Contents lists available at ScienceDirect

Journal of Cleaner Production

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

Evolutionary game based real-time scheduling for energy-efficient distributed and flexible job shop

Jin Wang [a], Yang Liu [b, c, *], Shan Ren [a, **], Chuang Wang [a], Wenbo Wang [d]

[a] School of Modern Posts, Xi'an University of Posts & Telecommunications, Xi'an, 710061, PR China
[b] Department of Management and Engineering, Linköping University, SE-581 83 Linköping, Sweden
[c] Department of Production, University of Vaasa, 65200 Vaasa, Finland
[d] School of Mechanical Engineering, Jiangsu University, Zhenjiang, 212013, PR China

ARTICLE INFO

Article history:
Received 29 June 2020
Received in revised form
5 January 2021
Accepted 22 January 2021
Available online 26 January 2021

Handling editor: Yutao Wang

Keywords:
Energy efficiency
Distributed and flexible job shop
Real-time scheduling
Real-time data
Evolutionary game

ABSTRACT

With the global energy crisis and environmental issues becoming severe, more attention has been paid to production scheduling considering energy consumption than ever before. However, in the context of intelligent manufacturing, most studies apply the industrial internet of things (IIoT) to improve energy efficiency. It may cause the real-time data in the workshop unable to be collected and treated timely, thus affecting the real-time decision-making of the scheduling system. Edge computing (EC) can make full use of embedded computing capabilities of field devices to process real-time data and reduce the response time of making production decisions. Therefore, in this study, an overall architecture of the EC-IIoT based distributed and flexible job shop real-time scheduling (DFJS-RS) is proposed to enhance the real-time decision-making capability of the scheduling system. The DFJS-RS method, which consists of the task assignment method of the shop floor layer and the RS method of the flexible manufacturing units (FMUs) layer, is designed and developed. An evolutionary game-based solver method is adopted to obtain the optimal allocation. Finally, a case study is employed to validate the DFJS-RS method. The results show that compared with the existing production scheduling method, the DFJS-RS method can improve energy efficiency by up to 26%. This improvement can further promote cleaner production (CP) and sustainable societal development.

1. Introduction

At present, energy conservation and emission reduction is an important issue facing by the industrial enterprise (Yao et al., 2019). According to the 2019 international energy outlook, energy use in the world's industrial sector grows by more than 30% between 2018 and 2050, reaching an estimated 315 quadrillion British thermal units by 2050 (Energy Information Administration, 2019). In China, industrial enterprises used at least 50% of the country's electricity energy and emitted more than 26% of carbon dioxide (Y. Liu et al., 2014). Therefore, it is extremely urgent to study efficient technologies and methods of energy conservation and emission reduction to enhance energy efficiency for achieving cleaner production (CP) and sustainable societal development.

In general, there are two main aspects of the study on improving the energy efficiency of the manufacturing enterprise: equipment-level and production management-level (Huang and Yu, 2017; Zhang et al., 2020). Since upgrading the equipment requires a lot of investment, equipment-level energy-efficient approaches may not be suitable for some small businesses. Thus, more and more scholars are interested in studying how to use the production management method, especially production scheduling technology, to enhance energy efficiency (Dai et al., 2019a; Plitsos et al., 2017).

Recently, with the development of Industrial Internet of things (IIoT), big data and edge computing (EC), more and more manufacturing enterprises begin to utilise these advanced information technologies to realise real-time data-based workshop management and control (C. Wang et al., 2018; Liu et al., 2020; Wang and Zhang, 2020; Li et al., 2020). At present, by extending the EC and IIoT to the production scheduling fields, real-time data in the manufacturing process becomes more accessible, thus forming

* Corresponding author. Department of Management and Engineering, Linköping University, SE-581 83 Linköping, Sweden.
** Corresponding author.
E-mail addresses: yang.liu@liu.se (Y. Liu), renshan@xupt.edu.cn (S. Ren).

Abbreviations

CP	Cleaner production
DFJS	Distributed and flexible job shop
DFJS-RS	DFJS real-time scheduling
EC	Edge computing
EDSM	Existing dynamic scheduling method
FJS	Flexible job shop
FMU	Flexible manufacturing unit
IIoT	Industrial Internet of things
JP_l	Job pool l
JPW	Job pool of workshop
LPT	Longest processing time
MAR	Machine assignment rule
$RSJP_l$	RS job pool
SMJ	Set of the machined job
SMJ_l	SMJ of F_l
SPT	Shortest processing time
TPQ-CM	Temporary processing queue of the corresponding machine

a big data manufacturing environment (Zhang et al., 2018). Thus, some scholars try to integrate smart sensors technology, enhanced data analytics tools, visual monitoring and production scheduling to realise real-time data-driven energy-efficient production scheduling (Biel et al., 2018). For example, Kim et al. (2017) developed a workshop scheduling system based on real-time energy consumption monitoring for minimising energy consumption. W. Wang et al. (2018) proposed a real-time optimisation approach based on real-time energy data for high energy consumption manufacturing enterprises.

Although these researches have made great progress, there is an urgent problem that needs to be solved, i.e., how to realise energy-efficient distributed and flexible job shop (DFJS) scheduling based on real-time data in big data manufacturing environment. The specific problems are shown as follows.

(1) How to develop a production scheduling method that considers real-time manufacturing data to realise CP in DFJS? The traditional DFJS scheduling problem mainly focuses on production efficiency (i.e. local makespan and global makespan), and seldom involves other aspects, especially environmental problems (Ziaee, 2014). However, with the increasing prominence of energy deficiency in recent years, it is worth considering to design a DFJS scheduling method that can satisfy both time and energy objectives. Especially in the context of intelligent manufacturing, it is necessary to combine advanced information technologies (i.e. IIoT, big data and EC) with production scheduling technology to enhance DFJS production efficiency and realise energy-efficient production scheduling. Thus, a new DFJS real-time scheduling (DFJS-RS) paradigm should be developed to realise CP through the newest information technology.

(2) How to realise real-time assignment of production jobs based on real-time data in DFJS to enhance production efficiency and energy efficiency at the same time? At present, no research on dynamic DFJS scheduling problem has been found after a rigorous literature search. For the dynamic flexible job shop (FJS) scheduling, the following two dynamic scheduling strategies are generally adopted: periodic rescheduling policy and event-driven rescheduling policy (Pfund and Fowler, 2017; Zhang and Wong, 2017). However, in the face of the frequent occurrence of abnormal events in the manufacturing process, these existing methods have apparent defects. In the periodic rescheduling policy, the dynamic scheduling system cannot timely respond to the workshop dynamic disturbance. In the event-driven rescheduling policy, frequent rescheduling may make the dynamic scheduling system unstable. Moreover, the existing dynamic scheduling method (EDSM) is a centralised decision-making model, and the computational complexity is higher with the increase of scheduling scale. Thus, it is necessary to propose a new real-time data-driven RS method based on an evolutionary game for DFJS to reduce the scheduling complexity, improve production efficiency and realise sustainable production.

To solve the above issues, an evolutionary game-based DFJS-RS with EC-IIoT method was proposed, which provides a new paradigm for distributed RS problem. The main contributions of this study are in the following four aspects.

(1) The EC-IIoT is applied to the DFJS and an overall architecture of the EC-IIoT based DFJS-RS is proposed. This framework can effectively deal with the data explosion and shorten the device response time, thus better solving real-time decision-making problems.
(2) The RS method is adopted to assign operations to suitable machines based on real-time data. Compared with the traditional rescheduling strategy, the RS method does not generate the scheduling initially, and the real-time assignment of jobs is carried out once each time. Therefore, the production system will be more stable and continuous due to eliminating the deviation between the new and the original schedule.
(3) The evolutionary game-based allocation method is used to allocate the operations in real-time. Compared with the existing solution algorithm, the proposed method lets a machine only select one operation each time. Thus, the complexity of the scheduling problem is stable even when the numbers of jobs and machines increase.
(4) The evolutionary game equilibrium solution is the optimal result of the DFJS-RS problem. It can avoid the disadvantages of the traditional multi-objective optimisation method. For example, selecting feasible solutions in Pareto optimisation or determining weight coefficient in weight approach is challenging.

Therefore, the proposed EC-IIoT based DFJS-RS is not just another study dealing with general scheduling problems but provides a novel method for improving the RS efficiency and realising energy-efficient production scheduling.

The remainder of this study is arranged as follows. Section 2 reviewed the related works on energy-efficient production scheduling and real-time production scheduling. Section 3 proposed the overall architecture of the EC-IIoT based DFJS-RS. Section 4 described the DFJS-RS model. Section 5 proposed an evolutionary game based solve method for DFJS-RS. Section 6 presented a case study to demonstrate the effectiveness of the proposed DFJS-RS. Section 7 summarised the conclusions and future work.

2. Literature review

First, two relevant literature streams, i.e. energy-efficient production scheduling and real-time production scheduling, are reviewed. Then a literature analysis is conducted to summarise the research gaps.

2.1. Energy-efficient production scheduling

Since the 1950s, many scholars have started to study various strategies and methods to deal with the production scheduling problem (Giglio et al., 2017). Nevertheless, up to the beginning of the 21st century, the research on energy-efficient production scheduling problem has not appeared. Energy-efficient production scheduling, also known as energy-saving scheduling, considers CP or green manufacturing problem in existing production scheduling (Ding and Yang, 2013). The first attempts to improve energy efficiency employing a production scheduling method was proposed by Mouzon et al. (2007). They used an operation method to minimise the energy consumption of the production machine. Subsequently, with people's attention to environmental problems, energy-efficient production scheduling has gradually become a hot issue (Jiang et al., 2018). In recent years, there are more and more researches on the integration of production scheduling and energy consumption problem, aiming at promoting CP (Gao et al., 2018; Dai et al., 2019b; Nouiri et al., 2019). At present, the study on improving the energy efficiency of the production process through production scheduling is generally carried out in the following four kinds of workshops: single-machine shop, flow-shop, job shop and FJS.

For the single-machine shop, C. G. Liu et al. (2014) studied an operational decision-making problem to minimise completion time and energy consumption. A solution model to optimise the completion time and the energy consumption is proposed in a single-machine system by Yildirim and Mouzon (2012). At the same time, they developed a genetic algorithm to obtain the near-optimal solution in the multi-objective optimisation problem. For the flow-shop, a new particle swarm optimisation method was adopted by Tang et al. (2016), to address the dynamic flexible flow shop scheduling problem considering the makespan and energy consumption. Liu et al. (2017) introduced a fuzzy set theory to optimise tardiness and energy consumption in a flow shop system. For the job shop, to improving productivity and reduce carbon dioxide emissions, May et al. (2015) studied the production scheduling strategies in a job shop. Masmoudi et al. (2019) developed an integer linear programming method to solve job shop scheduling problem considering energy efficiency. For the FJS, Yin et al. (2017) presented a novel low-carbon scheduling method considering productive, energy consumption and noise for the FJS environment. Gong et al. (2019) proposed an integrated energy and labour perception multi-objective FJS scheduling approach that considers makespan, total energy consumption, labour cost and workload.

From the analysis of the above literature review, we can know that many experts and scholars have studied the energy-efficient production scheduling problem to improve energy efficiency and promote CP. However, compared with other types of workshop energy-efficient production scheduling problems, the research on energy-efficient FJS scheduling problem has just started in recent years, and there are still many research topics to be explored. More importantly, DFJS is an extension of FJS, but the DFJS scheduling problem considering energy consumption has not been discussed. DFJS is a very typical type of workshop in a manufacturing shop floor. With the increasing attention to environmental issues and the development of green manufacturing, it is particularly important to study energy-efficient DFJS scheduling related to energy consumption.

Besides, the existing research on real-time energy information-driven energy-efficient production scheduling is quite limited. Through a rigorous literature search, only a few papers have been found to deal with this problem. For example, Ding and Wu (2019) proposed a multi-objective fuzzy method based energy loss optimisation scheduling modelling in the IIoT environment. Tian et al. (2019a) proposed a rescheduling method in the IIoT environment to solve the energy-efficient production scheduling and real-time control problem in the FJS. However, these works still did not involve the energy-efficient DFJS scheduling problem.

