[Regular Paper]

A Modification Module for Robust Schedule Generation System of Batch Plants

Wei CHEN, Tetuo FUCHINO, and Masaaki MURAKI*

Dept. of Industrial Engineering and Management, Tokyo Institute of Technology,
2-12-1 Ookayama, Meguro-ku, Tokyo 152

(Received September 11, 1996)

The focus of this paper is to develop a Modification Module (MM) to support the robust schedule generation system in batch process management. Since the tightly coupled nature of schedules in the processes, any local improvement is not effective. Therefore, MM should generate a set of modification tactics to relax the detected multiple bottlenecks simultaneously based not only on the bottleneck information but also on scheduling one. This function is categorized in the pattern recognition problem, thus a backpropagation neural network (BPNN) is a powerful technology applicable to it. In this study, its application is discussed with the emphasis on the following three key aspects. First, to reflect the above tightly coupled nature, global and local features of schedule are examined as the scheduling information of the input layer. Second, due to the processing time variability, the available input data to MM become fuzzy information, and it is “discretized” based on the property of this scheduling phase. Third, to handle the discretized fuzzy input data, the standard backpropagation learning algorithm is extended. The effectiveness of the proposed MM is demonstrated through simulation experiments on a multiproduct batch process model.

1. Introduction

The interest in the scheduling methodology in the petroleum refining and petrochemical processing context has been intensified and stimulated particularly in the last decade, in part due to the major changes in the business environment brought about by the industry and also by changes in the information and computer technologies that can be deployed to solve these highly complex decision problems. The main objective of production scheduling is to satisfy the demand for different products under certain constraints (time horizon, limited resources) in the most efficient manner. Scheduling techniques usually consider that the plant status given at the time of planning will remain valid for a substantial period of time, while in real cases these conditions may undergo significant variations during the execution of plan; accordingly, the production schedules may become inefficient and eventually unfeasible. In this sense, robust schedules should have, first of all, a high probability of being performed; secondly, if plant conditions undergo variations (*e.g.*, processing time variability), they should be adapted

with as few modifications as possible to new conditions. Thus, a robust schedule will reduce the effect of uncertainty. These issues have also been addressed in the chemical processing industry and the chemical engineering profession^{1)~4)}.

Recently, authors have proposed a generic framework for that Robust Schedule generation System (RSS) of the following three modules⁵⁾: (1) Initial Schedule Generation Module (ISGM), (2) Check and Evaluation Module (CEM), (3) Modification Module (MM).

ISGM generates an initial schedule based on the most likely values of processing time. Several related research works^{2),3)} can be applicable. After the initial schedule is generated by ISGM, CEM checks the schedule feasibility, evaluates against robustness and performance, and then informs the evaluation results (*e.g.*, bottleneck information, scheduling information, etc.) to MM. Due to the variations in processing time and interaction among the tasks, a fuzzy-set-based method to evaluate the robustness and to detect the bottlenecks of the current schedule is proposed⁶⁾. In CEM, the robustness at each position between two adjacent tasks in the schedule is evaluated by the fuzzy-set-based possibility of conflict by processing time variability. When the robustness of a

* To whom correspondence should be addressed.

position is less than the threshold value given beforehand, the position is considered as a bottleneck. The robustness distribution is obtained by collecting the robustness values of all the positions in the whole schedule. Due-date constraint is also considered as one of time constraints in CEM.

MM then modifies the current schedule based on the evaluation results from CEM and activates CEM again. When the schedule is still unsatisfied or inefficient, CEM activates MM. Otherwise, CEM registers the modified schedule as the robust schedule for the on-line scheduling system^{3),4)}.

There exists a tightly coupled nature of schedules in batch processes, then such approaches as min-conflict heuristic⁵⁾, simulated annealing⁶⁾, improving the schedule by one modification tactic (e.g., time-shifting, swap, etc.) to a certain bottleneck, are ineffective in MM. Therefore, a basic strategy that selects a set of modification tactics based on the bottleneck and scheduling information to relax a group of bottlenecks simultaneously is necessary.

