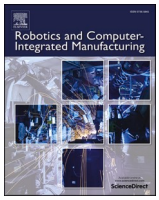
Robotics and Computer–Integrated Manufacturing 73 (2022) 102261

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim



Network-based dynamic dispatching rule generation mechanism for real-time production scheduling problems with dynamic job arrivals

Zilong Zhuang, Yue Li, Yanning Sun, Wei Qin *, Zhao-Hui Sun

School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, China

ARTICLE INFO

Keywords:
Production scheduling
Dispatching rule generation mechanism
Complex network
Multiple attribute decision making
Open shop scheduling

ABSTRACT

Although the concept of Industrial 4.0 has been well accepted, only few studies have dealt with real-time production scheduling of smart factories. Due to the advantages of simplicity, efficiency and quick response, heuristic rules have become the most promising technology to solve such problems. However, they suffer some drawbacks, such as high development and maintenance costs, low solution quality, and excessive emphasis on local information. To design heuristics from the perspective of system optimization and ensure the performance of heuristics in real-time production scheduling environments, this study develops a network-based dynamic dispatching rule generation mechanism. The complex network theory is introduced to extract a series of low-level heuristics from the perspective of system optimization, while the automatic heuristic generation problem is formulated as a multiple attribute decision making problem. Given that the dispersity of local features indicates their value for decision-making, the entropy weighting method is employed to automatically produce an adequate combination of the provided easy-to-implement low-level heuristics. Finally, the open shop scheduling problem with dynamic job arrivals is taken as an example to evaluate the effectiveness of the proposed algorithm. Numerical results demonstrate the excellent performance of the proposed algorithm in terms of algorithm effectiveness and computational time.

1. Introduction

In the era of Industrial 4.0, smart factories have become the key place for implementing a series of management activities [1,2]. As the crucial technology in production management, scheduling aims to optimize one or more production performance objectives by reasonably allocating resources among tasks within a given period of time [3,4]. However, with the development of manufacturing technologies and the refinement of manufacturing processes, the actual manufacturing system is evolving towards complexity, which brings great challenge to the scheduling optimization of manufacturing systems [5]. Besides, dynamic events in actual production also greatly substantiate the complexity of scheduling problems [6]. As the most frequent dynamic event in the actual production, the real-time production scheduling problems with dynamic job arrivals has attracted widespread attention [7,8]. Therefore, developing effective scheduling strategies for real-time production scheduling problems with dynamic job arrivals is of vital significance, which directly affects production efficiency, manufacturing costs and market competitiveness [9].

Various optimization methods have been proposed to solve production scheduling problems. However, in the case of dynamic order arrival requiring real-time responses, exact methods and metaheuristics have great difficulties in modeling, and require much more time to obtain suboptimal solutions, making them always perform poorly in real-time scheduling decisions [10,11]. Moreover, they will fail fundamentally as the problem size increases due to their inability to evaluate the performance of solutions in a timely manner. In comparison, heuristics are more practical and convenient in the highly dynamic production environment due to their ease of implementation and rapid response to dynamic changes [12].

Although plenty of heuristics have been proposed and successfully applied to various problems over the past few decades, there are still some difficulties in applying them to newly encountered problems or even new instances of similar problems [13]. Usually, heuristics are designed by experts in a tedious trial-and-error process, with candidate heuristics being tested, modified, and retested in a simulation environment until they meet the requirements for actual implementation in practice [14]. Therefore, heuristics are bespoke problem-specific

* Corresponding author.
E-mail address: wqin@sjtu.edu.cn (W. Qin).

https://doi.org/10.1016/j.rcim.2021.102261
Received 30 November 2020; Received in revised form 17 September 2021; Accepted 19 September 2021

methods and have to be specifically designed for the problem at hand, which are expensive to develop and maintain. Most of the existing heuristics are derived from practical production experience, focusing more on local characteristics than the overall structure of scheduling objects. In addition, heuristics are often developed to find solutions of acceptable quality within a reasonable time, but with few optimality guarantees [15]. Therefore, how to design heuristics from the perspective of system optimization and how to guarantee the performance of heuristics in dynamic scheduling environments remain to be studied [16].

As an attempt to bridge the gaps, this study develops a novel network-based dynamic dispatching rule generation mechanism for real-time production scheduling problems with dynamic order arrivals. It introduces complex network theory to describe production scheduling problems and design effective heuristic rules from the perspective of system optimization, and then formulates the automatic heuristic generation problem of generation constructive hyper-heuristic as a multiple attribute decision making (MADM) problem. The main novelties and contributions of this paper are three-fold. Firstly, complex network theory is introduced as a new tool to describe production scheduling problems, which can describe existing heuristic rules and systematically generate a series of low-level heuristics from the perspective of system optimization. Secondly, generation hyper-heuristic is utilized to realize dynamic adaptive scheduling, and the automatic heuristic generation problem of generation hyper-heuristic is formulated as a multiple attribute decision making problem. Thirdly, given that the attribute differences among nodes in complex scheduling networks largely reflect the value of this attribute for decision-making, the entropy weighting method is employed to automatically produce an adequate combination of the provided easy-to-implement low-level heuristics.

The remainder of the paper is organized as follows. Section 2 provides a brief literature review on real-time production scheduling. Section 3 elaborates on the proposed network-based dynamic dispatching rule generation mechanism. Section 4 presents the computation experiments and analysis. Finally, Section 5 draws some conclusions and introduces some interesting directions for future studies.

2. Literature review

In the past few decades, real-time production scheduling problems have attracted widespread attention due to the frequent occurrence of dynamic events in actual production [17,18]. To address such problems, dynamic scheduling is expected to make timely and reasonable decisions based on scheduling tasks and current system status, which requires full consideration of dynamic production demand and unstable production process. Therefore, it has become the common desire of scholars and enterprises to develop scheduling methods that can respond to dynamic events in a timely manner and maintain high performance in fluctuating machining conditions.

Dynamic scheduling has been studied extensively, which can be categorized into predictive-reactive scheduling, robust scheduling and completely reactive scheduling [19]. Predictive-reactive scheduling is usually divided into two stages. The first stage is the prescheduling stage that generates an initial scheduling scheme as the basis for the workshop to arrange production activities, while the second stage is the rescheduling stage that adjusts the scheme in response to real-time events [20]. Because of its simple principle and easy implementation, this method has been widely used in production environments with infrequent occurrences of dynamic events and low uncertainty. However, excessive rescheduling can easily interfere with the original production activities, resulting in production disorder and production performance decline, and therefore it is inapplicable to unstable scheduling systems with frequent occurrences of dynamic events [21]. Robust scheduling predicts the future dynamic events in advance under existing conditions, and generates a prescheduling scheme that can accommodate various possible emergencies, so as to ensure that the performance of the scheduling system will not deteriorate when dynamic events occur [22]. Due to the preventive measures taken against possible dynamic events, robust scheduling naturally has high robustness, but it cannot handle dynamic events that deviate from prediction and responds poorly to emergencies. Besides, in order to increase the robustness against disturbances, robust scheduling introduces slack time, which greatly increases the waiting time of some devices and reduces resource utilization [23].

Completely reactive scheduling is also known as online scheduling or real-time scheduling. Different from the first two types of scheduling, it is a real-time scheduling strategy that makes use of heuristic rules to respond quickly to changes of workshop state and provide guidance for the next operation of the actual production system [24]. Since this method makes scheduling decisions only upon the current system state information without prescheduling schemes, it has good practicability for the actual production environment with frequent dynamic events and high uncertainty. Heuristic rules have therefore become the primary solution to real-time production scheduling problems because of their advantages such as simplicity, efficiency and quick response [25]. However, the current heuristic rule design method is mainly based on manual experience, and knowledge related to complex production scheduling problems is difficult to obtain, making it extremely difficult to develop efficient scheduling rules [26]. Besides, most existing methods only take into account local information and can only perform local optimization. Therefore, it has become an urgent task to design effective scheduling rules through analyzing the influence of inherent constraints and dynamic events on the overall structure of the scheduling object.

Recently, genetic programming based approaches and gene expression programming based approaches have been widely used to design scheduling rules [27–29]. However, the rules learned by these approaches for a specific scheduling environment are fixed during the execution process, and may have weak adaptability in the new environment. Given that no scheduling rule performs dominantly better than any other in all scheduling environments, it will be a promising way to enhance the adaptability of scheduling rules by dynamically adjusting the rules used when the shop floor environment changes, which falls into the category of hyper-heuristic. As an ‘off-the-peg’ method rather than a ‘made-to-measure’ technique, hyper-heuristic aims to select or generate new heuristics from a set of potential heuristic components in a given situation rather than solving underlying combinatorial problems directly [30]. In terms of process nature, hyper-heuristics can be divided into selection hyper-heuristic and generation hyper-heuristic [31]. In comparison, generation hyper-heuristics search the space of heuristics constructed from components rather than the space of complete predefined heuristics, thus having a significantly larger search space and better potential performance. They are capable of solving given problem instances or a class of problems, due to their ability to automatically produce an adequate combination of low-level heuristics (LLHs) [15]. Obviously, LLHs play a crucial role in the performance of generation hyper-heuristic, which can be further categorized as constructive or perturbative [32]. Since perturbative heuristics are problem dependent and often appear in the form of moving operators designed for the problem domain, they belong to the category of metaheuristics and could hardly obtain suboptimal solutions for scheduling problems with dynamic events in an acceptable computational time. By contrast, constructive heuristics are usually generated using a given set of problem attributes, which can reduce the labor required to derive low-level heuristics and may lead to the induction of new constructive heuristics beyond imagination [30]. This allows constructive heuristics to be tailored for specific problem instances or induced for different types of problems [13]. However, how to dynamically generate effective dispatching rules based on a given set of problem attributes while ensuring the performance of generated rules has not been studied in the field of hyperheuristics.

