stochastic variables, such as the fluctuation smoothing (FS) rules, fluctuation smoothing policy for variance of cycle time (FSVCT), and fluctuation smoothing policy for mean cycle time (FSMCT), they use the average values of these variables and are in fact not responsive to environmental changes. Another example of dynamic/real-time scheduling is the dynamic bottleneck dispatching rule proposed by Zhang et al. [22].
3. Most data-mining-based approaches were developed for a relatively small-sized manufacturing system. Besides, these approaches attempted to simulate the best practices in the past for future scheduling applications. However, a semiconductor manufacturing factory is a highly dynamic environment in which the future conditions might be very different from those in the past. It is also very difficult to find out the so-called best practices for such a highly dynamic and complicated manufacturing system.

To solve these problems and to further improve the performance of scheduling jobs in a semiconductor manufacturing factory, a nonlinear scheduling rule incorporating fuzzy-neural remaining cycle time estimator is proposed in this study. The proposed methodology is modified from the well-known FS rules with the following treatment:

1. Improve the accuracy of estimating the remaining cycle time of a job with the look-ahead self-organization map (SOM) and fuzzy back-propagation network (FBPN) approach.
2. Smooth the fluctuation in the estimated remaining cycle time and balance its importance with that of the release time.
3. Magnify the difference in the slack.

The remainder of this paper is organized as follows. A literature review is given in Section 2. Section 3 is divided into two parts. The first one reviews the look-ahead SOM and FBPN approach. Then, it is applied to estimate both the cycle time and the step cycle time so as to derive the remaining cycle time, which is described in the second part. Finally, the estimated remaining cycle time and the release time of a job are normalized in order to calculate the slack of the job in a new manner. To evaluate the effectiveness of the proposed methodology, a simulation model simplified from an actual semiconductor manufacturing factory is constructed in Section 4 to generate some test data. Based on analysis results, some discussion points are made in Section 5. Finally, the concluding remarks and some directions for future research are given in Section 6.

2 Literature review

Parameters that will be used in this study are defined as follows:

1. CT_n: the cycle time (actual value) of job n;
2. CTE_n: the estimated cycle time of job n;
3. SCTE_{nj}: the estimated step cycle time of job n at step j;
4. SCT_{nj}: the step cycle time (actual value) of job n at step j;
5. RCTE_{nj}: the estimated remaining cycle time of job n at step j;
6. RCT_{nj}: the remaining cycle time (actual value) of job n at step j;
7. R_n: the release time of job n;
8. U_n: the average factory utilization at R_n;
9. Q_n: the total queue length on the processing route of job n at R_n;
10. BQ_n: the total queue length before bottlenecks at R_n;
11. FQ_n: the total queue length in the whole factory at R_n;
12. WIP_n: the amount of work-in-progress (WIP) in the factory at R_n;
13. $D_n^{(r)}$: the delay (waiting time) of the r-th recently completed job at R_n; r=1–3;
14. $FDW_n^{(f)}$: the future discounted workload on the processing route of job n at R_n [14]; f=1–3;
15. SK_n: the slack of job n;
16. E_n: the estimation error of job n;
17. ER_n: the estimation error rate of job n; and
18. λ: the mean release rate.

Gupta and Sivakumar [11] classified the existing scheduling approaches for a semiconductor manufacturing factory into four categories: heuristic rules, mathematical programming techniques, neighborhood search methods, and artificial intelligence techniques. Most of the existing scheduling rules are “deterministic,” namely, the information used in these scheduling rules (e.g., the release time, the total processing time, and the due date of a job) do not change over time. Conversely, Lu et al. [16] proposed two “stochastic” scheduling rules, FSVCT and FSMCT, in which RCT_{nj} was considered and therefore needed to be estimated. RCT_{nj} is a stochastic variable since it is highly dependent not only on the factory conditions but also on the progresses of the other jobs as well. Another useful information when implementing the scheduling rules is the remaining processing time. Scheduling rules considering the remaining processing time include CR and LS [17–18]. The remaining processing time is known, while RCT_{nj}

needs to be estimated. Theoretically, scheduling rules considering the remaining cycle time are more effective than those considering the remaining processing time. Both scheduling rules were shown to be effective in reducing the average cycle times and cycle time standard deviations. However, the problem was that RCTE_{nj} was difficult to estimate. Another way of designing a stochastic scheduling rule is to form a combination of multiple deterministic scheduling rules from which every time, only the most suitable one is chosen. For example, Hsieh et al. [12] used five scheduling rules including FSMCT, FSVCT, largest deviation first, one step ahead, and FIFO jointly. The problem was that extensive simulation experiment was required to estimate the performance of each candidate so as to determine the most suitable one.

Recently, agent technologies have been applied in this field. For example, Yoon and Shen [21] constructed a multiple-agent system for scheduling a semiconductor manufacturing factory in which four types of agents (scheduling agents, work cell agents, machine agents, and product agents) were designed and developed. The optimal scheduling plan was found by the scheduling agent through enumerating some possible scenarios. Their proposed methodology was only compared with the two basic scheduling rules, FIFO and EDD. Besides, the batch production commonly used in semiconductor factory was not considered in their study, and therefore, the case might be impractical. From a novel viewpoint, data mining has been applied in scheduling manufacturing systems recently. For example, in Li and Sigurdur [15], historic schedules were transformed into appropriate data files that can be mined to learn which past scheduling decisions are corresponded to the best practices. Youssef et al. [21] proposed a hybrid genetic algorithm (GA) and data mining approach to find out the optimal scheduling plan for a job shop in which GA was used to generate a learning population of good solutions. These good solutions were then mined to find out some decision rules that could later be form a scheduling heuristic. Koonce and Tsai [14] proposed a similar methodology. Simulation techniques for making data were widely used in the research of semiconductor factory. Very few studies used the data collected in the real semiconductor factory. Besides, jobs in a semiconductor manufacturing factory might have different priorities. Such issues have not explored in the previous studies. The performance measures commonly considered include the average cycle time, the cycle time standard deviation, the average tardiness, the number of tardy jobs, the maximal lateness, the hit rate, the average WIP level, and so on. From a different perspective, Hsieh and Hou [13] proposed a production-flow value-based scheduling rule by applying the theory-of-constraint to maximize the production flow value.

The nonlinear scheduling rule proposed in this study belongs to the FS rules. In the original FS rules, RCTE_{nj} is estimated with the average value that is generated after a lengthy iterative simulation. However, the estimation inaccuracy is so high that the following mis-scheduling problems might occur:

1. Miss: RCTE_{nj} is underestimated and the job is not assigned with a high priority for processing. As a result, the progress of the job will be delayed, which may further make the underestimation of RCTE_{nj} worse.
2. False alarm: RCTE_{nj} is overestimated and the job is assigned with a high priority. The progress of the job will be fast, which then makes the overestimation of RCTE_{nj} worse.

