Proceedings of the 2013 Winter Simulation Conference
R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds

INTELLIGENT DISPATCHING IN DYNAMIC STOCHASTIC JOB SHOPS

Tao Zhang
Oliver Rose

Universität der Bundeswehr München
Department of Computer Science
D-85577 Neubiberg, GERMANY

ABSTRACT

Dispatching rules are common method to schedule jobs in practice. However, they consider only limited factors which influence the priority of jobs. This limited consideration narrows the rules' scope of application. We develop a new hierarchical dispatching approach based on two types of factors: local factors and global factors, where each machine has its own dispatching rule setup. According to the global factors, the dispatchers divide the state of the manufacturing system into several patterns, and parameterize a neural network for each pattern to map the relationships between the local factors and the priorities of jobs. When making decisions, the dispatchers determine which pattern the current state belongs to. Then the appropriate neural network computes priorities according to the jobs' local factors. The job with the highest priority will be selected. Finally, the proposed approach is introduced on a manufacturing line and the performance is compared to classical dispatching rules.

1 INTRODUCTION

In dynamic stochastic job shops, jobs are released and arrive at the shop over time. The release date, processing time of jobs and machine breakdowns are stochastic and not known in advance. It is difficult and sometimes impossible to compute optimal schedules. In this case, scheduling is typically carried out by means of dispatching decisions: once a machine becomes free, we decide what it should do next. Detailed dispatching decisions in a job shop are usually determined by dispatching rules. A dispatching rule can find a reasonably good solution in a relatively short time and is very simple to implement. In this paper, we assume that no machine is kept idle while a job is waiting for processing.

At present, the study of dispatching rules focuses on two main fields: developing composite dispatching rules and selecting rules dynamically. The first field aims to develop new dispatching rules which are applicable to more complex manufacturing lines. The rule never changes when being used. Holthaus and Rajendran (1997) develop five new dispatching rules of this type for scheduling a job shop. Some of these rules make use of the process time and work-content in the queue of the next operation of a job, by following a simple additive approach, in addition to the arrival time and dynamic slack of a job. The proposed rules are not only simple in structure, but also quite efficient in minimizing several measures of performance. Chen and Matis (2013) present a dispatching rule called the Weight Biased Modified RRrule that minimizes the mean tardiness of weighted jobs in an m-machine job shop. It is a significant extension of the RRrule in that it has linear complexity and considers weighted jobs. The shortcoming of these dispatching rules is that they are usually effective in some specific situations but not in others.

Therefore, the second field focuses on finding better rules for a given situation. The rule changes over time according to the state of the manufacturing line. Scholz-Reiter et al. (2010) use Gaussian processes as a machine learning technique for the selection of dispatching rules in dynamic scenarios. Lian et al. (1998) propose a fuzzy inference-based selection of dispatching rules, adapting the scheduling decision to the dynamic changes in the manufacturing environment. The selection of dispatching rules takes the state of the manufacturing line into consideration. Each rule can be used in the right state. However, the selection just determines which information concerning jobs, such as processing time, waiting time and so on, plays the most important role in assigning jobs (computing priorities) in a given state. Only the primary information about jobs is involved in the decision-making. But sometimes effects of the secondary information about jobs should not be ignored. Theoretically, the more information that is considered, the better decision that is made and the wider scope the approach will have.

In addition, both the developed new rules and dynamically selected rules are used for all machines in the shop, i.e., all machines use the same rule at the same time. The position of machines in the layout of production processes also influences the performance of rules. It is possible that a rule performs well on one machine, but badly on other machines. Thus, the better way is to involve the position information while a decision is made or to build individual rules for each machine.

In this paper, using artificial intelligence techniques, we create an individual dispatcher for each machine. The dispatchers consider a large amount of information including global factors and local factors in the decision-making process. The paper is structured as follows. Section 2 presents the general idea of the intelligent dispatcher which contains two data-driven models: the pattern recognition model and the neural network model. Section 3 describes how to collect the data on global factors, local factors and the priority by using simulation. The data will then be used to build the two data-driven models. In Section 4, the pattern recognition model is developed by using spectral clustering. The spectral clustering classifies the state of manufacturing line into several patterns according to the global factors. In Section 5, a feed-forward neural network is built for each pattern and trained by the Levenberg-Marquardt algorithm based on the local data and the priority data. The networks map the relationships between the local factors and the priority of jobs. Section 6 provides an application of the proposed approach.