Based on the above brief review, we find that although a broad

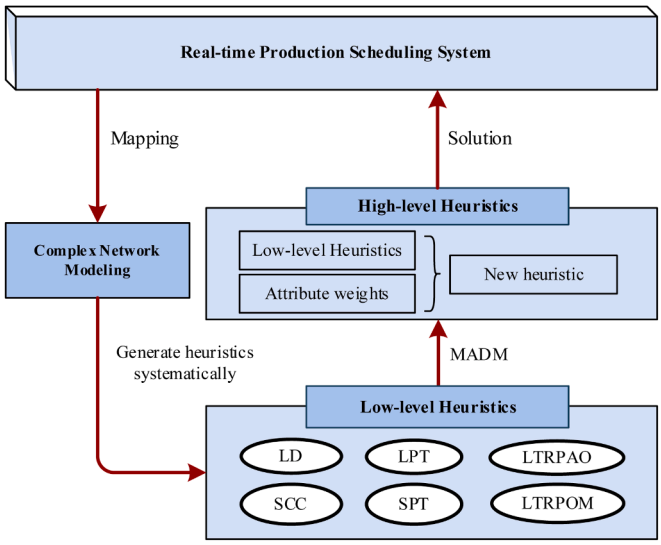


Fig. 1. Network-based dynamic dispatching rule generation method for real-time production scheduling problems.

literature on real-time production scheduling has been published, few studies have dealt with the design of low-level heuristics from the perspective of system optimization and the development of performance guarantee mechanism of hyper-heuristics in real-time production scheduling environments. This study ingeniously makes full use of complex network theory and multiple attribute decision making methods, and develops a network-based dynamic dispatching rule generation mechanism for real-time production scheduling problems that employs the complex network theory to systematically generate a series of low-level heuristics from the perspective of system optimization and utilizes the multiple attribute decision making method to solve the automatic heuristic generation problem of generation constructive hyper-heuristics.

3. Methodology

This section presents the proposed network-based dynamic dispatching rule generation mechanism for real-time production scheduling problems with dynamic job arrivals. As shown in Fig. 1, the proposed network-based dynamic dispatching rule generation (NDRG) method consists of three main modules, namely complex network modeling, low-level heuristics generation, and high-level heuristics generation. It needs to be emphasized that the first two are conducted in the form of offline learning, while the last one is implemented by the multiple attribute decision making method upon the real-time decision information.

3.1. Complex network modeling

To model and analyze a large number of complex dynamic systems in the real world from an overall perspective, complex network theory has been developed in the field of statistical physics since the 1990s [33,34]. A complex network consisting of numerous nodes and edges is used to describe various complex systems, where nodes represent elements of the system, and edges between nodes indicate the correlation between these elements, providing a new solution to complex production scheduling problems. Complex network theory holds that the overall behavioral characteristics of complex systems are mainly determined by the association patterns among nodes, rather than by the behavior characteristics of individual nodes themselves. Similarly, it is generally believed that the overall performance of complex scheduling problems largely depends on the complex relationship of scheduling elements such as tasks, resources and operations [35]. Therefore, complex network theory has great potential as a powerful tool to describe the complex relationships of complex production scheduling problems.

Before specifically introducing complex network modeling, it is necessary to briefly distinguish between complex network modeling and disjunction graphs. Both the complex network modeling and the disjunctive graph model belong to the graph model, where each node denotes an operation, and each edge represents a mutually exclusive constraint between two operation nodes. Their difference lies in the modeling dependence and decision-making mechanism. In terms of modeling dependence, the disjunctive graph model is often used to analyze static systems with all the information known in advance, which is difficult to apply to the uncertain real-time production scheduling problems, while the complex network modeling can handle the dynamic randomness of complex systems well through the random appearance and disappearance of nodes and edges in a complex network. As for the decision-making mechanism, the disjunctive graph aims to change the undirected disjunctive arc into a directed arc to improve the critical path, which is only suitable for solving small and medium-scale problems. The complex network modeling gradually constructs a complete scheduling plan in an event-driven manner, and maps the arrival and completion of tasks to the appearance and disappearance of nodes in the complex network, respectively. Its scheduling decision is equivalent to node selection in a series of task network scenarios, which can realize rapid response to large-scale real-time production scheduling problems.

In recent years, complex network theory has already found its way into production scheduling problems. The complex network model for production scheduling problems can be represented by a triad $G = (V, L, R)$, where V denotes the set of nodes, L denotes the set of edges, and R denotes the set of network evolution rules [36]. Each node denotes an operation with an attribute namely the processing time, and each edge represents a mutually exclusive constraint between two operation nodes. Specifically, an edge will link any pair of operation nodes of the same job because any two operations of the same job cannot be processed simultaneously, which is called “job edge”. Similarly, an edge will link any pair of operation nodes that require the same resource because each resource can only be used to process one operation at a time, which is called “resource edge”. All the edges together represent the complex relationships between the main elements of scheduling systems. It should be noted that the complex network is directed if the problem is a job shop scheduling problem (JSSP) or a flow shop scheduling problem (FSSP), while the complex network is undirected if the problem is an OSSP [37]. If the setup time or transportation time is considered, the complex network is weighted, otherwise, it is unweighted. Besides, a node will appear in the complex scheduling network when the job it belongs to is available and disappear when the operation is completed. Finally, the complex network can model various scheduling problems by transforming them into the reasonable arrangement of node traversal order, on the condition that one node can be selected after its precursor nodes have been completed and only unconnected nodes can be traversed simultaneously [38].

From the perspective of combinatorial optimization, OSSP has significantly larger solution space and higher solving difficulty because it needs to determine the processing routes of jobs in addition to the elements that need to be determined for the classic JSSP or FSSP solution [39]. However, as can be seen from the above description, it is obviously simpler to use complex network theory to describe OSSP than to describe JSSP and FSSP. In order to better reflect the advantages of complex network theory in describing and solving complex dynamic optimization problems without loss of generality, this study takes OSSP with dynamic job arrivals (OSSP-DJA) as the specific research object.

In classic OSSP, a set of n jobs $J = \{J_1, J_2, \ldots, J_n\}$ are to be performed by a set of m machines $M = \{M_1, M_2, \ldots, M_m\}$, with all jobs and machines available at the beginning. Each job J_j consists of m operations $O(j, i)$ ($i = 1,\ldots, m$) where $O(j, i)$ refers to the operation of job j on the machine i that needs to be processed for p_{ji} time units without preemption. At any

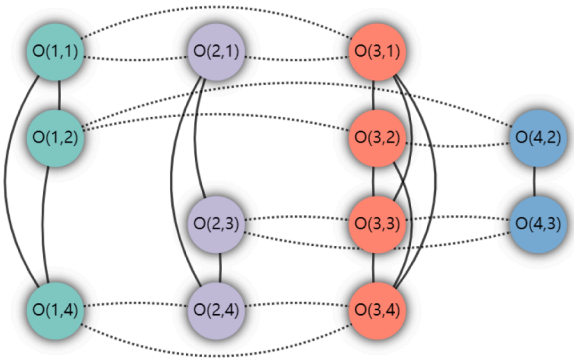


Fig. 2. A complex network model for an OSSP with four jobs and four machines.

time, each machine can process at most one job, and each job can be processed by at most one machine. Besides, setup times are negligible and preemption of operations is not allowed. The order in which each machine processes jobs (machine orders) and the order in which each job passes through machines (job orders) can be chosen arbitrarily. The goal of this problem is to find a feasible combination of the machine and job orders to optimize one or multiple performance measures. In OSSP-DJA, not all jobs are available at the beginning of the planning horizon, and some jobs arrive dynamically during execution.