These two mis-scheduling problems might impair the effectiveness of the FS rules. In this study, a fuzzy-neural approach is employed to estimate RCTE_{nj} instead. At first, Chen's look-ahead SOM and FBPN approach [7] is used to estimate both CTE_n and SCTE_n when job n is released into the semiconductor manufacturing factory. There are three steps for the look-ahead SOM-FBPN approach. The first step is to pre-classify jobs/examples with a SOM. After pre-classification, examples of different categories are then used to learn different FBPNs but with the same topology. Finally, the FBPN of every category can be used to estimate CTE_n or SCTE_n for new jobs belonging to the category. After estimating CTE_n and SCTE_n, RCTE_{nj} can be derived. With more accurate RCTE_{nj} estimation, the proposed fuzzy-neural approach is expected to achieve better scheduling performances.

Secondly, in the traditional FS rules, the importance of R_n (or n/λ) and the importance of RCTE_{nj} are not at the same scale, owing to the phenomenon that RCTE_{nj} might be much greater than R_n (or n/λ). As a result, SK_n might be determined solely by RCTE_{nj}. To deal with this problem, normalizing both of them to be in the range between 0 and 1 is required.

Thirdly, since the FS policies are based on differentiating SK_n's values, magnifying the differences in SK_n seems to be a good way of enhancing the performance of the FS policy. In this study, the "subtraction" operation in the traditional FS policy is replaced with the "division" operator for this purpose. This modification makes the proposed scheduling rule nonlinear.

3 Methodology

In the proposed methodology, firstly, the look-ahead SOM-FBPN approach proposed in our previous study is used to estimate RCTE_{nj} for every job in the semiconductor manufacturing factory.

3.1 Job remaining cycle time estimation with the look-ahead SOM-FBPN approach

There are three steps for implementing the look-ahead SOM-FBPN approach to estimate RCTE$_{nj}$. The first step is to pre-classify jobs/examples with a SOM as follows. The structure of the SOM is 10×10, and the number of output nodes is 100. Let x_n denote the 13-dimensional feature vector (U_n, Q_n, BQ$_n$, FQ$_n$, WIP$_n$, $D_n^{(1)}$, $D_n^{(2)}$, $D_n^{(3)}$, $FDW_n^{(1)}$, $FDW_n^{(2)}$, $FDW_n^{(3)}$, E_n, ER$_n$) corresponding to job n. Note that E_n and ER$_n$ are set to be zeros at the beginning. The feature vectors of all examples are fed into an SOM network using the following learning algorithm:

1. Set the number of output nodes and the number of input nodes. Initialize the values of learning rate, the neighborhood size, and the number of iterations.
2. Initialize the weights (w_{ij}) randomly where i=1–m and m stands for the maximum number of classes (job categories), j=1–10.
3. (Iteration) Provide an input vector to the network.
4. Find the output node (winner) based on the similarity between the input vector and the weight vector. For an input vector x_n, the winning unit can be determined by distance $\|x_n - w_c\| = \min_i \|x_n - w_i\|$, where w_i is the weight vector of the i-th unit and the index c refers to the winning unit.
5. Update the weight vector of the winner node using Kohonen's learning rule.

   $$w_i(t+1) = w_i(t) + \alpha(t)(x_n - w_i) \quad \text{for each } i \in N_c(t)$$

   where t is the discrete-time index of the variables; the factor $\alpha(t)\in[0, 1]$ is a scalar that defines the relative size of the learning step; and $N_c(t)$ specifies the neighborhood around the winner in the map array.
6. Provide the next input vector and go to step 4. Otherwise, go to step 7.
7. Stop if the predetermined number of iterations has been completed. Otherwise, go to step 3.

After the training is accomplished, input vectors that are topologically close are mapped to the same category, and then the classification result is post-processed, including eliminating isolated examples, merging small blocks, etc. Finally, the classification is finished and the value of 'm' is determined.

After pre-classification, examples of different categories are then obtained through learning with different FBPNs, but with the same topology described as follows:

1. Inputs: 11 parameters associated with the n-th example/job including U_n, Q_n, BQ$_n$, FQ$_n$, WIP$_n$, $D_n^{(r)}$ (r=1–3), and $FDW_n^{(f)}$ (f=1–3) that have to be normalized so that their values fall within [0, 1]. Then, some production execution/control experts are requested to express their beliefs (in linguistic terms) about the importance of each input parameter for estimating CTE$_n$ or SCTE$_n$. Linguistic assessments for an input parameter are converted into several prespecified fuzzy numbers. The subjective importance of an input parameter is then obtained by averaging the corresponding fuzzy numbers of the linguistic replies for the input parameter by all experts. The subjective importance obtained for an input parameter is multiplied to the normalized value of the input parameter. After such a treatment, all the inputs for the FBPN become triangular fuzzy numbers (TFNs), and the fuzzy arithmetic for TFNs is then used on training the FBPN.
2. Single hidden layer: Generally, one or two hidden layers are more beneficial for the convergence property of the FBPN.
3. Number of neurons in the hidden layer, which is chosen from 1 to 22 based on the preliminary analysis.
4. Output: the (normalized) CTE$_n$ or SCTE$_n$.
5. Network learning rule: Delta rule.
6. Transformation function: Sigmoid function,

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

7. Learning rate (η), 0.01–1.0.
8. Batch learning.
9. Number of epochs per replication, 75,000.
10. Number of initial conditions/replications, 100. Since the performance of a BPN or FBPN is highly sensitive to the initial condition, the training or testing process will be repeated for a certain number of times with various initial conditions that are randomly generated. Among the results, the best one is chosen for the subsequent analyses.

