Paper:

Deep Neural Network and Rule Based Framework for Distributed and Flexible Job-shop Scheduling

Guanghao Xu[*1], Huifang Li[*1], Ruitao Yang[*2], Fenxi Yao[*1]

[*1]School of Automation, Beijing Institute of Technology, Beijing 100081, P. R. China
E-mail: 3220200812@bit.edu.cn; huifang@bit.edu.cn; yaofenxi@bit.edu.cn
[*2]The 28th Research Institute of China Electronics Technology Group Corporation, Nanjing 210007, P. R. China
E-mail: 349705234@qq.com
[Received 00/00/00; accepted 00/00/00]

Abstract. With mass customization as a state-of-the-art production paradigm under the economic globalization situations, Distributed and Flexible Job-shop Scheduling (DFJS) becomes more attractive. Taking the idea of "data-driven intelligence", while considering the complexity of distributed and flexible manufacturing system environments under the mass customization or even one-of-a-kind production, this paper proposes an intelligent optimization algorithm based on Deep Neural Networks (DNNs) and heuristic dispatching rules. Firstly, a 3-stage scheduling model including job scheduling, operation scheduling and operation ranking, which constitutes an integrated scheduling framework is established. Then, a DNN-based scheduling algorithm is proposed to optimize job scheduling and operation scheduling. Finally, we introduce a rule-based method to optimize operation sequencing. By integrating the above two methods, the whole job-shop scheduling process can be optimized simultaneously, so as to minimize the average tardiness penalty for the whole job set.

Keywords: Distributed and flexible job-shop, Scheduling, One-of-a-kind production modes, Deep Neural Networks

1. INTRODUCTION

With the globalization of economy, Distributed Manufacturing (DM) emerges and gradually becomes the main trend of modern production [1]. DM means multiple Flexible Manufacturing Units (FMUs) are geographically distributed and each FMU equipped with several multi-purpose machines represents a factory or a cell with the capability of processing various types of jobs concurrently or independently. On the other hand, DM promotes the development of massive customized or even one-of-a-kind production as its high flexibility, agility and ease to meet the individualized needs of customer. All of above greatly increases the complexity of DM scheduling and decision problems, and how to allocate task to its most suitable resource becomes crucial to achieve the optimal resource usage. Therefore, Distributed and Flexible Job-shop Scheduling (DFJS) problem attracts more attentions from researchers and practitioners [2].

Many meta-heuristic based methods are proved to perform well in some scheduling scenarios [3], [4], [5] and [6], and obviously effective in optimization. Giovanni and Pezzella propose an improved genetic algorithm for DFJS by simply encoding job operations and giving an improved local searching method to optimize job makespan [7]. However, many research efforts focus on scheduling problems in the situation where all jobs are ready at the beginning, but ignore the uncertainty from the random job arrivals [8]. This lies mainly on long time taken during the optimization iteration process, and hence limits its adaptability to dynamic events in actual production systems.

Rule-based methods are widely investigated and exploited in complex production systems as they are relatively easy to implement in practical settings with a minimal computational effort. Giannelos et al. introduce a rule-based method to optimize job tardiness by chaos and nonlinear theory [9]. However, once the operating condition changes (e.g., more or fewer machines become available, due date modification etc.), the existing dispatching rules may no longer adapt to new application situations, or even lead to performance decrease.

Thus, some heuristic methods are investigated for dynamic manufacturing environments. Ziaee proposes an approach for DFJS, where the FMU with the least workload is selected for each job, and then 5 scheduling influencing factors relevant to task processing time and machine capability are put forward and summed weightedly up to get a comprehensive evaluation index for each machine so as to determine the assignment of each operation [10]. Nevertheless, its main obstacle is that these affecting factors are not modeled comprehensively and it is hard to get a better balance between them.