2 INTELLIGENT DISPATCHING

2.1 Dispatching Problem

There are two tasks for dispatching: job assignment or sequencing and machine allocation. If there is more than one job waiting for processing before a machine when the machine becomes free, we need to decide which job should be processed first. This is the first task. If there is more than one machine that can process a job when the job starts its operation, we need to select one machine to process the job. This is the second task. In this paper, we consider job assignment only. The job assignment problem is actually to determine the priority of jobs waiting before the machine. The determination of a job's priority is related to the objectives of the scheduling. A job may have different priorities to meet different objectives.
Once the objective is fixed, the priority completely depends on the information which influences the priority. We group the information related to the priority into two types: global factors and local factors. The priority depends on these two types of factors and can be described as follows,

$$prio = f(G_{local}, G_{global}) .$$

The global factors concern the state of the manufacturing line, including all online jobs' condition and all machines' condition. A machine's condition can be described by its state and the time that the state lasts. In addition to its state and the time that the state lasts, a job's condition includes its position and progress. The position describes which machine or transport line the job can be found on. The progress indicates which step the job reaches. The local factor is the information of the job whose priority we compute. The information contains the static information, such as the due date, processing time, step number, and so on, and the dynamic information, such as the waiting time, current step, and so on. Having a closer look at the variety of dispatching rules, we see that local dispatching rules just consider a few local factors, and global dispatching rules consider a few global factors; dynamical selection of a dispatching rule considers more global factors, but only a few local factors.

The global and local factors are too numerous to be considered directly for the determination of a job's priority. This information needs to be aggregated or reorganized before use. The reorganization has to correspond to the scheduling objective too.

In the following, we list the global and local factors after reorganization when the objective is to minimize cycle time. The global factors are

- Number of unavailable machines (breakdown and maintenance) in each machine group,
- Queue length before each machine group,
- Sum of jobs' processing time in the queue before each machine group,
- Mean waiting time of jobs in the queue before each machine group,
- Work in process (WIP) level of each product,
- Sum of jobs' progress ratio grouped by product type ($ratio = n_{finished\,step} / n_{total\,step}$).

The amount of global factors can be calculated by $4n_M + 2n_P$, where n$_m$ is the machine number and n_p is the product number. The local factors include

- Processing time,
- Setup time,
- Waiting time,
- Remaining step number,
- Total number of steps,
- Sum of processing time at remaining steps,
- Raw processing time,

The local factors determine the priority directly while the global factors influence the priority indirectly. The global factors set the weights of local factors first. The weights of local factors denote the importance of each local factor for decision-making. For example, with the same objective the processing time dominates the priority in one state, but the remaining processing time plays the most important role in another state. Sometimes it is possible that in one state the shorter the processing time leads to a higher priority, but in another state the longer processing time, results in a higher priority. These phenomena depend on the global factors. The relations between local and global factors are presented below.

$$A = f''(G_{global}) = \{a_1, a_2, a_3, ...\}$$
$$prio = f'(AG_{local}) = f'(a_1 g^1_{local}, a_2 g^2_{local}, a_3 g^3_{local}, ...)$$

where A is the weight vector. Different values of the global factors results in different A vectors. The dispatching problem is reformulated to determine the functions f' and f''.

2.2 Intelligent Dispatching

There is no mathematical approach available yet to obtain the functions f' and f''. We use a clustering method to replace function f''. According to the values of the global factors, the state of the manufacturing line is divided into several patterns. Consequently, the function f' is divided into the appropriate subfunctions, too. Each pattern has one relevant subfunction to calculate the priorities. We use neural networks as subfunctions. For each pattern, G_{global} varies slightly and A is nearly constant. So in the same pattern, the global factors can be ignored. For pattern *i*,

$$prio = f_i'(AG_{local}) \square f_i'''(G_{local}) = neuralnetwork_i(G_{local}).$$

We develop one dispatcher for each machine. According to the global factors, the dispatchers divide the state of the manufacturing system into several patterns, and we build a neural network for each pattern to map the relationship between the local factors and the priorities of jobs. When making a decision, the dispatchers collect the global factors and decide which pattern the state belongs to in the pattern pool. The pattern and the network have a one-to-one relationship. Thus the appropriate network is selected from the network pool and used to compute the jobs' priorities according to values of their local factors. The job with the highest priority will be assigned. The process flow is shown in Figure 1.