2.2. Real-time production scheduling

The first paper on dynamic production scheduling was written by Holloway and Nelson (1974). To solve the dynamic job shop scheduling problem, they developed a new heuristic scheduling method. Then, Muhlemann et al. (1982) proposed a job shop scheduling framework and studied job shop scheduling in a real environment. They discussed the periodic rescheduling policy. Later, Church and Uzsoy (1992) adopted two rescheduling policies to solve a dynamic workshop scheduling problem in the case of rush order arrivals. Since then, more and more people have begun to study the dynamic production scheduling problem (Kundakci and Kulak, 2016; Shahgholi Zadeh et al., 2019). To realise sustainable manufacturing, some scholars have studied how to use RS to realise CP (W. Wang et al., 2018; Wang et al., 2020; Zhang et al., 2017a). At present, there are three fundamental approaches to address the dynamic production scheduling problem in the workshop: reactive, proactive and proactive-reactive scheduling methods (Lou et al., 2012).

For the reactive scheduling method, Tay and Ho (2008) presented genetic programming based dispatching rules method to address the multi-objective FJS scheduling problem. Rahmani and Ramezanian (2016) developed a novel reactive model to solve a dynamic flexible flow shop scheduling problem considering rush order into the process as disruptions. For the proactive scheduling method, Zhang et al. (2016) presented a Pareto-optimal approach to obtain a robust schedule for an FJS scheduling problem with flexible workdays. Nouiri et al. (2017) used a new particle swarm optimisation algorithm to study the FJS scheduling problem. For the proactive-reactive scheduling method, Gao et al. (2015) studied the FJS rescheduling problem for new job insertion and proposed four heuristic algorithms. Zhang and Wong (2017) integrated an ant colony algorithm and multi-agent system to study the FJS rescheduling problem under a dynamic environment. However, because there is no real-time interaction between manufacturing resources, the accuracy of the rescheduling scheme produced by the methods in the above literature is easily affected.

Thanks to the advent of advanced information technology, the applications of IIoT in the workshop provide an opportunity to reduce the above gap. Through the establishment of an IIoT-enabled workshop, the status of manufacturing resource and job progress can be perceived in real-time, true, and accurate. At present, some papers have started to use the IIoT technology to realise RS of the workshop. For example, the authors' previous paper proposed some scheduling strategies to realise RS based on real-time data in an IIoT-enabled FJS (Zhang et al., 2017b; Wang et al., 2019). Turker et al. (2019) developed a real-time data-based decision support system to solve the dynamic job shop scheduling problem using dispatching rules. However, the application of IIoT technology in the RS can produce a lot of real-time manufacturing data. How to effectively store and process these real-time data in the RS process is a problem that needs to be considered.

To cope with the above challenges, this study presented an EC-IIoT based DFJS-RS to realise energy-efficient DFJS scheduling based on real-time data by using the evolutionary game.

2.3. Literature analysis

To further clarify the differences between this study and published works in a similar area, a brief review of recent literature on multi-objective workshop dynamic optimisation problem is conducted. We review the existing research from four aspects: application of smart technology, rescheduling strategy, solution algorithm and multi-objective optimisation method.

(1) A considerable part of the existing research does not involve smart technologies (Shen and Yao, 2015; Hosseinabadi et al., 2015; Salido et al., 2017). Although in recent years, more and more scholars began to adopt various smart technologies to promote the development of dynamic scheduling, most of these were only carried out in the context of IoT (Zhang et al., 2018; W. Wang et al., 2018; Turker et al., 2019; Tian et al., 2019b). The application of IoT leads to the generation of big data. How to effectively store and handle big data is a severe problem.

(2) Majority of the recent research on dynamic scheduling adopt event-driven rescheduling strategy (Sreekara Reddy et al., 2018; Z. Wang et al., 2019). For the event-driven rescheduling strategy, dynamic scheduling is performed when the previous schedule is modified to accommodate the new manufacturing environment. However, the new schedule may be completely different from the original one, meaning that the unprocessed operations in the original schedule would be processed earlier or later. It has a severe effect on other production activities planning related to the original schedule and reduces the stability of the production scheduling system.

(3) With the development of various solution algorithms, most of the research employs intelligent algorithm to solve scheduling problems (Valledor et al., 2018; Mourtzis and Vlachou, 2018; Zhang et al., 2019). These intelligent algorithms typically allocate all the unprocessed operations to the appropriate machines through a centralised allocation method. However, such a centralised method has high computational complexity.

(4) Current research dealing with multi-objective optimisation problems mainly include the Pareto optimisation and the weighted approach (Fang et al., 2019; Shi et al., 2019). However, Pareto optimisation requires decision-makers to choose from a large number of alternative solutions at each decision point, which is practically infeasible (Feng et al., 2020; W. Wang et al., 2020; Li and Wen, 2020). For the weighted approach, the decision-makers may not always be experienced or knowledgeable enough to define such specific weights for each objective (Ozturk et al., 2019).

Through the above analysis, it can be seen that there are still many deficiencies in the current research on multi-objective workshop dynamic scheduling. Therefore, in our study, an overall architecture of the EC-IIoT based DFJS-RS is proposed to enhance the real-time decision-making capability of the scheduling system.

In addition, the comparison results between this study and the existing literature on workshop multi-objective dynamic scheduling are summarised in Table 1. Based on the above analysis, our proposed method is superior to the existing multi-objective dynamic scheduling method in four aspects: application of smart technology, rescheduling strategy, solution algorithm and multi-objective optimisation method.

3. The overall architecture of the EC-IIoT based DFJS-RS

The overall architecture of the EC-IIoT based DFJS-RS is shown in Fig. 1. The purpose is to use EC-IIoT technology to establish a real-time data acquisition and processing model and realise real-time manufacturing-information-driven RS in the DFJS.

In the RS stage, each edge device obtains the real-time data of the corresponding machine through the IIoT device and processes these insignificant data to form real-time manufacturing information. The task of the edge device is to monitor and control the target machine and to transmit real-time manufacturing information from the machine to the cloud centre. Meanwhile, the cloud centre automatically obtains the real-time job information, e.g. the cutting time, setup time and cutting power etc. of each machine and its request of the operations when it is idle each time. Therefore, manufacturing resources can regularly interact with each other. Only one optimal operation is assigned each time to the requested machine according to their real-time manufacturing information. When the machine finishes the assigned operation, it automatically sends its current status and requests the operations until all the operations are finished. In the proposed method, at any time, each machine can obtain one optimal operation. The complexity of this problem is stable with increased operations because only one optimal operation is selected for one machine each time. Since the operation allocation is real-time data-driven and the proposed RS method is only started for the idle machine, the scheduling efficiency can be dramatically improved.

The implementation processes of DFJS-RS include two parts, namely shop floor layer and FMUs layer. The shop floor layer assigns all jobs to a suitable FMU and outputs an assignment result for each FMU. This layer aims to increase productivity and balance all FMUs workload. The FMUs layer contains some single FMU, which works simultaneously. In each FMU, the operations from upper-level assignment results are assigned to a suitable machine based on the real-time information of manufacturing resources. The aim of this layer is not only to increase productivity and balance workload for all machines by FMU itself but also to improve energy efficiency.

The detailed implementation of DFJS-RS consists of two steps:

Step 1: before the RS starts, all manufacturing data of the workshop can be known by the information management system. Then, products are divided into several independent jobs according to their machining characteristics. Next, all unallocated jobs are put into a job pool of workshop (JPW) and all FMUs request to undertake the jobs of JPW. By using the evolutionary game, each FMU can get a job from the JPW at a time. Repeat the above process until all jobs assigned to the most suitable FMU. Thus, which jobs should be processed in which FMU is determined.

Step 2: during the RS stage of FMU l, at time t_0, the first unallocated operations of all jobs of FMU l are added into an RS job pool l (RSJP_l). Here, FMU l denotes one of all FMUs. Then, each machine of FMU l requests an operation of the RSJP_l. Next, each machine continually interacts with operations and other manufacturing resources. At last, some operations of the RSJP_l are allocated to the corresponding machines based on the real-time status of machines using the evolutionary game. At each time t of the subsequent RS, the allocation processes in time t_0 are used to assign all operations to corresponding machines. In this step, the operations assignment is based on real-time manufacturing information. Thus, when abnormal events occur (e.g., machine breakdown, worker absenteeism), the adverse effects brought by abnormal events can be quickly removed and eliminated.

Step 3: If a rush order has happened, the new arriving jobs are assigned to the suitable FMU immediately according to the method of step 1. Then, the operations of the rush order of FMU l are allocated to the corresponding machines based on the method of step 2.

4. The DFJS-RS model

To implement the task assignment of the shop floor layer and the RS of the FMUs layer, this section presents the optimisation

Table 1
Summary of the literature on the multi-objective workshop dynamic optimisation problem.

Reference	Workshop category	Application of smart technology	Reschedule strategy	Solution algorithm	Multi-objective optimisation method
Shen and Yao (2015)	Flexible job shop	None	Event-driven	Multi-objective evolutionary	Pareto optimisation
Hosseinabadi et al. (2015)	Flexible job shop	None	Event-driven	Local search algorithm	Weighted approach
Salido et al. (2017)	Job shop	None	Match-up technique	Memetic algorithm	Weighted approach
Sreekara Reddy et al. (2018)	Flexible job shop	None	Event-driven	Teacher learning-based algorithm	Pareto optimisation
Valledor et al. (2018)	Flow shop	None	Period-driven	Knee point method	Pareto optimisation
Mourtzis and Vlachou (2018)	Job shop	Cloud-based CPS	Depending on the state of machines	Adaptive scheduling algorithm	Weighted approach
Zhang et al. (2018)	Unknown	IoT	Event-driven	Particle swarm optimisation	Pareto optimisation
W. Wang et al. (2018)	Hybrid flow shop	IoT	Event-driven	NSGA-II	Pareto optimisation
Ozturk et al. (2019)	Flexible job shop	None	Priority rules	Evolutionary method	Weighted approach
Z. Wang et al. (2019)	Job shop	None	Event-driven	Particle swarm optimisation	Weighted approach
Zhang et al. (2019)	Flexible job shop	RFID	Event-driven	Mixed quantum algorithm	Weighted approach
Fang et al. (2019)	Job shop	Digital twin	Event-driven	NSGA-II	Pareto optimisation
Turker et al. (2019)	Job shop	IoT	Depending on the state of a system	Dispatching rules	unknown
Tian et al. (2019b)	Flexible job shop	IoT	Period-event-driven	Dynamic game	Nash equilibrium
Shi et al. (2019)	hybrid flow shop	IoT	Event-driven	Indicators-genetic algorithm	Weighted approach
Feng et al. (2020)	Flexible job shop	IoT, Edge computing	Period-event-driven	Improved GDA	Pareto optimisation
W. Wang et al. (2020)	Unknown	CPS, Digital twin	Event-driven	NSGA-II	Pareto optimisation
Li and Wen (2020)	Job shop	None	Event-driven	Particle swarm optimisation	Pareto optimisation
This study	Distributed and flexible job shop	IoT, Edge computing	Real-time scheduling	Evolutionary game	Evolutionary game equilibrium