A complex network model for an OSSP with four jobs and four machines is illustrated in Fig. 2, where job edges are represented by solid lines and resource edges by dashed lines. Besides, the stage skipping is taken into account because not all jobs need to go through all stages in practical production [40]. The formal description for OSSP-DJA using complex network theory is as follows: (1) A node $O(j, i)$ represents the operation of job j on the machine i, assigned two variables respectively representing the required processing time p_{ji} and release date r_j. The node $O(j, i)$ is available only when the actual execution time of the scheduling system exceeds r_j; (2) $O(j, i)$ and $O(k, i)$ respectively represent the operation of job j and k that needs to be processed by the machine i. Since a machine can handle at most a job at any time, all pairs of operations requiring the same machine (e.g. $O(j, i)$ and $O(k, i)$) will be linked by an edge; (3) $O(j, i)$ and $O(j, l)$ respectively represent the operation of job j on the machine j and k, respectively. Since a job can be processed by at most one machine at any time, all pairs of operations belonging to the same job (e.g. $O(j, i)$ and $O(j, l)$) will be linked by an edge; (4) Then, OSSP-DJA can be naturally transformed into how to arrange the node traversal order so that all the nodes in the network can be traversed as quickly as possible, on the condition that each node has a traversal time and only the disconnected nodes can be traversed simultaneously.

As is known to us all, network evolution rules R can be divided into two types. One type is the growth rule (GR) that controls the increase of network nodes and edges caused by the dynamic arrival of jobs. The other type is the cutting rule (CR) that determines the disappearance of network nodes and edges with the completion of operations. For OSSP-DJA, GR is embedded in the law of job arrival that is fundamentally uncontrollable, while CR is controllable by adopting different scheduling algorithms. Specifically, it is required to decide which nodes are selected to add into the set of nodes being processed whenever a new job arrives at the system or a selected node is completed, and it continues until all nodes have been traversed and no new jobs arrive. The decisions made under all network evolution scenarios constitute a cutting rule, and the node traversal order caused by the decisions is the scheduling scheme obtained by the cutting rule.

3.2. Low-level heuristics generation

A complex scheduling network can fully represent the complex relationships between the main elements in the scheduling problem. Therefore, analyzing the network structural characteristics is helpful to simplify complex scheduling problems and inspire the design of heuristic rules. Besides, the scheduling decision process based on heuristic rules is usually event-driven, that is, decisions are made only when a new job arrives or an operation ends. Due to the uncertainty and unpredictability of job arrival, scheduling decisions are usually obtained by using data-driven optimization methods upon the current scenario information. From the perspective of system optimization, given that more mutually exclusive constraints between nodes indicate stronger coupling and lower traversal efficiency, higher priority should be given to certain nodes, whose removal can significantly reduce the network coupling [41]. These nodes are called vital nodes in the context of complex networks, and they build a bridge between production scheduling optimization and vital node identification, so that the latest advances in the field of vital node identification provide solutions for scheduling optimization problems [35].

Since complex scheduling problems with different optimization objectives are essentially reflected in the node traversal order on the complex scheduling network model, the design ideas of adaptive generation constructive hyper-heuristic are consistent. A unified design method of heuristic rules based on network structural characteristics can be divided into four steps:

a) Establish a complex scheduling network model for the given scheduling object;
b) Extract global features related to scheduling optimization objectives;
c) Construct local features related to global features;
d) Design heuristic rules based on local features.

It should be noted that global features are used to measure the differences between different scheduling networks and different evolutionary moments of the same scheduling network, while local features are used to measure the differences or status between nodes or edges in the network. It is this difference that causes the difference in the effects of different traversal strategies. Given that network average degree and network average efficiency can distinctly reflect this coupling, two heuristic rules based on local topological properties namely the *Largest Degree first* (LD) rule and the *Smallest Cluster Coefficient first* (SCC) rule have been established for OSSP [36]. However, these two rules only apply to imbalanced network topologies obtained by setting jobs with different numbers of operations or multiprocessor tasks, and their performance of applying to classic balanced complex scheduling problems is greatly compromised due to their inability to identify valid initial traversal nodes. It is well known that the time information is a kind of heuristic information that is as important as the structure of scheduling objects, especially in a balanced complex scheduling network. Therefore, Zhuang et al. [38] reformulated two classic heuristic rules based on time information in the context of complex networks, namely the *Longest Processing Time first* (LPT) rule and the *Shortest Processing Time first* (SPT) rule. Besides, two new heuristic rules that integrate the heuristic information from network structural characteristics and node time attributes have been developed, namely the *Longest Total Remaining Processing on Adjacent Operations first* (LTRPAO) rule and the *Longest Total Remaining Processing on Other Machines first* (LTRPOM) rule. This study is an extraordinary extension of Zhuang et al. [38] and focuses more on how to develop generation constructive hyper-heuristic through real-time data analysis after introducing the complex network theory to describe production scheduling problems and systematically generating a series of heuristic rules. Therefore, these six heuristic rules developed in the context of complex networks can be used directly as low-level heuristic rules, and each rule can be regarded as a decision attribute of complex scheduling network nodes. It should be noted that the proposed hyper-heuristic can easily accept more low-level heuristic rules.

For OSSP-DJA, the scheduling optimization algorithm based on node attributes can be summarized as follows: First, all available operation

Table 1
Notations used in the whole algorithm.

Notation	Description
Plist	List of nodes being processed
Mlabel	List of machine labels for nodes being processed
Jlabel	List of job labels for nodes being processed
Wtopo	Adjacency matrix in the current scheduling scenario
Pline	A rule for calculating the priority value of all nodes
JDone	Node just finished
Done	Nodes that have been finished
Cnode	List of nodes that are available in the current scheduling scenario
Unode	List of nodes that have not been processed by a machine
Tc	Processing time of available nodes in the current scheduling scenario
Plist_CN	Node selected based on the SPT rule
Plist_CT	Processing time of node selected based on the SPT rule

nodes form a set of candidate nodes, from which a batch of nodes are selected one by one as the initial traversal nodes according to some central measure, while satisfying the traversal constraints. Once an operation node has been added into the set of processing nodes (SP), all its neighbors will be added to the set of currently prohibited processing nodes (SCP); Second, whenever a selected node is completed or a new job arrives in the system, the complex scheduling network will evolve into a new network scenario requiring decision-making. Specifically, when an operation finishes its processing on one machine, the operation node will be deleted from the SP, and its neighbors will be released from the SCP, while another operation with the optimal attribute measure from the set of candidate nodes will be added into the SP. For the appearance of a new job, all operations of the job will be added into the SCP if all machines are busy, while an operation node from the corresponding operation nodes will be added to the SP with the remaining operation nodes added into the SCP if some machines are idle. The evolution process continues until all nodes have been traversed and no new jobs arrive. The beginning time of each operation obtained during the entire decision-making process constitutes a complete scheduling scheme. Before presenting the detailed algorithm, the notations used in the whole algorithm are shown in Table 1.

Taking the SPT rule as an example, Algorithm 1 shows how to select a batch of initial traversal nodes to start the execution of the task on the machine at the beginning of the schedule. Algorithm 2 illustrates how to select an operation node from different jobs for the machine just released when an operation finishes being processed on the machine, which can improve the machine utilization and accelerate the completion of jobs. During the execution of the tasks on the machines, especially at the end of the entire process, it is often the case that the completion of an operation can activate other machines that are in an idle state. Algorithm 3 illustrates how to select an operation node for the idle machine when an operation is completed. Besides, when a new job arrives, Algorithm 4 shows how to make decisions. Finally, Algorithm 5 shows the pseudocode of the SPT rule for OSSP-DJA.

3.3. High-level heuristics generation

Many previous studies have shown that any elevated performance of an algorithm over one class of problems is offset by its performance over another class, which means no single dispatching rule performs dominantly better than any other rules in all scheduling environments [42]. Thus, it may not be the best strategy to focus solely on designing an optimal rule that can be applied to dynamic processing environments. With the arrival of new jobs and the completion of operation nodes, the production status changes over time. Therefore it is necessary to change the heuristic rules dynamically based on the real-time status of shop floor production. To realize the adaptive dynamic scheduling for OSSP-DJA, each heuristic rule can be regarded as an evaluation index of all available operations. Then the automatic generation problem of generation constructive hyper-heuristics at each decision point can be naturally transformed into a data-driven MADM problem.

MADM problems widely exist in various fields, such as manufacturing, services, economics, management, etc., [43–45], especially in evaluating the node importance in complex networks [46,47]. In MADM problems, attribute weights play an important role, making it a key issue to find suitable attribute weights. The methods of obtaining attribute weights can be divided into three categories: subjective method, objective method, and hybrid method [48]. The objective method is employed in this study to determine the attribute weights by using objective decision matrix information, which can reduce the dependence on empirical knowledge and avoid the influence of decision maker's preference on the determination of attribute weights [49].