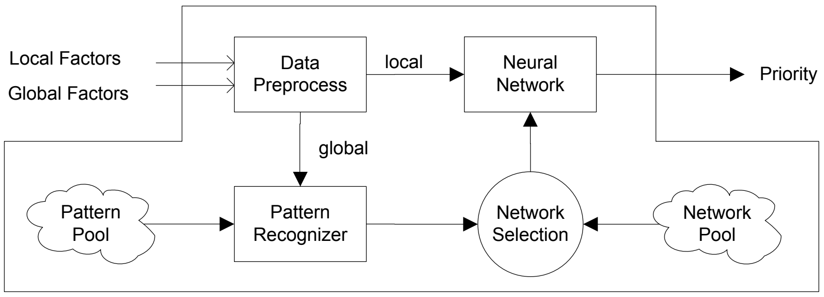


Figure 1: A dispatcher for one machine

The pattern pool is made up by each pattern's centroid which is determined by clustering plenty of data on global factors. Mean values of data which are classified into the same class compose the class' (pattern's) centroid. The pattern recognizer calculates the distances from current values of global factors to each centroid. The state will fit the pattern whose centroid is closest to the current values of the global factors. As mentioned above, for each pattern we create one network. All trained networks make up the network pool. The networks are trained by certain amount of data on local factors and corresponding data on the priorities. The dispatching problem is therefore converted into building the appropriate pattern pool and network pool.

3 DATA ACQUISITION

The pattern pool and the network pool are built by two data-driven models, a clustering model and a neural network model. How to obtain the required data is the first question in our work.

3.1 Required Data

Three types of data need to be collected: global factors, local factors and the priorities corresponding to each record of local factors (i.e., each job's information). Because there must be more than one job and only one priority is determined for each job when we make decision, one record of the global factor data must correspond to more than one record of the local factor data. One record of the local data has only one priority value. The global factor data is used for clustering. The local factor data and the priority data are used for supervised training of neural networks. All of data is grouped by machines. Clustering and training of neural networks, will be carried out in each group respectively.

3.2 Collect Data from the Simulation Model

The global factor data and local factor data are easy to obtain from the manufacturing system. But the priority data is impossible to determine from a real manufacturing system. For research purpose, we use a simulation approach to get all required data.

The simulation approach is shown in Figure 2. A main simulation is used to simulate the manufacturing line. At each decision making point, the main simulation is interrupted and current values of global factors (one record) and local factors (many records) are saved. Then a subsimulation with the same model and the same state is built for each job in the queue. One subsimulation selects one job from the queue and starts the simulation. The sub simulations end when the effects of the decision on the manufacturing line disappear. The results of each simulation will be used to calculate the priority of each related job. During the sub simulation, the dispatching decision is made by a base dispatching rule, and each stochastic event such as job release, machine breakdown occurs in all subsimulations at the same time. After all subsimulations end, the main simulation continues to the next timepoint of interest. The simulation continues in the same manner until the main simulation reaches the maximal simulation time.

