Manufacturing Services Scheduling with Supply-Demand Dual Dynamic Uncertainties Toward Industrial Internet Platforms

Ying Cheng, Yifan Xie, Dongxu Wang, Fei Tao, *Senior Member*, *IEEE*, Ping Ji

***Abstract*—As a series of industrial Internet platforms have been launched, manufacturing facilities in physical world, although distributed in different enterprises, are interconnected as the form of manufacturing services (MSs) in cyber space. In this context, it makes possible for on-demand sharing of MSs as well as corresponding cross-enterprise collaboration. However, many dynamic uncertainties of both MSs and the submitted demands occur unpredictably, which seriously hinders the platforms' applications. To cope with the problem of MSs scheduling with supply-demand dual dynamic uncertainties, a three-stage approach based on an evolutionary hypernetwork model is proposed. In which, six of nine kinds of dynamic events and eighteen specific conditions are considered, and an event-condition-act mechanism is designed to guide local/global re-scheduling if needed. Experimental results show the effectiveness of the proposed approach, as well as the potential of a platform employing the approach in response to different dynamic events in its application.**

***Index Terms*—Manufacturing service, dynamic scheduling, evolutionary hypernetwork, industrial Internet platform, event-condition-act (ECA) mechanism**

NOMENCLATURE

Time-independent constants		
Task related	en_p_T	Set of tasks of the enterprise en_p
	$t_i\_arrivaltime$	Arrival or recovered time of the task t_i
	$t_i\_deadline$	Expected completion time of the task t_i
	r_{ti}	Probability of the task t_i's malfunction
	R_t	Threshold of tasks' malfunction
	${R_t}^*$	Threshold of tasks' recovering
Service related	en_p_S	Set of services of the enterprise en_p
	$s_k\_arrivaltime$	Arrival time or the time to be re-available of the service s_k
	$s_k\_deadline$	Withdrawn time or the time to be unavailable of the service s_k
	$M^{S\_T}(t_i)$	Services that can be selected by t_i
	r_{sk}	Probability of s_k's malfunction
	R_s	Threshold of services' malfunction
	${R_s}^*$	Threshold of services' recovering
Hyper-edge related	$e_{ki}^{S\_T}\_arrivaltime$	Arrival time or the time to be re-available of the hyper-edge $e_{ki}^{S\_T}$
	$e_{ki}^{S\_T}\_deadline$	Withdrawn time or the time to be unavailable of the hyper-edge $e_{ki}^{S\_T}$
	$r_{ki}^{S\_T}$	Probability of the hyper-edge $e_{ki}^{S\_T}$'s malfunction
	$u_{ki}^{S\_T}$	Utility value when s_k performs t_i
	$time_{ki}^{S\_T}$	Execution time when s_k performs t_i
	R_{S_T}	Threshold of hyper-edges' malfunction
	${R_{S\_T}}^*$	Threshold of hyper-edges' recovering
	w_r	The weight of reliability
	w_u	The weight of utility
	λ_{*i}	Utility attenuation coefficient of t_i
Time-related variables		
Task related	t_T	Set of tasks at the time t
	$t\_en_p\_T$	Set of tasks of the enterprise en_p at the time t
	$t\_Influence\_T$	Set of tasks influenced by dynamic events during the period $[t-1, t)$
	$t\_Complete\_T$	Set of tasks that have completed during the period $[t-1, t)$
	t_Delete_T	Set of deleted tasks during the period $[t-1, t)$
	t_Add_T	Set of added/recovered tasks during the period $[t-1, t)$
	p_{ti}^t	Chosen probability of the task t_i's malfunction during the period $[t-1, t)$
Service related	t_S	Set of services at the time t
	$t\_en_p\_S$	Set of services of the enterprise en_p at the time t
	$t\_M^{S\_T}(t_i)$	Services that can be selected by the task t_i at the time t
	t_Delete_S	Set of deleted services during the period $[t-1, t)$
	t_Add_S	Set of added/recovered services during the period $[t-1, t)$
	p_{sk}^t	Chosen probability of the service s_k's malfunction during the period $[t-1, t)$
Hyper-edge related	$t\_E^{S\_T}$	Set of hyper-edges at the time t
	$t\_Delete\_E^{S\_T}$	Set of deleted hyper-edges during the period $[t-1, t)$
	$t\_Add\_E^{S\_T}$	Set of added/recovered hyper-edges during the period $[t-1, t)$
	$p_{e_{ki}^{S\_T}}^t$	Chosen probability of the hyper-edge $e_{ki}^{S\_T}$'s malfunction during the period $[t-1, t)$

Manuscript received XXX; revised XXX; accepted XXX. Date of publication XXX; date of current version XXX. This work is partly supported by National Natural Science Foundation of China under Grants 51705014 and 51875030, and Beijing Science Fund for Distinguished Young Scholars under Grant JQ19011. Paper no. TII-19-2510. (Corresponding author: Fei Tao.)

Ying Cheng, Yifan Xie, Dongxu Wang and Fei Tao are with the School of Automation Science and Electrical Engineering, Beihang University (BUAA), Beijing 100191, P. R. China (e-mail: ycheng@buaa.edu.cn; xyfan1234@163.com; wdxzsq@buaa.edu.cn; ftao@buaa.edu.cn).

Ping Ji is with the Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong, P. R. China (e-mail: p.ji@polyu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digsital Object Identifier XXX.

I. INTRODUCTION

ALONG with the gradual development and applications in industry of some new-generated information technologies,

a series of industrial Internet platforms or also called industrial IoT (IIoT) platforms have been brought forward based on the architecture of cyber physical systems (CPS) and have drawn wide attention from both academia and industry [1-4]. For example, there have existed MindSphere platform of Siemens, Predix platform of GE, ABB Ability platform, and so on. The manufacturing facilities and capabilities of different enterprises that distribute in different locations are interconnected via the platforms, and then provided as manufacturing services (MSs) for users. Consequently, it is an inevitable trend for enterprises to participate in the derived distributed collaboration and create value through the platforms.

However, the industrial Internet platforms are still facing problems. One major problem is that it is difficult to schedule the distributed MSs according to demand. The sensor & cloud-based system architecture of the platforms is explored widely for their cyber-physical integration. On the assumption that both sensor-based environment in shop floors within an enterprise and cloud-based environment among lots of networked enterprises are deployed, the connection, interconnection as well as real-time information exchange between each two facilities in shop floors are achieved based on sensor-based environment in physical world, furthermore, any manufacturing facility or resource provided by each participated enterprise is treated as a cloud service which is also called a MS based on cloud-based environment. By sharing real-time information, the distributed manufacturing facilities in the logical form of linked or aggregated MSs could be easily utilized or even highly shared by users at anywhere and anytime. That is to say, the available MSs of any networked enterprise that is participated in could be selected or outsourced, while it is not limited only within one enterprise. Therefore, it supports distributed collaboration among lots of networked enterprises with enough flexibility but without the limitations of both scale and scope. To further popularize the platforms and make it accepted and used by more enterprises, the problem of distributed MSs scheduling for cross-enterprise collaboration is one of the key problems to be solved urgently.

Furthermore, in real world, there is a quite common phenomenon, which is called dynamic uncertainties, that dynamic events or a myriad of unanticipated interruptions, such as, machine breakdowns, rush orders, yield variations, and hot jobs, etc., occurring stochastically in the platforms. They would lead to new situations that are unknown and unpredictable and they produce doubt because of a lack of assurance and confidence. In details, the dynamic uncertainties, which we called supply-demand dual dynamic uncertainties, are mainly reflected in two scenes. Before the production, there would be uncertain and stochastic demand and supply of the available MSs to be dispatched for pre-scheduling at any moment under cloud-based environment. During the subsequent operation processes, some unpredictable or even unavoidable events would occur dynamically and may trigger some faults as well as destroy the original pre-scheduling schemes, so that either local adjustment or global re-scheduling with real-time information under sensor-based environment is much more desired. However, majority of the related studies were prepared for a static environment, where all dispatched resources are assumed rightly available and unlimited at all times. It results that the schedules experience large changes when new conditions occur. In addition, the efficiency, stability, sustainability, flexibility, as well as consistency should be considered and addressed in the changes of original schedules adapting to dynamic events. Therefore, there is a need to understand and tackle dynamic uncertainties in the platforms, which bring challenges to the abovementioned problem.