Nowadays, with the development of Artificial Intelligence (AI), an interesting idea is to cast scheduling as a learning task and treat it as a machine learning problem. Mouelhi-Chibani and Pierreval present an NN-based intelligent method for sequencing [11]. Through continuous scheduling simulation, related scheduling knowledge can be discovered and then used to train a NN model, and

finally the most suitable dispatching rules, corresponding to job and resource related data, can be dynamically selected by the well-trained NN. Gabel and Riedmiller investigate reinforcement learning based scheduling algorithms, and define each machine as an agent [12]. According to agent's real-time status and a small set of parameters, the probability of selecting dispatching policies for agents is obtained, where the parameters are initialized randomly and optimized by a strategy gradient reinforcement learning algorithm.

As mentioned above, traditional scheduling methods all have different limitations. Meta-heuristic based algorithms are usually static algorithms, which require a lot of iterations to obtain near-optimal solutions, thus they are usually powerless for real-time scheduling. Besides, rule-based methods and heuristic algorithms are difficult to make effective decisions in a complex scheduling environment. With the popularization of the Industrial Big Data and Internet of Things, massive production process data can be easily collected and used to mine scheduling related knowledge. Data driven scheduling is expected to be a promising method in production process optimization and can break through the bottlenecks of traditional methods. Despite model training is time-consuming, it can be done offline. Once trained, the scheduler is capable of making quick and online decisions.

Motivated by the idea of "data-driven intelligence", this paper studies massive customized and one-of-a-kind production mode oriented DFJS problems. Firstly, taking average tardiness penalty as the optimization objective, an integrated framework including a 3-stage scheduling model for DFJS is established. Then, for job scheduling and operation scheduling, we propose the corresponding scheduling strategy based on DNN models. Finally, we introduce a rule-based method to optimize operation sequencing.

This paper is organized as follows. Section 2 gives the architecture of our 3-stage scheduling problem of DFJS. In Section 3, DNN-based scheduling algorithms for selecting the most suitable FMU for each job and the most suitable machine for each operation are proposed, respectively. Then, we introduce a rule-based method to set priority for operations to be executed in the machine. We conclude our paper in Section 4 and present our further research on DFJS problems.

2. PROBLEM FORMULATION

2.1. 3-stage Scheduling Framework

A flexible distributed manufacturing system is comprised of multiple geographically distributed FMUs, represented as $U = \{U_1, U_2, \ldots, U_k, \ldots, U_{NK}\}$, where NK is the total number of FMUs. Each FMU includes several multi-purpose machines equipped with interchangeable tools to perform different operations. Machines in U_k can be expressed as $M^k = \left\{M_1^k, M_2^k, \ldots, M_m^k, \ldots, M_{NM_k}^k\right\}$, where NM_k is the number of machines. Suppose the job set arriving at a manufacturing system within a specific period is $J = \left\{J_1, J_2, \ldots, J_j \ldots, J_{NJ}\right\}$, where NJ is the number of jobs. Usually J_j consists of several operations, and its operation set for FMU U_k can be denoted as $OP^{jk} = \left\{OP_1^{jk}, OP_2^{jk}, \ldots, OP_i^{jk}, \ldots, OP_{NO_j}^{jk}\right\}$, where NO_j is the total number of operations in J_j. In order to meet the deadline constraints and guarantee the Service Level Agreement (SLA), any jobs should be completed as soon as possible. Otherwise, a penalty or punishment will be introduced for deadline violation according to the predesigned SLA contract. Therefore, the optimization objective considered here is to minimize the average tardiness penalty $\overline{\pi_J}$ of job set J, as shown in (1):

$$Min(\overline{\pi_J}) = Min\left(\frac{\sum_{j=1}^{NJ}\left(\pi_j^k\right)}{NJ}\right) \quad \ldots\ldots\ldots \quad (1)$$

where π_j^k is the tardiness penalty of J_j processed by U_k, which can be denoted as:

$$\pi_j^k = PU_j \times TAR_j^k \quad \ldots\ldots\ldots\ldots \quad (2)$$

where PU_j represents the unit loss due to the tardiness of J_j. TAR_j^k is the tardiness of J_j in U_k, which is described as:

$$TAR_j^k = \max\left\{0, \left(ET_j^k - D_j\right)\right\} \quad \ldots\ldots\ldots \quad (3)$$

where D_j and ET_j^k represent the deadline and actual finish time of J_j, respectively. Taking the production life cycle into account, ET_j^k is formulated as:

$$ET_j^k = ST_j + dw_j^k + de_j^k + dtr_j^k \quad \ldots\ldots\ldots \quad (4)$$

where ST_j is the arrival time of J_j. dw_j^k and de_j^k express the whole operation waiting and processing time of J_j in U_k, respectively. dtr_j^k represents the finished product transportation time of J_j from U_k to its end customer.