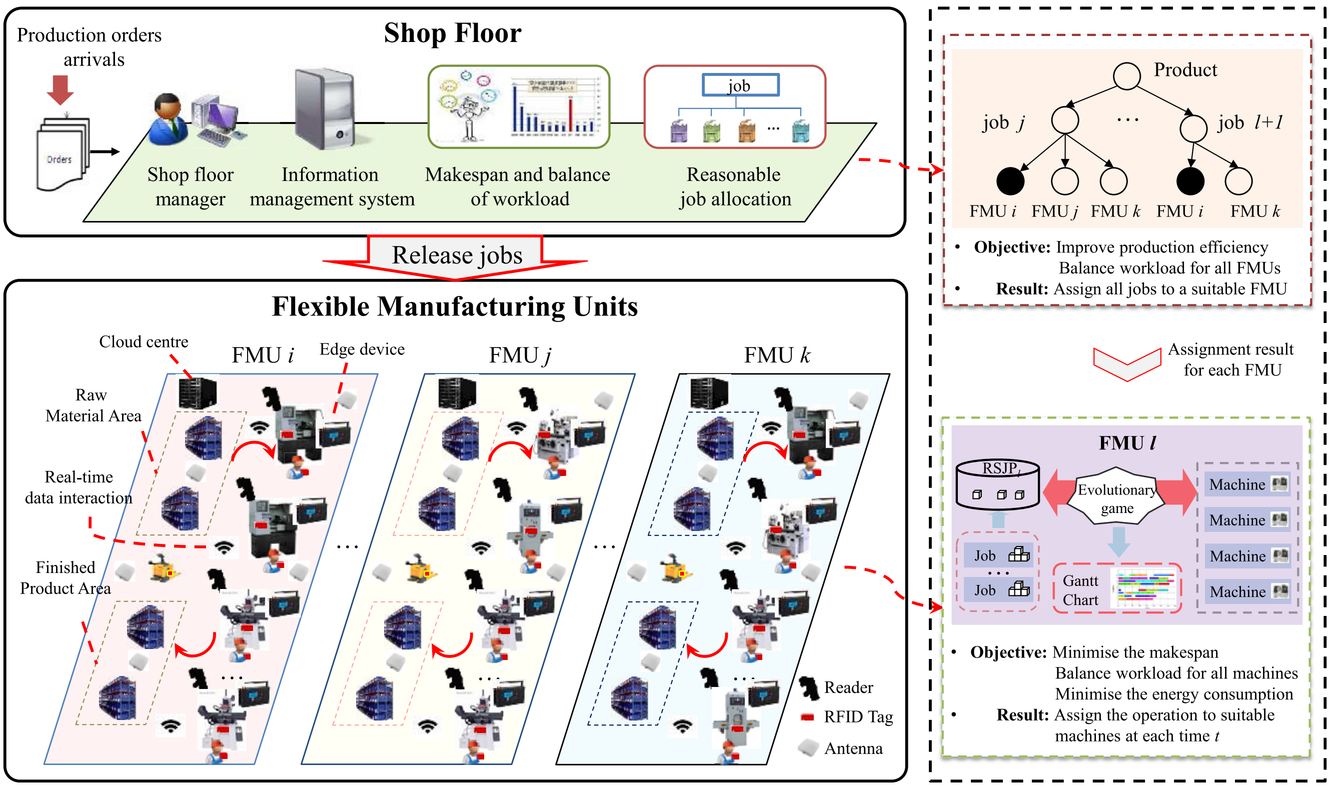

Production orders arrivals
Shop Floor
Orders
Shop floor manager
Information management system
Makespan and balance of workload
job
Reasonable job allocation
Release jobs
Flexible Manufacturing Units
Cloud centre
FMU i
Edge device
FMU j
FMU k
Raw Material Area
Real-time data interaction
Finished Product Area
Reader
RFID Tag
Antenna
Product
job j
job l+1
FMU i
FMU j
FMU k
FMU i
FMU k
Objective: Improve production efficiency
Balance workload for all FMUs
Result: Assign all jobs to a suitable FMU
Assignment result for each FMU
FMU l
RSJP
Evolutionary game
Machine
Job
Gantt Chart
Objective: Minimise the makespan
Balance workload for all machines
Minimise the energy consumption
Result: Assign the operation to suitable machines at each time t

Fig. 1. The overall architecture of the EC-IIoT based DFJS-RS.

objectives of the shop floor layer and FMUs layer, respectively.

4.1. Problem statement

The DFJS-RS problem is shown as follows in this study. The workshop has n jobs and p FMUs. The job i contains n_i operations and the lth FMU has m_l machines. An operation of each job can only be processed on one machine in one FMU. The primary purpose of the DFJS-RS is to assign each job to the suitable FMU and determine the optimal machine for each operation based on real-time manufacturing information so that the objectives of the shop floor layer and FMUs layer can be satisfied. Table 2 describes the Notations used in this study.

During the process of the DFJS-RS, we made the following assumptions:

(1) All FMUs can process all jobs.
(2) Transport costs between machines are ignored.
(3) At the initial moment, all machines are working properly.
(4) If a job is assigned to an FMU, all operations of this job are processed on that FMU.
(5) If abnormal events occur, the processing of the operation can be interrupted.

4.2. Formulation of DFJS-RS

4.2.1. Shop floor layer

In the shop floor layer, all FMUs negotiate to work simultaneously and output job assignment results for each FMU at the beginning of RS. The objective of the shop floor layer is to minimise the maximum completion time for jobs completed on the FMU as short as possible. At the same time, the workload balance for each FMU should be taken into account.

Objectives:

(1) Minimising the maximum completion time of all assigned jobs (makespan):

$$Minf_1^s = makespan = \max C_i \tag{1}$$

During the job assignment, each FMU can only obtain one job at a time. Therefore, in each job assignment, as long as the maximum completion time of the assigned job on all FMUs is minimised, the requirements of objective 1 of the shop floor layer can be satisfied.

(2) Minimising the workload balance index (WBI):

$$Minf_2^s = WBI = \sum_{L=1}^{P} |W_L - AL| \tag{2}$$

In Equation (2), since W_L represents the total workload of FMU l and AL represents the average workload of all FMU, $|W_L\text{-}AL|$ represents the deviation between the total workload of FMU l and the average workload. Thus, the workload balance index is defined as the sum of the absolute values of the deviation between the total workload of each FMU and the average workload, denoted as WBI, which should be minimised when assigning jobs.

Subject to:

$$W_L = \sum_{k=1}^{m_l} \sum_{i=1}^{n} \sum_{j=1}^{n_i} \left[x_{ij}^{lk} \cdot \left(t_{ijlk}^{c} + t_{ijlk}^{s} + t_{ijlk}^{t} \right) \right] \tag{3}$$

It can be seen from Equation (3) that the total workload of FMU l is the sum of the cutting time, workpiece setup time and tool changing time of all operations assigned on all machines within FMU l, denoted as W_L.

$$\sum_{k=1}^{m_l} x_{ij}^{lk} = 1 \tag{4}$$

Equation (4) indicates that if an operation O_{ij} is processed on a machine M^{lk}, then $x_{ij}^{lk} = 1$; otherwise $x_{ij}^{lk} = 0$.

$$AL = \frac{\sum_{L=1}^{p} W_L}{p} \tag{5}$$

Table 2
Notations.

Notations	Description
$J = \{j_1, j_2, \cdots, j_n\}$	Set of jobs
$j_i = \{O_{i1}, O_{i2}, \cdots, O_{in_i}\}$	Operation set of job i
$F = \{F_1, F_2, \cdots, F_i, \cdots, F_p\}$	Set of FMUs
$M^l = \{M^{l1}, M^{l2}, \ldots, M^{lm_l}\}$	Set of machines of FMU l
C_i	Completion time of j_i, where j_i is the job that has been assigned to FMUs
AL	The average workload of all FMU
W_L	The total workload of FMU l
WBI	Workload balance index
x_{ij}^{lk}	1, if M^{lk} is used for the O_{ij}; 0, otherwise
C_{ijlk}	Completion time of $O_{i,j}$ on $_{Mlk,}$ where $O_{i,j}$ is the operation that has been assigned to the machine on the FMU l
W_M^L	Critical machine workload on FMU l, which is the machine with the most workload
W_{lk}	Workload of M^{lk}
P_{idle}^{lk}	Idle power of M^{lk} kW
t_{idle}^{lk}	Idle time of M^{lk}
$P_{cutting}^{lk}$	Cutting power of M^{lk} kW
t_{ijlk}^{c}	Cutting time of O_{ij} operated on M^{lk}
$P_{changing}^{lk}$	Tool changing power of M^{lk} kW
t_{ijlk}^{t}	Tool changing time of O_{ij} operated on M^{lk}
t_{ijlk}^{s}	Workpiece setup time of O_{ij} operated on M^{lk}
E^l	Production energy consumption of FMU l
C_{ij}	Completion time of O_{ij}

It can be seen from Equation (5) that the average workload of all FMUs is the average of the sum of the total workload of each FMU.

Equation (1) guarantees the minimisation of the maximum completion time of all assigned jobs. Equation (2) ensures the balance of workload for each FMU. Equation (3) defines the total workload of FMU *l*. Equation (4) is the resource constraint, which means that an operation can only be allocated to one machine of one FMU. Equation (5) expresses the average workload of all FMUs.

4.2.2. FMUs layer

In the FMUs layer, each FMU makes a schedule by FMU itself. The real-time manufacturing information and the job assignment results from the shop floor layer are obtained as decision input. To improve production efficiency, minimising makespan^l and the critical machine workload are taken into account. Besides, to achieve green manufacturing, production energy efficiency is also considered. All objectives are related to a single FMU *l*.

Objectives:

(1) Minimising the maximum completion time of all assigned operations on the FMU *l* (makespan^l):

$$Minf_1^{f,l} = makespan^l = \max C_{ijlk} i\in[1,n], j\in[1,n_i], lk\in[l_1, lm_l] \tag{6}$$

During the operation assignment, each machine can only get one operation at a time. Therefore, in each operation assignment, as long as the maximum completion time of the assigned operation on all machines is minimised, the requirements of objective 1 of the FMUs layer can be satisfied.

(2) Minimising the critical machine workload of FMU *l* (W_M^L), which is the machine with the most workload:

$$Minf_2^{f,l} = W_M^L = \max\{W_{lk}\} lk\in[l_1, lm_l] \tag{7}$$

In Equation (7), since W_{lk} represents the workload of M^{lk}, the maximum value of W_{lk} ($k\in[1,m_l]$) on the FMU *l* is considered as the critical machine workload, denoted as W_M^L.

(3) Minimising the production energy consumption of FMU *l*, which can be divided into four types: cutting energy consumption, idle energy consumption, tool changing energy consumption and workpiece setup consumption, is defined as:

$$Minf_3^{f,l} = E^l = \sum_{k=1}^{m_l}\left(t_{idle}^{lk}\cdot P_{idle}^{lk}\right) + \sum_{i=1}^{n}\sum_{j=1}^{n_i}\sum_{k=1}^{m}\Big[\times\left(t_{ijlk}^{c}\cdot P_{cutting}^{lk} + t_{ijlk}^{t}\cdot P_{changing}^{lk} + t_{ijlk}^{s}\cdot P_{idle}^{lk}\right)\cdot x_{ij}^{lk}\Big] \tag{8}$$

where the first part of Equation (8) is the idle energy consumption and the second part of Equation (8) is the process energy consumption, such as cutting energy consumption, tool changing energy consumption and workpiece setup consumption.

Subject to:

$$W_{lk} = \sum_{i=1}^{n}\sum_{j=1}^{n_i}\left[x_{ij}^{lk}\cdot\left(t_{ijlk}^{c} + t_{ijlk}^{s} + t_{ijlk}^{t}\right)\right] \tag{9}$$

It can be seen from Equation (9) that the workload of M^{lk} is defined as the sum of the cutting time, workpiece setup time and tool changing time of all operations assigned on M^{lk}, denoted as W_{lk}.

$$C_{ij} - C_{i,j-1} \geq \left(t_{ijlk}^{c} + t_{ijlk}^{s} + t_{ijlk}^{t}\right)\cdot x_{ij}^{lk} \tag{10}$$

Equation (10) indicates that when an operation $O_{i,j-1}$ begins to be processed, the next operation $O_{i,j}$ cannot be processed before the completion of the operation $O_{i,j-1}$.

Equation (6) ensures that the makespan^l is minimised. Equation (7) denotes that the critical machine workload of FMU *l* is minimised. Equation (8) represents the minimisation of the production energy consumption of FMU *l*. Equation (9) gives the workload of M^{lk}. Inequity (10) guarantees the constraints of operation precedence.

5. Evolutionary game based solve method for DFJS-RS

In this section, an evolutionary game based solve method is designed and developed to improve workshop productivity and energy efficiency. Employing an evolutionary game, all tasks can be assigned to the appropriate machine in real-time.

5.1. Evolutionary game model for DFJS-RS problem

Since the evolutionary game can transform a multi-objective optimisation problem into solving the game equilibrium problem and use the dynamic game evolution process to obtain the optimal solution, in this study, we model the DFJS-RS problem by employing an evolutionary game. For the evolutionary game, the Nash equilibrium is considered as the solution, which ensures satisfactory returns for all players.

The evolutionary game can be formulated by $G = \{P, S, U, \zeta, \tau\}$, where, P represents the set of players, S is the strategy space of players, U is the payoff of players, ζ is the interference operator, τ is the maximum number of evolutionary iterations. The detailed introductions are as follows.

- Players P: in this study, the DFJS-RS problem needs to assign jobs to the suitable FMUs, and the RS within each FMU needs to be determined. Thus, the evolutionary game is used in both the shop floor layer and the FMUs layer. Each FMU is considered as a player at the shop floor layer, and the corresponding machines in each FMU are considered as players at the FMUs layer.
- Strategy S: in the shop floor layer, the strategy of each player corresponds to the unallocated jobs in the JPW. In the FMUs layer, the strategy of each player corresponds to the first unallocated operations of all jobs of corresponding FMU.
- Payoff U: to optimise the objectives at the two layers, the reciprocal of the objective function in the corresponding layer is the payoff of the player.
- Interference operator ζ: it imposes stochastic interference to the "stable" state, and the interference probability is set to p_d so that the original "stable" state is broken.
- The maximum number of evolutionary iterations τ: after several rounds of the game, the initial strategy combination reached a "stable" state, which is called the generation of evolution game. In this study, $\tau = T$ and the maximum number of evolutionary iterations is *T*.