To select a set of tactics to relax the group of bottlenecks based on evaluation results from CEM, MM based on a neural network, which is a potential technology for pattern recognition, is proposed in this paper. On the basis of the characteristics of the current problem such factors as (1) selection of input attributes from available information, (2) discretization of input data, (3) extension of standard backpropagation algorithm, are addressed. The effectiveness of the proposed module is demonstrated through simulation experiments in a multiproduct batch plant.

The remainder of this paper is organized as follows. In section 2., the basic strategy of MM is examined. In section 3., the principal components of the current MM architecture are described. In section 4., the simulation results are discussed. Finally, in section 5. we conclude this paper.

2. Basic Strategy of MM

Figure 1 shows a part of the modified initial schedule from MM in the experiment¹⁾ in which all processing times take on their most likely values. In this example, the critical bottleneck threshold value is given as 0.5, i.e., a position between the tasks with a lower robustness 0.5 meaning a bottleneck. There are two bottlenecks ($R(C_p)=0.24$ and $R(C_q)=0.29$) as shown in **Fig. 1** in hatched circles. $R(C_p)$ means the robustness of a constraint (C_p) at position p . This information is transferred to MM to relax the bottlenecks of the

Fig. 1 Part of the Current Schedule in MM

Fig. 2 Part of the Modified Schedule by Only Considering Bottleneck p

Fig. 3 Part of the Modified Schedule by Considering Bottleneck p and q

current schedule. **Figure 2** shows part of the modified schedule by shifting batch 2 of product 5 right with 6 time units to relax bottleneck p only. Although the bottleneck p is removed individually, bottlenecks l , m , n , and q have appeared. **Figure 3** shows part of the modified schedule by swapping batch 1 of product 5 and 2 for bottleneck p and swapping batch 1 of product 6 for batch 2 of product 5 for bottleneck q . Bottlenecks p and q are relaxed, and their values of robustness then become 0.62 and 0.64, respectively. Only one bottleneck r appeared newly while three bottlenecks l , m , n , appeared as shown in **Fig. 2**.

From this example, it is clear that relaxing the group of bottlenecks simultaneously is effective. As a basic modification strategy, it is necessary for MM to select a set of modification tactics for corresponding bottlenecks simultaneously, and it is also important to consider the scheduling information, which reflects the regularities of modification tactics. The selection of a set of suitable modification tactics to a group of bottlenecks in MM is essentially a pattern recognition problem that maps each set of scheduling information into a set of modification tactics. For example, when a specific set of information is given, corresponding tactics will be presented. It is ideal to find a perfect classification model necessary to classify every set of information into a set of corresponding modification tactics. Therefore, it will be necessary to obtain a proper classification model to minimize the average errors between desired and obtained sets. The neural network approach (NN) is a potential and anticipated technology applicable to MM due to the nature of their parallel structure of simple computational elements and good classification performance.

In this research, an extended backpropagation neural network (E-BPNN) based MM is proposed in the next section for selecting a set of tactics. Three aspects should be addressed on its application. Firstly, to reflect the modification regularities, features (*e.g.*, operation time of each equipment, slack time between tasks, etc.)⁷⁾ should be considered for scheduling information. Secondly, the above information has a fuzzy nature, but it is sufficient for this problem to change the fuzzy information to discrete data since the robust schedule generation system is in the stage of planning. α -Level set representation⁸⁾ is used to represent the input data because it can represent various type of information, and the computation load is light. Finally, the standard backpropagation learning algorithm is also extended to handle the fuzzy input represented by α -level set representation.

3. Application of BPNN to MM

3.1. Information Data Structure of MM

An information data structure to describe the characteristics of a set of particular modification tactics to relax bottlenecks is required to support the robust scheduling system. The bottleneck information directly reflects the robustness distribution of the current schedule, and the scheduling information reflects the flexibility of modification, defined with room for moving a task to a new

position in the current schedule. As mentioned by Chen and Muraki¹⁾, the scheduling information is of global and local features in which the former feature reflects the potential flexibility of modification for the schedule as a whole. Four global features (system congestion, equipment utilization, equipment utilization deviation, batch late factor)^{1),8)} are utilized in this research. For example, high equipment utilization often indicates a tight schedule without much flexibility for modification. High standard deviation of equipment utilization indicates the presence of high contended-for equipment that in turn indicates low flexibility of modification. The latter features reflect flexibility for schedule modification within the limited modification time horizon. The modification time horizon is defined as the interval that temporally spans the task in bottleneck. Two local features (task slack, task tardiness)^{1),8)} are used in this research. For example, high slack predicts that it may have opportunities for resequencing to relax the bottlenecks. Each value of input vector is obtained as illustrated by Chen and Muraki^{1),8)}.