As shown in Fig. 1, a DFJS problem mainly includes the following three steps:

1. Selecting the most suitable FMU for arriving jobs.
2. Selecting the most suitable machine for each operation of assigned jobs.
3. Determining the most suitable sequence of operations in the waiting queue of each machine.

2.2. Job Scheduling

Job scheduling is to select the most suitable FMU for each job that arrives at the manufacturing system, aiming at minimizing the tardiness penalty of each job. Once a new job J_j arrives, its Available Resource Pool (ARP) will be built, which is represented as $S^j = \left\{S_1^j, S_2^j, \ldots, S_a^j, \ldots, S_{NS_j}^j\right\}$, where $S^j \subseteq U$, and NS_j is the total number of resources in ARP for J_j.

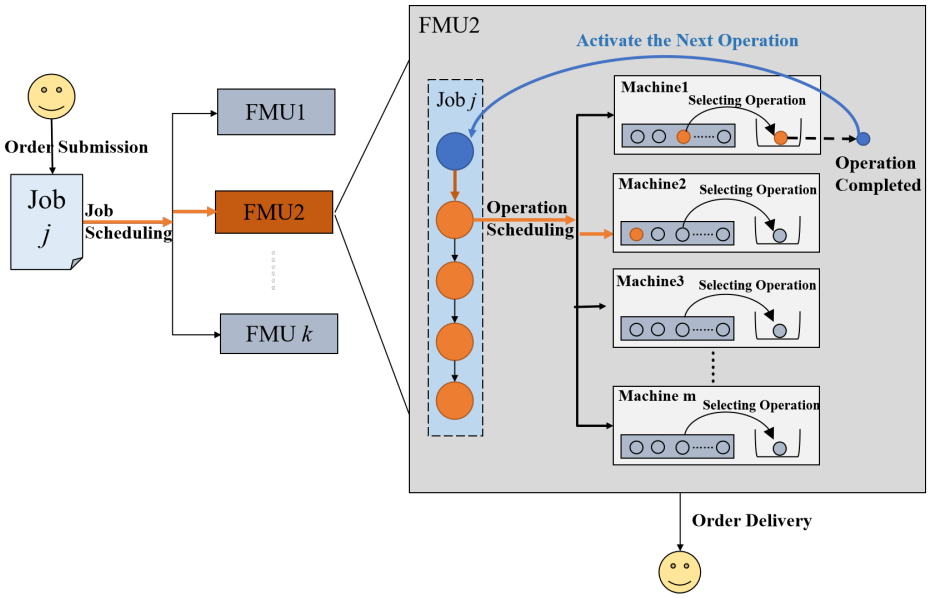

Fig. 1. Distributed and flexible manufacturing model in one-of-a-kind production

It should be noted that only one FMU can be selected for a particular job at a time. The target of job scheduling is to select a suitable resource S_a^j for job J_j to minimize its delay time $\widehat{\pi}_j^a$, which is shown as:

$$\underset{a}{Min}\left(\widehat{\pi}_j^a\right)=\underset{a}{Min}\left(ET_j^a-D_j\right) \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (5)$$

where ET_j^a is the finish time of J_j on S_a^j.

2.3. Operation Scheduling

Once a job J_j is determined to be processed in a particular FMU S_a^j, the operation scheduling will be needed to select the most suitable machine for each operation OP_i^{ja}. In this paper, we only restrict the job processing workflow to a serial structure. Each operation starts to be processed only upon the completion of its immediate predecessor. This kind of operation constraint can be formulated as follows:

$$\beta_i^{ja}=\begin{cases}0, & i=1\\ \delta_{i-1}^{ja}, & i>1\end{cases} \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (6)$$

where β_i^{ja} is the earliest start time of OP_i^{ja}, δ_{i-1}^{ja} is the actual completion time of OP_{i-1}^{ja}.