As a result, in order to promote the wider applications of industrial Internet platforms, how to solve such a problem of MSs scheduling considering the environment with supply-demand dual dynamic uncertainties, so as to support efficient distributed collaboration and dynamic control among therein networked enterprises, as shown in Fig. 1, becomes a big challenge but one of the most important issues which must be settled. To deal with this problem, an evolutionary hypernetwork model is introduced to describe the MSs scheduling problem with dynamic uncertainties, and a three-stage approach based on the evolutionary hypernetwork model is proposed. In particular, to meet the requirements of stability and efficiency preferentially, the objective function of the problem is established with a comprehensive consideration of reliability and utility, and three algorithms are employed for solving this scalable and dynamic optimization problem.

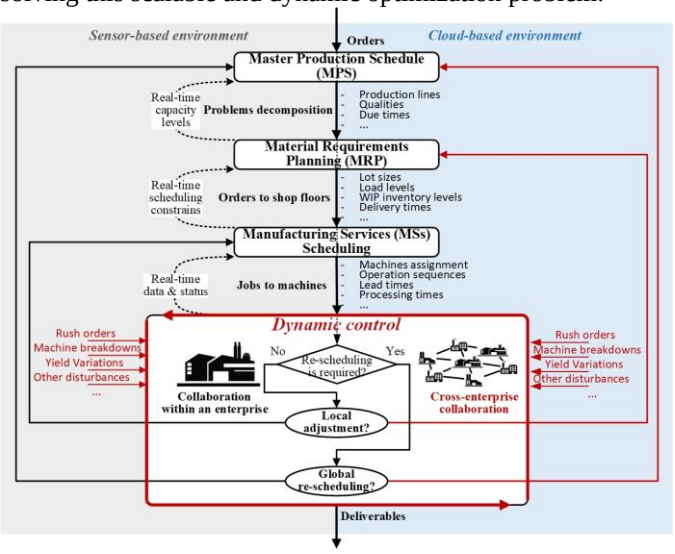


Fig. 1. Features on the MSs scheduling problem with dynamic uncertainties.

The remainder of this article is organized as follows. In Section 2, a review on related work is provided. An evolutionary hypernetwork based approach is proposed in Section 3 and its three stages are illustrated in detail in Section 4. Experiments by employing three algorithms are conducted successively in Section 5 to show feasibility of the proposed approach, and discussion and conclusions are finally carried out based on the experimental results.

II. RELATED WORK

A. *Existing MSs scheduling problem models*

The dynamic and uncertain environment is usually divided into two stages. At the initial state, the number of participated users is uncertain, which indicates that the numbers of services and tasks submitted by each user are uncertain. During the execution process, various dynamic events of both services and tasks occur unpredictably. Therefore, the MSs scheduling problems with dynamic uncertainties can be divided into eight situations, namely situations A-H which are shown in Table 1.

TABLE I
DIFFERENT MSS SCHEDULING PROBLEMS IN THE II PLATFORMS

At initial state		Demand		during execution process
		Certain	Uncertain	
Supply	Certain	A/E	B/F	Dynamics do not exist
	Uncertain	C/G	D/**H**	/ Dynamics exist

Most related studies optimize the scheduling problems just considering static and certain environment in manufacturing systems. The scheduling or its similar problems with situation A have been widely researched in the existing studies, including MS selection [5], composition and scheduling [6], multi-task scheduling of distributed services [7], and so on. Usually, more than one criterion is considered in their functional objectives modelling, such as time, quality of service [8], cost [8, 9], energy [9], utilization [10, 11], etc. However, manufacturing systems always operate in dynamic environment where usually some unpredictable real-time events may cause a change in the scheduled plans, and a previously feasible scheme may turn infeasible when it is released to the shop floor [12]. There are still few studies aiming to explore the problems of dynamic MS scheduling, which are specific to situations B-H. Therefore, the following analysis extends to the dynamic scheduling problems of distributed manufacturing. Apart from the criteria used in the related problems with situation A, many different criteria are put forward for dynamic scheduling, such as robustness and stability [13], flexibility [14], efficiency, consistency, etc. To measure the performance of re-scheduling, some typical time-based measures such as makespan, mean flow time and total tardiness [15], are sometimes transformed into cost-based measures such as revenue, production cost and penalty cost. Measures of the time taken to generate a new feasible scheme upon a disruption, namely response speed, and measures of stability, including the deviation from the original schedule, are also of importance and considered [11]. Regardless of which criteria are considered, the problems are usually taken as a linear programming model or mixed integer programming model, constraint programming model, discrete simulation models, or divide and allocate resources or tasks hierarchically based on their flow and path [16].

B. Existing dynamic scheduling approaches

Abovementioned performances are considered in the changes of the original plans and schemes adapting to the dynamic events. Some algorithms are selected in accordance with some specific dispatching mechanisms when addressing the problems. The explored dynamic scheduling solutions responding to events or interruptions can be generally classified into four groups, including (a) re-generating the complete schedule, (b) switching to dispatching mode, (c) modifying the existing schedule, and (d) continuing to follow the existing schedule and letting the production system gradually absorb the impact of the occurred dynamic events or interruptions [17]. For dispatching mechanisms, they allow the dynamic response or reactions that are adapting to the dynamic events. The complete changes driven by the dispatching mechanisms usually offer the advantage of having a mechanism which makes response and adaption to the dynamic events in real time [18]. In order to modify the detailed schedules following the occurrence of an interruption or an event, some self-adaptive [19] or self-organizing [3] strategies are also explored.

For solving problems, many algorithms are applied to search better solutions with less time consumption. For example, a hybrid multi-agent system negotiation and an ant colony optimization (ACO) approach are developed to discuss the flexible job-shop scheduling and re-scheduling problem with different types of disruptions [20], a dynamic scheduling strategy based on predictive-reactive scheduling approach is proposed for dynamic flexible flow shop problem [15], and an event-triggered dynamic scheduling method is put forward to solve the scheduling problem in cloud environment with randomly arriving tasks [21].

C. A brief summary

A vast majority of researchers have dealt with scheduling methods focused on finding optimal schedules for simple and static scheduling models. However, there are still some limitations: (a) The traditional problems are mainly with the assumption of static conditions in manufacturing systems, only either dynamic events of the demands such as arrival of new orders, or dynamic events of the supplied resources such as machine breakdown are considered. There are few studies addressing the dual dynamic events of both supply and demand. (b) Most of the existing studies apply linear or mixed integer programming models, constraint programming models and discrete simulation models. Based on these models, the related studies mainly discuss and obtain the final schemes for finite scale and static environment by solutions of various kinds of algorithms. But they ignore the interrelations among the supplies and demands. Furthermore, when the quantities of both services and demands change rapidly, these models are inefficient and unable to meet the dual dynamic requirements.

Therefore, aiming to accomplish modelling, analysis and solution for MSs scheduling problems with situation H, the contributions in this article include:

a) An evolutionary hypernetwork model is defined to depict the MSs scheduling problem, especially its dual dynamic uncertainties and the derived scalabilities.
b) All dynamic uncertainties are extracted as nine kinds of dynamic events in the defined model, six of them and eighteen specific conditions are further discussed.
c) A three-stage approach based on the defined model is proposed for settling the MSs scheduling problems.
d) Considering reliability and utility evaluation, experimental results show the efficiency of the proposed approach for the platforms responding to any dynamic event.

III. AN EVOLUTIONARY HYPERNETWORK BASED APPROACH FOR MSS SCHEDULING WITH DUAL DYNAMIC UNCERTAINTIES

For the MSs scheduling problem discussed in this article, an iteratively evolutionary hypernetwork based approach is proposed. In order to describe the problem during a certain period, any time range considered could be divided into N units. When defining $t = 0$ as the initial state, the process picked out for the problem should be considered covering $t = 1 \sim N$. As illustrated in Fig. 2, the general workflow includes two parts:

(a) initialization and pre-scheduling when $t = 0$; and

(b) dynamic-event-driven re-scheduling when $t \geq 1$, if necessary.