After pre-classification, a portion of the adopted examples in each category is fed as "training examples" into the FBPN to determine the parameters for the category. Two phases are involved at the training stage. In the forward phase, the inputs are multiplied with weights, summated, and transferred to the hidden layer. Then, the activated signals are outputted from the hidden layer as:

$$\tilde{h}_j = \left(h_{j1}, h_{j2}, h_{j3}\right) = \frac{1}{1+e^{-\tilde{n}_j^h}}$$
$$= \left(\frac{1}{1+e^{-n_{j1}^h}}, \frac{1}{1+e^{-n_{j2}^h}}, \frac{1}{1+e^{-n_{j3}^h}}\right),$$

where

$$\tilde{n}_j^h = \left(n_{j1}^h, n_{j2}^h, n_{j3}^h\right) = \tilde{I}_j^h(-)\tilde{\theta}_j^h$$
$$= \left(I_{j1}^h - \theta_{j3}^h, I_{j2}^h - \theta_{j2}^h, I_{j3}^h - \theta_{j1}^h\right),$$

$$\widetilde{I}_j^h = \left(I_{j1}^h, I_{j2}^h, I_{j3}^h\right) = \sum_{all\, i} \widetilde{w}_{ij}^h(\times)\widetilde{x}_{(i)} \cong \left(\sum_{all\, i} \min\left(w_{ij1}^h x_{(i)1}, w_{ij3}^h x_{(i)3}\right), \sum_{all\, i} w_{ij2}^h x_{(i)2}, \sum_{all\, i} \max\left(w_{ij1}^h x_{(i)1}, w_{ij3}^h x_{(i)3}\right) \right),$$

and (−) and (×) denote fuzzy subtraction and multiplication, respectively; $\widetilde{h}_j$s are also transferred to the output layer with the same procedure. Finally, the output of the FBPN is generated as:

$$\widetilde{o} = (o_1, o_2, o_3) = \frac{1}{1 + e^{-\widetilde{n}^o}} = \left(\frac{1}{1 + e^{-n_1^o}}, \frac{1}{1 + e^{-n_2^o}}, \frac{1}{1 + e^{-n_3^o}}\right),$$

where

$$\widetilde{n}^o = \left(n_1^o, n_2^o, n_3^o\right) = \widetilde{I}^o(-)\widetilde{\theta}^o = \left(I_1^o - \theta_3^o,\ I_2^o - \theta_2^o,\ I_3^o - \theta_1^o\right),$$

$$\widetilde{I}^o = \left(I_1^o,\ I_2^o,\ I_3^o\right) = \sum_{\text{all}\, j} \widetilde{w}_j^o(\times)\widetilde{h}_j \cong \begin{pmatrix} \sum\limits_{\text{all}\, j} \min\left(w_{j1}^o h_{j1}, w_{j3}^o h_{j3}\right), \\ \sum\limits_{\text{all}\, j} w_{j2}^o h_{j2},\ \sum\limits_{\text{all}\, j} \max\left(w_{j1}^o h_{j1}, w_{j3}^o h_{j3}\right) \end{pmatrix}.$$

To improve applicability of the FBPN and to facilitate the comparisons with conventional techniques, the fuzzy-valued output $\widetilde{o}$ is defuzzified according to the centroid-of-area (COA) formula:

$$o = \text{COA}(\widetilde{o}) = \frac{o_1 + 2o_2 + o_3}{4}.$$

Then, the output o is compared with the actual value a (the normalized CT$_n$ or SCT$_n$) to determine RMSE, E_n, and ER$_n$:

$$\text{RMSE} = \sqrt{\frac{\sum\limits_{\text{all trained example}} (o - a)^2}{\text{number of trained examples}}},$$

$E_n = o - a$,

ER$_n = E_n/a$.

Subsequently, in the backward phase, the deviation between o and a is propagated backward, and the error terms of neurons in the output and hidden layers can be obtained, respectively, as:

$$\delta^o = o(1 - o)(a - o),$$

$$\begin{aligned}\widetilde{\delta}_j^h &= \left(\delta_{j1}^h, \delta_{j2}^h, \delta_{j3}^h\right) = \widetilde{h}_j(\times)\left(1 - \widetilde{h}_j\right)(\times)\widetilde{w}_j^o\delta^o \\ &\cong \Big(\min\Big(\min\left(h_{j1}(1 - h_{j3})w_{j1}^o, h_{j3}(1 - h_{j1})w_{j1}^o\right)\delta^o, \\ &\quad \max\left(h_{j3}(1 - h_{j1})w_{j3}^o, h_{j1}(1 - h_{j3})w_{j3}^o\right)\delta^o\Big), \\ &\quad h_{j2}(1 - h_{j2})w_{j2}^o\delta^o, \max\Big(\min\Big(h_{j1}(1 - h_{j3})w_{j1}^o, \\ &\quad h_{j3}(1 - h_{j1})w_{j1}^o\Big)\delta^o, \max\left(h_{j3}(1 - h_{j1})w_{j3}^o, h_{j1}(1 - h_{j3})w_{j3}^o\right)\delta^o\Big)\Big).\end{aligned}$$

Based on these errors obtained above, adjustments for the connection weights and thresholds then can be made as:

$$\begin{aligned}\Delta\widetilde{w}_j^o &= \left(\Delta w_{j1}^o, \Delta w_{j2}^o, \Delta w_{j3}^o\right) = \eta\delta^o\widetilde{h}_j \\ &= \eta\delta^o\left(\min(h_{j1}, h_{j3}),\ h_{j2},\ \max(h_{j1}, h_{j3})\right),\end{aligned}$$

$$\begin{aligned}\Delta\widetilde{w}_{ij}^h &= \left(\Delta w_{ij1}^h, \Delta w_{ij2}^h, \Delta w_{ij3}^h\right) = \eta\widetilde{\delta}_j^h(\times)\widetilde{x}_i \\ &\cong \eta\Big(\min\left(\delta_{j1}^h x_{i1}, \delta_{j1}^h x_{i3}, \delta_{j3}^h x_{i1}, \delta_{j3}^h x_{i3}\right), \delta_{j2}^h x_{i2}, \\ &\quad \max\left(\delta_{j1}^h x_{i1}, \delta_{j1}^h x_{i3}, \delta_{j3}^h x_{i1}, \delta_{j3}^h x_{i3}\right)\Big),\end{aligned}$$

$$\Delta\theta^o = -\eta\delta^o,$$

$$\begin{aligned}\Delta\widetilde{\theta}_j^h &= \left(\Delta\theta_{j1}^h,\ \Delta\theta_{j2}^h,\ \Delta\theta_{j3}^h\right) = -\eta\widetilde{\delta}_j^h \\ &= \left(-\eta\delta_{j3}^h, -\eta\delta_{j2}^h, -\eta\delta_{j1}^h\right).\end{aligned}$$

To accelerate convergence, a momentum can be added to the learning expressions. For example,

$$\begin{aligned}\Delta\widetilde{w}_j^o &= \eta\delta^o\widetilde{h}_j + \alpha\left(\widetilde{w}_j^o(t) - \widetilde{w}_j^o(t-1)\right) \\ &= \Big(\eta\delta^o h_{j1} + \alpha w_{j1}^o(t) - \alpha w_{j3}^o(t-1), \eta\delta^o h_{j2} + \alpha w_{j2}^o(t) \\ &\quad - \alpha w_{j2}^o(t-1), \eta\delta^o h_{j3} + \alpha w_{j3}^o(t) - \alpha w_{j1}^o(t-1)\Big).\end{aligned}$$