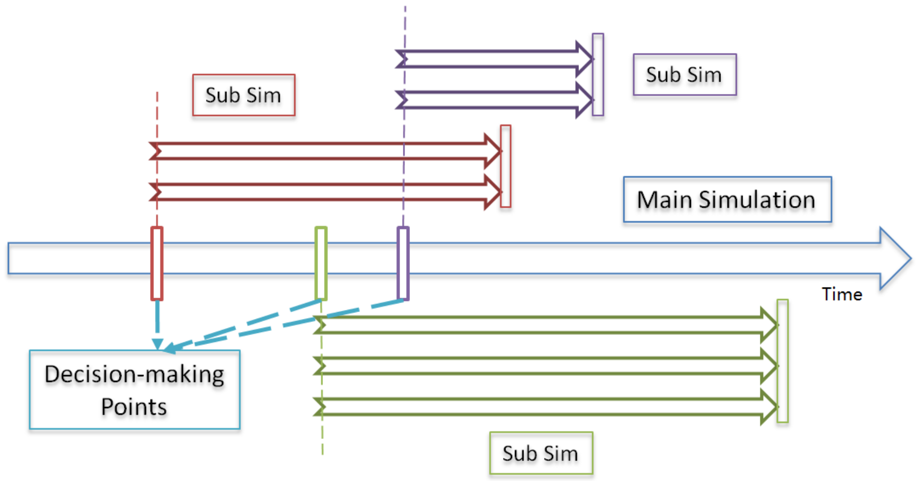


Figure 2: Data acquisition from the simulation

3.3 Two Important Issues of the Data Acquisition

There are two important issues of the data acquisition. (1) How long should the subsimulation run? (2) How should we get the priority from the subsimulation result. Theoretically, the subsimulation ends when the effect of current decision disappears. But it is hard to determine this time. Thus, the period of sub simulation is decided by the number of completed jobs n after the decision timepoint.

$$n = m + WIP/2$$

where m is the number of completed jobs from the decision-making timepoint to the time that jobs in the queue at the decision timepoint are finished. WIP is the work in process level at the decision point. The priority is related to the scheduling objective. Here, we give two formulas under two common objectives. While the objective is to minimize the cycle time,

$$prio = 1/\sum_{p\in P}(\omega_p \sum_{j\in J_p} c_j / n_{J_p})$$

where p denotes a product; P is the set of products. ω_p is the weight of product p. j is a job. J_p is the set of completed jobs whose product type is p. c_j is job j's cycle time. n_{J_p} is the number of jobs in the set J_p.

While the objective is to minimize total weighted tardiness

$$prio = 1/\sum_{p\in P}(\omega_p \sum_{j\in J_p} \max(C_j - d_j, 0)/n_{J_p})$$

where C_j is job j's completion time and d_j is job j's due date.

4 CLUSTERING

In Section 3, all required data is collected by the simulation. This section will discuss how to cluster the global factor data so as to classify the state of manufacturing line into several patterns. Because the amount of global factors is in direct proportion to the number of machines and products, it may be a considerable effort. Thus, spectral clustering (Luxburg 2007) is used to reduce the dimension first, and then the K-means approach is used for clustering. Spectral clustering techniques make use of the spectrum (eigenvalues) of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions. The Euclidean distance is selected to be the similarity function. The similarity graph is built by k-nearest neighbor method. The normalized graph Laplacian is used. The algorithm of the spectral clustering is given here.

4.1 Building the Similarity Graph

We treat each record of global factor data as a data point. The collected data is represented by a matrix X_{global}. Rows of X_{global} correspond to points *x*, columns correspond to global factors *g*. *N* is the number of points needed to be clustered; *m* is the number of global factors. The *k*-nearest neighbor method is used to build the similarity graph *W*. x_i is connected to x_j if x_j is one of *k*-nearest neighbors. In this case, $w_{i,j}$ is the Euclidean distance between points x_j and x_i,

$$w_{i,j} = \sqrt{\sum_{k=1}^{M}(g_{i,k} - g_{j,k})^2} .$$

If x_j is not in the *k*-nearest neighbors of x_i, $w_{i,j} = 0$.

$$X_{global} = \begin{array}{c|cccc} & & & & \\ x_1 & g_{1,1} & g_{1,2} & \cdots & g_{1,M} \\ x_2 & g_{2,1} & g_{2,2} & \cdots & g_{2,M} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N & g_{N,1} & g_{N,2} & \cdots & g_{N,M} \end{array} \qquad W = \begin{array}{c|cccc} & x_1 & x_2 & \cdots & x_N \\ x_1 & 0 & w_{1,2} & \cdots & w_{1,N} \\ x_2 & w_{2,1} & 0 & \cdots & w_{2,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N & w_{N,1} & w_{N,2} & \cdots & 0 \end{array}$$