At any time when $t \geq 1$, once dynamic-event-driven re-scheduling is triggered, there are three stages carried out,

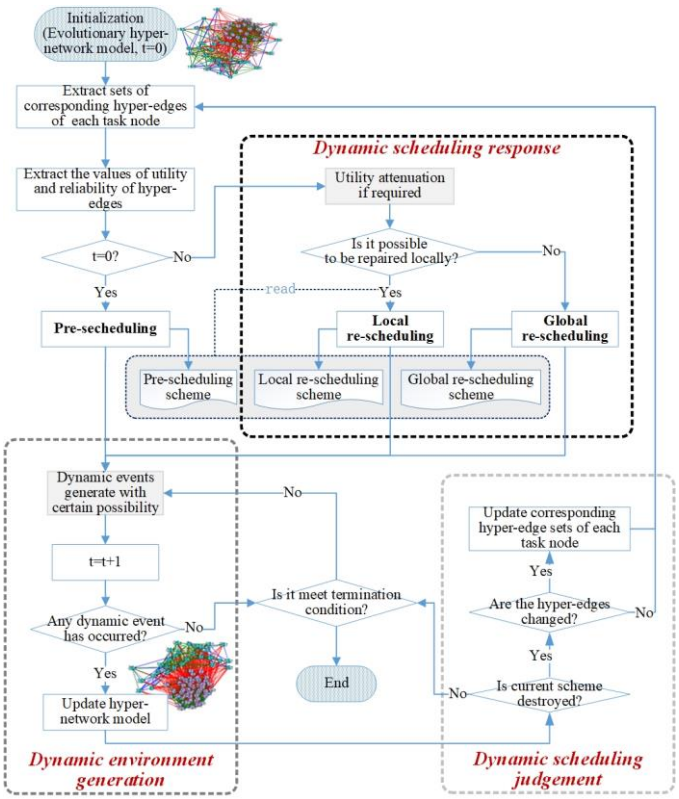


Fig. 2. Workflow of the proposed approach.

including dynamic environment generation (*Stage G*) based on the evolutionary hypernetwork, dynamic scheduling judgement (*Stage J*) and dynamic scheduling response (*Stage R*). It is specifically defined that, when $t = n$, check and judge whether there are dynamic events occurred during the interval $t = [n-1, n)$. The workflow of the approach is described as below.

Step 1: Initialization. Collect the supply and demand information of MSs in an industrial Internet platform and generate the initial hypernetwork model when $t = 0$.
Step 2: Extract the corresponding hyper-edge sets of each task node from the initial hypernetwork model, namely collection of the matchable services to be selected for each task.
Step 3: Extract the values of reliability and utility of each hyper-edge.
Step 4: Judge the current moment is at the initiate state or during the subsequent process. If $t = 0$, turn to next step; If $t \geq 1$, jump to ***Step R1*** in *Stage R*.
Step 5: Pre-scheduling, and then turn to ***Step G1*** in *Stage G*.
Step 6: Termination condition judgment. If executions of all scheduled tasks are finished, the scheduling process ends; otherwise, jump to ***Step G1*** in *Stage G*.
Stage G
Step G1: Specific operation during a time unit, dynamic events generate with probability in the evolutionary hypernetwork model.
Step G2: Time counting is going, $t = t + 1$.
Step G3: Judge whether dynamic events generate during this time unit. If it happens, turn to the next step; otherwise, jump to ***Step 6***.
Step G4: Update the hypernetwork model and turn to ***Step J1*** in *Stage J*.
Stage J
Step J1: Judge whether the updated hypernetwork breaks the existing scheme. If not, jump to ***Step 6***; if the existing scheme is destroyed, turn to the next step.
Step J2: Judge whether the hyper-edges in the updated hypernetwork are changed. If changed, turn to the next step; otherwise, jump to ***Step 2***.
Step J3: Update the corresponding set of hyper-edges and jump to ***Step 2***.
Stage R
Step R1: Due to the updating of hyper-edges caused by dynamic events, the utility values of corresponding hyper-edges are attenuated regularly.
Step R2: Judge whether the existing scheme can be partially restored. If it can be restored, turn to the next step, otherwise, jump to ***Step R4***.
Step R3: Local re-scheduling based on the existing scheme, and turn to ***Step G1*** in *Stage G*.
Step R4: Global re-scheduling, and turn to ***Step G1*** in *Stage G*.

In the abovementioned workflow, the initialization of hypernetwork model for MSs scheduling problem and its pre-scheduling when $t = 0$ are the basic premise of the proposed approach. Before expounding the detailed three stages of dynamic-event-driven re-scheduling during the operation process, the initial hypernetwork model and its pre-scheduling are illustrated here at first.

(1) The initial hypernetwork model

Specific to the problem discussed in this article, the initial model is improved based on the static model of MS supply-demand matching hypernetwork, which is proposed in previous work and donated as $Matching\_Net$ [22]. As shown in Fig. 3, the previously proposed hypernetwork model consists of manufacturing service network (S_Net), manufacturing task network (T_Net), and the hyper-edges E^{S_T} between these two networks [22]. For the problem discussed in this article, an enterprises collaborative network E_Net is accordingly supplemented, so that the initial hypernetwork model is improved as $Matching\_Net = \{\boldsymbol{V}(S, T, EN), \boldsymbol{E}(E^S, E^T, E^{EN}, E^{S\_T})\} = \{\boldsymbol{S\_Net}(S, E^S), \boldsymbol{E^{S\_T}}, \boldsymbol{T\_Net}(T, E^T); \boldsymbol{E\_Net}(EN, E^{EN})\}$. In which, the three sets of nodes $S = \{s_k | k = 1,2,\dots,NoS\}$, $T = \{t_i | i = 1,2,\dots,NoT\}$ and $EN = \{en_p | p = 1,2,\dots,NoE\}$ are respectively standing for services, tasks and the participated enterprises; the first three sets of edges E^S, E^T and E^{EN} are the specific correlations among those services, tasks, and different enterprises; and the fourth set of interlayer edges $E^{S\_T} = \{e_{ki}^{S\_T} | k = 1,2,\dots,NoS; i = 1,2,\dots,NoT\}$ indicates the matchable hyper-edges between services and tasks.

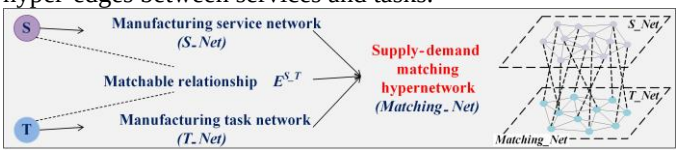


Fig. 3. The schematic of the hypernetwork $Matching\_Net$ proposed in [22].

To describe the environment with dynamic uncertainties, some risk-related attributes are supplemented, such as probability of a task's malfunction r_{ti}, probability of a service's malfunction r_{sk}, and probability of a hyper-edge's malfunction $r_{ki}^{S\_T}$ and the utility value $u_{ki}^{S\_T}$ of a hyper-edge. Moreover, the descriptive parameters for uncertainty including the arrival time $t_i\_arrivaltime$ and the expected completion time $t_i\_deadline$ of each task and the actual execution time $time_{ki}^{S\_T}$ when the service s_k performs the task t_i, are also supplemented. The actual execution of each task can be arbitrary but should be between its arrival time and expected completion time. The attributes $r_{ki}^{S\_T}$, $u_{ki}^{S\_T}$ and $time_{ki}^{S\_T}$ are part of the weight attributes of each hyper-edge. Only when a hyper-edge exists, namely $e_{ki}^{S\_T} = 1$, its corresponding attributes make sense.

In particular, to specify whether the service s_k is selected for the task t_i and what time it will start to perform in the scheme scheduled at the time t, the decision coefficients are donated as $\langle a_{ki}{}^t, a_{ki}\_starttime^t \rangle$. $a_{ki}{}^t$ is assigned as 0 or 1, and $a_{ki}\_starttime^t$ is set as the natural numbers from the time t to N. If $a_{ki}{}^t = 1$ in the scheme generated at the time t, s_k is invoked to perform t_i, and it begins to perform at the moment that $a_{ki}\_starttime^t$ reveals.