Once OP_{i-1}^{ja} completes its processing, an ARP will be built for its immediate successor OP_i^{ja} in S_a^j. The ARP for OP_i^{ja} is as follows: $S^{a,ji}=\{S_1^{a,ji},S_2^{a,ji},\ldots,S_b^{a,ji},\ldots,S_{NS_a^{ji}}^{a,ji}\}$, where $S^{a,ji}\subseteq M^a$, and NS_a^{ji} is the total number of available resources for OP_i^{ja} in S_a^j. In the real world, the actual processing time of OP_i^{ja} will be affected not only by the capability but also health condition of the selected machine. Therefore, the actual completion time δ_{ib}^{ja} of a certain operation OP_i^{ja} depends on not only the machine selected, but also its waiting time and transmitting time from its predecessor, which can be formulated as:

$$\delta_{ib}^{ja}=dtr_{ib}^{ja}+dw_{ib}^{ja}+\beta_i^{ja}+de_{ib}^{ja} \quad . \quad . \quad . \quad . \quad . \quad . \quad (7)$$

where dtr_{ib}^{ja} is the transmission time of OP_i^{ja} from its immediate predecessor operation OP_{i-1}^{ja}, dw_{ib}^{ja} is the actual waiting time of OP_i^{ja}, and de_{ib}^{ja} expresses the processing time of OP_i^{ja} on $S_b^{a,ji}$, respectively. At the stage of operation scheduling, the most suitable resource $S_b^{a,ji}$ will be selected for each operation OP_i^{ja} so as to minimize the flow time $\widetilde{\pi}_{ib}^{ja}$ of OP_i^{ja}, represented as:

$$\underset{b}{Min}\left(\widetilde{\pi}_{ib}^{ja}\right)=\underset{b}{Min}\left(\delta_{ib}^{ja}-\beta_i^{ja}\right) \quad . \quad . \quad . \quad . \quad . \quad . \quad (8)$$

2.4. Operation Ranking

Multiple ongoing jobs often coexist in a job-shop system and several operations from different jobs may be assigned to the same machine and wait to be processed in its input buffer. This needs to rank all the operations in the input buffer for selecting the next operation to be processed. Suppose an event occurs at time t, the waiting queue $OP_{km}(t)$ of the machine M_m^k can be represented as $OP_{km}(t)=\left\{OP_{km}^{j_1i_1},\ldots,OP_{km}^{j_ni_n},\ldots,OP_{km}^{j_{NW_{km}(t)}i_{NW_{km}(t)}}\right\}$, where $NW_{km}(t)$ is the total number of operations at time t. Once a particular machine M_m^k is released, the next most suitable operation should be selected.

To minimize the average tardiness penalty for a job set J, all its job tardiness penalties should be avoided or minimized as much as possible. Due to job tardiness penalty closely relates to the sub tardiness penalties of its operations, at this stage, we aim at minimizing the average sub tardiness penalty of all the operations in $OP_{km}(t)$, which

is expressed as:

$$Min\left(\widehat{\pi}_{km}(t)\right) = Min\left(\frac{\sum_{n=1}^{NW_{km}(t)} \max\left(0,\left(\delta_{km}^{j_n i_n} - SUD_{km}^{j_n i_n}\right)\times PU_{j_n}\right)}{NW_{km}(t)}\right) \quad . \quad (9)$$

where $\delta_{km}^{j_n i_n}$ is the actual completion time of operation $OP_{km}^{j_n i_n}$, PU_{j_n} is the tardiness penalty coefficient of the job J_{j_n}. $SUD_{km}^{j_n i_n}$ represents the sub deadline of $OP_{km}^{j_n i_n}$, which can be obtained as:

$$SUD_{km}^{j_n i_n} = \begin{cases} ST_{j_n} + \frac{(D_{j_n} - ST_{j_n})\times v_k^{j_n i_n}}{v_k^{j_n}}, i_n = 1 \\ SUD_{km}^{j_n (i_n-1)} + \frac{(D_{j_n} - ST_{j_n})\times v_k^{j_n i_n}}{v_k^{j_n}}, i_n \neq 1 \end{cases} \quad . \quad (10)$$