4.2 Reducing Dimension According to Jianbo and Jitendra (2000)

The dimensionality reduction is achieved by calculating eigenvectors of a Laplacian matrix *L*, $L = D - W$, where *D* is a diagonal matrix and $d_{i,i} = \sum_{j=1}^{N} w_{j,i}$. Then we compute the first *k* generalized eigenvectors *u*$u_1,\ldots,u_k$ corresponding to the first *k* smallest eigenvalues of the generalized eigen-problem $Lu = \lambda Du$. Let *V* be the matrix containing *k* eigenvectors as column. The matrix *V* is the result of the dimensionality reduction.

$$V = \begin{array}{c|cccc} & u_1 & u_2 & \cdots & u_k \\ x_1 & v_{1,1} & v_{1,2} & \cdots & v_{1,k} \\ x_2 & v_{2,1} & v_{2,2} & \cdots & v_{2,k} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N & v_{N,1} & v_{N,2} & \cdots & v_{N,k} \end{array}$$

4.3 K-means Clustering and Centroid Calculation

Now the *M*-dimension global factor data X_{global} has been reduced to *k*-dimension data *V*. Using *K*-means method to cluster *V*, in which each row denotes one point, into *k* classes, $V = \{V_1, V_2, \ldots, V_k\}$. The data X_{global} is divided into *k* classes, $X_{global} = \{X_1, X_2, \ldots, X_k\}$, where $X_j = \{x_i \mid v_{i,*} \in V_j\}$; $v_{i,*}$ denotes the *i*-th row of *V*. Then we calculate centroids *C* of classes according to the global factor data in the corresponding class, $C = \{C_1, C_{2,} ..., C_k\}$. The centroids make up the pattern pool and will be used in the pattern recognition. A point belongs to class *i* if the distance between the points and the centroids C_i is the shortest.

5 NEURAL NETWORKS

A neural network (Haykin 2009) is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on a connectionist approach to computation. In more practical terms neural networks are non-linear statistical data modeling or decision making tools. Feed forward networks can be used for any kind of input to output mapping. A feed forward network with one hidden layer and enough neurons in the hidden layers can fit any finite input-output mapping problem. Here, we use it to map the relationship between the local factor and the priority.

5.1 Structure of Networks

Three-layer feed forward networks are introduced into the model. The inputs *X* are the local factors and the output *Y* is the priority. The number of nodes *J* in the hidden layer can be calculated by $J = \sqrt{K+M} + b, 0 < b < 10$, where *K* is the output node number; *M* is the input node number. We use the tangent sigmoid transfer function $f_h(x) = 1/(1+e^{-x})$ in the hidden layer and linear function $f_o(x) = x$ in the output layer. The sum of square errors (SSE) is defined to evaluate the training process

$$SSE(w) = \sum_{n=1}^{N} e_n \text{ , where } e_n = \frac{1}{2}\sum_{k=1}^{K}(y_{n,k} - o_{n,k})^2 \text{ ,}$$

$$o_{n,k} = f_o(\sum_{j=1}^{J} w_{j,k} h_{n,j} + w_{0,k}) \text{, } h_{n,j} = f_h(\sum_{m=1}^{M} w_{m,j} x_{n,m} + w_{0,j}) \text{.}$$

where *n* is the index of training data, *N* is the data number. e_n denotes the error of the *n*-th training data. The symbols k , *j, m* denote the node in the output layer, in the hidden layer, and in the input layer respectively; $y_{n,k}$ is the target output of the node *k* in the output layer. $o_{n,k}$ is the actual output. $h_{n,j}$ is the node *j*'s output. $x_{n,m}$ is the data of the node *m* in the input layer. $w_{0,k}$ and $w_{0,j}$ are biases of the node *j* in the hidden layer and the node *k* in the output layer. $w_{j,k}$ is the weight between the node *j* in the hidden layer and the node *k* in the output layer.