Considering that the local feature of nodes can reflect the status of nodes in the network, the dispersity of local features among all nodes provides valuable information for distinguishing nodes and preventing misidentification [50]. The greater the dispersity, the higher the value of the local feature for decision-making. As a measure of system randomness and disorder, the entropy can also be employed to judge the dispersity of a local feature. Therefore, for the MADM problem of automatic heuristic generation, the entropy weighting method is the most suitable weighting method. It can make full use of the attribute differences among nodes in complex scheduling networks and objectively determine the attribute weights by evaluating the information provided by the observation value of each attribute [51]. Specifically, the attribute weights can be determined by the variation in each attribute. The

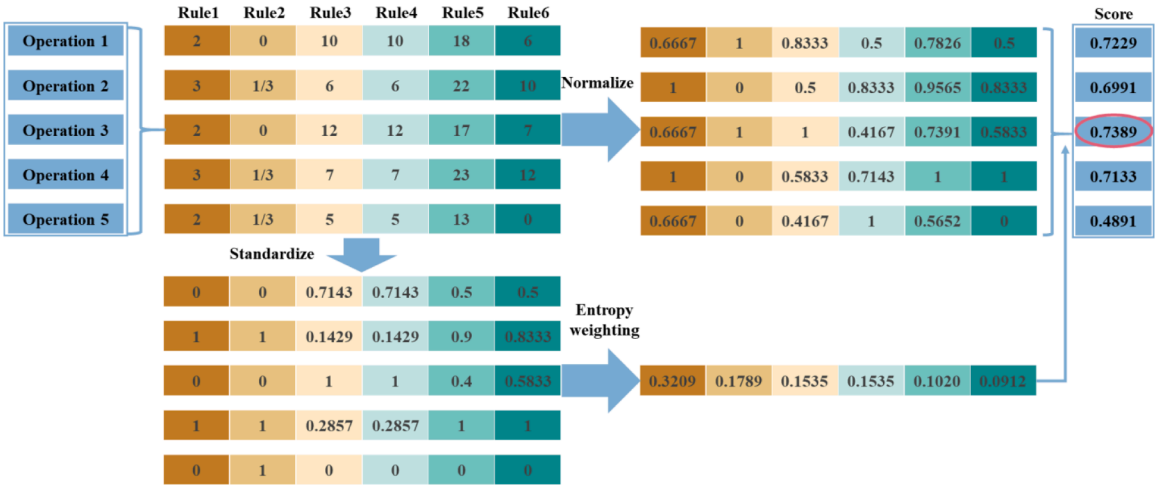


Fig. 3. An example of the entropy weighting method being applied to decision making.

greater the variation, the larger the attribute weight. The entropy weighting method has strong objectivity and thus can avoid the error caused by the subjective judgment of evaluators [41]. Fig. 3 shows an example of the entropy weighting method being applied to a decision situation with five candidate operations and six heuristic rules.

The steps to determine the attribute weights and weighted attribute values of candidate nodes in a decision scenario are as follows:

Step 1 Construct a decision matrix $X=(x_{ij})_{m\times n}$, where a_{ij} represents the evaluated value of ith object under the jth indicator. The decision system has m objects and n evaluating indicators. The decision matrix X will be:

$$X=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \tag{1}$$

Step 2 Standardize the decision matrix X to establish a standardized matrix $V=(v_{ij})_{m\times n}$, where v_{ij} is the standardized value of x_{ij}.

$$v_{ij}=\frac{x_{ij}-\min\limits_{1\le i\le m}(x_{ij})}{\max\limits_{1\le i\le m}(x_{ij})-\min\limits_{1\le i\le m}(x_{ij})} \tag{2}$$

Step 3 Calculate the weight of the ith evaluated object under the jth indicator p_{ij}, where $0\le p_{ij}\le 1$.

$$p_{ij}=\frac{v_{ij}}{\sum_{i=1}^{m} v_{ij}} \tag{3}$$

Step 4 Calculate the entropy value e_j under the jth indicator. When $p_{ij}=0$ or $p_{ij}=1$, $p_{ij}\times\ln(p_{ij})=0$.

$$e_j=-\frac{1}{\ln(m)}\sum_{i=1}^{m} p_{ij}\times\ln(p_{ij}) \tag{4}$$

Step 5 Calculate the difference coefficient d_j under the jth indicator. According to the Eq. (4) of entropy value, the higher similarity of the jth indicator indicates the higher value of e_j and the lesser information provided by the corresponding indicator.

$$d_j=1-e_j \tag{5}$$

Step 6 Determine the entropy weight for each indicator.

$$w_j=\frac{d_j}{\sum_{k=1}^{n} d_k} \tag{6}$$

Step 7 Considering that the priority of decision objects is proportional to the attribute value in some rules and inversely proportional to the attribute value in other rules, the decision matrix X needs to be normalized as $\overline{X}$.

$$\overline{x_{ij}}=\begin{cases} x_{ij}\Big/\max\limits_{1\le i\le m}(x_{ij}) & j\text{thindicatoristhepositiveindicator} \\ \min\limits_{1\le i\le m}(x_{ij})\Big/x_{ij} & j\text{thindicatoristhenegativeindicator} \end{cases} \tag{7}$$

Step 8 Calculate the weighted attribute value of each decision object. The higher the weighted attribute value of the decision object is, the higher priority the decision object has.

Table 2
Comparison of three OSSP benchmarks.

	Taillard [52]	Bruckner et al. [53]	Guéret and Prins [54]
Number of jobs	{4,5,7,10,15,20}	{3,4,5,6,7,8}	{3,4,5,6,7,8,9,10}
Processing times	U[1, 99]	U[1, 499]	U[1, 999]
Instances/size	10	8~9	10
Total instances	60	52	80
Arrival intervals	U[10, 50]	U[10, 250]	U[10, 500]

$$S_i=\sum_{i=1}^{n}\overline{x_{ij}}\times w_j \tag{8}$$

4. Experimental study

To investigate the effectiveness of proposed network-based dynamic dispatching rule generation method for real-time production scheduling problems, computational experiments are designed and performed. For comparison purposes, experimental results obtained by the proposed NDRG are compared with those obtained by each low-level heuristic rule. All the algorithms are coded in MATLAB 2017a on a PC with Intel Core CPU 3.40 GHz and 8 GB of RAM.

4.1. Description of benchmark instances

In the open shop scheduling literature, three sets of benchmark instances have been published by Taillard [52], Brucker et al. [53], and Guéret and Prins [54]. As shown in Table 2, the main differences among these three datasets lie in the range of problem size, the distribution law of processing times, and the number of instances per size. Specifically, the first set of instances proposed by Taillard [52] (denoted by tai_* × *_*) were divided in m∈{4,5,7,10,15,20} with uniformly distributed between 1 and 100 for the processing times. For each set of instances, there are 10 different test problems, totaling 60 instances. These instances are considered easy to solve due to the uniform distribution of processing times. Brucker et al. [53] presented 52 harder instances ranging from 9 operations to 64 operations (denoted by j*-per*-*). During the generation of these instances, the total processing times of each job j ($\sum_{i=1}^{m} p_{ji}$) and each machine i ($\sum_{j=1}^{n} p_{ji}$) are equal to or very close to 1000 time units [55]. The third set provided by Guéret and Prins [54] (denoted by gp*-*) consists of 80 problem instances, which are said even harder than those previously generated by Brucker et al. [53].

Regarding the arrival time, the instances of Taillard [52], Brucker et al. [53], and Guéret and Prins [54] do not provide this information. Due to the lack of benchmark instances for OSSP-DJA, new benchmark instances are generated by employing these existing OSSP benchmarks and modifying them into dynamic problems through performing the simulation. For the original instance with m jobs and m machines, other m jobs following the same distribution law of processing times are available with their dynamic arrival. The arrival intervals of these newly arrived jobs are randomly generated, with a distribution U[10, 50] for Taillard's instances, U[10, 250] for Bruckner et al.'s instances, and U[10, 500] for Guéret and Prins's instances.

The indicator commonly used for evaluating the quality of algorithms in computational experiments is the relative percentage deviation (*RPD*) for ith instance between the solution sol_{ik} obtained by kth algorithm and the lower bound LB_i, as shown in Eq. (9). For this problem, a rough lower bound '*LB*' can be calculated by the Eq. (10), where *Set1* denotes a set of jobs available at the beginning, *Set2* denotes a set of jobs arriving successively.

Table 3
Average RPD obtained by each algorithm on these three modified benchmarks.