Theoretically, the learning process of network terminates either when the RMSE falls below a prespecified level or the improvement in the RMSE becomes negligible with more epochs or a large number of epochs have already been run. Then, test examples are fed into the FBPN to evaluate the accuracy of the network that is also measured with the RMSE. However, the accumulation of fuzziness during the training process continuously increases the lower bound, the upper bound, and the spread of the fuzzy-valued output $\tilde{o}$ (and those of many other fuzzy parameters) and might prevent the RMSE (calculated with the defuzzified output o) from converging to its minimal value. Conversely, the centers of some fuzzy parameters are becoming smaller and smaller due to network learning. It is possible that a fuzzy parameter becomes invalid in the sense that the lower bound is higher than the center. To deal with this problem, the lower and upper bounds of all fuzzy numbers in the FBPN will no longer be modified if the following index converges to a minimal value

$$\alpha\sqrt{\frac{\sum_{\text{all examples}}\min\left((o_1-a)^2,(o_3-a)^2\right)}{\text{number of examples}}}+(1-\alpha)\sqrt{\frac{\sum_{\text{all examples}}\max\left((o_1-a)^2,(o_3-a)^2\right)}{\text{number of examples}}},$$

$0<\alpha<1$.

After training and testing, the FBPN of every category is then employed to estimate CTE_n and SCTE_n for the examples belonging to the category. Then, E_n and ER_n are calculated for every example and are fed back to the SOM classifier. Subsequently, classification is done again. If the classification result is the same as the previous one, then stop; otherwise, the FBPN of every category has to be retrained and retested with the above procedure again.

The look-ahead SOM-FBPN approach introduced previously is used to estimate both CTE_n and SCTE_n when job n is released into the semiconductor manufacturing factory, namely, there will be two groups of FBPNs to estimate CTE_n and SCTE_n, respectively. A product in a semiconductor manufacturing factory usually requires up to hundreds of processes to undergo, and we can estimate the SCTE_n for each step. Subsequently, RCTE_{nj} is derived in the following way:

$$\mathrm{RCTE}_{nj}=\left(\mathrm{CTE}_n-\mathrm{SCT}_{nj}\right)\times\mathrm{SCT}_{nj}\big/\mathrm{SCTE}_{nj}.$$

The formula provides the adjustments of RCTE_{nj} by considering the difference between SCTE_n and SCT_n in a proportional manner.

3.2 Nonlinear scheduling rules with new job slack

Subsequently, RCTE_{nj} has to be embedded into the FS rule. There are two different ways of formulation, depending on the purpose of scheduling. One way is aimed at minimizing the variance of cycle time with the FSVCT rule:

$$\mathrm{SK}_n=R_n-\mathrm{RCTE}_{nj}.$$

The other way is to minimize the mean cycle time with the FSMCT rule:

$$\mathrm{SK}_n=n/\lambda-\mathrm{RCTE}_{nj}.$$

They are a subclass of the LS policies. The FSVCT rule attempts to make every job equally late or equally early. Thus, it will reduce the standard deviation of lateness. On the other hand, the FSMCT rule effectively diminishes the burst of arrivals to all buffers simultaneously. Therefore, it is expected to reduce the average cycle time. However, RCTE_{nj} might be much greater than R_n or n/λ. As a result, SK_n might be determined solely by RCTE_{nj}. To solve this problem, in the proposed methodology, all terms for the FS rules are normalized:

$$N\left(\mathrm{RCTE}_{nj}\right)=\left(\mathrm{RCTE}_{nj}-\min_{\text{all }n}\left(\mathrm{RCTE}_{nj}\right)\right)\Big/\left(\max_{\text{all }n}\left(\mathrm{RCTE}_{nj}\right)-\min_{\text{all }n}\left(\mathrm{RCTE}_{nj}\right)\right).$$

$$N(R_n)=\left(R_n-\min_{\text{all }n}(R_n)\right)\Big/\left(\max_{\text{all }n}(R_n)-\min_{\text{all }n}(R_n)\right).$$

$$Nor(n/\lambda)=\left(n/\lambda-\min_{\text{all }n}(n/\lambda)\right)\Big/\left(\max_{\text{all }n}(n/\lambda)-\min_{\text{all }n}(n/\lambda)\right).$$

On the other hand, since the FS policies are based on differentiating SK_n's, magnifying the difference in SK_n seems to be a good way of improving the performance of the FS policy. For this reason, in the proposed methodology, the "division" operator is applied to calculate SK_n instead:

$$\underset{n}{SK}=N(R_n)\Big/N\left(\underset{nj}{RCTE}\right).$$

Obviously, it becomes a nonlinear scheduling rule. The new way helps in breaking possible ties when selecting jobs in the traditional FS policies. An example illustrated in Table 1 demonstrates the new way of calculating SK_n and makes comparison with the traditional way. Note that there is a tie with the traditional FS policy (job no. 2 and job no. 3

Table 1 The new way of calculating the slack of a job

Job no.	Release time	Estimated remaining cycle time	Slack (traditional)	$N(R_n)$	$N(\text{RCTE}_{nj})$	Slack (new)
1	3	15	−12	0.0000	0.0000	0.0000
2	4	23	−19[a]	0.1111	0.2286	0.4859
3	5	24	−19[a]	0.2222	0.2571	0.8639
4	7	32	−25	0.4444	0.4857	0.9148
5	8	44	−36	0.5556	0.8286	0.6704
6	10	47	−37	0.7778	0.9143	0.8506
7	11	46	−35	0.8889	0.8857	1.0035
8	12	50	−38	1.0000	1.0000	0.9999
Max	12	50				
Min	3	15				

[a] Tie

have the same slack values) that is broken by the proposed methodology. By breaking possible ties, the "division" operator in the proposed methodology indeed increases the degree of differentiation. To further investigate these issues, 20 examples in which each consists of the data of 100 jobs were randomly generated. R_n's ranged from 1 to 200, while RCTE_n's were between 50 and 500. All of these times follow uniform distributions. The numbers of ties with the traditional FS policy in all examples are shown in Fig. 1. Conversely, in all examples, there were no ties with the proposed methodology. In practice, the formula can be slightly adjusted in the following way to avoid "divided by zero" and "value too large or small":

$$\text{Nonlinear FSVCT rule}: \quad \text{SK}_n = \ln\left(N(R_n)/\left(N\left(\text{RCTE}_{nj}\right)+10^{-5}\right)+10^{-5}\right).$$