5.2. Evolutionary game based DFJS-RS method

In this section, an evolutionary game based DFJS-RS method is introduced in detail, which includes two layers: shop floor layer and FMUs layer. The shop floor layer is used to assign each job to a suitable FMU considering the makespan and balance of workload.

The FMUs layer is used to solve a real-time FJS scheduling problem for each FMU, with the objectives of makespan, critical machine workload and production energy consumption. The evolutionary game based DFJS-RS method aims to optimise all jobs that need to be processed in the workshop based on the real-time status information of manufacturing resources. The detailed instructions for these two layers are shown as follows.

5.2.1. Shop floor layer

The shop floor layer can produce a job assignment result for each FMU at the beginning of RS in the static shop floor environment and it also assigns rush order to the suitable FMU during the RS stage. Fig. 2 describes the procedure of the shop floor layer and the specific implementation consists of seven steps.

Step 1: The two optimisation objectives are assigned to all FMUs of the workshop in turn. For example, f_1^s is assigned to the FMU 1 and f_2^s is assigned to the FMU 2. If an FMU 3 exists, f_1^s is assigned to the FMU 3 again. Each FMU is a player in the evolutionary game. The reciprocal of the objective function assigned to each player is the corresponding player's payoff.

Step 2: Pick out the unallocated jobs from all jobs and put these jobs into a JPW. These jobs in the JPW called the strategies of evolutionary game. Thus, all FMUs can request processing jobs in the JPW, and one job can be assigned to one FMU at a time. The assigned job is added to the set of the machined job (SMJ) of the corresponding FMU. Besides, during the RS stage, the unallocated jobs from rush orders can also be placed into the JPW so that all rush orders are assigned to the suitable FMUs.

Step 3: If an F_l requests processing J_i, the allocation process within F_l is triggered, which uses the following methods.

(1) Based on the previous job assignment result of F_l, the operations in the SMJ of F_l (SMJ$_l$) can be known. Thus, put the operations of J_i and the operations in the SMJ$_l$ into a job pool l (JP$_l$).
(2) To assign these operations in the JP$_l$ to the corresponding machine, we use a machine assignment rule (MAR) and a dispatching rule. The MAR allocates each operation to the machine with the minimum processing time. The dispatching rule uses the Shortest Processing Time (SPT). The processing time is equal to the sum of tool changing time, cutting time and workpiece setup time.

Therefore, the schedule within each FMU can be obtained by

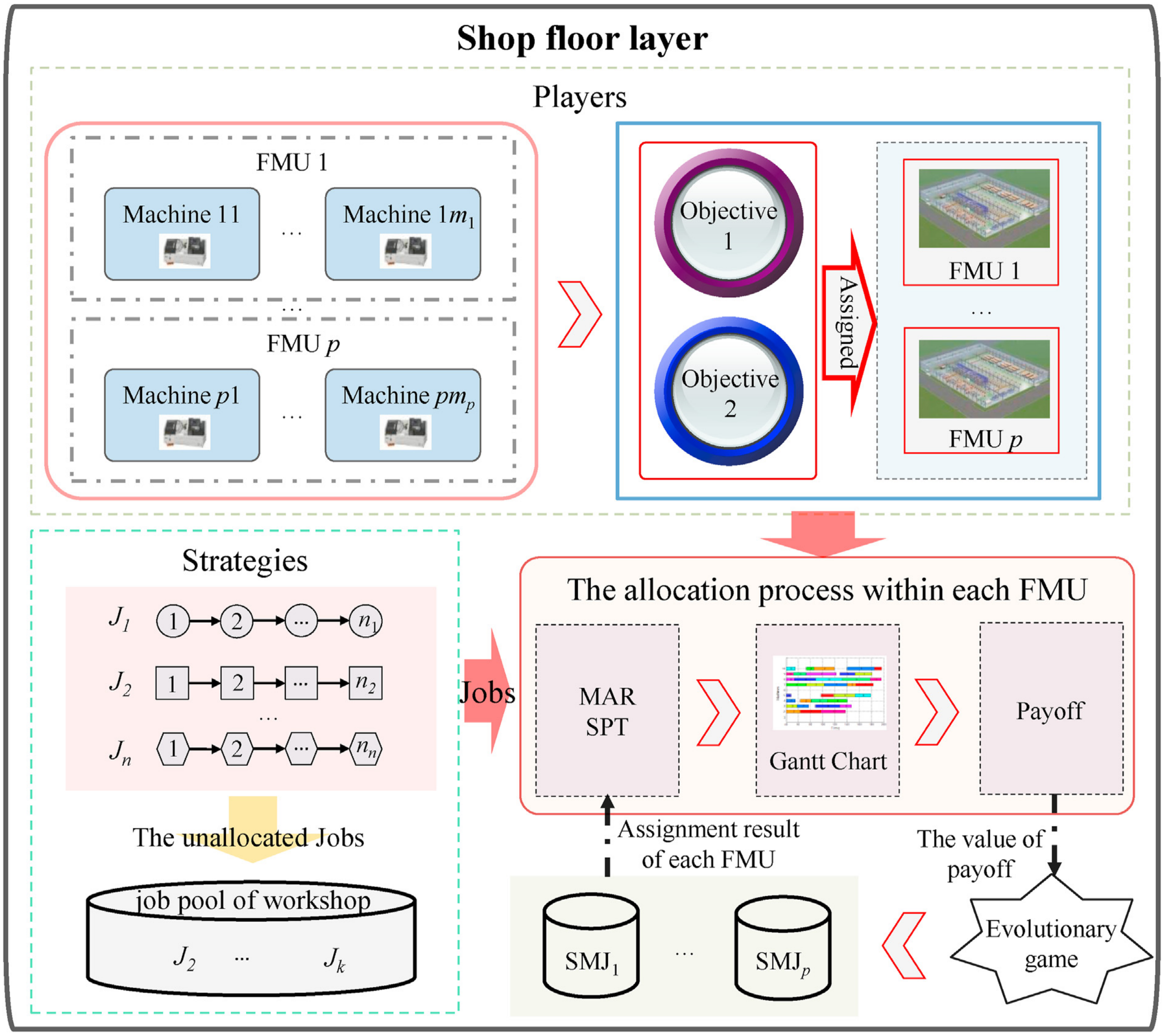


Fig. 2. The procedure of shop floor layer.

using the above method.

Step 4: Based on Step 3 and Eqs. (1) and (2), the payoff of each player is calculated under various feasible strategy combinations. To ensure each FMU can choose one corresponding strategy, if the number of jobs in a strategy set is less than the number of FMUs, the empty set will be added to the strategy set.

Step 5: Solve the Nash equilibrium by a best-response dynamics based solver method, which is described in Section 5.3.

Step 6: According to the Nash equilibrium results, each FMU can obtain one job, which is added to the corresponding SMJ.

Step 7: Repeat the above process until all jobs are assigned to the SMJ of the corresponding FMU.

In this layer, the output is the assignment result of each FMU. The result shows which job needs to be processed in which FMU during the RS stage.

5.2.2. FMUs layer

The FMUs layer is used to select the most suitable operations for each machine according to their real-time status at each time t within a given FMU. It could reduce the complexity of the computation because each machine can only assign at most one operation at each time t. Fig. 3 shows the procedure of the FMUs layer within F_l at each time t, and the specific implementation consists of eight steps.

Step 1: Three optimisation objectives of the FMUs layer are assigned to all machines of F_l in turn. All machines of F_l correspond to players of the evolutionary game. Thus, the reciprocal of the objective function assigned to each player is the corresponding player's payoff. If each machine is unavailable at time t, this step is terminated, and RS goes to the next time t ($t = t+1$).

Step 2: The first unallocated operations of all jobs of the SMJ$_l$ are put into the RSJP$_l$. The operations in the RSJP$_l$ correspond to strategies of the evolutionary game. Thus, each machine can request to process the operations in the RSJP$_l$.

Step 3: Each player can choose the corresponding strategy in the RSJP$_l$ to form different strategy combinations. The payoff of each player is calculated under different strategy combinations, according to Eqs. (6)–(8).

Step 4: Find the Nash equilibrium according to the best-response dynamics based solver method, which is described in Section 5.3.

Step 5: Based on the result of the Nash equilibrium, some operations could be allocated to the corresponding machines. However, to guarantee optimal allocation, only one operation allocated is determined at a time, which is considered to be a true operation. Other operations are considered as false operations. The rules for distinguishing a true operation and a false operation are shown as follows in Fig. 4. The true operation is added into a temporary processing queue of the corresponding machine (TPQ-CM) and put the false operations into the RSJP$_l$.

Step 6: Repeat the above processes until all machines have a new operation added to their TPQ.

Step 7: Select the operations that can be processed at time t from the TPQ of each machine based on the real-time manufacturing information. Then, add these operations to the real-time processing queue of the corresponding machines.

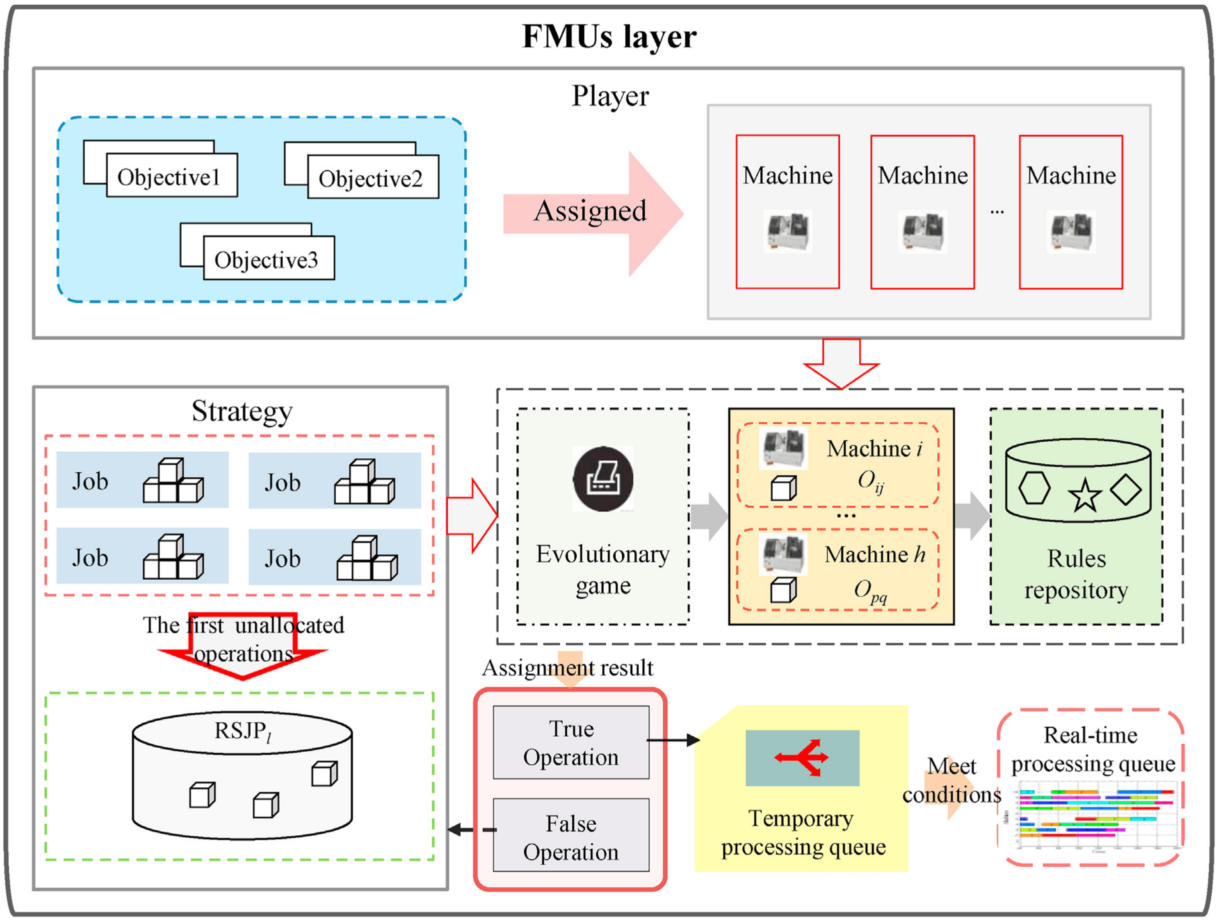
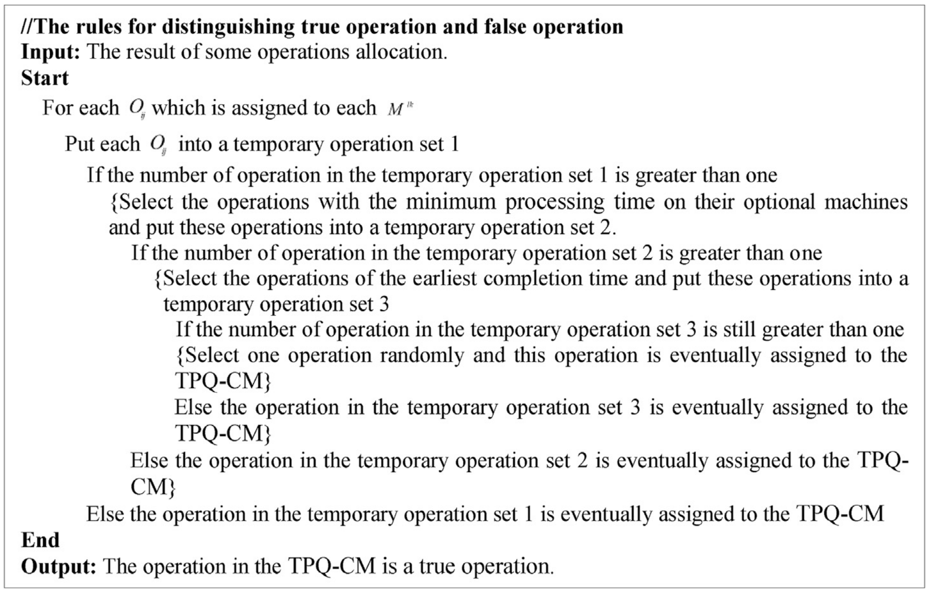