(2) The objective of pre-scheduling

For pre-scheduling when $t = 0$, the decision coefficients are simplified as $\langle a_{ki}, a_{ki}\_starttime \rangle$. Any task's executable time interval $[t_i\_arrivaltime, t_i\_deadline]$ and its set of optional services $M^{S\_T}(t_i) = \{s_k | e_{ki}^{S\_T} = 1; k = 1,2, \dots, NoS\}$, as well as its actual execution time $time_{ki}^{S\_T}$ spending by different services, are known from the initial hypernetwork. If all of enterprises are small-and-medium-sized enterprises, the MSs owned by any enterprise are indicated as $en_p\_S = \{s_{k_p} | k_p = 1,2, \dots, en_p\_NoS\}$. To achieve the average utility and reliability of the whole system as higher as possible, the objective of pre-scheduling is to maximize the comprehensive value of all of enterprises' average utility and average reliability, as shown in formula 1.

$$\max \frac{\sum_{en_p=1}^{NoE} \left(\frac{\omega_r \left(\sum_{i=1}^{NoT} \sum_{k_p}^{|en_p\_S|} a_{k_p i} \left(1 - r_{k_p i}^{S\_T}\right) \right) + \omega_u \left(\sum_{i=1}^{NoT} \sum_{k_p}^{|en_p\_S|} a_{k_p i} u_{k_p i}^{S\_T} \right)}{\sum_{i=1}^{NoT} \sum_{k_p}^{|en_p\_S|} a_{k_p i}} \right)}{NoE} \quad (1)$$

Subject to

$$\{a_{ki}\}_{NoS \times NoT} \subseteq E^{S\_T} \quad (2)$$

$$t_i\_arrivaltime \le a_{ki}\_starttime < a_{ki}\_starttime + time_{ki}^{S\_T} \le t_i\_deadline \quad (3)$$

$$\sum_{k=1}^{NoS} a_{ki} = 1 \quad (4)$$

$$[a_{ki}\_starttime, a_{ki}\_starttime + time_{ki}^{S\_T}] \cap [a_{kj}\_starttime, a_{kj}\_starttime + time_{kj}^{S\_T}] = \emptyset \quad (5)$$

The constraints are described as formulas 2-5. Formula 2 reveals that the service invoked by each task must be chosen from its corresponding set of optional matchable services. Formula 3 shows that the actual execution interval of the invoked service should satisfy the executable time interval of the corresponding task. Formula 4 and 5 respectively indicate that each task can invoke only one service, and each service can only perform one task at the same time, namely when a service is assigned to different tasks its actual execution intervals for different tasks are not superimposable.

IV. THREE STAGES OF DYNAMIC-EVENT-DRIVEN RE-SCHEDULING IN THE PROPOSED APPROACH

Based on the initialization of hypernetwork model and the solution of pre-scheduling when $t = 0$, three stages of the subsequent dynamic-event-driven re-scheduling when $t \ge 1$ in the proposed approach are illustrated in this section.

A. Dynamic environment generation (Stage G)

In *stage G*, it aims to iteratively generate the time-related evolutionary hypernetwork model to describe the MSs scheduling problem with supply-demand dual dynamic uncertainties. To make the hypernetwork model evolutionary, some dynamic events caused by the dynamic uncertainties in the platforms need to be firstly explored.

1) Dynamic uncertainties in industrial Internet platforms

Usually some inevitable unpredictable events may cause a change in the schemes. These dynamics lead to uncertainties that need to be understood and dealt with in real world. The uncertainty is defined as the lack of complete certainty, that is, the existence of more than one possibility [23]. Since the platforms are based on the architecture of CPS, their dynamic uncertainties are analyzed from the following two perspectives.

The dynamics and uncertainties in physical world, are classified into two categories: supply-related and demand-related. The supply-related dynamics and uncertainties include machine breakdown, operator illness, unavailability or toll failures, loading limits, delay in the arrival or shortage of materials, defective material, etc. The demand-related dynamics and uncertainties include rush jobs, job cancellation, due date changes, early or late arrival of jobs, changes in job priority, changes in job processing time [12], etc. Moreover, the number of jobs and facilities are changing all the time [12], and the specific execution time of different jobs are also uncertain.

The dynamics and uncertainties in cyber space, are also classified into two parts including which originated by physical world or which originating in cyber space spontaneously. Based on the hypernetwork model, tasks are treated as nodes in the layer of T_Net, and facilities or other kinds of services are also seen as nodes in the layer of S_Net. Once there are disruptions in real world, the task/service nodes and the matchable relationships between tasks and services are changing with these disruptions. The platform guarantees to deliver services to tasks under any circumstances, but sometimes enterprises cannot access their data and resources due to network outage and system failures. The outage may be permanent, as a provider has gone out of business, or temporary. Either way, a failure to provide data and resources can be a disaster for enterprises. In addition, the platform takes charge to match tasks and services and decides the execution order and time. Once there is something wrong with the platform, the existing executions would be changed.

2) The evolutionary hypernetwork model with dynamic uncertainties

According to above analysis, those dynamic uncertainties in the platforms result in some dynamic events in the evolutionary hypernetwork model. In total, there are the following nine kinds of dynamic events, namely E1-E9.

E1: deletion of a service node,
E2: recovery or adding of a service node,
E3: change of some attributes of a service node,
E4: deletion or accomplishment of a task node,
E5: recovery or adding of a task node,
E6: change of some attributes of a task node,
E7: deletion of a hyper-edge,
E8: reconnection or adding of a hyper-edge, and
E9: change of attributes such as the weight of a hyper-edge.

Totally six dynamic events including E1, 2, 4, 5, 7 and 8, which make a big difference on the structure of the hypernetwork model, are discussed in priority in this article. The occurrence criterion of the dynamic events to be discussed are illustrated as follows.

E1: When $r_{sk} p_{sk}^t \ge R_s$, the service s_k is defined as malfunction; or when $s_k\_deadline \in [t-1, t)$, the service s_k is deleted or unavailable at the time t, it means $s_k \in t\_Delete\_S$. For this situation, s_k and all of its associated hyper-edges are deleted, that is, if $s_k \in t\_Delete\_S$, then $e_{k*}^{S\_T} \in t\_Delete\_E^{S\_T}$.

E2: For a service s_k that has been already deleted, it recovers at the time t when $(1 - r_{sk}) p_{sk}^t \ge R_s{}^*$; or when $s_k\_arrivaltime \in [t-1, t)$, it is added or recovered at the time t. For this situation, $s_k \in t\_Add\_S$, and all of its associated hyper-edges are added, namely, if $s_k \in t\_Add\_S$, then $e_{k*}^{S\_T} \in t\_Add\_E^{S\_T}$.

E4: The task t_i is defined as malfunction when $r_{ti} p_{ti}^t \ge R_t$, or

t_i is deleted at the time t when $t_i\_deadline \in [t-1, t)$. For this situation, $t_i \in t\_Delete\_T$, and all of its associated hyper-edges are deleted, namely, if $t_i \in t\_Delete\_T$, then $e_{*i}^{S\_T} \in t\_Delete\_E^{S\_T}$. Moreover, for the task t_i which is accomplished at the time t, $t_i \in t\_Complete\_T$ when $t-1 < \alpha_{*i} starttime + time_{*i}^{S\_T} \le t$. There are mainly two situations: (a) when $a_{*i} starttime < t-1 < \alpha_{*i} starttime + time_{*i}^{S\_T} \le t$, this is a general case and will be discussed in this article; and (b) when $t-1 < a_{*i} starttime < \alpha_{*i} starttime + time_{*i}^{S\_T} \le t$, this case is not considered as we assume that the execution time for a service is much longer than the setting time unit.

E5: For a task node t_i that has already been deleted, it recovers at the time t when $(1-r_{ti})p_{ti}^{t} \ge R_t{}^*$, or it is added at the time t when $t_i\_arrivaltime \in [t-1, t)$. For this situation, $t_i \in t\_Add\_T$. The task t_i and all of its associated hyper-edges are added, namely, if $t_i \in t\_Add\_T$, then $e_{*i}^{S\_T} \in t\_Add\_E^{S\_T}$.

E7: When $r_{ki}^{S\_T} p_{e_{ki}^{S\_T}}^{t} \ge R_{S\_T}$, the hyperedge $e_{ki}^{S\_T}$ is defined as malfunction, $e_{ki}^{S\_T} \in t\_Delete\_E^{S\_T}$.

E8: When both the service s_k and the task t_i exist and $(1-r_{ki}^{S\_T})p_{e_{ki}^{S\_T}}^{t} \ge R_{S\_T}{}^*$, the matchable hyperedge $e_{ki}^{S\_T}$ recovers, that is, $e_{ki}^{S\_T} \in t\_Add\_E^{S\_T}$.