On the other hand, the nonlinear FSMCT rule can be derived in the same way:

$$\text{Nonlinear FSMCT rule}: \quad \underset{n}{SK} = \ln\left(N(n/\lambda)\Big/\left(N\left(\text{RCTE}_{nj}\right)+10^{-5}\right)+10^{-5}\right).$$

4 Test data from a simulated semiconductor manufacturing factory

In practice, the history data of each job is only partially available in the semiconductor manufacturing factory. Further, some information of the previous jobs such as Q_n, BQ_n, FQ_n, and SCT_{nj} is not easy to collect on the shop floor. Therefore, a simulation model is often developed to simulate the manufacturing processes of a real semiconductor manufacturing factory [1–11]. Then, such information can be derived from the shop floor status collected from the simulation model [3]. To generate test data, a simulation program coded using Microsoft Visual Basic 6.0 is constructed to simulate a semiconductor manufacturing environment with the following assumptions:

1. Jobs are uniformly released into the semiconductor manufacturing factory.
2. The distributions of the inter-arrival times of machine downs are exponential.
3. The distribution of the time required to repair a machine is uniform.
4. The percentages of jobs with different product types in the factory are predetermined. As a result, this study is focused on fixed-product-mix cases.
5. The percentages of jobs with different priorities released into the semiconductor manufacturing factory are loosely controlled.
6. The priority of a job cannot be changed during fabrication.
7. A job has equal opportunities for processing on each alternative machine/head available at a step.
8. A job cannot proceed to the next step until every wafer of this batch has been finished at the current step.
9. No preemption is allowed.

The basic configuration of the simulated semiconductor manufacturing factory is simplified from a real-world semiconductor manufacturing factory located in the Science

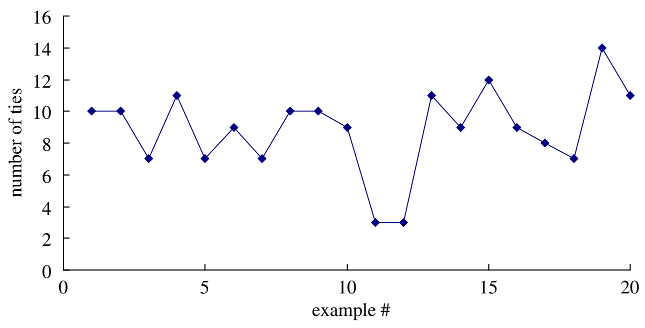

Fig. 1 The number of ties in all examples with the traditional FSVCT rule

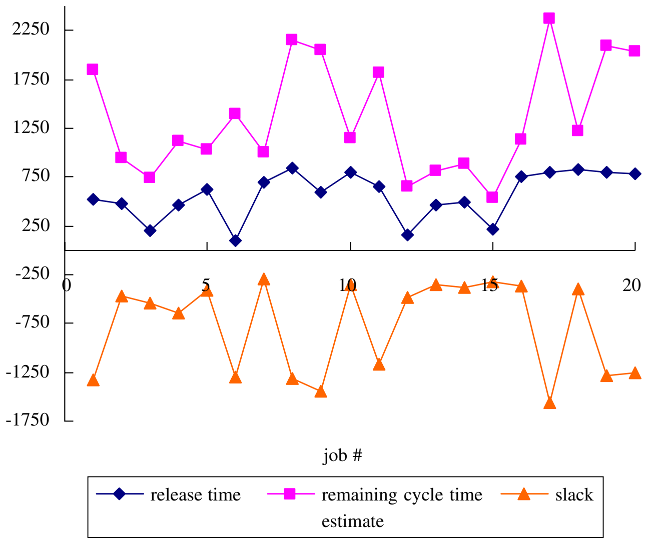

Fig. 2 The slack values of 20 jobs calculated in the traditional way

Park of Hsin-Chu, Taiwan. The simulation model is not only large and complicated but also capable of demonstrating the characteristics of the real semiconductor manufacturing factory. The conclusions drawn here are therefore meaningful to the control of the real semiconductor manufacturing factory. Assumptions 1–3 and 7–9 are commonly adopted in related researches [3–4, 6], while assumptions 5 and 6 are made to simplify the situation. There are five products (labeled as A–E) in the simulated semiconductor manufacturing factory. A fixed product mix is assumed. The percentages of these products in the factory's product mix are assumed to be 35%, 24%, 17%, 15%, and 9%, respectively. The simulated semiconductor manufacturing factory has a monthly capacity of 20,000 pieces of wafers and is expected to be fully utilized (utilization=100%). Jobs are released into the semiconductor manufacturing factory one by one every 0.85 h, namely, the mean release rate $\lambda=1/0.85=1.18$ jobs per hour. Three types of priorities (normal, hot, and super hot) are randomly assigned to jobs. The percentages of jobs with these priorities released into the semiconductor manufacturing factory are restricted to be approximately 60%, 30%, and 10%, respectively. Each product has 150–200 steps and six to nine reentrances to the most bottleneck machine. The singular production characteristic "reentry" of the semiconductor industry is clearly reflected in the example. It also shows the difficulty for the production planning and scheduling staff to provide an accurate due date for the product with such a complicated routing. A total of 102 machines (including alternative machines) are provided to process single-wafer or batch operations in the semiconductor manufacturing factory. Thirty replications of the simulation are successively run. The time required for each simulation replication is about 15 min on a PC with 256 MB RAM and Athlon™ 64 Processor 3000+ CPU. A horizon of 24 months is simulated. The maximal cycle time is less than 3 months. Therefore, 4 months and an initial WIP status (obtained from a pilot simulation run) seemed to be sufficient to drive the simulation into a steady state. The statistical data were collected starting at the end of the fourth month. For each replication, data of 30 jobs are collected and classified by their product types and priorities. In total, data of 900 jobs can be collected. A traced report was generated every simulation run for verifying the simulation model. The simulated average cycle times have also been compared with the actual values to validate the simulation model. An eightfold evaluation process was used to conduct cross data validations, namely, if there are R records, then an eightfold decomposition process makes each fold contain $R/8$ records. Each fold is used to be the testing data and the rest of the folds are merged as the training data.

Taking machine no. 4 as an example, the release times and estimated remaining cycle times of 20 jobs simultaneously gathered at the machine are summarized in Fig. 2. The slacks of these jobs can be obtained in the traditional way and are shown in the same figure. Conversely, after normalizing the release times and estimated remaining cycle times of these jobs, their slack values were also found in the new way. The results are shown in Fig. 3. The degree of differentiation with the proposed methodology was more obvious than with the traditional approach. The scheduling results using the two approaches are compared in Table 2.