Fig. 3. The procedure of FMUs layer.



```
//The rules for distinguishing true operation and false operation
Input: The result of some operations allocation.
Start
  For each $O_{ij}$ which is assigned to each $M^{ik}$
    Put each $O_{ij}$ into a temporary operation set 1
      If the number of operation in the temporary operation set 1 is greater than one
        {Select the operations with the minimum processing time on their optional machines
        and put these operations into a temporary operation set 2.
          If the number of operation in the temporary operation set 2 is greater than one
            {Select the operations of the earliest completion time and put these operations into a
            temporary operation set 3
              If the number of operation in the temporary operation set 3 is still greater than one
              {Select one operation randomly and this operation is eventually assigned to the
              TPQ-CM}
              Else the operation in the temporary operation set 3 is eventually assigned to the
              TPQ-CM}
          Else the operation in the temporary operation set 2 is eventually assigned to the TPQ-
          CM}
      Else the operation in the temporary operation set 1 is eventually assigned to the TPQ-CM
End
Output: The operation in the TPQ-CM is a true operation.
```

Fig. 4. The rules for distinguishing a true operation and a false operation.

Step 8: At the next t ($t = t+1$), repeat the above step until all assignments are completed.

In this layer, the output is a real-time processing queue for each machine. When abnormal events occur, the influence of abnormal events can be timely reduced by corresponding methods and strategies.

5.3. Evolutionary game solution

For the evolutionary game, Nash equilibrium is a general concept of its solution. Nash Equilibrium is reached when each player cannot further improve his payoff by changing his strategy, meaning that the payoff of each player has reach maximum. By determining the five elements of the evolutionary game in Section 5.1, the DFJS-RS problem can be transformed into an evolutionary game problem. In this section, a best-response dynamics based solution algorithm is proposed to solve the Nash equilibrium of the evolutionary game.

Definition 1. For $G = [P,S,U]$, set $S_{-i} = S_1 \times S_2 \times \cdots \times S_k \times \cdots \times S_n, k \neq i$,If $BR_i(s_{-i}) = \{s_i^* \in S_i : u_i(s_i^*, s_{-i}) \geq u_i(s^{(i)}, s_{-i}), \forall s^{(i)} \in S_i\}$Then $B_i : S_{-i} \rightarrow S_i$is the best-response correspondence for player i.

Thus, the best-response dynamics are defined as follows. Under a specific strategy combination, if other players keep their strategies unchanged, a strategy that a player chooses to maximise its payoff is called the best-response of the player in this combination. Starting from a particular strategy combination$s = \{s_i, s_{-i}\}$, the dynamic process by which all players alternately choose their best-response is called best-response dynamics. If all players adopt the same strategy combination s^* every time after a finite time in the best-response dynamics, the strategy combination s^* is considered a “stable” state.

To get the Nash equilibrium of the evolutionary game, stochastic interference is needed to cause the game to deviate from the original “stable” state. Then, the new “stable” state is achieved by the sequence best-response dynamics of players. Repeat the above process so that the maximum number of evolutionary iterations reaches the set value. The final “stable” state is the Nash equilibrium of the evolutionary game.

The interference operator in the evolutionary process is defined as follows: for the strategy s_i of S, if the uniform random number rand $(0, 1)$ is less than the interference probability p_d, a strategy is randomly selected from the ith player’s strategy set to replace the current strategy s_i. Otherwise, it remains unchanged. Thus, the strategy combination after interference can be obtained through this way for all strategies of S.

From the above analysis, the best-response dynamics based solution algorithm is summarised in Fig. 5.

6. Case study

In this section, a DFJS-RS case is described and analysed to verify the effectiveness and feasibility of the presented DFJS-RS.

6.1. Case scenario

Here, an industrial case from a collaboration company in Xi’an is used. The company is a typical discrete manufacturer for engine production. With a two-week investigation at its workshops, it observed that the manufacturing information could not accurately and timely reflect the real situation. When the abnormal event occurred, it further intensified production interference. Therefore, the company is in great need of scheduling method based on real-time manufacturing information to realise real-time production optimisation and management. Since the implementation of the proposed DFJS-RS into a real-life company is a challenging and complex task, a proof of concept experiment is designed. For simplicity of discussion but without losing generality, a hypothetical case scenario is considered that represents the configuration of a real-life workshop in the company.

This case scenario is a configuration of DFJS containing several FMUs. To make this case scenario close to a real EC-IIoT-enabled DFJS, some common manufacturing resources are selected to establish the case scenario in each FMU. As shown in Fig. 6, a demo case scenario of an FMU is presented. It is mainly composed of two parts, namely a manufacturing area with some machines, each machine can sense and interact with each other; a storage area with some shelves, each shelf can store raw materials and finished products.

To realise the active sensing of real-time data in the production

```
//Algorithm for evolutionary game solution
Input: The five elements of the evolutionary game.
Start
Step 1. Each player chooses their strategies $s_i$ from the $S$ stochastically and the initial strategy set $s^{(t)}$ can be determined. Here, $t$=0.
Step 2. Through many iterations of sequential best-response dynamics can reach a "stable" state $s^{*(t)}$.
Step 3. A stochastic interference which represents noise is imposed to the state $s^{*(t)}$ and this "stable" state is broken.
Step 4. Through many iterations of sequential best-response dynamics again can reach another "stable" state $s^{*(t')}$.
Step 5. Compare the payoff function values of strategy combinations $s^{*(t)}$ and $s^{*(t')}$.
Step 6. Assign the strategy combinations with large payoff to the strategy set $s^{*(t+1)}$, $t=t+1$.
Step 7. Repeated above Step 3 to Step 6 until the maximum number of evolutionary iterations is satisfied.
END
Output: A Nash equilibrium of $G$.
```

Fig. 5. Algorithm for evolutionary game solution.

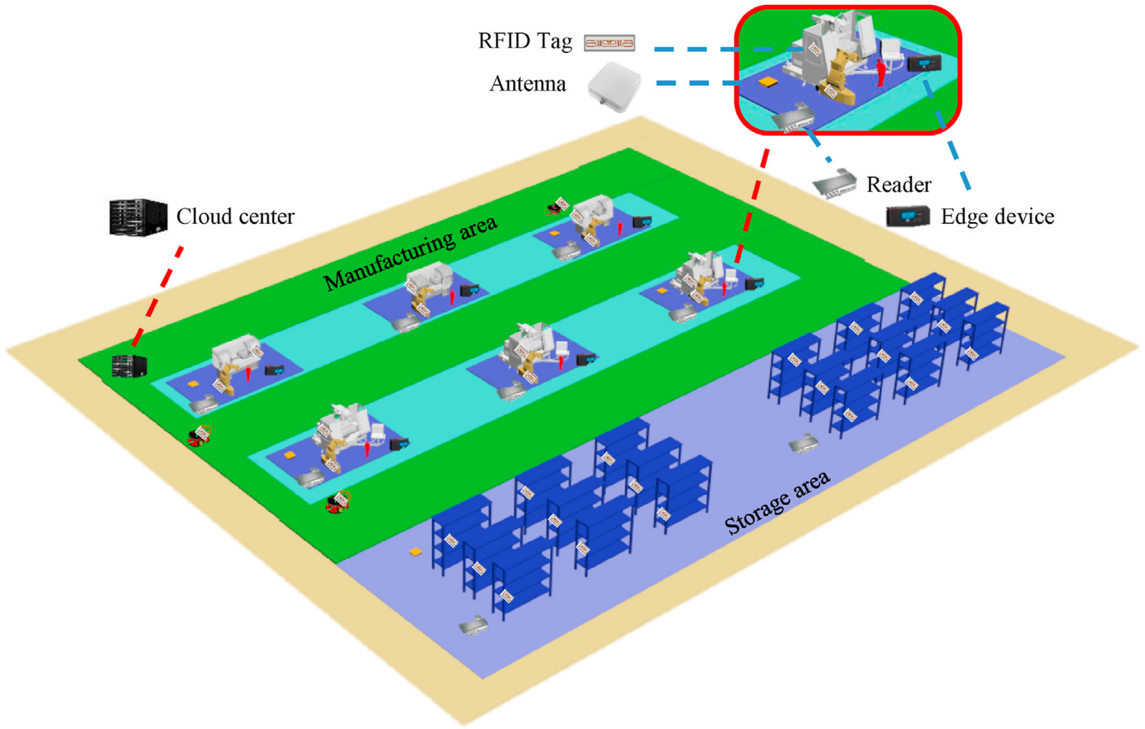


Fig. 6. The case scenario of an FMU.

process, RFID tags are pasted into some manufacturing resources. For example, RFID tags are attached to machines so that their real-time status data can be collected; RFID tags are attached to each pallet so that real-time material data can be obtained; each worker has an employee card for providing personal information. Besides, the edge device is placed next to each machine, and the device captures real-time data through the IIoT device and pre-processes these real-time data. Then, real-time manufacturing information is transmitted to the cloud centre. Finally, the cloud centre can make a detailed scheduling plan based on real-time manufacturing information.

6.2. *Simulation experiment*

Based on the above-mentioned case scenario, a simulation experiment is discussed and studied. The optimisation results are obtained by running software Matlab 7.0 on a computer with a frequency of 2.2 GHz CPU and 16 GB RAM.

The original data of the simulation experiment are given by Chan's paper (Chan et al., 2006), which is a two-FMUs DFJS scheduling problem. The form of this data is similar to the form of data in a real-life company. Both these two FMUs contain the same number and type of machines. Compared with Chan's paper, since our study focuses on the RS problem, job 11 is added to the simulation data as a rush order during the production execution stage. Besides, to lower the production energy consumption of each FMU, compared with Chan's paper, the cutting power is presented and the processing time divided into three parts: tool changing time, workpiece setup time and cutting time. The detailed data is shown in Table 3. In Table 3, the numbers (A/B/C/D) of row O_{ij} and column M^{lk} mean that if O_{ij} is processed on M^{lk}, the tool changing time, workpiece setup time, cutting time and cutting power are 'A', 'B', 'C' and 'D' respectively. Table 4 shows the idle power and tool changing power of machines, which are based on (J. Wang et al., 2020). Time is measured in hours and power in kW. In this study, the value of p_d and T are 0.3 and 100, respectively.

In this case, the flows of the proposed DFJS-RS include three steps. They are shown as follows.