5.2 Training

We adopt the Levenberg-Marquardt algorithm to train the networks in batches. The Levenberg-Marquardt algorithm combines the steepest descent method and the Gauss-Newton algorithm. It inherits the speed advantage of the Gauss-Newton algorithm and the stability of the steepest descent method. A mathematical description of the LM neural network training algorithm has been presented by Hagan and Menhaj (1994). The formula of updating weights and biases is

$$w_{s+1} = w_s - (Jac_s^T Jac_s + \mu I)^{-1} Jac_s^T E_s$$

where *I* is an identity matrix, μ is the combination coefficient. The combination coefficient μ is adjusted automatically according to the error during the training. If the error increases after weights and biases updates, μ is increased by a certain factor; if the error decreases, μ is reduced by the same factor. Jacobian matrix Jac is shown as following. It can be computed by using the chain rule of calculus and the first derivatives of the transfer functions.

In addition, the early stopping technique is used to avoid over-fitting. The data is divided into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error will normally decrease during the initial phase of training. However, when the network begins to overfit the data, the error on the validation set will typically begin to rise. When the times of validation error increasing reach a specified number of iterations, the training is stopped, and the weights and biases at the minimum of the validation error are returned.

$$
Jac = \begin{array}{c} \begin{array}{cc} \textit{Hidden Layer} & \textit{Output Layer} \end{array} \\ \begin{bmatrix} \partial e_1/\partial w_{0,j} & \partial e_1/\partial w_{i,j} & \partial e_1/\partial w_{0,k} & \partial e_1/\partial w_{j,k} \\ \partial e_2/\partial w_{0,j} & \partial e_2/\partial w_{i,j} & \partial e_2/\partial w_{0,k} & \partial e_2/\partial w_{j,k} \\ \vdots & \vdots & \vdots & \vdots \\ \partial e_N/\partial w_{0,j} & \partial e_N/\partial w_{i,j} & \partial e_N/\partial w_{0,k} & \partial e_N/\partial w_{j,k} \end{bmatrix} \end{array}, \quad E = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix}.
$$

In Section 4, the global factor data is divided into k classes. Consequently, the local factor data and the priority data are grouped into *k* groups due to one record of local factor data relates to one record of global factor data. Based on each group of local factor data and the priority data, one neural network is trained. These networks make up the network pool mentioned in Section 2. Combining all above components solves the dispatching problem.

6 APPLICATION

In this section, the intelligent dispatching approach is used in a manufacturing system. The system contains 6 machine groups, 24 machines and produces 4 products with 4 process flows and 4 different outputs. There are no batch processing machines. Interval times of releasing jobs follow the normal distribution. Both sequence dependent and independent setups are needed. The interval between two breakdowns is subject to the exponential distribution and the repairing time follows an exponential distribution too. The objective of scheduling is to minimize the cycle time.

6.1 Data Acquisition, Clustering and Neural Network Training

32 global factors and 5 local factors are collected at each decision point. The priority of each job is calculated from the subsimulation results. The main simulation runs 100 days; there are 15800 decision-making points; subsimulations run 58980 times; the base dispatching rule is SPT during the subsimulations. We obtain a total of 15800 records of global factor data and 58980 records of local factor data and priority data. The data is grouped by machine groups. For each machine group one dispatcher is built. We use the try-method to obtain the suitable class number of the global factor data for each machine group, shown in Table 1.