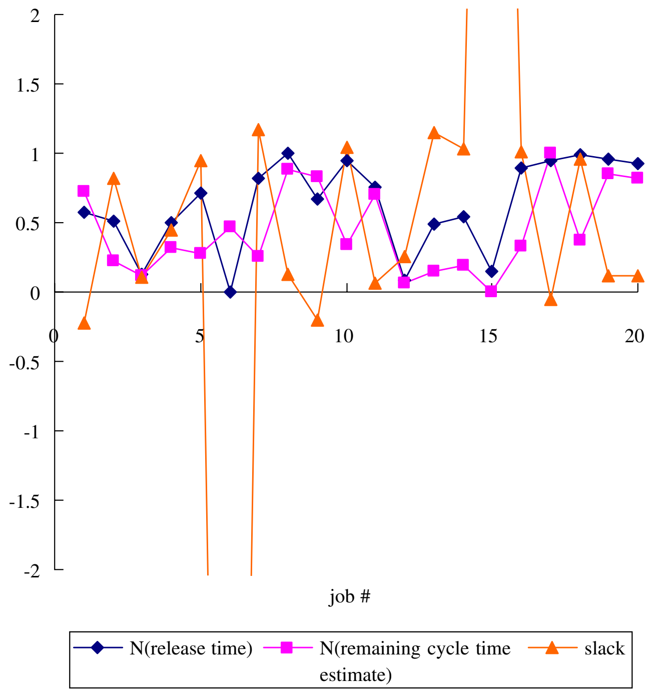

Fig. 3 The slack values of 20 jobs calculated in the new way

Table 2 The scheduling results by the two approaches

Approach	Scheduling result
The traditional FS policy	9->11->2->16->3->18->20->15->17->1->12->4->8->13->14->19->10->6->5->7
The proposed methodology	3->2->11->9->15->1->18->20->16->12->17->4->8->13->19->4->6->10->7->5

Besides, taking the data of product type A, normal priority, full size (25 wafers per job) as an example, the remaining cycle times at various steps are summarized in Table 3. The data were collected only once every four steps to avoid delaying the simulation progress. The variation in the remaining cycle time of each step is also shown in the same table. Note that the variation is very large. The traditional way of using the average value might result in mis-scheduling.

5 Experimental results and discussion

To evaluate the effectiveness of the proposed methodology and to make comparison with four existing approaches—p-FIFO, p-EDD, p-SRPT, and p-FS—all these methods were applied to ten test cases containing the data of full-size jobs with different product types and various priorities. Some cases were not analyzed because their data were not enough for job classification and network training in the SOM-FBPN approach. As a result, the performance of the proposed methodology was not optimized in such cases. The results are summarized in Tables 4 and 5.

For p-FIFO, jobs were sequenced on each machine first by their priorities, then by their arrival times at the machine.

For p-EDD, jobs were also sequenced first by their priorities, then by their due dates. The performance of p-EDD is dependent on the way of determining the due date of a job. In the experiment, the due date of a job was determined as follows:

$$\text{Due date} = \text{release time} + (\zeta - 1.5 \times \text{p}) \times \text{total processing time}$$

where ζ indicates the cycle time multiplier. To consider the requirement that a job with higher priority has to be completed faster, ζ was subtracted by 1.5 times p. The value of p for a job with normal, hot, and super hot priority is 1, 2, and 3, respectively.

For p-FS, there were two stages. First, jobs were scheduled with the p-FIFO policy for which the remaining cycle times at each step of all jobs were recorded and averaged. Subsequently, the p-FS policy was applied to schedule jobs instead based on the average remaining cycle times obtained previously. Namely, jobs were sequenced on each machine first by their priorities, then by their slack values that were equal to their release times minus the average remaining cycle times.

In the proposed methodology, the SOM-FBPN approach was used to estimate both the cycle time and the step cycle time so as to derive the remaining cycle time. To determine the optimal number of the hidden-layer nodes in each FBPN, a preliminary analysis was done. An example is given in Fig. 4 in which the minimal RMSEs with various numbers of hidden-layer nodes were compared. The optimal choice was to use 11 hidden-layer nodes if 50,000 or 75,000 epochs were run. However, if only 25,000 epochs could be run owing to the limitation in time, then the optimal number of hidden-layer nodes was 10. Taking product A (normal priority) as an example, the averages and standard deviations of the remaining cycle times at some steps are summarized in Table 3. Obviously, using the average value of the remaining cycle time in the traditional FS rule was far from accurate and might result in mis-scheduling.

For the average cycle time, the p-FIFO policy was adopted as the comparison basis, and the percentage of improvement in the performance measure by applying another approach is enclosed in parentheses following the performance measure. On the other hand, the p-FS policy was adopted as the comparison basis on the cycle time standard deviation.

According to experimental results, the following points are made:

1. In the cycle time standard deviation respect, the proposed methodology outperformed the other approaches in almost all cases. The average advantage over the basis p-FS policy was up to 41%.

Table 3 The remaining cycle time at each step (product type A, normal priority)

Step no.	7	11	15	18	22	25	29	33	36	…	116
Remaining cycle time (h)	992	978	674	652	554	511	473	454	428	…	10
Variation (h)	330	321	312	303	294	285	276	268	261	…	21

Table 4 The performances of various approaches in reducing the average cycle times

Average cycle time (h)	A (normal)	A (hot)	A (super hot)	B (normal)	B (hot)	C (normal)	C (hot)	D (normal)	D (hot)	E (normal)
p-FIFO	1,256	401	320	1,278	457	1,418	574	1,244	692	1,670
p-EDD-5.0	1,087 (−13%)	346 (−14%)	306 (−4%)	1,433 (+12%)	478 (+5%)	1,755 (+24%)	585 (+2%)	1,118 (−10%)	702 (+1%)	1,621 (−3%)
p-EDD-5.5	1,074 (−14%)	346 (−14%)	302 (−6%)	1,464 (+15%)	464 (+2%)	1,822 (+28%)	611 (+6%)	1,135 (−9%)	675 (−2%)	1,671 (+0%)
p-EDD-6.0	1,047 (−17%)	350 (−13%)	298 (−7%)	1,488 (+16%)	481 (+5%)	1,863 (+31%)	590 (+3%)	1,132 (−9%)	700 (+1%)	1,648 (−1%)
p-EDD-6.5	1,033 (−18%)	347 (−13%)	304 (−5%)	1,556 (+22%)	484 (+6%)	1,928 (+36%)	580 (+1%)	1,121 (−10%)	698 (+1%)	1,645 (−1%)
p-SRPT	966 (−23%)	350 (−13%)	309 (−3%)	1,737 (+36%)	483 (+6%)	1,971 (+39%)	580 (+1%)	1,107 (−11%)	695 (+0%)	1,614 (−3%)
p-FS	1,046 (−17%)	385 (−4%)	317 (−1%)	1,745 (+37%)	519 (+14%)	1,884 (+33%)	606 (+6%)	1,076 (−14%)	704 (+2%)	1,701 (+2%)
The proposed methodology	1,368 (+9%)	365 (−9%)	299 (−7%)	1,370 (+7%)	413 (−10%)	1,425 (+1%)	495 (−14%)	1,281 (+3%)	671 (−3%)	1,795 (+7%)