where ST_{j_n} is the arrival time of job J_{j_n}. $v_k^{j_n i_n}$ is the estimated processing time of operation $OP_{km}^{j_n i_n}$ in U_k, i.e., the average processing time of $OP_{km}^{j_n i_n}$ on each machine. $v_k^{j_n}$ represents the estimated processing time of J_{j_n} on U_k, which can be formulated as:

$$v_k^{j_n} = \sum_{i_n=1}^{NO_{j_n}} v_k^{j_n i_n} \quad . \quad (11)$$

where NO_{j_n} is the total number of operations in J_{j_n}.

3. DNN AND RULE BASED DFJS SCHDULER

This paper uses DNN to discover some implicit association knowledge among multiple factors by extracting scheduling features from massive and multi-dimensional sample data, and finally constructs DNN-based schedulers to select the most suitable resources for jobs and operations. Besides, we use a rule-based method to rank operations in the waiting queues of machines dynamically.

3.1. Job Scheduling Algorithm

Job scheduling aims to map a job to the most suitable FMU. Based on the work in [11] and [13], and taking the characteristics of one-of-a-kind production into consideration, job and FMU related data attributes are designed first.

Job J_j related attribute set JP^j contains job process composition and job deadline information etc., which can be denoted as $JP^j = \left\{JP_1^j, JP_2^j, \ldots, JP_{NJP}^j\right\}$, where NJP is the total number of attributes. While FMU S_a^j related attribute set is divided into two parts, SS^{aj} and SD^{aj}. SS^{aj} describes the static information of FMU, such as its resource configuration, processing capability and address. SD^{aj} depicts the real-time status of FMU, such as its current workload and health status. SS^{aj} and SD^{aj} can be defined as $SS^{aj} = \left\{SS_1^{aj}, SS_2^{aj}, \ldots, SS_{NSS}^{aj}\right\}$, $SD^{aj} = \left\{SD_1^{aj}, SD_2^{aj}, \ldots, SD_{NSD}^{aj}\right\}$, respectively, where NSS and NSD are the total number of static configuration and real-time status attributes, respectively.

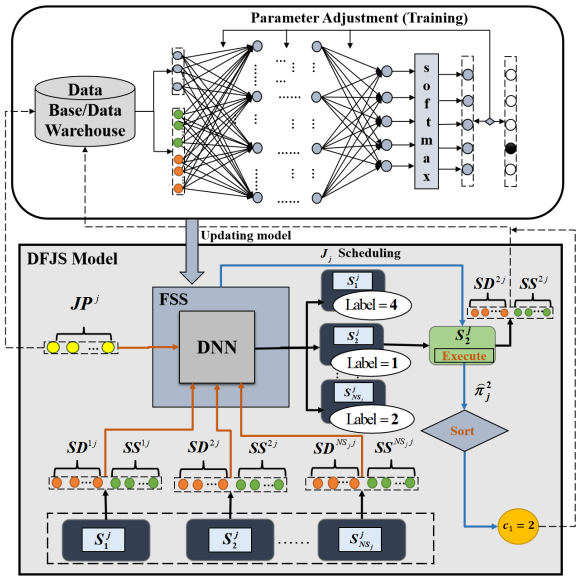

Fig. 2. DNN-based job scheduling algorithm

In order to obtain sample data for DNN-based job scheduler construction, the operation scheduler and operation ranker should be predetermined first. After that, we can use existing dispatching rules to select the most suitable FMU S_a^j for job J_j and produce job scheduling sample data by simulation. Then we can get the sample data $\left\{JP^j, SS^{aj}, SD^{aj}\right\}$, as well as the delay time $\widehat{\pi}_j^a$ of job J_j.

We sort the whole sample data according to $\widehat{\pi}_j^a$, and divide it into several clusters. Each cluster is labeled with an integer c_1, where $c_1 = 1, 2, \ldots NFC$, and NFC is the number of clusters. Here we put one label for all sample data from the same cluster, and set a larger number for the cluster with bigger delay time accordingly.