Taking the above six dynamic events into consideration, the update models of the iteratively evolutionary hypernetwork are constructed as formulas 6-11. As defined in Section 3, all dynamic events occurred during $[t-1, t)$ are described in the update models at the time t. Formulas 6, 7 and 8 indicate the update sets of services, tasks, and hyper-edges at the time t. Formulas 9 and 10 describe the sets of available services and tasks of the enterprise en_p. Formula 11 shows the set of available services that can be chosen by the task t_i at the time t.

$$t\_S = (t-1)\_S - t\_Delete\_S + t\_Add\_S \quad (6)$$

$$t\_T = (t-1)\_T - t\_Complete\_T - t\_Delete\_T + t\_Add\_T \quad (7)$$

$$t\_E^{S\_T} = (t-1)\_E^{S\_T} - t\_Delete\_E^{S\_T} + t\_Add\_E^{S\_T} \quad (8)$$

$$t\_en_p\_S = en_p\_S \cap t\_S \quad (9)$$

$$t\_en_p\_T = en_p\_T \cap t\_T \quad (10)$$

$$t\_M^{S\_T}(t_i) = \{s_k | e_{ki}^{S\_T} \in t\_E^{S\_T}, e_{ki}^{S\_T} = 1, k = 1, 2, \ldots, NoS\} \quad (11)$$

B. Dynamic scheduling judgement (Stage J)

In *stage J*, it aims to conduct the judgement whether the current scheme is destroyed and dynamic scheduling is triggered by any dynamic event at the time t. In fact, all kinds of dynamic events make the model evolutionary iteratively. But not all of them would destroy the current scheme and trigger the operation of re-scheduling dynamically.

Specific to each dynamic event occurs during $[t-1, t)$, different conditions exist that could cause different results at the time t. Before the analysis on those specific conditions of different dynamic events, it is assumed that once a scheme is generated, there is a contract between those related enterprises. Unless there is an irresistible factor, such as the deletion of a selected service or a hyper-edge, the original scheme is retained without triggering re-scheduling dynamically, although some new added services or hyper-edges may bring out a better scheme or a new task comes out. In addition, the effect of deleting an existing task is ignored, and the recovery of a previously deleted task is regarded as adding a new task. Therefore, it is necessary to raise a specific analysis on the dynamic events of deleting a service or a hyper-edge.

Specifically, for the events of deleting a service or a hyper-edge depicted above, i.e., E1 and E7, different conditions of these two dynamic events are distinguished by considering the following four questions in order: (a) has the service or hyper-edge been selected and assigned for a task; (b) if it has been assigned, has it started execution; (c) is it recoverable; and (d) if it is recoverable, will it recover in time? Based on the state of a service or a hyper-edge at the time t, it can be divided into the following three categories: (a) a service or a hyper-edge has been assigned but has not yet started execution, (b) a service or a hyper-edge has been assigned and is in execution, and (c) a service or a hyper-edge has not been assigned. Furthermore, if a service or a hyper-edge is deleted, it can be also divided into three situations, including (a) timely recovery, namely the service or hyper-edge is invalidated when it has been invoked and can still complete the original assigned task as needed after recovery; (b) not timely recovery, namely the service or hyper-edge is invalidated when it has been invoked and cannot complete the original assigned task as needed after recovery; and (c) permanent invalidation.

According to above analysis, there totally are 18 kinds of conditions specific to those six dynamic events based on the mentioned judging criteria in Table 2. After conducting the judgement of different conditions specific to dynamic events, there are 9 conditions, including C3, C4, C6, C7, C10, C13, C14, C16 and C17 which are marked with "☆" in Table 2, destroying the current scheme and triggering dynamic scheduling. As to the remaining other 9 kinds of conditions, the current scheme is not destroyed so that it will keep proceeding.

C. Dynamic scheduling response (Stage R)

In *Stage R*, it aims to carry out the effective response efficiently for dynamic events in the evolutionary hypernetwork with different conditions. There are two problems need to be settled in this stage, namely, (a) which response act is appropriate responding to any dynamic event with different conditions; and (b) as two alternative response acts for dynamic scheduling, local and global re-scheduling, how to make decision on the alternative response acts.

Based on the dynamic events and specific conditions discussed in the first two stages, the event-condition-act (ECA) mechanism for local or global re-scheduling based on the evolutionary hypernetwork is proposed and adopted in this stage, which are illustrated in Fig. 4.

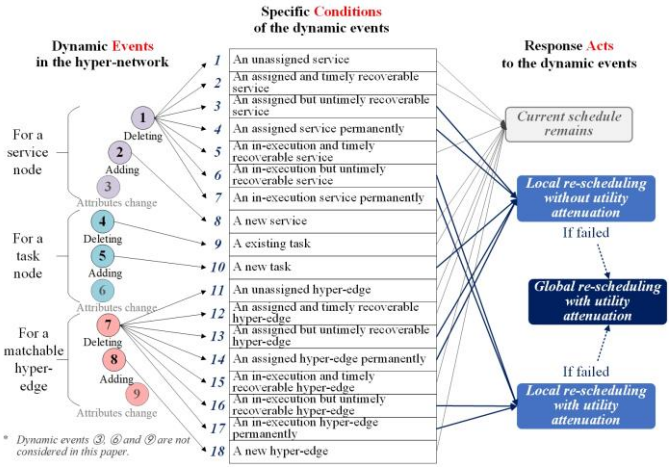


Fig. 4. The event-condition-act mechanism for dynamic scheduling response.

When the conditions C3, C4, C6 and C7 of E1, conditions

TABLE II
THE DYNAMIC EVENTS AND ITS CORRESPONDING JUDGEMENT CONDITIONS

Dynamic events in the hypernetwork	Specific conditions and judging criteria of different dynamic events				No.
	Has it been selected?	Has it started execution?	Is it recoverable?	Does it recover in time?	
E1 $s_k \in t\_Delete\_S$	**(N)** $a_{k*}{}^{t-1} = 0$			/	C1
	(Y) $a_{k*}{}^{t-1} = 1$	**(N)** $a_{k*}\_starttime^{t-1} > t$	**(Y)** $s_k\_arrivaltime \neq \infty$	**(Y)** $s_k\_arrivaltime \leq a_{k*}\_starttime^{t-1}$	C2
				(Y) $s_k\_arrivaltime > a_{k*}\_starttime^{t-1}$ and $s_k\_arrivaltime + time_{k*}^{S\_T} \leq t_*\_deadline$	C3☆
				(N) $s_k\_arrivaltime > a_{k*}\_starttime^{t-1}$ and $s_k\_arrivaltime + time_{k*}^{S\_T} > t_*\_deadline$	C3☆
			(N) $s_k\_arrivaltime = \infty$	/	C4☆
		(Y) $a_{k*}\_starttime^{t-1} \leq t < a_{k*}\_starttime^{t-1} + time_{k*}^{S\_T}$	**(Y)** $s_k\_arrivaltime \neq \infty$	**(Y)** $s_k\_arrivaltime + time_{k*}^{S\_T} - t + a_{k*}\_starttime^{t-1} \leq t_*\_deadline$	C5
				(N) $s_k\_arrivaltime + time_{k*}^{S\_T} - t + a_{k*}\_starttime^{t-1} > t_*\_deadline$	C6☆
			(N) $s_k\_arrivaltime = \infty$	/	C7☆
	An invalid service node recovers, this condition is discussed in dynamic event E1. $s_k \in \sum_{t=1}^{t-1} t\_Delete\_S$				/
E2 $s_k \in t\_Add\_S$	$s_k \notin \sum_{t=1}^{t-1} t\_Delete\_S$				C8
E4	$t_i \in t\_Delete\_T$ or $t_i \in t\_Complete\_T$				C9
E5	$t_i \in t\_Add\_T$; when $t_i \in \sum_{t=1}^{t-1} t\_Delete\_T$, the recovery of a previously deleted task node is regarded as adding a new task node.				C10☆
E7 $e_{ki}^{S\_T} \in t\_Delete\_E^{S\_T}$	**(N)** $a_{ki}{}^{t-1} = 0$			/	C11
	(Y) $a_{ki}{}^{t-1} = 1$	**(N)** $a_{ki}\_starttime^{t-1} > t$	**(Y)** $e_{ki}^{S\_T}\_arrivaltime \neq \infty$	**(Y)** $e_{ki}^{S\_T}\_arrivaltime \leq a_{ki}\_starttime^{t-1}$	C12
				(Y) $e_{ki}^{S\_T}\_arrivaltime > a_{ki}\_starttime^{t-1}$ and $e_{ki}^{S\_T}\_arrivaltime + time_{ki}^{S\_T} \leq t_{i}\_deadline$	C12
				(N) $e_{ki}^{S\_T}\_arrivaltime > a_{ki}\_starttime^{t-1}$ and $e_{ki}^{S\_T}\_arrivaltime + time_{ki}^{S\_T} > t_{i}\_deadline$	C13☆
			(N) $e_{ki}^{S\_T}\_arrivaltime = \infty$	/	C14☆
		(Y) $a_{ki}\_starttime^{t-1} \leq t < a_{ki}\_starttime^{t-1} + time_{ki}^{S\_T}$	**(Y)** $e_{ki}^{S\_T}\_arrivaltime \neq \infty$	**(Y)** $e_{ki}^{S\_T}\_arrivaltime + time_{ki}^{S\_T} - t + a_{ki}\_starttime^{t-1} \leq t_{i}\_deadline$	C15
				(N) $e_{ki}^{S\_T}\_arrivaltime + time_{ki}^{S\_T} - t + a_{ki}\_starttime^{t-1} > t_{i}\_deadline$	C16☆
			(N) $e_{ki}^{S\_T}\_arrivaltime = \infty$	/	C17☆
	An invalid matchable hyper-edge recovers, this condition is discussed in dynamic event E7. $e_{ki}^{S\_T} \in \sum_{t=1}^{t-1} t\_Delete\_E^{S\_T}$				/
E8 $e_{ki}^{S\_T} \in t\_Add\_E^{S\_T}$	$e_{ki}^{S\_T} \notin \sum_{t=1}^{t-1} t\_Delete\_E^{S\_T}$				C18