At first, all jobs are assigned to the most suitable FMU at the beginning of RS. Here, FMUs consider both optimisation objectives, namely the makespan and workload balance index, to complete the assignment of jobs based on the steps described in Section 5.2.1. The result of the assignment determines the most suitable FMU for each job.

When all jobs are assigned to the suitable FMU, each FMU will conduct RS according to the real-time manufacturing information of the corresponding FMU. At each time t of RS in each FMU, the evolutionary game based RS approach (see Section 5.2.2) is used to allocate the operation to the optimal machine based on their real-time status information.

During the manufacturing execution, if an abnormal event occurs, the corresponding FMU can obtain this information timely. Then, through the method proposed in Section 5, the adverse effects brought by these abnormal events can be eliminated in time.

6.3. Performance analysis for a static environment

To demonstrate the performance of proposed DFJS-RS, this method is compared to four existing static scheduling methods, including genetic algorithm with dominant gene (GADG) presented by Chan et al. (2006), improved genetic algorithm (IGA) proposed by De Giovanni and Pezzella (2010), genetic algorithm (called GA_CL) proposed by Chang and Liu (2017) and genetic algorithm (called GA_JS) proposed by Lu et al. (2018). All these existing scheduling methods are based on the data in Chan et al. (2006) as simulation data. Thus, for comparison in this study, we also use the same simulation data given by Chan et al. (2006). The optimisation objective results of our proposed method cannot be directly compared with the results of existing scheduling methods. The corresponding reasons and handling methods are as follows.

Table 4
Idle power and tool changing power of machines.

Machines	Idle power [KW]	Tool changing power [KW]
M^{l1}	0.995	1.450
M^{l2}	1.485	1.672
M^{l3}	1.910	2.919

(1) For the makespan, since these existing scheduling methods take the maximum completion time of all jobs on all FMUs as the scheduling objective, it is different from the maximum completion time proposed in our study. Thus, we use the maximum value of $f_1^{f,l}(l = 1, 2)$ (denote as $f_{1,\max}^{f}$) to compare with the makespan of existing scheduling methods.
(2) For the critical machine workload, the existing scheduling methods do not consider to optimise. To compare with our proposed method, for the GADG proposed by Chan et al. (2006), we calculate the critical machine workload for each FMU according to the Gantt chart in their paper. The maximum value of $f_2^{f,l}(l = 1, 2)$ (denote as $f_{2,\max}^{f}$) is compared with the maximum value of critical machine workload for all FMUs in the GADG.
(3) For the production energy consumption, the existing scheduling methods still do not take into account. For the GADG proposed by Chan et al. (2006), we calculate the total production energy consumption of all FMUs based on the Gantt chart. The sum of $f_3^{f,l}(l = 1, 2)$ (denote as $f_{3,total}^{f}$) is compared with the total production energy consumption of the GADG.

Since the Gantt chart of the other three methods is not given in the corresponding literature, the values of critical machine

Table 3
The instance of DFJS-RS.

Jobs	Operations	Available machine		
		M^{l1}	M^{l2}	M^{l3}
J_1	O_{11}	0.5/2.5/4/2.3	–	0.3/0.7/3/3.4
	O_{12}	1/3/4/2.5	0.2/0.8/2/1.9	–
	O_{13}	0.3/0.7/2/2.9	–	0.8/1.2/4/3.7
	O_{14}	0.2/0.8/1/3.1	0.6/1.4/2/2.4	–
J_2	O_{21}	1.2/1.8/5/2.5	1.6/2.4/8/5.2	–
	O_{22}	–	2/4/8/2.4	0.4/0.6/3/4.2
	O_{23}	1.1/1.9/4/3.2	0.8/2.2/11/1.9	–
	O_{24}	0.8/1.2/6/3.1	–	0.3/1.7/2/3.9
J_3	O_{31}	1.1/2.9/6/3.4	0.8/2.2/12/2.7	0.6/1.4/6/4.3
	O_{32}	–	0.2/0.8/1/2.4	0.7/1.3/4/4.6
	O_{33}	0.2/0.8/1/2.1	–	0.4/1.6/2/3.4
	O_{34}	0.3/0.7/5/2.6	0.2/0.8/2/3.4	–
J_4	O_{41}	–	1.2/0.8/7/2.5	0.5/1.5/3/4.7
	O_{42}	1.7/1.3/3/3.5	–	0.5/0.5/1/4.6
	O_{43}	–	0.9/2.1/4/3.1	1.3/1.7/9/3.8
	O_{44}	1.4/2.6/5/3.8	1.1/1.9/3/2.2	0.8/1.2/1/3.2
J_5	O_{51}	1.3/2.7/6/3.1	–	2.1/2.9/10/4.3
	O_{52}	–	1.3/1.7/4/2.9	2.1/3.9/8/3.4
	O_{53}	1.3/1.7/2/2.1	1.2/1.8/5/3.2	–
	O_{54}	0.5/1.5/2/3.5	1.3/1.7/3/2.5	1.4/1.6/5/5.1
J_{11}	$O_{11,1}$	1.2/2.8/4/2.4	2.1/2.9/3/3.1	–
	$O_{11,2}$	0.1/0.9/1/3.5	–	0.8/1.2/3/4.3
J_6	O_{61}	1.4/1.6/4/2.2	–	0.9/1.1/2/3.5
	O_{62}	2.2/2.8/4/3.1	1.1/0.9/1/2.1	–
	O_{63}	0.5/0.5/2/2.9	–	1.1/1.9/3/3.6
	O_{64}	0.3/0.7/1/2.3	0.7/1.3/2/3.2	2.2/2.8/3/4.4
J_7	O_{71}	1.4/1.6/5/3.4	1.3/2.7/8/3.6	–
	O_{72}	2.3/3.7/9/3.5	–	1.3/0.7/2/3.3
	O_{73}	1.3/1.7/4/2.4	1.1/2.9/10/2.4	–
	O_{74}	1.2/2.8/4/3.7	2.2/2.8/11/2.5	0.9/1.1/2/4.9
J_8	O_{81}	2.3/2.7/7/2.5	–	1.1/2.9/4/3.5
	O_{82}	–	0.3/0.7/1/3.2	1.3/1.7/3/4.9
	O_{83}	0.2/0.8/1/2.1	1.3/1.7/7/2.4	0.8/1.2/2/3.1
	O_{84}	1.1/1.9/3/4.2	0.3/0.7/2/3.2	–
J_9	O_{91}	–	1.2/1.8/10/2.4	1.1/1.9/2/3.2
	O_{92}	1.4/1.6/3/3.9	2.2/2.8/3/3.1	0.2/0.8/1/4.2
	O_{93}	–	2.1/1.9/3/3.2	2.3/3.7/9/3.9
	O_{94}	–	1.2/1.8/3/2.2	1/1/1/3.2
J_{10}	$O_{10,1}$	1.3/1.7/7/2.5	–	3.5/5.5/9/4.6
	$O_{10,2}$	–	2.1/2.9/2/3.2	2.6/3.4/9/4.3
	$O_{10,3}$	1.4/1.6/2/2.6	1.3/1.7/5/2.4	–
	$O_{10,4}$	1.1/1.9/1/3.2	1.2/1.8/3/2.1	1.3/1.7/5/4.1
J_{11}	$O_{11,3}$	0.8/1.2/2/2.3	1.2/1.8/3/3.1	–
	$O_{11,4}$	–	0.9/2.1/4/2.4	1.4/1.6/4/3.5

Table 5
Comparison results for each method.

Objectives	GADG		IGA		GA_CL		GA_JS		DFJS-RS	
	BE.	AV.	BE.	AV.	BE.	AV.	BE.	AV.	BE.	AV.
$f^f_{1,\max}$ [hour]	42	43.1	37	38.6	37	37.6	38	38.0	38	38.8
$f^f_{2,\max}$ [hour]	37	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	36	36.5
$f^f_{3,total}$ [kWh]	557.7	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	480.2	510.2

workload and total production energy consumption cannot be obtained. Thus, the corresponding optimisation results cannot be compared. The best (BE.) and average (AV.) of results for each method after 50 runs are shown in Table 5. The values of critical machine workload and total production energy consumption calculated from the Gantt chart are the best results in the GADG.

As shown in Table 5, the best makespan in the DFJS-RS is 38 h. The maximum value and the minimum value of the best makespan in the existing scheduling method are 42 h and 37 h, respectively. Thus, the maximum improvement can be achieved to 9.5% and the minimum to −2.6%. Although the DFJS-RS method is not always superior to the existing scheduling methods in terms of the best makespan, the result of DFJS-RS has the better value of the best critical machine workload compared with the GADG. For our proposed method, the best critical machine workload is 36 h while in the GADG it is 37 h. Thus, the improvement can be as much as 2.7%. Besides, the best total production energy consumption got by the DFJS-RS is 480.2 kWh, which indicates an improvement of 13.9% compared to the GADG. Therefore, in general, the scheduling performance of DFJS-RS in this study is superior to the existing scheduling methods in a static environment.

6.4. Performance analysis considering abnormal events

To further illustrate the performance of the DFJS-RS method considering dynamic disturbance, we compare the DFJS-RS method with the EDSM, which includes the heuristic, right-shift rescheduling, periodic rescheduling and event-driven rescheduling. For the heuristics, in the machine route decision problem, SPT and random assignment (RA) are adopted, and in the operation sequencing problem, SPT and longest processing time (LPT) are adopted. RA means that each operation can be randomly allocated to a machine. For the right-shift rescheduling, the DFJS-RS method is used to generate the initial schedule. For the periodic rescheduling, NSGA-II (W. Wang et al., 2018) is used as the rescheduling method. The population size is set as 100, the maximum number of iterations is set as 600, the mutation probability is set as 0.25, and the crossover probability is randomly selected as the number between 0 and 1. The rescheduling interval is set to 5 h. For the event-driven rescheduling, NSGA-II (W. Wang et al., 2018) is still used as the rescheduling method. Because the above EDSM cannot assign jobs to suitable FMU in the DFJS, our proposed DFJS-RS method is used to assign jobs to suitable FMUs. The EDSM only conducts dynamic scheduling within the FMU.

Three scenarios for the occurrence of abnormal events are presented to compare the performances. Scenario 1: M^{12}, M^{13}, M^{21} and M^{22} breakdown at time $t_1 = 3$, $t_2 = 5$, $t_3 = 4$ and $t_4 = 6$ respectively during the manufacturing execution stage. The machine repair time is $t_5 = 5$, $t_6 = 7$, $t_7 = 6$ and $t_8 = 8$, respectively. Scenario 2: job 11 is added to the FMU 1 as a rush order at time $t_9 = 12$. Scenario 3: the abnormal events in scenario 1 and scenario 2 are considered simultaneously in scenario 3. The simulation results of scenario 1, scenario 2 and scenario 3 are given in Table 6, Table 7 and Table 8, respectively.

It can be known from Table 6, that the DFJS-RS method is entirely superior to the EDSM. For example, the value of $f^f_{1,\max}$obtained by the DFJS-RS method is 40 h. Compared with the EDSM, the maximum and minimum improvements are 31.0% and 2.4% respectively. The maximum value of $f^f_{2,\max}$ got by the EDSM is 42 h and the minimum value of $f^f_{2,\max}$ is 38 h. Therefore, the maximum improvement can be achieved to 11.9% and the minimum to 2.6% in the DFJS-RS method. Compared with the EDSM, the DFJS-RS method can still achieve maximum improvement of 24.3% and a minimum improvement of 0.7% on the value of $f^f_{3,total}$.

The data in Table 7 further illustrate that the DFJS-RS method provides better solutions than the EDSM. From Table 7, the value of $f^f_{1,\max}$ obtained by the DFJS-RS method is 45 h, and the maximum and minimum values obtained by the EDSM are 64 h and 47 h, respectively. Compared with the EDSM, the maximum improvement can be achieved to 29.7% and the minimum to 4.3%. For $f^f_{2,\max}$, the value obtained by the DFJS-RS method is 44 h, with a maximum improvement of 12.0% and a minimum improvement of 2.2% compared with the EDSM. The DFJS-RS method improves$f^f_{3,total}$by 26.0% and 1.2%, respectively, for its maximum and minimum values than the existing one.