Instance	LD	SCC	LPT	SPT	LTRPAO	LTRPOM	NDRG
Taillard							
tai_4 × 4	44.9	44.9	43.4	**40.8**	44.1	48.0	**36.9**
tai_5 × 5	55.8	55.8	56.4	54.5	**53.9**	57.2	**50.3**
tai_7 × 7	50.2	50.2	48.8	**44.9**	49.2	48.1	**44.2**
tai_10 × 10	46.4	46.4	46.8	**45.9**	47.8	46.7	**43.4**
tai_15 × 15	48.5	48.5	47.3	48.0	**47.0**	47.7	**45.3**
tai_20 × 20	47.1	47.1	47.7	**46.7**	47.9	47.0	**45.2**
Brucker et al.							
j3	60.9	60.9	59.9	**55.6**	**55.6**	58.3	**50.9**
j4	43.2	43.2	46.2	**42.2**	42.9	47.4	**37.3**
j5	29.5	29.5	27.9	**26.7**	28.3	28.1	**20.6**
j6	28.0	28.0	27.6	26.8	27.7	**26.6**	**23.7**
j7	18.3	18.3	16.9	**16.2**	19.5	19.8	**12.8**
j8	16.2	16.2	15.2	**14.1**	16.4	14.6	**11.7**
Guéret and Prins							
GP03	32.2	32.2	35.3	28.7	**28.3**	31.3	**26.4**
GP04	13.9	13.9	17.3	**12.5**	13.0	17.3	**8.6**
GP05	15.4	15.4	16.8	15.4	15.4	**14.0**	**10.7**
GP06	14.6	14.6	12.7	12.7	**12.0**	14.3	**9.8**
GP07	9.9	9.9	**8.7**	9.1	9.0	8.8	**5.6**
GP08	**5.8**	**5.8**	6.1	7.0	6.0	6.0	**3.2**
GP09	11.1	11.1	9.7	9.2	**8.6**	8.8	**6.6**
GP10	6.5	6.5	6.6	**5.5**	6.2	6.0	**3.7**
Average	29.92	29.92	29.87	**28.13**	28.94	29.80	**24.85**

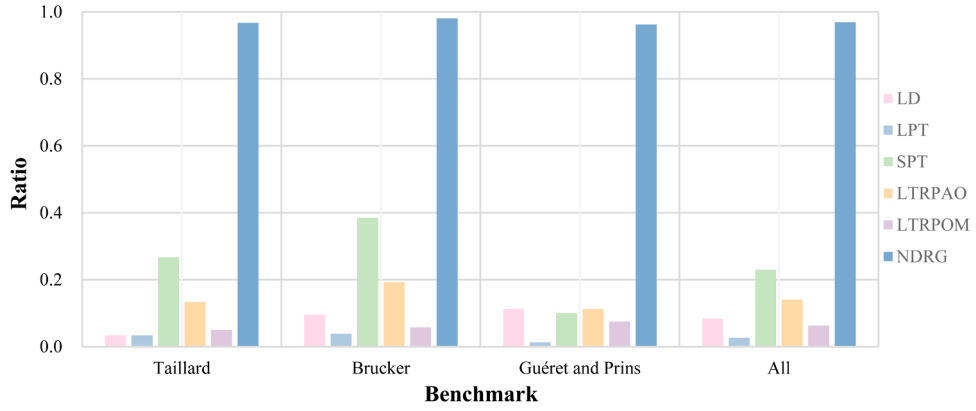


Fig. 4. Percentage of optimal solutions obtained by each algorithm on each benchmark dataset.

$$PRD_{ik} = \frac{sol_{ik} - LB_i}{LB_i} \times 100 \quad (9)$$

$$LB = \max\left(\max_{i \in M} \sum_{j \in Set1 \cup Set2} p_{ji}, \max_{j \in Set1 \cup Set2} \sum_{i=1}^{m} p_{ji} + r_j\right) \quad (10)$$

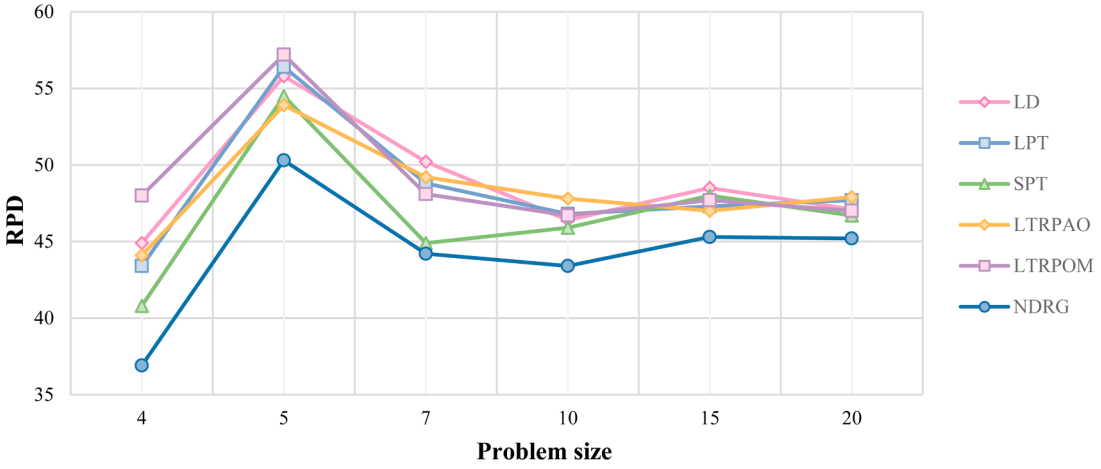


Fig. 5. Average RPD of each algorithm under different problem sizes.

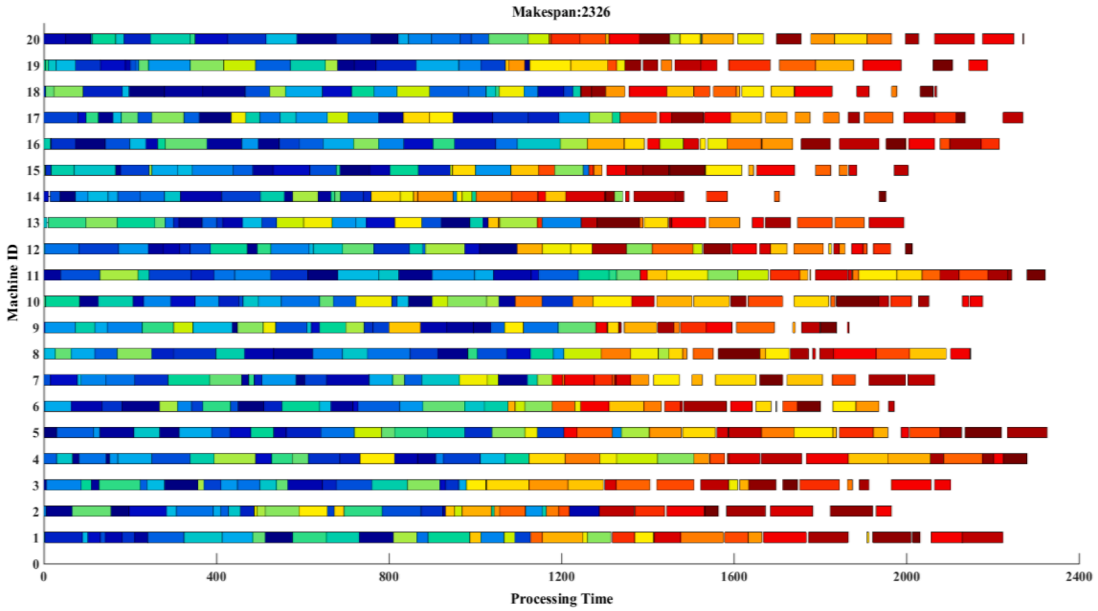


Fig. 6. Gantt chart for instance 'tai_20 × 20_8' obtained by SPT.

4.2. Comparison with other heuristics

The proposed NDRG is compared with its six constructive heuristics, namely LD, SCC, LPT, SPT, LTRPAO, and LTRPOM, in terms of algorithm effectiveness and computational time. Table 3 reports the average RPD obtained by each algorithm on these three modified benchmarks. In the modified Taillard's benchmark, NDRG provides the best results among all algorithms, while SPT also provides competitive results and yields the lowest average RPD in four groups out of six, regardless of NDRG. In the modified Bruckner et al.'s benchmark, NDRG is still the best performing algorithm, and SPT yields the lowest average RPD in five groups out of six, regardless of NDRG. In the modified Guéret and Prins's benchmark, LTRPAO narrowly beat SPT and ranks second with the lowest average RPD in three groups out of eight. Judging from these three benchmarks as a whole, NDRG is the best performing algorithm with the average RPD of 24.85%, and SPT provides the second-best result with the RPD of 28.13%. LD and SCC are the worst-performing algorithms with the RPD of 29.92% because they don't take into account the time attribute of nodes. Besides, it is easy to find out that LD and SCC are equivalent and always obtain the same results for these instances. The main reasons are as follows: (1) At the beginning, topological differences among nodes in the complex scheduling network for balanced $O_m||C_{max}$ do not work; (2) In the subsequent scheduling process, the local attributes of nodes to be selected are sufficiently similar, and LD is approximately equivalent to SCC, subject to Eq. (11). Therefore, the experimental results of SCC are no longer presented in the following text.