Bottlenecks are chosen as a bad order on robustness. The number to be relaxed simultaneously can be adjusted according to the magnitude of the scheduling problem. In this research, four bottlenecks are selected in one repetition.

3.2. Fuzzy Data

3.2.1. Input Description

Due to the processing time variability, the above information data to NN from CEM are fuzzy, and can be represented using triangular, trapezoidal or Gaussian functions for this scheduling phase.

When a fuzzy pattern (S_i ; $i=1, 2, \dots, m$) is informed to i -th input node, these patterns can be described by α -level sets, which provide the means to depict any fuzzy number as a collection of intervals. As shown in Fig. 4, given a fuzzy number S_i with an associated membership function μ_{S_i} and $\alpha_c \in [0, 1]$, ordinary subsets of level α_c of S_i is called:

$$S_{i\alpha_c} = \{x: \mu_{S_i}(x) \geq \alpha_c\} \quad c = 1, 2, \dots, h \quad (1)$$

This corresponds to the interval of values for which the evaluated membership function is greater than α_c . For the purpose of neural network training, a number h of α -level sets is defined and the corresponding intervals are generated for each fuzzy number. This process creates additional sample points around the original training patterns depending on the degree of fuzziness associated with them.

It is clear from the definition of Eq. (1), $\mu_{S_i\alpha_c}$ can

Fig. 4 Fuzzy Description of the Input Data

Fig. 5 Fuzzy Description of the Output Data

be converted to Eq. (2) by representing $S_{i\alpha_c}$ for each level α_c as shown in Eq. (3).

$$\mu_{S_i}(x) \approx \bigcup_{c=1}^h S_{i\alpha_c} \quad (2)$$

$$S_{i\alpha_c} = [S_{i\alpha_c}^L, S_{i\alpha_c}^U] \quad i=1, 2, \dots, m; c=1, 2, \dots, h \quad (3)$$

where the superscripts L and U are used to represent the lower and upper limit of an interval, respectively.

Thus, it becomes possible to apply the E-BPNN in which the traditional BPNN is modified by adding two connections to the processing units (neurons), representing the lower and upper limits of an interval passing through the network. Each pair of connections has an associated weight value, which represents the strength of the link.

3. 2. 2. Output Description

For the k output node O_k of level α_c :

$$O_{k\alpha_c} = [O_{k\alpha_c}^L, O_{k\alpha_c}^U] \quad k=1, 2, \dots, n; c=1, 2, \dots, h \quad (4)$$

where $\alpha_c \in [0, 1]$, $O_{k\alpha_c}^L$, $O_{k\alpha_c}^U$ represent the lower and upper limits of E-BPNN outputs O_k and correspond to the fuzzy input pattern (S_i ; $i=1, 2, \dots, m$) on the α_c ($c=1, 2, \dots, h$) level as shown in Fig. 5.

During the recall phase, a pattern (S_i ; $i=1, 2, \dots, m$) is presented to the input layer using α_c level sets ($c=1, 2, \dots, h$). The E-BPNN generates intervals of number h in each output neuron representing the degree of membership of the input pattern to each class.

Once a set of intervals is produced in the output layer, a defuzzifying algorithm known as the "centroid method" is applied to generate a single value in each output neuron. The centroid is computed by finding a variable value of the center of gravity of the membership function for the fuzzy number. Thus, given a fuzzy output response O_k

whose membership function is represented by number h of α -level sets as shown in Fig. 5, the centroid method produces:

$$O_{kk} = \frac{\sum_{c=1}^h \alpha_c [O_{k\alpha_c}^L + O_{k\alpha_c}^U]}{\sum_{c=1}^h \alpha_c} \quad (5)$$

where O_{kk} is the centroid of the output neuron k .