2. On the other hand, though the performance of the proposed methodology in reducing the average cycle time was sometimes good and sometimes bad, on average, the proposed methodology still surpassed the other approaches. The average advantages over p-FIFO, p-EDD-5.0, p-EDD-5.5, p-EDD-6.0, p-EDD-6.5, p-SRPT, and p-FS were 1%, 1%, 2%, 2%, 3%, 4%, and 7%, respectively.
3. The effect of the proposed methodology on reducing the average cycle times was more obvious to jobs with higher priorities (see Fig. 5).
4. Conversely, the proposed methodology shows the most obvious effect in reducing the cycle time standard deviations when the priorities of the jobs are the lowest "normal" (see Fig. 6).
5. The traditional p-FS policy performed poorly in the simulation experiments. The reason might be owing to the diversification in the product type and priority that made the remaining cycle time of a job highly uncertain and very difficult to estimate. As a result, estimating the remaining cycle time using their average value might cause inaccurate results and further impair the scheduling performance of the traditional p-FS policy.
6. As expected, p-SRPT performed well in reducing the average cycle times, but might show exceedingly bad performance in the standard deviation.

Table 5 The performances of various approaches in reducing the cycle time variation

Cycle time standard deviation (h)	A (normal)	A (hot)	A (super hot)	B (normal)	B (hot)	C (normal)	C (hot)	D (normal)	D (hot)	E (normal)
p-FS	319	35	28	222	55	290	54	172	64	322
p-FIFO	56 (−82%)	24 (−31%)	23 (−18%)	87 (−61%)	40 (−27%)	72 (−75%)	31 (−43%)	133 (−23%)	71 (+11%)	306 (−5%)
p-EDD-5.0	130 (−59%)	25 (−29%)	23 (−18%)	50 (−77%)	39 (−29%)	134 (−54%)	23 (−57%)	89 (−48%)	82 (+28%)	294 (−9%)
p-EDD-5.5	103 (−68%)	34 (−3%)	17 (−39%)	60 (−73%)	28 (−49%)	147 (−49%)	60 (+11%)	81 (−53%)	60 (−6%)	310 (−4%)
p-EDD-6.0	101 (−68%)	31 (−11%)	22 (−21%)	41 (−82%)	49 (−11%)	144 (−50%)	34 (−37%)	76 (−56%)	41 (−36%)	306 (−5%)
p-EDD-6.5	90 (−72%)	25 (−29%)	20 (−29%)	38 (−83%)	53 (−4%)	141 (−51%)	36 (−33%)	79 (−54%)	53 (−17%)	306 (−5%)
p-SRPT	246 (−23%)	32 (−9%)	23 (−18%)	106 (−52%)	30 (−45%)	250 (−14%)	37 (−31%)	90 (−48%)	51 (−20%)	307 (−5%)
The proposed methodology	44 (−86%)	28 (−20%)	18 (−36%)	31 (−86%)	21 (−62%)	146 (−50%)	27 (−50%)	181 (+5%)	84 (+31%)	126 (−61%)

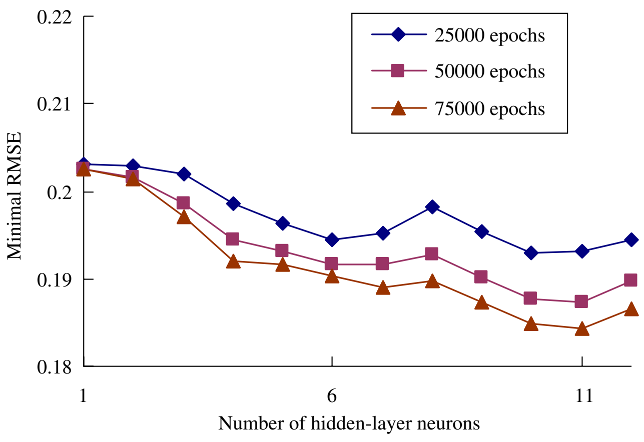

Fig. 4 Determining the optimal number of hidden-layer nodes

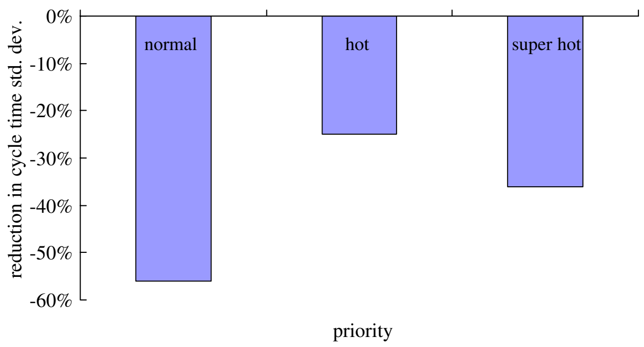

Fig. 6 The performance of the proposed methodology in reducing the cycle time standard deviations with various priorities

7. Among various EDD rules, the performance of p-EDD-5.0 was the best one in reducing the average cycle times, while p-EDD-6.0 was the best choice if the cycle time standard deviations were to be minimized.

The look-ahead SOM-FBPN approach was also applied to the traditional FSVCT rule. Taking product type B (normal priority) as an example, the results are shown in Table 6. We noticed that with better remaining cycle time estimation, the performances of the traditional FSVCT rule were indeed improved. However, incorporating the look-ahead SOM-FBPN approach with the nonlinear scheduling rule could reduce the cycle time standard deviation significantly.

6 Conclusions and directions for future research

To further improve the performance of job scheduling in a semiconductor manufacturing factory, a nonlinear scheduling rule incorporating a fuzzy-neural remaining cycle time estimator is proposed in this study. The proposed methodology is modified from the traditional FS policy with three treatments. At first, Chen's look-ahead SOM-FBPN approach is used to estimate the remaining cycle time of every job in the semiconductor manufacturing factory. Subsequently, the release time and remaining cycle time of a job are both normalized to balance their importance in the FS rule. Finally, the normalized release time is divided by the normalized remaining cycle time to calculate the slack instead, and in this way, the proposed scheduling rule becomes a nonlinear one.