C13, C14, C16 and C17 of E7, and condition C10 of E5, occur at the time t, it is bound to make the current scheme destroyed. So that dynamic scheduling is required, and its alternative response acts include:

(1) *local re-scheduling*: once a task was influenced by the dynamic events occur in the evolutionary hypernetwork, the local re-scheduling would be performed preferentially; and

(2) *global re-scheduling*: if local re-scheduling is impossible to deal with this situation, global re-scheduling should be operated to handle the iteratively evolutionary hypernetwork.

No matter local or global re-scheduling it is performed as needed, the utility is attenuated which is created by the corresponding service execution for each influenced but executed task, such as the response for E1 with conditions C6 and C7, and E7 with conditions C16 and C17.

If the original invoked service has been in execution, the longer the execution time, the greater the attenuation of utility generated by new invocation of another service; if the original assigned service has not started execution, the utility of new service invocation does not attenuate. Consequently, when t_i is influenced and needs to invoke another service, the corresponding utility attenuation coefficient is calculated by formula 12.

$$\lambda_{*i} = \begin{cases} 0, \text{when } a_{ki}\_starttime^{t-1} > t \\ (t - a_{ki}\_starttime^{t-1})/time_{ki}^{S\_T}, \text{when } a_{ki}\_starttime^{t-1} \leq t < a_{ki}\_starttime^{t-1} + time_{ki}^{S\_T} \end{cases} \quad (12)$$

Specific to those conditions which make dynamic scheduling triggered, each influenced task may have three states, including (a) still in the hypernetwork and has been executed before the time t, (b) still in the hypernetwork and has not been executed yet at the time t, and (c) is deleted from the hypernetwork. For the first kind of state, if the influenced task should be assigned

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2020.3004248, IEEE Transactions on Industrial Informatics

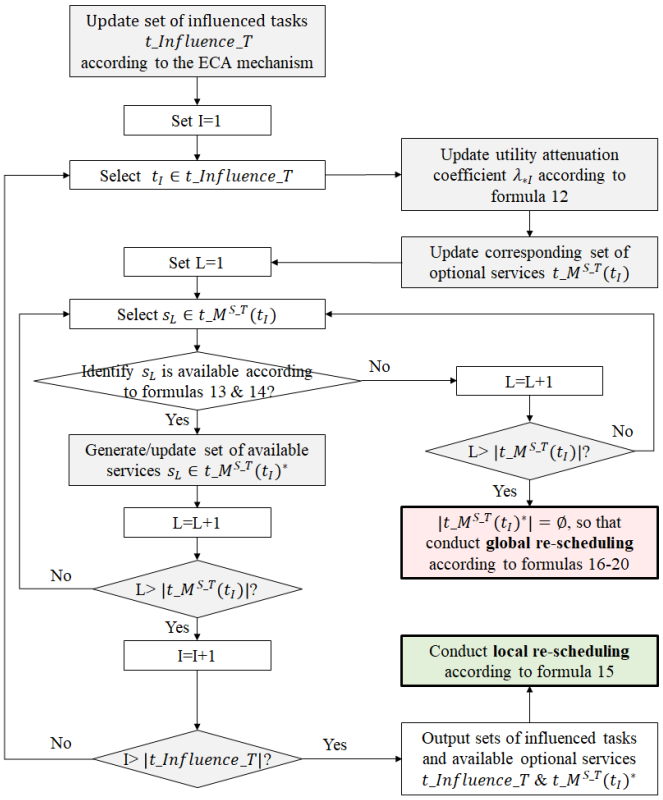


Fig. 5. The algorithm-based approach workflow of local or global re-scheduling in *Stage R*.

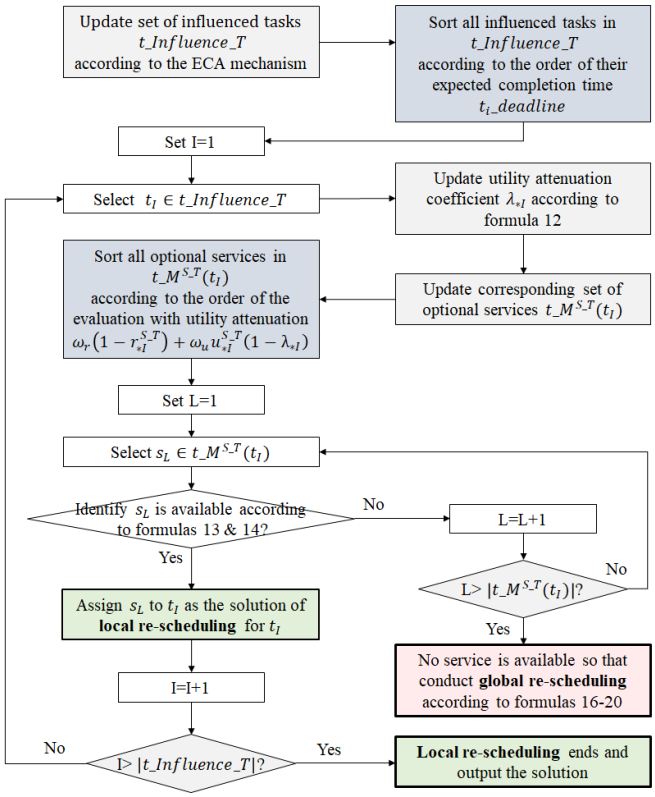


Fig. 6. The priority ordering-based approach workflow of local or global re-scheduling in *Stage R*.

to another service, the utility of previous execution which has been conducted for this task will attenuate. The numbers of tasks with above three states are defined as t_NOT1, t_NOT2 and t_NOT3. When conducting local re-scheduling, there are $t\_NOT1 + t\_NOT2$ tasks either in the first or second kind of states, need to be considered comprehensively. Illustrated in Fig. 5 and 6, here are two comparative workflows in *Stage R* to perform the specific response, local or global re-scheduling, adapting to any dynamic event and specific condition.