After completing the defuzzifying process, a threshold value is assigned, and the neuron with the highest activation is declared as a winner. In the case of patterns in overlapping zones, the network produces more than one winning neuron representing the degree of the uncertainty in the decision. The neuron with the highest activation is declared as the absolute winner and the second best is taken as an option.

The following four modification tactics adopted here are: *left/right shift* task on the same (or substitutable) unit to move a task in the bottleneck as much as possible within the modification time horizon; *swap* the task in the bottleneck within the time horizon with the task on the same (or substitutable) unit, while maximizing robustness; *give up* a further modification of current task. Note that the modification tactics can be extended or reduced, if necessary. Then, the four candidate modification tactics are considered for one bottleneck relaxation. There are 16 output nodes in the current research.

3. 3. Learning Mechanism

Input fuzzy numbers are represented by interval vectors, so that to handle them in E-BPNN, the learning method should be extended from the standard backpropagation algorithm.

The error function to be minimized during learning is defined as the least mean square error to the fuzzy labels. Thus, for given the input fuzzy pattern (S_i ; $i=1, 2, \dots, m$), the error function should be computed by Eq. (6):

$$E = \sum_{c=1}^h \sum_{k=1}^n \frac{\alpha_c [\overline{\mu_{O_k}}(x) - O_k]^2}{2} \quad (6)$$

where $\overline{\mu_{O_k}}(x)$ represents the output target of the fuzzy input pattern ($S_i: i=1, 2, \dots, m$) to class k , and O_k is the output neuron k activation of one bottleneck.

The learning rate and the slope of the activation function are parameters of the standard back-propagation algorithm, which are defined empirically.

4. Evaluation of the Approach

In this section, the proposed MM is tested on a multiproduct batch process model.

4.1. Test Problems

Nine test cases of the same problem as described by Ishii and Muraki³⁾ are used here. In these experiments, a simple multiproduct batch process model consisting of four tasks as shown in **Fig. 6** is employed. Each task is assumed to be operated in a different unit. Eight kinds of products, which have the same task network but different processing time are specified. The mean processing time of each task for a product is defined randomly in each experiment based on a uniform distribution between 5 and 35 time units. The due date of each production order is specified randomly in each experiment based on a uniform distribution between 200 and 600 time units. For waiting time constraints, the following three scenarios referred to WTC1, WTC2, and WTC3 are specified.

- WTC1: Unlimited Wait (UW) constraint for any task
- WTC2: task 2 has a Zero Wait (ZW) constraint
- WTC3: all tasks except task 4 have a ZW constraint

In addition, a process model is developed by using a discrete event simulation technique. A feasible product list is generated based on the stock data and production orders. The production start timing of each order selected from the feasible order list is specified by a time-shifting logic⁹⁾ to assure waiting time constraints in constructing the initial schedule.

The processing time of each task is assumed fuzzy functions in which the variations have a randomly variability up to $\pm 25\%$ of the mean processing time.

4.2. Constructing the E-BPNN

The neural network is responsible for identifying a set of promising modification tactics to corresponding bottlenecks. The neural network

Fig. 6 A Multiproduct Process Model

structure consists of an input layer of 10 nodes and an output layer of 16 nodes including several hidden layers. Several parameters must be determined to obtain a complete neural network. Those parameters include a number of hidden layers, a number of hidden neurons at each layer, and a learning rate. Additionally, all of the weights on the arcs will empirically be determined from the training samples. 100 training samples are generated by simulation. The threshold value used to establish whether a neuron is firing or not is set at 0.7. The number of α -level sets is fixed at 2 (0.9 and 1.0). The stopping criterion is taken as an error value below 0.01. All weight values are initialized with random numbers between 0.0 and 1.0.

The experimental design and related simulation results used to obtain neural network parameters are summarized in **Table 1**. All experiments with one hidden layer are eliminated since they do not satisfy the stopping criteria. Therefore, the number of hidden layers is set at two. Four series of hidden neurons are then repeated. The learning rates are changed to 0.45, 0.50, 0.55, and 0.60.

As a result, several experiments satisfy the stopping criteria. Experiment 5, 6, 7, and 13 give 100% correct classification accuracy. Among these experiments, experiment 5 are selected as the final neural network since it offered the lowest mean system error.