$$C_i = \frac{2 * E_i}{k_i * (k_i - 1)} \tag{11}$$

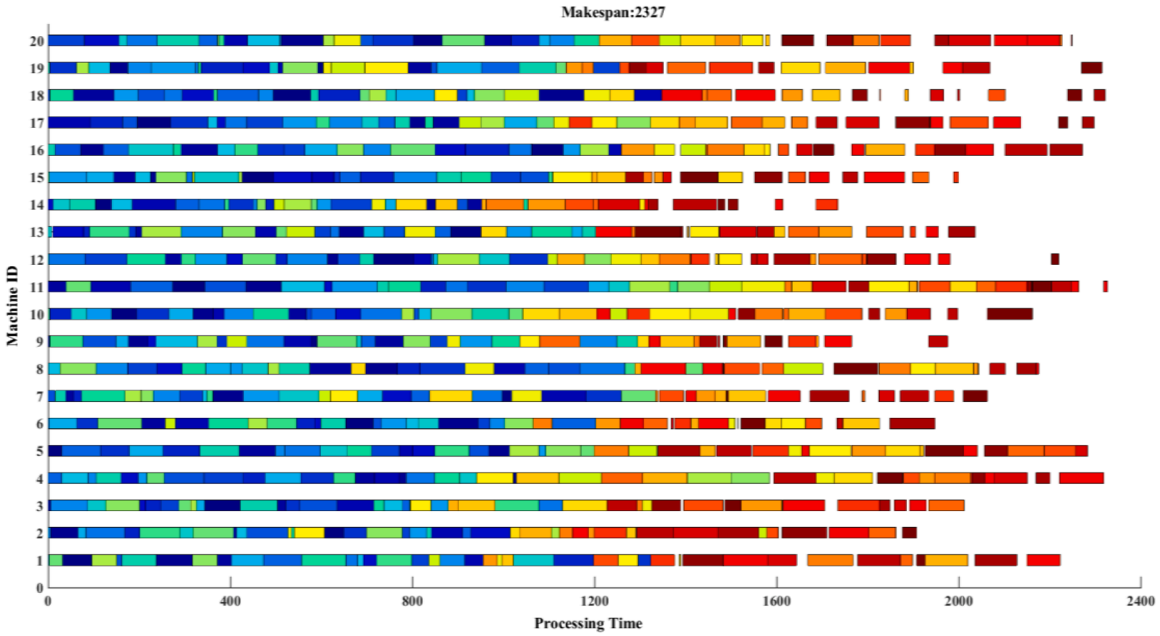


Fig. 7. Gantt chart for instance 'tai_20 × 20_8' obtained by NDRG.

Table 4
Computational times required by each algorithm for these three modified benchmarks.

Instance	LD	LPT	SPT	LTRPAO	LTRPOM	NDRG
Taillard						
tai_4 × 4	0.10	0.10	0.10	0.10	0.10	0.10
tai_5 × 5	0.28	0.28	0.28	0.28	0.28	0.28
tai_7 × 7	1.44	1.36	1.37	1.39	1.37	1.39
tai_10 × 10	7.80	7.75	7.74	7.75	7.80	7.76
tai_15 × 15	56.57	58.29	57.04	57.25	57.53	57.81
tai_20 × 20	233.99	235.69	233.23	231.74	233.57	232.70
Brucker et al.						
j3	0.03	0.03	0.03	0.03	0.03	0.03
j4	0.12	0.12	0.11	0.11	0.11	0.11
j5	0.31	0.32	0.33	0.32	0.31	0.31
j6	0.73	0.75	0.75	0.74	0.74	0.76
j7	1.55	1.54	1.53	1.52	1.53	1.54
j8	2.94	2.93	2.97	2.94	2.97	2.99
Guéret and Prins						
GP03	0.04	0.03	0.03	0.03	0.03	0.03
GP04	0.11	0.11	0.11	0.11	0.11	0.11
GP05	0.32	0.32	0.32	0.32	0.32	0.32
GP06	0.75	0.76	0.76	0.75	0.74	0.75
GP07	1.60	1.59	1.61	1.57	1.63	1.61
GP08	3.04	3.03	3.03	3.01	3.04	3.12
GP09	5.31	5.37	5.40	5.37	5.38	5.31
GP10	8.99	9.09	9.05	9.00	9.10	9.13

Algorithm 1
Select initial traversal nodes.

Input: *wtopo, pline*
Output: *Plist, Mlabel, Jlabel*
Begin
1: *Plist*←∅, *Mlabel*(1:*m*)=0, *Jlabel*(1:*n*)=0
2: *deg*=sum(*wtopo*,2)
3: **while** *deg*>0
4: *num*←min(*pline*)
5: *Plist*=[*Plist num*]
6: *Mlabel*(*col*)=*num*, *Jlabel*(*row*)=*num*
7: *wtopo*(:, col)=0, *wtopo*(col,:)=0
8: *deg*=sum(*wtopo*,2)
9: **end for**
End

Algorithm 2
Select an operation node for the machine just released.

Input: *JDone, Done, Plist, Mlabel, Jlabel,* T_1
Output: *Plist, Mlabel, Jlabel*
Begin
1: [*row, col*]←*JDone, Cnode*←∅, *Tc*←∅
2: *Unode*←setdiff(*col: m: n*m, Done*)
3: **for** *i*=0 **to** *i*=len(*Unode*)
4: *Urow*=(*Unode*(*i*)-*col*)/*m*+1
5: **if** *Jlabel*(*Urow*)==0
6: *Cnode*←[*Cnode Unode*(*i*)]
7: *Tc*←[*Tc* T_1(*Urow, col*)]
8: **end if**
9: **end for**
10: *Plist_CN, Plist_CT*←min(*Tc*)
11: *Plist*←[*Plist_CN*], [*row*1, *col*1]←*Plist_CN*
12: *Jlabel*(*row*1)←*Plist_CN, Mlabel*(*col*1)←*Plist_CN*
End

To further statistically analyze the experimental results, Fig. 4 illustrates the percentage of optimal solutions obtained by each algorithm on each benchmark dataset. We may safely conclude that the proposed NDRG exceeds the other heuristics in the number of optimal solutions obtained on these three benchmarks, which fully proves the superiority of the proposed algorithm in algorithm effectiveness. Besides, it may be

Algorithm 3
Select an operation node for the idle machine.

Input: *JDone, Done, Plist, Mlabel, Jlabel,* T_1
Output: *Plist, Mlabel, Jlabel*
Begin
1: [*row, col*]←*JDone, Cnode*←∅, *Tc*←∅
2: *Fm*←find(*Mlabel* = 0)
3: **for** *i*=0 **to** *i*=len(*Fm*)
4: **if** T_1(*row, Fm*(*i*))>0
..5: *Cnode*←[*Cnode* (*row*-1)*m+*Fm*(*i*)]
..6: *Tc*←[*Tc* T_1(*row, Fm*(*i*))]
..7: **end if**
..8: **end for**
..9: *Plist_CN*2, *Plist_CT*2←min(*Tc*)
10: *Plist*←[*Plist_CN*2], [*row*2, *col*2]←*Plist_CN*2
11: *Jlabel*(*row*2)←*Plist_CN*2, *Mlabel*(*col*2)←*Plist_CN*2
End

Algorithm 4
Select an operation node when a new job arrives.

Input: *JDone, Done, Plist, Mlabel, Jlabel,* T_1, T_0
Output: *Plist, Mlabel, Jlabel*
Begin
1: T_1=[T_1; T_0], *Cnode*←∅, *Tc*←∅
2: *Row, Col*←size(T_1)
3: *Fm*←find(*Mlabel* = 0)
4: **if** len(*Fm*)>0
5: **for** *i*=0 **to** *i*=len(*Fm*)
6: **if** T_1(*Row, Fm*(*i*))>0
7: *Cnode*←[*Cnode* (*Row*-1)*m+*Fm*(*i*)]
8: *Tc*←[*Tc* T_1(*Row, Fm*(*i*)]
9: **end if**
10: **end for**
11: **end if**
12: *Plist_CN3, Plist_CT3*←min(*Tc*)
13: *Plist*←[*Plist_CN*3], [*row*3, *col*3]←*Plist_CN*3
14: *Jlabel*(*row*3)←*Plist_CN*3, *Mlabel*(*col*3)←*Plist_CN*3
End

interesting to study the possible interaction between the problem size and the algorithm effectiveness. Fig. 5 shows the average RPD obtained by these algorithms in different problem sizes. It can be seen that the proposed NDRG outperforms the other heuristics in all sizes of the modified Taillard's benchmark, which indicates that the proposed algorithm has a higher probability of obtaining the best results for new instances. But it doesn't mean that the algorithm produces the best results in any instance. For example, SPT slightly beats NDRG for instance 'tai_20 × 20_8', and the scheduling schemes obtained by these two heuristics are presented in Fig. 6 and 7, respectively. The same results can be obtained on the other two benchmarks.

Regarding the computational time required by each algorithm, Table 4 presents the average computational time for all considered methods for each instance size. It can be safely concluded that the computational efficiency of all methods is extremely similar, which indicates that the proposed NDRG does not significantly increase the computation time on the basis of its constructive heuristics. The main reason may be that all low-level heuristic rules are performed in parallel and the time spent in the multiple attribute decision making process is negligible compared with the calculation process of node attributes. Besides, the computation time required by NDRG may be lower in some cases compared with the single low-level heuristic rule due to the improved computational efficiency caused by better traversal order. This fully confirms the excellent property of the proposed algorithm in computational time.

Algorithm 5
SPT for OSSP-DJA.