To make sure the execution interval of any optional service s_l satisfies the executable time interval of the influenced task t_i, the identification on whether s_l is available responding to local re-scheduling for t_i is conducted by formula 13. The judgement on whether there is $a_{li}\_starttime^t \geq t$ making formula 14 is established for any existing execution time assigned for s_l is also need to be considered. The objective of local re-scheduling is shown in formula 15.

$$t_i\_arrivaltime \leq a_{li}\_starttime < a_{li}\_starttime + time_{li}^{S\_T} \leq t_i\_deadline \tag{13}$$

$$[a_{li}\_starttime^t, a_{li}\_starttime^t + time_{li}^{S\_T}] \cap [a_{lj}\_starttime^{t-1}, a_{lj}\_starttime^{t-1} + time_{lj}^{S\_T}] = \emptyset \tag{14}$$

$$\max \left(\begin{matrix} \sum_{i=1}^{t\_NOT1} w_r(1-r_{li}^{S\_T}) + \sum_{i=1}^{t\_NOT1} w_u u_{ki}^{S\_T}(1-\lambda_{li}) \\ + \sum_{i=1}^{t\_NOT2} w_r(1-r_{li}^{S\_T}) + \sum_{i=1}^{t\_NOT2} w_u u_{ki}^{S\_T} \end{matrix} \right) \Big/ (t\_NOT1 + t\_NOT2) \tag{15}$$

$$\max \sum_{en_p=1}^{NoE} \frac{\left(\begin{matrix} \omega_r \left(\sum_{i=1}^{|t\_T|} \sum_{k_p}^{|t\_en_p\_S|} {a_{k_pi}}^t \left(1 - r_{k_pi}^{S\_T}\right) \right) \\ +\omega_u \left(\sum_{i=1}^{|t\_T|} \sum_{k_p}^{|t\_en_p\_S|} {a_{k_pi}}^t u_{k_pi}^{S\_T} \left(1 - \lambda_{k_pi}\right) \right) \end{matrix} \right)}{\sum_{i=1}^{|t\_T|} \sum_{k_p}^{|t\_en_p\_S|} {a_{k_pi}}^t} \Big/ NoE \tag{16}$$

Subject to

$$\{{a_{ki}}^t\}_{|t\_S| \times |t\_T|} \subseteq t\_E^{S\_T} \tag{17}$$

$$t_i\_arrivaltime \leq a_{ki}\_starttime^t < a_{ki}\_starttime^t + time_{ki}^{S\_T} \leq t_i\_deadline \tag{18}$$

$$\sum_{k=1}^{|t\_S|} {a_{ki}}^t = 1 \tag{19}$$

$$[a_{ki}\_starttime^t, a_{ki}\_starttime^t + time_{ki}^{S\_T}] \cap [a_{kj}\_starttime^t, a_{kj}\_starttime^t + time_{kj}^{S\_T}] = \emptyset \tag{20}$$

When it is impossible to perform local re-scheduling, global re-scheduling is required. It is equivalent to solve the pre-scheduling problem which are illustrated in formulas 6-11, but with the updated models at the time t. Consequently, the decision coefficients at the time t are $\langle {a_{ki}}^t, a_{ki}\_starttime^t \rangle$, and the objective and constraints of global re-scheduling are shown in formula 16 and formulas 17-20.

V. Experiments and Results

A simulation system for the operation of industrial Internet platforms with dual dynamic uncertainties is developed. It is coded in Matlab and implemented on a PC with a 2.60-GHz i7-9750H CPU, 8.00GB of RAM and Windows 10 of 64 bits.

A. Experiments setup and algorithm design

1) Experiments setup

Toward the dual dynamic uncertainties of industrial Internet platforms, the experiment conditions are set as below.

(1) When $t = 0$, for each enterprise, its number of services and tasks are uncertain. The original arrival time and deadline of each existing service are fixed, and the arrival time and deadline of a task are uncertain and generated at initial time. Whether a hyper-edge exists is also uncertain. The parameters of a hyper-edge, such as $u_{ki}^{S\_T}$ and $r_{ki}^{S\_T}$, are generated if the

hyper-edge exists at initial time. In addition, other notations such as $time_{ki}^{S\_T}$, r_{sk} and r_{ti} are generated at the arrival time of hyper-edges, services and tasks. These parameters need to be randomly generated. The detailed parameters setting is listed in Table 3. There are totally 105 services and 82 tasks randomly generated and distributed to 8 enterprises and 5729 hyper-edges existed between those services and tasks, namely $NoS = 105$, $NoT = 82$, and $|E^{S\_T}| = 5729$.

TABLE III
THE PARAMETERS SETTING

Parameters	Values or ranges
N	50
NoE	8
$s_k\_arrivaltime$	0
$s_k\_deadline$	50
$R_s, R_t, R_{S\_T}, R_s^*, R_t^*, R_{S\_T}^*, w_r, w_u$	0.5
en_p_NoS	[10,15]
en_p_NoT	[8,12]
$t_i\_arrivaltime$	[0,35]
$t_i\_deadline$	$t_i\_arrivaltime$ + [10,15]
$e_{ki}^{S\_T}\_arrivaltime$	max $(s_k\_arrivaltime, t_i\_arrivaltime)$
$e_{ki}^{S\_T}\_deadline$	min $(s_k\_deadline, t_i\_deadline)$
$time_{ki}^{S\_T}$	[1,10]
$r_{sk}, r_{ti}, u_{ki}^{S\_T}, r_{ki}^{S\_T}$	[0,1]

(2) During t = 1~N, six kinds of dynamic events listed in Table 2 may happen. If happened, count the numbers of dynamic events with which specific conditions, and judge whether trigger dynamic scheduling. Conduct local or global re-scheduling if necessary. In order to simulate the environment with supply-demand dual dynamic uncertainties, different dynamic events of services, tasks and hyper-edges occurred during the experiment process are generated according to Table 4, including the event details of both services and tasks and the number of changing hyper-edges at t = 11, 21, 31 and 41.

TABLE IV
DYNAMIC EVENTS OF SERVICES, TASKS, AND HYPER-EDGES

t	Occurred E1 of s_k and its recover t $k\langle s_k\_arrivaltime \mid \text{recover } t\rangle$		Occurred E5 of t_i and its recover t $i\langle t_i\_arrivaltime \mid \text{recover } t\rangle$		Number of occurred E7 $\lvert t\_Delete\_E^{S\_T}\rvert$
11	6⟨0|29⟩	11⟨0|15⟩	17⟨10|20⟩		3014
	18⟨0|39⟩	21⟨0|inf⟩	37⟨7|16⟩		
	26⟨0|35⟩	31⟨0|41⟩	38⟨8|40⟩		
	36⟨0|49⟩	38⟨0|23⟩	44⟨2|19⟩		
	39⟨0|20⟩	40⟨0|48⟩			
	44⟨0|38⟩	52⟨0|45⟩			
	72⟨0|34⟩	73⟨0|36⟩			
	77⟨0|25⟩	83⟨0|13⟩			
	93⟨0|29⟩	103⟨0|22⟩			
21	5⟨0|22⟩	11⟨15|39⟩	27⟨15|22⟩		1026
	22⟨0|inf⟩	45⟨0|38⟩	51⟨1|inf⟩		
	49⟨0|44⟩	53⟨0|36⟩	53⟨10|inf⟩		
	57⟨0|inf⟩	58⟨0|32⟩			
	62⟨0|22⟩	63⟨0|37⟩			
	67⟨0|36⟩	75⟨0|40⟩			
	83⟨13|inf⟩	98⟨0|46⟩			
31	2⟨0|44⟩	3⟨0|31⟩	17⟨20|inf⟩		1305
	9⟨0|inf⟩	14⟨0|38⟩	32⟨16|35⟩		
	16⟨0|inf⟩	39⟨20|43⟩	34⟨11|inf⟩		
	46⟨0|36⟩	55⟨0|48⟩			
	62⟨22|40⟩	68⟨0|49⟩			
	69⟨0|49⟩	74⟨0|inf⟩			
	77⟨25|45⟩	87⟨0|48⟩			
	93⟨29|42⟩				
41	1⟨0|47⟩	3⟨31|inf⟩	6⟨4|41⟩	7⟨32|inf⟩	1206
	5⟨22|41⟩	11⟨39|48⟩	8⟨32|inf⟩	37⟨16|42⟩	
	18⟨39|inf⟩	44⟨38|45⟩	39⟨2|inf⟩	46⟨12|42⟩	
	46⟨36|41⟩	50⟨0|49⟩	48⟨24|inf⟩	52⟨33|41⟩	
	53⟨36|43⟩	58⟨32|45⟩	57⟨26|inf⟩	67⟨7|inf⟩	
	61⟨0|44⟩	62⟨40|49⟩	70⟨8|44⟩	76⟨29|44⟩	
	63⟨37|41⟩	67⟨36|47⟩	77⟨17|41⟩		
	75⟨40|43⟩	97⟨0|46⟩			

inf: means the value exceeds the time unit interval considered in experiments, the corresponding nodes or edges are treated as unrecoverable.