To evaluate the effectiveness of the proposed methodology, production simulation is also employed in this study to generate some test data. According to experimental results,

1. On average, the proposed methodology outperformed the other approaches in reducing both the average cycle times and cycle time standard deviations. The advantage was especially obvious when the cycle time standard deviations were to be minimized.
2. A nonlinear scheduling rule reveals its capability in reducing the cycle time standard deviations. However, that might rise the average cycle times. To solve this problem, Chen's look-ahead SOM-FBPN approach can be used to estimate the remaining cycle time of every job in the semiconductor manufacturing factory instead, which significantly improved the performance of the nonlinear scheduling rule in reducing the average cycle times.

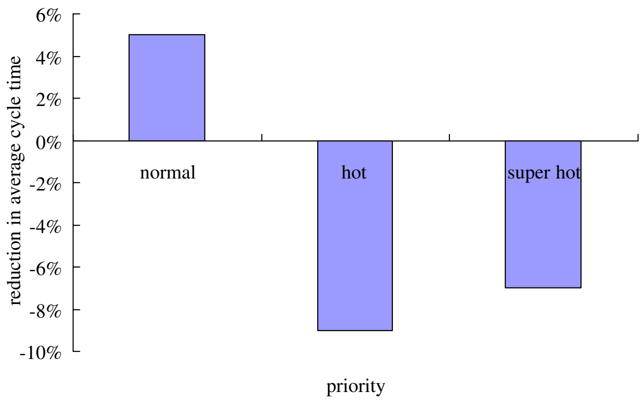

Fig. 5 The performance of the proposed methodology in reducing the average cycle times with various priorities

Table 6 The results of applying the look-ahead SOM-FBPN approach to the traditional FSVCT rule

Approach	Average cycle time (h)	Cycle time standard deviation (h)
FSVCT	1,745	222
FSVCT + SOM-FBPN	1,448 (−17%)	163 (−27%)
Nonlinear + SOM-FBPN	1,370 (−21%)	31 (−86%)

However, to further evaluate the effectiveness and efficiency of the proposed methodology, it has to be applied in a real semiconductor manufacturing factory. In other words, a field study is necessary. Besides, there are also some other scheduling rules based on estimated remaining cycle times, improving such rules in the same way can be investigated in the future. Finally, other nonlinear forms of the scheduling rule can be developed to further enhance the scheduling performance. These constitute some directions for future research.

References

1. Barman S (1998) The impact of priority rule combinations on lateness and tardiness. IIE Trans 30:495–504 doi:10.1080/07408179808966489
2. Chang P-C, Hsieh J-C (2003) A neural networks approach for due-date assignment in a wafer fabrication factory. Int J Ind Eng 10(1):55–61
3. Chang P-C, Hsieh J-C, Liao TW (2001) A case-based reasoning approach for due date assignment in a wafer fabrication factory. Proceedings of the International Conference on Case-Based Reasoning (ICCBR 2001), Vancouver, British Columbia, Canada
4. Chang P-C, Hsieh J-C, Liao TW (2005) Evolving fuzzy rules for due-date assignment problem in semiconductor manufacturing factory. J Intell Manuf 16:549–557 doi:10.1007/s10845-005-1663-4
5. Chang P-C, Liao TW (2006) Combining SOM and fuzzy rule base for flow time prediction in semiconductor manufacturing factory. Appl Soft Comput 6:198–206 doi:10.1016/j.asoc.2004.12.004
6. Chen T (2003) A fuzzy back propagation network for output time prediction in a wafer fab. Appl Soft Comput 2(3F):211–222 doi:10.1016/S1568-4946(02)00066-2
7. Chen T (2006) A hybrid look-ahead SOM-FBPN and FIR system for wafer lot output time prediction and achievability evaluation. Int J Adv Manuf Technol 35(5–6):575–586 doi:10.1007/s00170-006-0741-x
8. Chen T (2006) A hybrid SOM-BPN approach to lot output time prediction in a wafer fab. Neural Process Lett 24(3):271–288 doi:10.1007/s11063-006-9027-4
9. Chen T (2006) A look-ahead back propagation network to predict wafer lot output time. WSEAS Trans Comput 5(5):910–915
10. Chen T, Tsai HR, Wu HC (2006) Wafer lot output time prediction with a hybrid artificial neural network. WSEAS Trans Comput 5 (5):817–823
11. Gupta AK, Sivakumar AI (2006) Job shop scheduling techniques in semiconductor manufacturing. Int J Adv Manuf Technol 27:1163–1169 doi:10.1007/s00170-004-2296-z
12. Hsieh B-W, Chen C-H, Chang S-C (2001) Scheduling semiconductor wafer fabrication by using ordinal optimization-based simulation. IEEE Trans Robot Autom 17(5):599–608 doi:10.1109/70.964661
13. Hsieh S, Hou KC (2006) Production-flow-value-based job dispatching method for semiconductor manufacturing. Int J Adv Manuf Technol 30(7–8):727–737 doi:10.1007/s00170-005-0105-y
14. Koonce DA, Tsai S-C (2000) Using data mining to find patterns in genetic algorithm solutions to a job shop schedule. Comput Ind Eng 38(3):361–374 doi:10.1016/S0360-8352(00)00050-4
15. Li X, Sigurdur O (2004) Data mining for best practices in scheduling data. Proceedings of IIE Annual Conference and Exhibition
16. Lu SCH, Ramaswamy D, Kumar PR (1994) Efficient scheduling policies to reduce mean and variation of cycle time in semiconductor manufacturing plant. IEEE Trans Semicond Manuf 7 (3):374–388 doi:10.1109/66.311341
17. Qi C, Sivakumar AI, Gershwin SB (2008) Impact of production control and system factors in semiconductor wafer fabrication. IEEE Trans Semicond Manuf 21(3):376–389 doi:10.1109/TSM.2008.2001214
18. Rose O (2002) Some issues of the critical ratio dispatch rule in semiconductor manufacturing. Winter Simul Conf Proc 2:1401–1405
19. Uzsoy R, Church LK, Ovacik IM (1992) Dispatching rules for semiconductor testing operations—a computational study. Proceedings of IEEE/CHMT International Electronics Manufacturing Technology Symposium
20. Wein LM (1998) Scheduling semiconductor wafer fabrication. IEEE Trans Semicond Manuf 1:115–130 doi:10.1109/66.4384
21. Yoon HJ, Shen W (2008) A multiagent-based decision-making system for semiconductor wafer fabrication with hard temporal constraints. IEEE Trans Semicond Manuf 21(1):83–91 doi:10.1109/TSM.2007.914388
22. Zhang H, Jiang Z, Guo C (2009) Simulation-based optimization of dispatching rules for semiconductor wafer fabrication system scheduling by the response surface methodology. Int J Adv Manuf Technol. doi:10.2007/s00170-008-1462-0