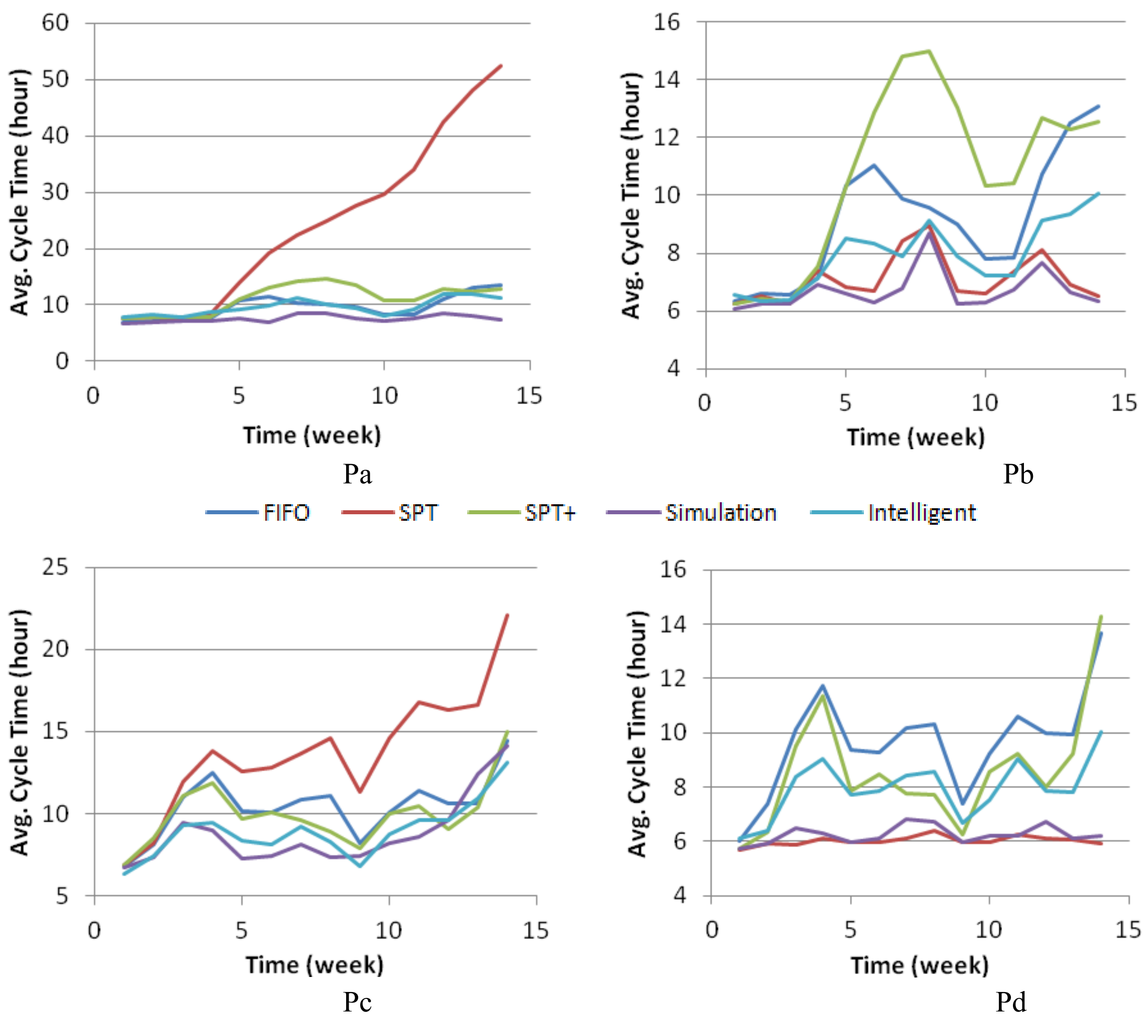
The neural networks have 5 inputs and 1 output. The number of nodes in the hidden layer is 6. After testing, the average accuracy of the 117 networks is 75.32%.

Table 1: class number for each machine group

Machine	MA	MB	MC	MD	ME	MF
Class Num	12	14	25	28	18	20

6.2 Results and Comparison

We run the simulation one more time using the built dispatchers to obtain the performance measure cycle time (statistics values and trend chart). Then we compare the intelligent dispatching approach with the dispatching rules including first in first out (FIFO), shortest processing time (SPT) and SPT+. SPT+ is an improved version of SPT. In the SPT+, we set a maximal waiting time. If a job's waiting time is longer than the maximal waiting time, the job will be dispatched first so as to avoid that the job with the long processing time will have to wait too long. In addition, after data acquisition, we receive an integrated schedule in which the job with highest priority is always selected at each decision making point. We put it into the comparison too and call the approach "simulation". The proposed approach is called "intelligent" approach here. The results are show in Figure 3 and Table 2.