It can be known from Table 8, that the results got by the DFJS-RS method are still better than the EDSM. The values of $f^f_{1,\max}$,$f^f_{2,\max}$ and $f^f_{3,total}$ obtained by the DFJS-RS method are 46 h, 44 h and 550.1 kWh respectively. Compared with the EDSM, the maximum improvement of$f^f_{1,\max}$,$f^f_{2,\max}$ and $f^f_{3,total}$ can be calculated as 29.2%, 15.4% and 22.6% respectively, and the minimum improvement is 2.1%, 4.3% and 1.7% respectively. It can be found from the simulation results that the RS performance can be significantly improved by using the proposed EC-IIoT based DFJS-RS method.

To illustrate the efficiency of the proposed method, Table 9 lists the mean CPU time of all methods at each time t or rescheduling point. Although the mean CPU time of the SPT + SPT method is shorter, it can be seen from Table 6, Tables 7 and 8 that the proposed method can achieve a better scheduling solution. Meanwhile, compared with other existing methods, the proposed method has better time efficiency. Therefore, the proposed method is in overall more efficient than the existing methods.

The differences between the EC-IIoT based DFJS-RS method compared to EDSM mainly lie in the following two aspects. Firstly, by applying EC and IIoT technology to RS, real-time status information of the manufacturing resources such as machine status can be sensed and captured. Jobs can be assigned to a suitable machine according to their real-time status. For the EDSM, the assignment of jobs is only based on the original status of manufacturing resources, and the real-time status of manufacturing resources is not considered. Thus, in the process of manufacturing execution, the EDSM often appears frequent interruption, etc. occur. Secondly, for EC-IIoT enabled DFJS-RS, operations are allocated to the suitable

Table 6
Results of the comparison between the DFJS-RS method and EDSM (Scenario 1).

Scheduling methods	$f^f_{1,\max}$ [hour]	$f^f_{2,\max}$ [hour]	$f^f_{3,total}$ [kWh]
SPT + SPT	44	41	522.6
SPT + LPT	44	41	568.8
RA + SPT	58	40	664.1
RA + LPT	57	40	684.8
Game + Right-shift rescheduling	41	38	526.5
NSGAII + Periodic rescheduling	48	42	557.9
NSGAII + Event-driven rescheduling	43	40	537.4
DFJS-RS	40	37	518.7

Table 7
Results of the comparison between the DFJS-RS method and EDSM (Scenario 2).

Scheduling methods	$f^f_{1,\max}$ [hour]	$f^f_{2,\max}$ [hour]	$f^f_{3,total}$ [kWh]
SPT + SPT	47	47	570.7
SPT + LPT	50	47	592.4
RA + SPT	64	49	734.9
RA + LPT	63	49	724.2
Game + Right-shift rescheduling	–	–	–
NSGAII + Periodic rescheduling	51	50	638.9
NSGAII + Event-driven rescheduling	47	45	550.6
DFJS-RS	45	44	543.8

Table 8
Results of the comparison between the DFJS-RS method and EDSM (Scenario 3).

Scheduling methods	$f^f_{1,\max}$ [hour]	$f^f_{2,\max}$ [hour]	$f^f_{3,total}$ [kWh]
SPT + SPT	47	47	559.8
SPT + LPT	50	47	588.4
RA + SPT	65	49	702.7
RA + LPT	59	49	699.4
Game + Right-shift rescheduling	–	–	–
NSGAII + Periodic rescheduling	55	52	710.4
NSGAII + Event-driven rescheduling	48	46	568.2
DFJS-RS	46	44	550.1

Table 9
CPU time comparisons of all scheduling methods.

Scheduling methods	Mean CPU time (s)
SPT + SPT	4.312
SPT + LPT	7.621
RA + SPT	8.634
RA + LPT	15.413
NSGAII + Periodic rescheduling	40.134
NSGAII + Event-driven rescheduling	42.631
Proposed method	5.312

machines at every time, which is based on the real-time manufacturing information. Thus, the impact of abnormal events can be eliminated in time. For the EDSM, such as periodic rescheduling, if the rescheduling period is too long, the manufacturing system is not able to respond to abnormal events in time. Even if the event-driven rescheduling can respond to abnormal events in time, frequent rescheduling also affects the stability of the dynamic scheduling system.

7. Conclusions and future work

In today's environmentally friendly society, reducing energy consumption is one of the important social responsibility of manufacturing enterprises. Thus, more and more enterprises regard energy conservation and emission reduction as one of its production objectives. However, the integration of advanced information technology and CP is still in its infancy. This study provides a theoretical basis for the application of EC-IIoT in CP. These theories can help enterprises use the newest information technology to promote the realisation of energy conservation and emission reduction. At the same time, these theories also fill the gap of applying EC-IIoT-driven RS technologies to realise sustainable production. Besides, energy-efficient DFJS-RS provides detailed insights into the implementation of EC-IIoT. Integrating EC-IIoT with existing production management methods can further improve product quality, productivity, worker health and safety, and customer satisfaction, which is beneficial to ethical, sustainable societal development strategies that aim to improve environmental, economic, and social equity. Therefore, both theoretically and practically, the research in this study can improve the integration of EC-IIoT and RS, and further increase production efficiency and achieve CP.

This study has three main contributions. Firstly, an overall architecture of EC-IIoT based DFJS-RS can provide theoretical and practical insights for the academic and business communities to realise sustainable production. Secondly, the design of the shop floor layer and FMUs layer can be used as a technology for energy efficiency optimisation considering CP. Thirdly, an evolutionary game optimisation method for DFJS-RS is proposed to improve energy efficiency and promote sustainable production.

In this study, EC is introduced into the RS, and the real-time monitoring of machine state is realised with the EC-IIoT manufacturing environment established. One core benefit of the proposed method is establishing the architecture of the EC-IIoT based DFJS-RS, which enables the big data to be collected and treated timely and effectively. The machine status can be accessed easily in real-time from adjacent wireless sensors, and the relevant scheduling can be obtained synchronously. With the proposed architecture, the data filtering and calculation can be conducted locally, so it can dramatically reduce the amount of data sent to the cloud, thus reduce the cloud computing load and energy consumption of data processing. Therefore, compared with the existing IoT-based dynamic scheduling framework, this study utilises the edge cloud synergy to deal with data explosion more effectively, shorten the device response time, and save time, energy and cost, even with increased data.

The shortage of case analysis is one of the main limitations of this study. Since the real-world production process contains lots of manufacturing resources and production jobs, more case needs to be done, and more data need to be analysed to test the methods and to improve the flexibility and efficiency of the DFJS-RS system.

Future research should combine more advanced information technology with production management methods to achieve CP. Besides, this study only uses real-time manufacturing information to drive DFJS-RS. However, how to realise RS based on prediction information is a problem that should be future discussed.

CRediT authorship contribution statement

Jin Wang: Writing - original draft, Methodology, Software. **Yang Liu:** Supervision, Validation, Writing - review & editing. **Shan Ren:** Supervision, Validation, Writing - review & editing. **Chuang Wang:** Writing - review & editing. **Wenbo Wang:** Validation, Software.

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.

Acknowledgements

The authors would like to thank the financial supports of the National Natural Science Foundation of China (Grant No. 52005408), the Scientific Research Program Funded by Shaanxi Provincial Education Department (Program No.20JK0920).

References

Biel, K., Zhao, F., Sutherland, J.W., Glock, C.H., 2018. Flow shop scheduling with grid-integrated onsite wind power using stochastic MILP. Int. J. Prod. Res. 56, 2076–2098. https://doi.org/10.1080/00207543.2017.1351638.

Chan, F.T.S., Chung, S.H., Chan, P.L.Y., 2006. Application of genetic algorithms with dominant genes in a distributed scheduling problem in flexible manufacturing systems. Int. J. Prod. Res. 44, 523–543. https://doi.org/10.1080/00207540500319229.

Chang, H.C., Liu, T.K., 2017. Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms. J. Intell. Manuf. 28, 1973–1986. https://doi.org/10.1007/s10845-015-1084-y.

Church, L.K., Uzsoy, R., 1992. Analysis of periodic and event-driven rescheduling policies in dynamic shops. Int. J. Comput. Integrated Manuf. 5, 153–163. https://doi.org/10.1080/09511929208944524.

Dai, M., Tang, D., Giret, A., Salido, M.A., 2019a. Multi-objective optimisation for energy-efficient flexible job shop scheduling problem with transportation constraints. Robot. Comput. Integrated Manuf. 59, 143–157. https://doi.org/10.1016/j.rcim.2019.04.006.

Dai, M., Zhang, Z., Giret, A., Salido, M.A., 2019b. An enhanced estimation of distribution algorithm for energy-efficient job-shop scheduling problems with transportation constraints. Sustain 11. https://doi.org/10.3390/su11113085.

De Giovanni, L., Pezzella, F., 2010. An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. Eur. J. Oper. Res. 200, 395–408. https://doi.org/10.1016/j.ejor.2009.01.008.

Ding, X., Wu, J., 2019. Study on energy consumption optimisation scheduling for internet of things. IEEE Access 7, 70574–70583. https://doi.org/10.1109/ACCESS.2019.2919769.

Ding, Y., Yang, H., 2013. Promoting energy-saving and environmentally friendly generation dispatching model in China: phase development and case studies. Energy Pol. 57, 109–118. https://doi.org/10.1016/j.enpol.2012.12.018.

Energy Information Administration, U., 2019. International Energy Outlook 2019.

Fang, Y., Peng, C., Lou, P., Zhou, Z., Hu, J., Yan, J., 2019. Digital-twin-based job shop scheduling toward smart manufacturing. IEEE Trans. Ind. Informat. 15, 6425–6435. https://doi.org/10.1109/TII.2019.2938572.

Feng, Y., Hong, Z., Li, Z., Zheng, H., Tan, J., 2020. Integrated intelligent green scheduling of sustainable flexible workshop with edge computing considering uncertain machine state. J. Clean. Prod. 246 https://doi.org/10.1016/j.jclepro.2019.119070.

Gao, K.Z., Suganthan, P.N., Tasgetiren, M.F., Pan, Q.K., Sun, Q.Q., 2015. Effective ensembles of heuristics for scheduling flexible job shop problem with new job insertion. Comput. Ind. Eng. 90, 107–117. https://doi.org/10.1016/j.cie.2015.09.005.

Gao, Y., Wang, Q., Feng, Y., Zheng, H., Zheng, B., Tan, J., 2018. An energy-saving optimisation method of dynamic scheduling for disassembly line. Energies 11. https://doi.org/10.3390/en11051261.

Giglio, D., Paolucci, M., Roshani, A., 2017. Integrated lot sizing and energy-efficient job shop scheduling problem in manufacturing/remanufacturing systems. J. Clean. Prod. 148, 624–641. https://doi.org/10.1016/j.jclepro.2017.01.166.

Gong, X., De Pessemier, T., Martens, L., Joseph, W., 2019. Energy- and labor-aware flexible job shop scheduling under dynamic electricity pricing: a many-objective optimisation investigation. J. Clean. Prod. 209, 1078–1094. https://doi.org/10.1016/j.jclepro.2018.10.289.

Holloway, C.A., Nelson, R.T., 1974. Job shop scheduling with due dates and variable processing times. Manag. Sci. 20, 1264–1275. https://doi.org/10.1287/mnsc.20.9.1264.

Hosseinabadi, A.A.R., Siar, H., Shamshirband, S., Shojafar, M., Nasir, M.H.N.M., 2015. Using the gravitational emulation local search algorithm to solve the multi-objective flexible dynamic job shop scheduling problem in Small and Medium Enterprises. Ann. Oper. Res. 229, 451–474. https://doi.org/10.1007/s10479-014-1770-8.

Huang, R.H., Yu, T.H., 2017. An effective ant colony optimisation algorithm for multi-objective job-shop scheduling with equal-size lot-splitting. Appl. Soft Comput. J. 57, 642–656. https://doi.org/10.1016/j.asoc.2017.04.062.

Jiang, T., Zhang, C., Zhu, H., Gu, J., Deng, G., 2018. Energy-efficient scheduling for a job shop using an improved whale optimisation algorithm. Mathematics 6. https://doi.org/10.3390/math6110220.

Kim, S., Meng, C., Son, Y.J., 2017. Simulation-based machine shop operations scheduling system for energy cost reduction. Simulat. Model. Pract. Theor. 77, 68–83. https://doi.org/10.1016/j.simpat.2017.05.007.

Kundakci, N., Kulak, O., 2016. Hybrid genetic algorithms for minimising makespan in dynamic job shop scheduling problem. Comput. Ind. Eng. 96, 31–51. https://doi.org/10.1016/j.cie.2016.03.011.