Input: T_1, T_2, RT
Output: *FET*, *ET*
Begin
1: *wtopo*←graph is built based on T_1
2: *pline*←reshape(T_1',:,1), *Done*←∅, *JDone*←∅
3: R_1,M←size(T_1), R_2=size(T_2,1), *Tnode*=(R_1+R_2)*M
4: *Plist*, *Mlabel*, *Jlabel* ←Algorithm 1 (*wtopo*, *pline*)
5: value0, ind0←min(*pline*(*node*)) for *node* in *Plist*
6: *ST_d*←value0
7: *i*=0
8: **while** *i*<=R_2
9: *i*=*i*+1
10: **if** size(*Done*,2)=*Tnode*
11: break
12: **end if**
13: **if** len(*Plist*)>0 and (*i*>R_2 or *ST_d*<*RT*(*i*))
14: *ST* = *ST_d*, *JDone*←*Plist*(ind0), *Done*←[*Done JDone*]
15: *Plist*←set diff(*Plist*, *JDone*), *ET*(*JDone*)←*ST*
16: *wtopo*(:,*Done*)=0, *wtopo*(*Done*,:)=0
17: *row_d*, *col_d*←*JDone*, *Jlabel*(*row_d*)=0, *Mlabel*(*col_d*)=0
18: **for** *i*=0 **to** *i*=len(*Plist*)
19: *pline*(*Plist*(*i*))←*pline*(*Plist*(*i*))-value0
20: **end for**
21: *Plist*, *Mlabel*, *Jlabel*←Algorithm 1 (*JDone*, *Done*, *Plist*, *Mlabel*, *Jlabel*, T_1)
22: **if** len(*Plist*)<*m*
23: *Plist*, *Mlabel*, *Jlabel*←Algorithm 3 (*JDone*, *Done*, *Plist*, *Mlabel*, *Jlabel*, T_1)
24: **end if**
25: **else if** len(*Plist*)=0
26: *wtopo*(:,*Done*)=0, *wtopo*(*Done*,:)=0
27: *Plist*←Algorithm 1 (*wtopo*, *pline*), *Plist*←setdiff(*Plist*, *Done*)
28: value0, ind0←min(*pline*(*node*)) for *node* in *Plist*
29: *ST_d*=*ST*+value0, *i*=*i*-1
30: **else if** (*i*≤R_2 and *ST_d*>*RT*(*i*))
31: *ST*=*RT*(*i*)
32: *Plist*, *Mlabel*, *Jlabel*←Algorithm 4 (*JDone*, *Done*, *Plist*, *Mlabel*, *Jlabel*, T_1, T_2(*i*,:))
33: **end if**
34: value0, ind0← min(*pline*(*node*)) for *node* in *Plist*
35: *ST_d*← *ST_d*+value0
36: **end while**
End

5. Conclusions and Future work

This study deals with real-time production scheduling problems with dynamic job arrivals, and develops a network-based dynamic dispatching rule generation mechanism by employing the complex network theory to systematically generate a series of low-level heuristics from the perspective of system optimization and utilizing the multiple attribute decision making method to solve the automatic heuristic generation problem of generation constructive hyper-heuristics. Numerical results indicate that the proposed method can effectively aggregate the information of each heuristic rule, and obtain an approximate solution that is significantly better than that of any single heuristic rule, without significantly increasing the calculation time. The study on this subject makes an important contribution to the overall knowledge in the field of real-time production scheduling. From another point of view, the proposed automatic generation mechanism based on multiple attribute decision making bridges the fields of generation constructive hyper-heuristic and multiple attribute decision making, and the development of network-based dynamic dispatching rule generation mechanism aslo expands the application of complex newtork in the complex dynamic production scheduling.

Taking all points into consideration, future work is still required to further improve the performance of the proposed network-based dynamic dispatching rule generation method for real-time production scheduling problems. The first extension is to develop more effective low-level heuristic rules from the perspective of system optimization. The other extension is to take into account more effective multiple attribute decision making methods. Besides, the proposed network-based dynamic dispatching rule generation method is required to be investigated in more practical production scheduling cases.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors would like to acknowledge the financial support of the National Natural Science Foundation of China (No. 51775348), and National Key Research and Development Program of China (No. 2019YFB1704401).

References

[1] J. Zhang, G. Ding, Y. Zou, S. Qin, J. Fu, Review of job shop scheduling research and its new perspectives under Industry 4.0, J. Intell. Manuf. 30 (4) (2019) 1809–1830, https://doi.org/10.1007/s10845-017-1350-2.
[2] E. Oztemel, S. Gursev, Literature review of Industry 4.0 and related technologies, J. Intell. Manuf. 31 (1) (2020) 127–182, https://doi.org/10.1007/s10845-018-1433-8.
[3] A. Kusiak, Smart manufacturing, Int. J. Prod. Res. 56 (1-2) (2018) 508–517, https://doi.org/10.1080/00207543.2017.1351644.
[4] F. Tao, Q. Qi, A. Liu, A. Kusiak, Data-driven smart manufacturing, J. Manuf. Syst. 48 (2018) 157–169, https://doi.org/10.1016/j.jmsy.2018.01.006.
[5] E.M. Frazzon, M. Kück, M. Freitag, Data-driven production control for complex and dynamic manufacturing systems, CIRP Ann. Manuf. Technol. 67 (1) (2018) 515–518, https://doi.org/10.1016/j.cirp.2018.04.033.
[6] D. Bai, J. Liang, B. Liu, M. Tang, Z.H. Zhang, Permutation flow shop scheduling problem to minimize nonlinear objective function with release dates, Comput. Ind. Eng. 112 (2017) 336–347, https://doi.org/10.1016/j.cie.2017.08.031.
[7] P. Zheng, J. Wang, J. Zhang, C. Yang, Y. Jin, An adaptive CGAN/IRF-based rescheduling strategy for aircraft parts remanufacturing system under dynamic environment, Robot. Comput.-Integr. Manuf. 58 (2019) 230–238, https://doi.org/10.1016/j.rcim.2019.02.008.
[8] T. Zhou, D. Tang, H. Zhu, Z. Zhang, Multi-agent reinforcement learning for online scheduling in smart factories, Robot. Comput.Integr. Manuf. 72 (2021), 102202, https://doi.org/10.1016/j.rcim.2021.102202.
[9] C. Qian, Y. Zhang, C. Jiang, S. Pan, Y. Rong, A real-time data-driven collaborative mechanism in fixed-position assembly systems for smart manufacturing, Robot. Comput.Integr. Manuf. 61 (2020), 101841, https://doi.org/10.1016/j.rcim.2019.101841.
[10] Y. Zhou, J.J. Yang, Z. Huang, Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming, Int. J. Prod. Res. 58 (9) (2020) 2561–2580, https://doi.org/10.1080/00207543.2019.1620362.
[11] M. Rauf, Z. Guan, S. Sarfraz, J. Mumtaz, E. Shehab, M. Jahanzaib, M. Hanif, A smart algorithm for multi-criteria optimization of model sequencing problem in assembly lines, Robot. Comput.Integr. Manuf. 61 (2020), 101844, https://doi.org/10.1016/j.rcim.2019.101844.
[12] M.M. Nasiri, R. Yazdanparast, F. Jolai, A simulation optimisation approach for real-time scheduling in an open shop environment using a composite dispatching rule, Int. J. Comput. Integr. Manuf. 30 (12) (2017) 1239–1252, https://doi.org/10.1080/0951192X.2017.1307452.
[13] E.K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: a survey of the state of the art, J. Oper. Res. Soc. 64 (12) (2013) 1695–1724, https://doi.org/10.1057/jors.2013.71.
[14] J. Branke, S. Nguyen, C.W. Pickardt, M. Zhang, Automated design of production scheduling heuristics: A review, IEEE Trans. Evol. Comput. 20 (1) (2015) 110–124, https://doi.org/10.1109/TEVC.2015.2429314.
[15] M.G. Epitropakis, E.K. Burke, Hyper-heuristics. Handbook of Heuristics, 2018, pp. 1–57, https://doi.org/10.1007/978-3-319-07153-4_32-1.
[16] H.R. Topcuoglu, A. Ucar, L. Altin, A hyper-heuristic based framework for dynamic optimization problems, Appl. Soft. Comput. 19 (2014) 236–251, https://doi.org/10.1016/j.asoc.2014.01.037.
[17] C.W. Pickardt, T. Hildebrandt, J. Branke, J. Heger, B. Scholz-Reiter, Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems, Int. J. Prod. Econ. 145 (1) (2013) 67–77, https://doi.org/10.1016/j.ijpe.2012.10.016.
[18] S. Luo, Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning, Appl. Soft. Comput. 91 (2020), 106208, https://doi.org/10.1016/j.asoc.2020.106208.
[19] Y. Liu, Y. Mei, M. Zhang, Z. Zhang, A Predictive-reactive approach with genetic programming and cooperative coevolution for the uncertain capacitated arc routing problem, Evol. Comput. 28 (2) (2020) 289–316, https://doi.org/10.1162/evco_a_00256.