Finger 3: Tendency chart of cycle time grouped by product types (Pa, Pb, Pc, and Pd)

Table 2: Average cycle time of jobs grouped by product type

Rule \ Product	Pa	Pb	Pc	Pd	Summary
FIFO	8.78	8.28	9.51	8.80	8.85
SPT	24.34	6.09	12.94	5.02	13.80

SPT+	10.25	9.83	9.07	7.76	9.49
Simulation	6.55	5.68	7.87	5.25	6.50
Intelligent	8.23	7.07	7.98	7.03	7.38

SPT is an optimal rule for minimizing the cycle time in the single machine problem. In this case, we can see that SPT performs worse than any other approaches for products Pa and Pc. The reason is that Pa and Pc have a longer raw processing time than Pb and Pd. When we use the SPT+ rule, the cycle times of products Pa and Pc go down, but go up for products Pb and Pd. The simulation approach is the best for products Pa, Pb and Pc and the second best for the product Pd. The intelligent approach performs more stably than FIFO, SPT and SPT+ among the products. From the summary column in Table 2, we can see that the intelligent approach is the second best one. The single machine optimal rule SPT is the worst one.

7 CONCLUSION

Using artificial intelligence techniques, we create a specific dispatcher for each machine. The dispatchers consider a very large amount of information – including global factors and local factors – in the decision-making process. Thus the dispatchers have a wide scope. Because the dispatchers are built from a data-driven model, for a given manufacturing line we just need to create its simulation model and to collect required data. A complicated theoretical analysis can be avoided. The spectral clustering is used for dimensionality reduction. It improves the efficiency of the clustering. We use neural networks to map the relationship between the local factors and the priorities of jobs. The neural network can easily be converted to online learning which is on our agenda for further work. Finally, the proposed approach is introduced into a manufacturing line and performs better and more steadily than dispatching rules FIFO, SPT and SPT+. Because the intelligent approach actually learns its behavior from the simulation model and there must be error in the neural networks, it is impossible that the intelligent approach performs as well as the simulation approach. But the intelligent approach is more efficient than the simulation approach due to the time which has to be spent on the subsimulations.

REFERENCES

Chen, B. and T. I. Matis. 2013. "A Flexible Dispatching Rule for Minimizing Tardiness in Job Shop Scheduling." *International Journal of Production Economics*. 141: 360-365.

Hagan, M. T. and M. B. Menhaj. 1994. "Training Feedforward Networks with the Marquardt Algorithm." *IEEE Transactions on Neural Networks*. 5: 989-993.

Haykin, S. S. 2009. Neural Networks and Learning Machines. 3rd ed. New York: Pearson Prentice Hall.

Holthaus, O. and C. Rajendran. 1997. "Efficient Dispatching Rules for Scheduling in a Job Shop." *International Journal of Production Economics*. 48: 87-105.

Jianbo, S. and M. Jitendra. 2000. "Normalized Cuts and Image Segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 22: 888-905.

Lian, Y., H. M. Shih and T. Sekiguchi. 1998. "Dynamic Selection of Dispatching Rules by Fuzzy Inference".In *proceedings of the IEEE International Conference on Fuzzy Systems*, 979-984, Piscataway, New Jersey: IEEE, Inc.

Luxburg, U. 2007. "A Tutorial on Spectral Clustering." *Statistics and Computing*. 17: 395-416.

Scholz-Reiter, B., J. Heger and T. Hildebrandt. 2010. "Gaussian Processes for Dispatching Rule Selection in Production Scheduling: Comparison of Learning Techniques".In *proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW)*, 631-638, Piscataway, NJ: IEEE Inc.

AUTHOR BIOGRAPHIES

TAO ZHANG is a Ph.D. student working on production planning and scheduling at the Department of Computer Science of the Universität der Bundeswehr München, Germany. From 2007 to 2009 he received his Master in metallurgical engineering with the subject of production planning and scheduling in iron and steel industry from Chongqing University, China. He is involved in modeling and simulation of complex system and intelligent optimization algorithms. His email address is tao.zhang@unibw.de.

OLIVER ROSE holds the Chair for Modeling and Simulation at the Department of Computer Science of the Universität der Bundeswehr, Germany. He received a M.S. degree in applied mathematics and a Ph.D. degree in computer science from Würzburg University, Germany. His research focuses on the operational modeling, analysis and material flow control of complex manufacturing facilities, in particular, semiconductor factories. He is a member of IEEE, INFORMS Simulation Society, ASIM, and GI, and has been the General Chair of WSC 2012. His email address is oliver.rose@unibw.de.