Li, J., Zhang, Y., Qian, C., Ma, S., Zhang, G., 2020. Research on recommendation and interaction strategies based on resource similarity in the manufacturing ecosystem. Adv. Eng. Inf. 46 https://doi.org/10.1016/j.aei.2020.101183.

Li, J.X., Wen, X.N., 2020. Construction and simulation of multi-objective rescheduling model based on pso. Int. J. Simulat. Model. 19, 323–333. https://doi.org/10.2507/IJSIMM19-2-CO8.

Liu, C.G., Yang, J., Lian, J., Li, W.J., Evans, S., Yin, Y., 2014. Sustainable performance oriented operational decision-making of single machine systems with deterministic product arrival time. J. Clean. Prod. 85, 318–330. https://doi.org/10.1016/j.jclepro.2014.07.025.

Liu, G.S., Zhou, Y., Yang, H.D., 2017. Minimising energy consumption and tardiness penalty for fuzzy flow shop scheduling with state-dependent setup time. J. Clean. Prod. 147, 470–484. https://doi.org/10.1016/j.jclepro.2016.12.044.

Liu, Y., Dong, H., Lohse, N., Petrovic, S., Gindy, N., 2014. An investigation into minimising total energy consumption and total weighted tardiness in job shops. J. Clean. Prod. 65, 87–96. https://doi.org/10.1016/j.jclepro.2013.07.060.

Liu, Y., Zhang, Y., Ren, S., Yang, M., Wang, Y., Huisingh, D., 2020. How can smart technologies contribute to sustainable product lifecycle management? J. Clean. Prod. 249 https://doi.org/10.1016/j.jclepro.2019.119423.

Lou, P., Liu, Q., Zhou, Z., Wang, H., Sun, S.X., 2012. Multi-agent-based proactive-reactive scheduling for a job shop. Int. J. Adv. Manuf. Technol. 59, 311–324. https://doi.org/10.1007/s00170-011-3482-4.

Lu, P.-H., Wu, M.-C., Tan, H., Peng, Y.-H., Chen, C.-F., 2018. A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems. J. Intell. Manuf. 29, 19–34. https://doi.org/10.1007/s10845-015-1083-z.

Masmoudi, O., Delorme, X., Gianessi, P., 2019. Job-shop scheduling problem with energy consideration. Int. J. Prod. Econ. 216, 12–22. https://doi.org/10.1016/j.ijpe.2019.03.021.

May, G., Stahl, B., Taisch, M., Prabhu, V., 2015. Multi-objective genetic algorithm for energy-efficient job shop scheduling. Int. J. Prod. Res. 53, 7071–7089. https://doi.org/10.1080/00207543.2015.1005248.

Mourtzis, D., Vlachou, E., 2018. A cloud-based cyber-physical system for adaptive shop-floor scheduling and condition-based maintenance. J. Manuf. Syst. 47, 179–198. https://doi.org/10.1016/j.jmsy.2018.05.008.

Mouzon, G., Yildirim, M.B., Twomey, J., 2007. Operational methods for minimisation of energy consumption of manufacturing equipment. Int. J. Prod. Res. 45, 4247–4271. https://doi.org/10.1080/00207540701450013.

Muhlemann, A.P., Lockett, A.G., Farn, C.K., 1982. Job shop scheduling heuristics and frequency of scheduling. Int. J. Prod. Res. 20, 227–241. https://doi.org/10.1080/

00207548208947763.
Nouiri, M., Bekrar, A., Jemai, A., Trentesaux, D., Ammari, A.C., Niar, S., 2017. Two stage particle swarm optimisation to solve the flexible job shop predictive scheduling problem considering possible machine breakdowns. Comput. Ind. Eng. 112, 595–606. https://doi.org/10.1016/j.cie.2017.03.006.
Nouiri, M., Bekrar, A., Trentesaux, D., 2019. An energy-efficient scheduling and rescheduling method for production and logistics systems†. Int. J. Prod. Res. 58 (11), 3263–3283. https://doi.org/10.1080/00207543.2019.1660826.
Ozturk, G., Bahadir, O., Teymourifar, A., 2019. Extracting priority rules for dynamic multi-objective flexible job shop scheduling problems using gene expression programming. Int. J. Prod. Res. 57, 3121–3137. https://doi.org/10.1080/00207543.2018.1543964.
Pfund, M.E., Fowler, J.W., 2017. Extending the boundaries between scheduling and dispatching: hedging and rescheduling techniques. Int. J. Prod. Res. 55, 3294–3307. https://doi.org/10.1080/00207543.2017.1306133.
Plitsos, S., Repoussis, P.P., Mourtos, I., Tarantilis, C.D., 2017. Energy-aware decision support for production scheduling. Decis. Support Syst. 93, 88–97. https://doi.org/10.1016/j.dss.2016.09.017.
Rahmani, D., Ramezanian, R., 2016. A stable reactive approach in dynamic flexible flow shop scheduling with unexpected disruptions: a case study. Comput. Ind. Eng. 98, 360–372. https://doi.org/10.1016/j.cie.2016.06.018.
Salido, M.A., Escamilla, J., Barber, F., Giret, A., 2017. Rescheduling in job-shop problems for sustainable manufacturing systems. J. Clean. Prod. 162, S121–S132. https://doi.org/10.1016/j.jclepro.2016.11.002.
Shahgholi Zadeh, M., Katebi, Y., Doniavi, A., 2019. A heuristic model for dynamic flexible job shop scheduling problem considering variable processing times. Int. J. Prod. Res. 57, 3020–3035. https://doi.org/10.1080/00207543.2018.1524165.
Shen, X.-N., Yao, X., 2015. Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. Inf. Sci. (Ny) 298, 198–224. https://doi.org/10.1016/j.ins.2014.11.036.
Shi, L., Guo, G., Song, X., 2019. Multi-agent based dynamic scheduling optimisation of the sustainable hybrid flow shop in a ubiquitous environment. Int. J. Prod. Res. 59 (2), 576–597. https://doi.org/10.1080/00207543.2019.1699671.
Sreekara Reddy, M.B.S., Ratnam, C., Rajyalakshmi, G., Manupati, V.K., 2018. An effective hybrid multi objective evolutionary algorithm for solving real time event in flexible job shop scheduling problem. Meas. J. Int. Meas. Confed. 114, 78–90. https://doi.org/10.1016/j.measurement.2017.09.022.
Tang, D., Dai, M., Salido, M.A., Giret, A., 2016. Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimisation. Comput. Ind. 81, 82–95. https://doi.org/10.1016/j.compind.2015.10.001.
Tay, J.C., Ho, N.B., 2008. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Comput. Ind. Eng. 54, 453–473. https://doi.org/10.1016/j.cie.2007.08.008.
Tian, S., Wang, T., Zhang, L., Wu, X., 2019a. An energy-efficient scheduling approach for flexible job shop problem in an internet of manufacturing things environment. IEEE Access 7, 62695–62704. https://doi.org/10.1109/ACCESS.2019.2915948.
Tian, S., Wang, T., Zhang, L., Wu, X., 2019b. The Internet of Things enabled manufacturing enterprise information system design and shop floor dynamic scheduling optimisation. Enterp. Inf. Syst. 14, 1238–1263. https://doi.org/10.1080/17517575.2019.1609703.
Turker, A.K., Aktepe, A., Inal, A.F., Ersoz, O.O., Das, G.S., Birgoren, B., 2019. A decision support system for dynamic job-shop scheduling using real-time data with simulation. Mathematics 7. https://doi.org/10.3390/math7030278.
Valledor, P., Gomez, A., Priore, P., Puente, J., 2018. Solving multi-objective rescheduling problems in dynamic permutation flow shop environments with disruptions. Int. J. Prod. Res. 56, 6363–6377. https://doi.org/10.1080/00207543.2018.1468095.
Wang, C., Jiang, P., Lu, T., 2018. Production events graphical deduction model enabled real-time production control system for smart job shop. Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci. 232, 2803–2820. https://doi.org/10.1177/0954406217728531.
Wang, J., Yang, J., Zhang, Y., Ren, S., Liu, Y., 2020. Infinitely repeated game based real-time scheduling for low-carbon flexible job shop considering multi-time periods. J. Clean. Prod. 247 https://doi.org/10.1016/j.jclepro.2019.119093.
Wang, J., Zhang, Y., Liu, Y., Wu, N., 2019. Multiagent and bargaining-game-based real-time scheduling for internet of things-enabled flexible job shop. IEEE Internet Things J. 6, 2518–2531. https://doi.org/10.1109/JIOT.2018.2871346.
Wang, S., Zhang, Y., 2020. A credit-based dynamical evaluation method for the smart configuration of manufacturing services under Industrial Internet of Things. J. Intell. Manuf. https://doi.org/10.1007/s10845-020-01604-y. In press.
Wang, W., Yang, H., Zhang, Y., Xu, J., 2018. IoT-enabled real-time energy efficiency optimisation method for energy-intensive manufacturing enterprises. Int. J. Comput. Integrated Manuf. 31, 362–379. https://doi.org/10.1080/0951192X.2017.1337929.
Wang, W., Zhang, Y., Zhong, R.Y., 2020. A proactive material handling method for CPS enabled shop-floor. Robot. Comput. Integrated Manuf. 61, 101849 https://doi.org/10.1016/j.rcim.2019.101849.
Wang, Z., Zhang, J., Yang, S., 2019. An improved particle swarm optimisation algorithm for dynamic job shop scheduling problems with random job arrivals. Swarm Evol. Comput. 51 https://doi.org/10.1016/j.swevo.2019.100594.
Yao, Y., Jiao, J., Han, X., Wang, C., 2019. Can constraint targets facilitate industrial green production performance in China? Energy-saving target vs emission-reduction target. J. Clean. Prod. 209, 862–875. https://doi.org/10.1016/j.jclepro.2018.10.274.
Yildirim, M.B., Mouzon, G., 2012. Single-machine sustainable production planning to minimise total energy consumption and total completion time using a multiple objective genetic algorithm. IEEE Trans. Eng. Manag. 59, 585–597. https://doi.org/10.1109/TEM.2011.2171055.
Yin, L., Li, X., Gao, L., Lu, C., Zhang, Z., 2017. A novel mathematical model and multi-objective method for the low-carbon flexible job shop scheduling problem. Sustain. Comput. Informatics Syst. 13, 15–30. https://doi.org/10.1016/j.suscom.2016.11.002.
Zhang, C., Yao, X., Tan, W., Zhang, Y., Zhang, F., 2019. Proactive scheduling for job-shop based on abnormal event monitoring of workpieces and remaining useful life prediction of tools in wisdom manufacturing workshop. Sensors (Switzerland) 19. https://doi.org/10.3390/s19235254.
Zhang, J., Yang, J., Zhou, Y., 2016. Robust scheduling for multi-objective flexible job-shop problems with flexible workdays. Eng. Optim. 48, 1973–1989. https://doi.org/10.1080/0305215X.2016.1145216.
Zhang, S., Wong, T.N., 2017. Flexible job-shop scheduling/rescheduling in dynamic environment: a hybrid MAS/ACO approach. Int. J. Prod. Res. 55, 3173–3196. https://doi.org/10.1080/00207543.2016.1267414.
Zhang, Y., Liu, S., Liu, Y., Yang, H., Li, M., Huisingh, D., Wang, L., 2018. The 'Internet of Things' enabled real-time scheduling for remanufacturing of automobile engines. J. Clean. Prod. 185, 562–575. https://doi.org/10.1016/j.jclepro.2018.02.061.
Zhang, Y., Ren, S., Liu, Y., Si, S., 2017a. A big data analytics architecture for cleaner manufacturing and maintenance processes of complex products. J. Clean. Prod. 142, 626–641. https://doi.org/10.1016/j.jclepro.2016.07.123.
Zhang, Y., Wang, J., Liu, Y., 2017b. Game theory based real-time multi-objective flexible job shop scheduling considering environmental impact. J. Clean. Prod. 167, 665–679. https://doi.org/10.1016/j.jclepro.2017.08.068.
Zhang, Y., Zhang, D., Wang, Z., Qian, C., 2020. An optimal configuration method of multi-level manufacturing resources based on community evolution for social manufacturing. Robot. Comput. Integrated Manuf. 65 https://doi.org/10.1016/j.rcim.2020.101964.
Ziaee, M., 2014. A heuristic algorithm for the distributed and flexible job-shop scheduling problem. J. Supercomput. 67, 69–83. https://doi.org/10.1007/s11227-013-0986-8.