4.3. Results of the Proposed MM

To illustrate the effectiveness of the E-BPNN based MM described above, a total of 1000 patterns are used for testing. The following parameters are measured:

- (1) *Crisp pattern*: A percentage value indicating the amount of patterns correctly classified by the system without having more than one output neuron firing above 0.7. This statistics measures the accuracy of the networks to correctly classify the patterns without any uncertainty.
- (2) *Fuzzy pattern*: a value indicating the percentage of patterns correctly identified by one of the two output neurons firing above 0.5 when the second option neuron is taken as the correct pattern. In this case, the neuron with the

Table 1 Experimental Results to Obtain the Final Neural Network Weights

Index	Number of hidden neurons	Learning rate	Minimum error	Average error	Maximum error	Accepted samples [%]
1	16×16	0.45	0.00000	0.04132	0.50192	48
2	16×16	0.50	0.00000	0.04294	0.50467	47
3	16×16	0.55	0.00002	0.04347	0.50219	38
4	16×16	0.60	0.00000	0.04358	0.51002	39
5	18×18	0.45	0.00000	0.00080	0.01000	100
6	18×18	0.50	0.00003	0.00179	0.01000	100
7	18×18	0.55	0.00001	0.00284	0.01273	100
8	18×18	0.60	0.00000	0.00682	0.06732	67
9	20×20	0.45	0.00000	0.03017	0.49899	78
10	20×20	0.50	0.00002	0.04810	0.54106	49
11	20×20	0.55	0.00000	0.03722	0.49011	61
12	20×20	0.60	0.00000	0.03017	0.49003	58
13	16×18	0.45	0.00006	0.00114	0.01000	100
14	16×18	0.50	0.00002	0.03407	0.49082	78
15	16×18	0.55	0.00000	0.03571	0.48018	76
16	16×18	0.60	0.00002	0.04188	0.48002	66

Experiment 5 is chosen as the final neural network.

Table 2 Results Obtained by Proposed MM (%)

Test case	1	2	3	4	5	6	7	8	9	Min	Ave.	Max
Crisp patterns	81.8	82.6	83.3	85.8	86.4	81.2	82.3	83.4	84.8	81.2	83.5	86.4
Fuzzy patterns	12.2	14.8	14.1	11.3	11.7	14.8	14.1	14.6	12.1	11.3	13.3	14.8
Total	94.0	97.4	97.4	97.1	98.1	96.0	96.4	98.0	97.9	94.0	96.9	98.1

highest activation has failed to declare the correct pattern. The statistics shows the ability of the network to manage uncertainty during the classification process.

Table 2 illustrates the performance of proposed MM on varying test problems. As can be seen from **Table 2**, MM gives not only good performance on a crisp pattern but can also suggest an alternative solution. It performs with average 83.5% on crisp patterns and with average 4.3 iterations. Because the highest output neuron has failed to recognize the appropriate modification pattern, MM is capable of suggesting an alternative solution in this case. The results obtained from **Table 2** also show that for these particular problems, on average up to 96.9% of the patterns can be identified in either the first or the second best firing neuron.

Thus, the effectiveness of the proposed MM to recognize a set of modification tactics to relax a group of bottlenecks is clearly demonstrated.

4.4. Comparison Study on Robust Schedule

To illustrate the effectiveness of MM, we should evaluate the robust schedule generated by RSS against the one from ISGM on both makespan ($M(S)$) and the schedule delay ($\delta(S)$). Due to the dynamic nature at schedule execution in real cases, thus, expected values $E[\delta(S)]$ and $E[M(S)]$ as criteria are estimated through simulation. The experiments consist of three main steps as follows:

Step 1: Generate a schedule S by ISGM or RSS.

Step 2: Simulate S under disturbance, and estimate $E[M(S)]$ and $E[\delta(S)]$ through 500 simulation runs.

Step 3: Compare and save $M_0(S)$, $E[M(S)]$, and $E[\delta(S)]$.

These experiments are set up to answer the following question:

- Is the robust schedule generated by proposed RSS after modification effective on maintaining good performance in the expected makespan $E[M(S)]$ while reducing the “dispersion” of makespan (*i.e.*, expected delay $E[\delta(S)]$). For this purpose, a timing-shifting policy is used as the controlling policy during the schedule execution.