Then, the above labeled sample data can be used to train a DNN model, and the well-trained model is embedded into the First Stage Scheduler (FSS). Once a new job J_j arrives, FSS collects information of the available resource S_a^j as an input $\left\{JP^j, SS^{aj}, SD^{aj}\right\}$ of our DNN-based scheduler. J_j will be executed on the resource with the smallest output classification value. The main procedure of proposed algorithm is illustrated in Fig. 2.

If more outputs with the same smallest output classification value occur, two optional strategies are suggested:

1. Compare the output with its classification probabilities, and select the one with the largest probability for J_j.

2. Divide each sample data cluster into several sub-clusters according to its job delay time, and train sub-classification models for obtaining sub-classification outputs, finally select one resource with the smallest

sub-classification value for job J_j.

3.2. Operation Dispatching Strategy

Similar to job scheduling, the attributes considered for operation scheduling include operation OP_i^{ja} and machine $S_b^{a,ji}$ related attributes. Operation related attributes set is represented as $OPP^i = \{OPP_1^i, OPP_2^i, \ldots, OPP_{NOP}^i\}$, where NOP is the total number of operation related attributes. Machine related attributes can be divided into two sets MS^{bi} and MD^{bi}. MS^{bi} depicts the static machine configuration information, such as capabilities and locations, while MD^{bi} reflects the real-time machine status, such as current workload and health status, where $MS^{bi} = \{MS_1^{bi}, MS_2^{bi}, \ldots, MS_{NMS}^{bi}\}$ and $MD^{bi} = \{MD_1^{bi}, MD_2^{bi}, \ldots, MD_{NMD}^{bi}\}$, NMS and NMD are the total number of static configuration and real-time status attributes.

Similar to the first stage, for obtaining sample data to construct our DNN-based operation scheduling model, the job scheduler and operation ranker should be set first. Then, operation OP_i^{ja} will be assigned to a suitable machine $S_b^{a,ji}$ under the guidance of different dispatching rules, so as to generate different sample data $\{OPP^i, MS^{bi}, MD^{bi}\}$ and obtain the flow time $\widetilde{\pi}_{ib}^{ja}$ of OP_i^{ja}. After that, we sort the sample data according to $\widetilde{\pi}_{ib}^{ja}$, and divide it into multiple subsets. Each subset is assigned the same integer label c_2, $c_2 = 1, 2, \ldots NSC$, where NSC represents the number of subsets. The DNN-based scheduler trained by the above sample data is called Second Stage Scheduler (SSS). SSS collects the corresponding information of the operation to be performed and its available machines, and outputs an integer value for each machine. The process will eventually be assigned to the machine with the lowest integer value.

In the case of more than one outputs with the same smallest output classification value, two optional strategies are suggested, the same as in Section 3.1.

3.3. Operation Sequencing

When an operation is assigned to a particular machine, it can be processed immediately if the corresponding machine is ready, otherwise the assigned operation will be added into its waiting list. Once the machine releases from the current task or recovers from breakdown while only one operation waits for processing, the machine will perform this operation directly. However, there often are multiple operations waiting to be processed by the same machine. Different operation processing sequences will result in different scheduling performance.

In this section, we introduce a rule-based scheduler to determine the priority of operations in the waiting list. Suppose there are two operations OP_x and OP_y to be executed, and their corresponding attributes $\left\{\delta_{km}^{j_x i_x}, SUD_{km}^{j_x i_x}, PU_{j_x}\right\}$ and $\left\{\delta_{km}^{j_y i_y}, SUD_{km}^{j_y i_y}, PU_{j_y}\right\}$ are already known. At this time, there are two cases: executing OP_x first and executing OP_y first. We calculate the average sub tardiness penalty $\widehat{\pi}_{km}(t)$ in different cases according to (9). If $\widehat{\pi}_{km}^x(t) < \widehat{\pi}_{km}^y(t)$, OP_x has a higher priority, otherwise OP_y has a higher priority, where $\widehat{\pi}_{km}^x(t)$ and $\widehat{\pi}_{km}^y(t)$ represent the average sub tardiness penalty of executing OP_x and OP_y first, respectively.