[20] B. Yang, J. Geunes, Predictive–reactive scheduling on a single resource with uncertain future jobs, Eur. J. Oper. Res. 189 (3) (2008) 1267–1283, https://doi.org/10.1016/j.ejor.2006.06.077.
[21] D. Ouelhadj, S. Petrovic, A survey of dynamic scheduling in manufacturing systems, J. Sched. 12 (4) (2009) 417–431, https://doi.org/10.1007/s10951-008-0090-8.
[22] J. Long, Z. Sun, P.M. Pardalos, Y. Bai, S. Zhang, C. Li, A robust dynamic scheduling approach based on release time series forecasting for the steelmaking-continuous casting production, Appl. Soft. Comput. 92 (2020), 106271, https://doi.org/10.1016/j.asoc.2020.106271.
[23] F.E. Minguillon, N. Stricker, Robust predictive–reactive scheduling and its effect on machine disturbance mitigation, CIRP Ann. Manuf. Technol. 69 (1) (2020) 401–404, https://doi.org/10.1016/j.cirp.2020.03.019.
[24] B. Mihoubi, B. Bouzouia, M. Gaham, Reactive scheduling approach for solving a realistic flexible job shop scheduling problem, Int. J. Prod. Res. (2020) 1–19, https://doi.org/10.1080/00207543.2020.1790686.
[25] D. Spensieri, E. Åblad, R. Bohlin, J.S. Carlson, R. Söderberg, Modeling and optimization of implementation aspects in industrial robot coordination, Robot. Comput. Integr. Manuf. 69 (2021), 102097, https://doi.org/10.1016/j.rcim.2020.102097.
[26] S. Chand, Q. Huynh, H. Singh, T. Ray, M. Wagner, On the use of genetic programming to evolve priority rules for resource constrained project scheduling problems, Inf. Sci. 432 (2018) 146–163, https://doi.org/10.1016/j.ins.2017.12.013.
[27] M. Đumić, D. Jakobović, Ensembles of priority rules for resource constrained project scheduling problem, Appl. Soft. Comput. 110 (2021), 107606, https://doi.org/10.1016/j.asoc.2021.107606.
[28] F. Zhang, Y. Mei, S. Nguyen, M. Zhang, Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling, IEEE T. Cybern. (2021) 1–15, https://doi.org/10.1109/TCYB.2021.3050141.
[29] G. Ozturk, O. Bahadir, A. Teymourifar, Extracting priority rules for dynamic multi-objective flexible job shop scheduling problems using gene expression programming, Int. J. Prod. Res. 57 (10) (2019) 3121–3137, https://doi.org/10.1080/00207543.2018.1543964.
[30] N. Pillay, R. Qu, Hyper-Heuristics: Theory and Applications, Springer International Publishing, 2018.
[31] E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, J.R. Woodward, A classification of hyper-heuristic approaches, in: Handbook of Metaheuristics, 146, 2010, pp. 449–468, https://doi.org/10.1007/978-1-4419-1665-5_15.
[32] S.S. Choong, L.P. Wong, C.P. Lim, Automatic design of hyper-heuristic based on reinforcement learning, Inf. Sci. 436 (2018) 89–107, https://doi.org/10.1016/j.ins.2018.01.005.
[33] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440, https://doi.org/10.1038/30918.
[34] A.L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512, https://doi.org/10.1126/science.286.5439.509.
[35] Z. Zhuang, Y. Chen, Y. Sun, W. Qin, Complex scheduling network: an objective performance testing platform for evaluating vital nodes identification algorithms, Int. J. Adv. Manuf. Technol. 111 (2020) 273–282, https://doi.org/10.1007/s00170-020-06145-5.
[36] Q. Xuan, T.J. Wu, Network model and heuristic scheduling rule designing method for complex open shop problems, J. Zhejiang Univ. (Eng. Sci.) 45 (6) (2011) 961–968, https://doi.org/10.3785/j.issn.1008-973X.2011.06.001.
[37] F. Zhang, Y. Mei, S. Nguyen, M. Zhang, Correlation coefficient-based recombinative guidance for genetic programming hyperheuristics in dynamic flexible job shop scheduling, IEEE Trans. Evol. Comput. 25 (3) (2021) 552–566, https://doi.org/10.1109/TEVC.2021.3056143.
[38] Z. Zhuang, Z. Huang, L. Chen, W. Qin, A heuristic rule based on complex network for open shop scheduling problem with sequence-dependent setup times and delivery times, IEEE Access 7 (2019) 140946–140956, https://doi.org/10.1109/ACCESS.2019.2944296.
[39] J. Pempera, C. Smutnicki, Open shop cyclic scheduling, Eur. J. Oper. Res. 269 (2) (2018) 773–781, https://doi.org/10.1016/j.ejor.2018.02.021.
[40] W. Qin, Z. Zhuang, Y. Liu, O. Tang, A two-stage ant colony algorithm for hybrid flow shop scheduling with lot sizing and calendar constraints in printed circuit board assembly, Comput. Ind. Eng. 138 (2019), 106115, https://doi.org/10.1016/j.cie.2019.106115.
[41] C. Yang, S. Lan, L. Wang, Research on coordinated development between metropolitan economy and logistics using big data and Haken model, Int. J. Prod. Res. 57 (4) (2019) 1176–1189, https://doi.org/10.1080/00207543.2018.1503427.
[42] J.W. Kang, H.J. Park, J.S. Ro, H.K. Jung, A strategy-selecting hybrid optimization algorithm to overcome the problems of the no free lunch theorem, IEEE Trans. Magn. 54 (3) (2018) 1–4, https://doi.org/10.1109/TMAG.2017.2750204.
[43] U.K. Uz Zaman, M. Rivette, A. Siadat, S.M. Mousavi, Integrated product-process design: Material and manufacturing process selection for additive manufacturing using multi-criteria decision making, Robot. Comput.Integr. Manuf. 51 (2018) 169–180, https://doi.org/10.1016/j.rcim.2017.12.005.
[44] J.J. Liou, Y.C. Chuang, E.K. Zavadskas, G.H. Tzeng, Data-driven hybrid multiple attribute decision-making model for green supplier evaluation and performance improvement, J. Clean Prod. 241 (2019), 118321, https://doi.org/10.1016/j.jclepro.2019.118321.
[45] G.H. Mendonça, F.G. Ferreira, R.T. Cardoso, F.V. Martins, Multi-attribute decision making applied to financial portfolio optimization problem, Expert Syst. Appl. 158 (2020), 113527, https://doi.org/10.1016/j.eswa.2020.113527.
[46] Z. Liu, C. Jiang, J. Wang, H. Yu, The node importance in actual complex networks based on a multi-attribute ranking method, Knowl. Based Syst.. 84 (2015) 56–66, 10.1016/j.knosys.2015.03.026.
[47] L. Fei, H. Mo, Y. Deng, A new method to identify influential nodes based on combining of existing centrality measures, Mod. Phys. Lett. B 31 (26) (2017), 1750243, https://doi.org/10.1142/S0217984917502438.
[48] Y. Dong, Y. Liu, H. Liang, F. Chiclana, E. Herrera-Viedma, Strategic weight manipulation in multiple attribute decision making, Omega Int. J. Manage. Sci. 75 (2018) 154–164, https://doi.org/10.1016/j.omega.2017.02.008.
[49] C. Wang, M. Xu, G. Olsson, Y. Liu, Characterizing of water-energy-emission nexus of coal-fired power industry using entropy weighting method, Resour. Conserv. Recycl. 61 (2020), 104991, https://doi.org/10.1016/j.resconrec.2020.104991.
[50] A. Zareie, A. Sheikhahmadi, M. Jalili, Influential node ranking in social networks based on neighborhood diversity, Futur. Gener. Comp. Syst. 94 (2019) 120–129, https://doi.org/10.1016/j.future.2018.11.023.
[51] S. Lan, C. Yang, G.Q. Huang, Data analysis for metropolitan economic and logistics development, Adv. Eng. Inform. 32 (2017) 66–76, https://doi.org/10.1016/j.aei.2017.01.003.
[52] E. Taillard, Benchmarks for basic scheduling problems, Eur. J. Oper. Res. 64 (2) (1993) 278–285, https://doi.org/10.1016/0377-2217(93)90182-M.
[53] P. Brucker, J. Hurink, B. Jurisch, B. Wöstmann, A branch & bound algorithm for the open-shop problem, Discret Appl. Math. 76 (1-3) (1997) 43–59, https://doi.org/10.1016/S0166-218X(96)00116-3.
[54] C. Guéret, C. Prins, A new lower bound for the open-shop problem, Ann. Oper. Res. 92 (1999) 165–183, https://doi.org/10.1023/A:1018930613891.
[55] B. Naderi, M. Zandieh, Modeling and scheduling no-wait open shop problems, Int. J. Prod. Econ. 158 (2014) 256–266, https://doi.org/10.1016/j.ijpe.2014.06.011.