Figure 7 summarizes the results of the experiments on $E[M(S)]$ and $E[\delta(S)]$ *vs.* problem, respectively. When analyzing the performance of robust scheduling system, it must emphasize on both $E[M(S)]$ and $E[\delta(S)]$. Thus, from **Fig. 7**, it is quite clear that the robust schedules generated by MM dominate the schedules generated only by ISGM in bi-criterion objective. This is an encouraging result suggesting that our system is not only capable of achieving good production performance, but also have the property of yielding less dispersion in makespan.

5. Conclusions

A modification module of a robust schedule

(a) Expected Makespan ($E[M(S)]$), (b) Expected Schedule Delay ($E[\delta(S)]$).

Fig. 7 Results of Simulation Experiments on MM

generation system in batch process management is presented in this paper. It is clarified that to modify the current schedule for generating a robust one, MM should select a set of modification tactics based on the bottleneck and scheduling information simultaneously to relax corresponding bottlenecks. This selection becomes a pattern recognition problem, and BPNN is found to be potential. This study enables to function MM based on BPNN by considering the following three key aspects. First, to reflect the modification regularities, group and local features are considered for scheduling information besides bottleneck one. Second, since the above system is in the stage of planning, α -level set representation is used as a discretization method to represent the fuzzy input data. Finally, the backpropagation learning algorithm is also extended to handle the above input data. The performance of proposed MM is demonstrated through simulation

experiments on a multiproduct batch process.

References

- 1) Chen, W., Muraki, M., *J. Chem. Eng. Jpn.*, **30**, 260 (1997).
- 2) Kondili, E., Pantelides, C. C., Sargent, R. W., *Comput. Chem. Engng.*, **17**, 211 (1993).
- 3) Ishii, N., Muraki, M., *Int. J. Prod. Res.*, **34**, 329 (1996).
- 4) Chen, W., Muraki, M., Jiang, W. S., "Advances in Management Science," International Academic Press, (1994).
- 5) Minton, S., Johnston, M., Philips, A., Laird, P., *Proceedings AAAI-90*, Boston, MA, 1990, p. 17.
- 6) Zweben, M., *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, 1990, p. 251.
- 7) Zimmermann, H., "Fuzzy Sets, Decision Making, and Expert Systems," Kluwer Academic Publishers, (1987).
- 8) Chen, M., Muraki, M., *Int. J. Prod. Res.*, in press (JEM/SAC/IJPR 96052).
- 9) Ishii, N., Muraki, M., *Comput. Chem. Engng.*, **20**, 217 (1996).

要 旨

バッチプラントのロバストスケジューリングシステムのための修正モジュール

陳 偉, 瀧野 哲郎, 村木 正昭

東京工業大学社会理工学研究科経営工学専攻, 152 東京都目黒区大岡山 2-12-1

本論文はバッチプロセスの運転管理におけるロバストスケジューリング生成システムのための修正モジュールを提案した。バッチプロセスでは処理時間の変動に対する現行スケジュールのボトルネックが複数存在し、プロセスの特徴から各ボトルネックはスケジュールを介して強い相互作用がある。このため部分的なボトルネックの解消は新たなボトルネックを派生し、その繰り返しでは効果的にロバストスケジュールを作成することはできない。この問題を解消するためには、ボトルネック群に対する修正案群を同時に選択する必要がある、ボトルネック間の相互作用から、パターン認識の問題に帰着できる。したがって、BPNN (Backpropagation Neural Network) が有効となり、

本研究はロバストスケジューリングシステムの特徴から以下の3点を工夫することでBPNNを用いた修正モジュールを開発した。(1) ボトルネックの相互作用を学習に反映させるために、スケジュールの全体情報(稼働率等)および局部情報(タスク間のスラック時間等)をBPNNの入力データとした。(2) ファジー情報をスケジューリングフェーズの考察に基づき離散化し α -レベル表現を可能にした。(3) (2)の α -レベル表現によるファジーデータを効率良く学習するためのアルゴリズムを提案した。開発したモジュールの有効性は多製品バッチプロセスを用いたシミュレーション実験を通じて示した。

.....

Keywords

Computer, Production, Batch plant, Robust schedule, Modification module, Neural network