Suppose that at trigger time t, there are more than one operation in the waiting queue of machine M_m^k, i.e., $OP_{km}(t) = \left\{OP_{km}^{j_1 i_1}, \ldots, OP_{km}^{j_n i_n}, \ldots, OP_{km}^{j_{NW_{km}(t)} i_{NW_{km}(t)}}\right\}$, we will select operation with the highest priority by pairwise comparison. First, we compare $OP_{km}^{j_1 i_1}$ with $OP_{km}^{j_2 i_2}$, select the operation with a higher priority according to the above rules, compare it with $OP_{km}^{j_3 i_3}$, and so on until all operations are traversed once. The operation with the highest priority will be the next one to be executed.

4. CONCLUSION AND FUTURE WORK

In this paper, the DFJS problems under massive customized or even one-of-a-kind production mode are studied. DFJS contains 3-stage hierarchical optimization problems which makes it more complex than traditional scheduling problems and not easy to be solved. Motivated by "data-driven intelligence", we propose a DNN-based intelligent scheduling algorithm framework through integrating job scheduling and operation scheduling together to realize the near optimal mapping from jobs to FMUs, and from operations to machines as well. We also design a rule-based method to guide operation ranking in machines' waiting buffer queues so as to minimize the total tardiness of the whole job set to be scheduled.

As future work, we first need to implement our method by simulation to verify the effectiveness of our proposed DNN-based DFJS framework. After that, we will try to collect different attributes in the scheduling process to form and enhance our sample data and train the DNN-based scheduler, then evaluate its scheduling performance from several aspects through comparing with different benchmark algorithms to verify its advantages over all the peers.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61836001; and in part by the National Key Research and Development Program of China under Grant No. 2018YFB1003700.

References:

[1] J. Behnamian and S. M. T. Fatemi Ghomi, "A survey of multi-factory scheduling", Journal of Intelligent Manufacturing, 27:231 – 249, February 2016.

[2] J. S. Srai, M. Kumar, G. Graham, W. Phillips, J. Tooze, S. Ford, et al., "Distributed manufacturing: scope, challenges and opportunities.", International Journal of Production Research, 54:6917–6935, 2016.

[3] G. Abaza, I. Badr, P. Goehner, and S. Jeschke, "Extending an agent-based FMS scheduling approach with parallel genetic algorithms", In IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society, pp. 2689–2694, 2010.

[4] M. Khalid and U. K. Yusof, "Improved Immune Algorithm for

Optimizing Distributed Production Scheduling Problem in Flexible Manufacturing System Subject to Machine Maintenance", International Journal of Mathematics and Computers in Simulation, 8, 2014.

[5] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, and A. C. Ammari, "An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem", Journal of Intelligent Manufacturing, 29(3), 2015.

[6] P. H. Lu, M. C. Wu, H. Tan, Y. H. Peng, and C. F. Chen, "A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems", Journal of Intelligent Manufacturing, pp. 1–16, 2015.

[7] L. D. Giovanni and F. Pezzella, "An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem", European Journal of Operational Research, 200(2):395–408, 2010.

[8] Y. Morinaga, M. Nagao, and M. Sano, "Optimization of flexible job-shop scheduling with weighted tardiness and setup-worker load balance in make-to-order manufacturing", In 2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS), pp. 87–94, 2014.

[9] N. Giannelos, N. Papakostas, D. Mourtzis, and G. Chryssolouris, "Dispatching policy for manufacturing jobs and time-delay plots", International Journal of Computer Integrated Manufacturing, 20:329 – 337, 2007.

[10] M. Ziaee, "A heuristic algorithm for the distributed and flexible job-shop scheduling problem", The Journal of Supercomputing, 67:69–83, 2014.

[11] W. Mouelhi-Chibani and H. Pierreval, "Training a neural network to select dispatching rules in real time", Computers & Industrial Engineering, 58(2):249–256, 2010.

[12] T. Gabel and M. Riedmiller, "Distributed policy search reinforcement learning for job-shop scheduling tasks", International Journal of Production Research, 50(1):41–61, 2012.

[13] G. R. Weckman, C. V. Ganduri, and D. A. Koonce, "A neural network job-shop scheduler", Journal of Intelligent Manufacturing, 19(2):191–201, 2008.