2) Algorithm design

The MSs scheduling problem is a kind of complex dynamic multivariate optimization problems. Many algorithms are widely used to solve such problems within a short time, including GA [24, 25], ACO [25], and PSO [26]. For the problem explored in this article, the solution space is large, and the solution scheme changes with dynamic events. The ABC algorithm is an optimization algorithm based on the intelligent behavior of honeybee swarm, which can be used for optimizing multivariable functions [27]. It employs intelligent foraging behavior of honeybees to generate the offspring individuals. While the exploration process carried out by artificial scouts is good for global optimization, the exploitation process managed by artificial onlookers and employed bees is very efficient for local optimization. Compared with other swarm intelligence and population based algorithms such as GA and PSO, the performance of ABC algorithm has the ability to get out of a local minimum and can be efficiently used for multivariable and multimodal function optimization [27].

Therefore, ABC algorithm is quite suitable for the problem which needs to carry out global or local re-scheduling when dynamic events happen. The procedures and ingredients for the problem explored in this article are designed as follows and illustrated as shown in Fig. 7.

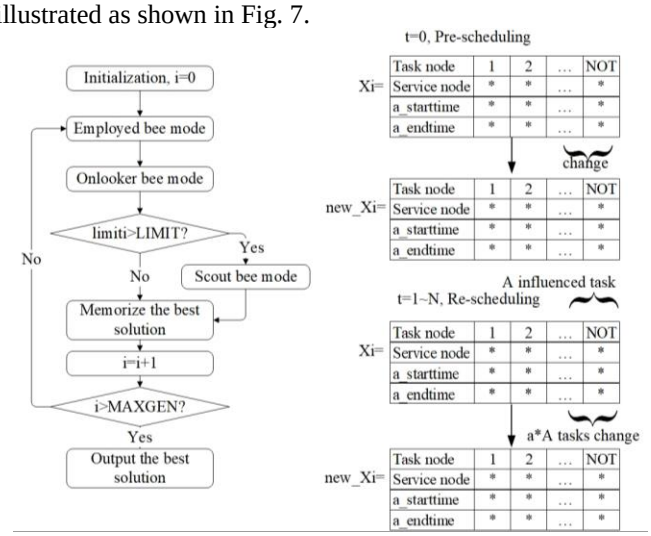


Fig. 7. Structure of ABC algorithm and mapping for the explored problem.

- *Food sources*: it means the candidate solutions. Each one is represented by $\text{Xi} = \begin{bmatrix} a_{*1} & \cdots & a_{*NoT} \\ a\_starttime_{*1} & \cdots & a\_starttime_{*NoT} \\ a\_endtime_{*1} & \cdots & a\_endtime_{*NoT} \end{bmatrix}$, which means the scheduling scheme of NoT tasks in this article. For any task, its corresponding assigned service and the execution time are decided in the scheduling scheme.
- *Nectar*: The nectar amount of a food source corresponds to the fitness of the associated solution, which is an optimization function $f(X)$ with NoT -dimension real-valued decision variables. In this article, the computation expression is shown in formula 1, 16 and 17.
- *Employed bee*: Each employed bee explores and tries to improve one solution Xi by performing an operation on it. In this article, there are many constraints for solutions. The new_Xi must satisfy these constraints. The employed bee chooses $\text{a} * A$ ($\text{a} \in [0,1]$) tasks randomly and changes the scheduling scheme of them each time. The greedy selection

is applied to choose the best solution. If the $f(new\_Xi) \geq f(Xi)$, then the previous one is replaced by the new one. If the solution Xi did not improve for limited time, the employed bee will find a new solution.

- *Onlooker bee*: After employed bee updates its solution, the onlooker decides on where to look for an acceptable food source according to the optimization function $f(X)$. They choose a solution by $p_i = f_i / \sum_{k=1}^{M} f_k$ and update it in the same way of employed bee mode. M is the number of candidate solutions.
- *Scout bee*: As mentioned earlier, in case that the limit is exceeded, a scout bee searches for new solutions randomly, which replaces the exhausted one.
- When $t = 0$, the ultimate solution X is optimized for all submitted tasks; when $t = 1 \sim N$, the solution X is optimized for influenced tasks but keeping others' scheme unchanged.

According to [28], the dynamic multi-objective PSO and dynamic nondominated sorting genetic algorithm II (DNSGA-II) have a mechanism of dealing with dynamism for optimization and DNSGA-II performs very well. These two algorithms are selected to verify the experiments and compare with the experimental results by ABC algorithm.

B. Solutions of pre-scheduling and subsequent re-scheduling

Based on the algorithms of ABC, DNSGA-II and PSO, the running results of pre-scheduling and the subsequent dynamic-event-driven re-scheduling at $t = 11$, 21, 31 and 41 are provided in Fig. 8(a)-(e). As ABC got higher fitness value, the solutions of pre-scheduling and re-scheduling by ABC algorithm are carried out in Fig. 9. For the three-dimensional Gantt chart in Fig. 9(a), the left facade with grey spots reveals the hyper-edges in the initial hypernetwork; the back facade with blank strips depicts the expected executable time intervals of the tasks, which is the scheduling Gantt chart need to be assigned with appropriate services and the corresponding execution period within the expected time interval; the solid columns in space show the selected service and its decided in-execution time for each task in the obtained solution of pre-scheduling. In the other four three-dimensional Gantt charts, the spots on the left facades evolve driven by dynamic events of the corresponding hyper-edges as well as their associated services and tasks, and the blank Gantt charts on the back facades also change if some dynamic events happen on a task.

C. Results discussion and conclusions

1) Comparison on algorithms' convergence

According to the results of both pre-scheduling and re-scheduling by different algorithms, their fitness values and iteration times are collected in Table 5. For the solutions of pre-scheduling and each re-scheduling, the highest values of fitness are carried out by ABC algorithm. In the first two groups of experiments, the values of fitness reduce from pre-scheduling to re-scheduling. During the re-scheduling process from $t = 11$ to 41, the fitness value by ABC algorithm almost keeps around 0.86, the final value by DNSGA-II becomes minimal among three algorithms. As to the solutions by PSO, the obtained values increase from pre-scheduling to re-scheduling, because it still does not achieve the terminal convergence although reaching the iteration threshold.

TABLE V
FITNESS VALUES AND CONVERGENCE ITERATIONS OF DIFFERENT ALGORITHMS

Time units	By ABC		By DNSGA-II		By PSO	
	Fitness	Iterations	Fitness	Iterations	Fitness	Iterations
t=0	0.8891	2603	0.8335	3572	0.7259	2728
t=11	0.8606	3087	0.7840	653	0.7433	1428
t=21	0.8593	525	0.7486	292	0.7486	112
t=31	0.8593	32	0.7380	1	0.7690	422
t=41	0.8595	93	0.7377	1	0.7702	14

As a result, ABC performs better in conducting both pre-scheduling and dynamic-event-driven re-scheduling, because of its higher and especially more stable fitness value adapting to any occurred dynamic events.

2) Comparison on solutions from pre- to re-scheduling

According to the results of pre-scheduling and re-scheduling by above three algorithms, the utility and reliability of each re-scheduling responding to those dynamic events at different time are summarized as t_{11}, t_{21}, t_{31} and t_{41} in Fig. 10(a). The rates of change of both utility and reliability compared to pre-scheduling results are calculated and shown in Fig. 10(b), in which, the results of pre-scheduling by these three algorithms, as the starting points of change, are represented by the origin.

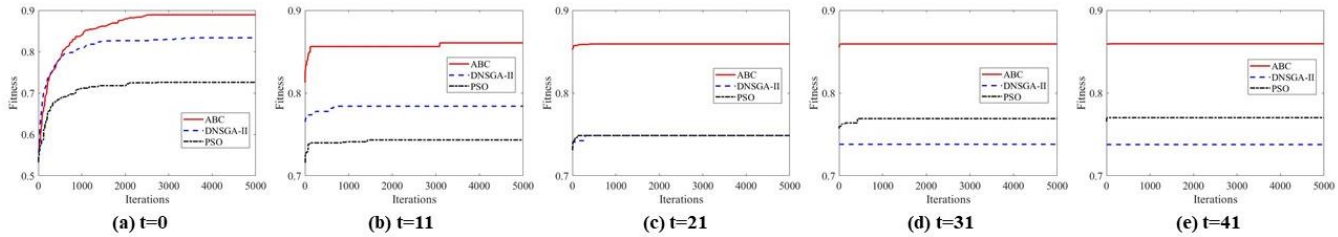


Fig. 8. Running results of pre-scheduling and subsequent re-scheduling.

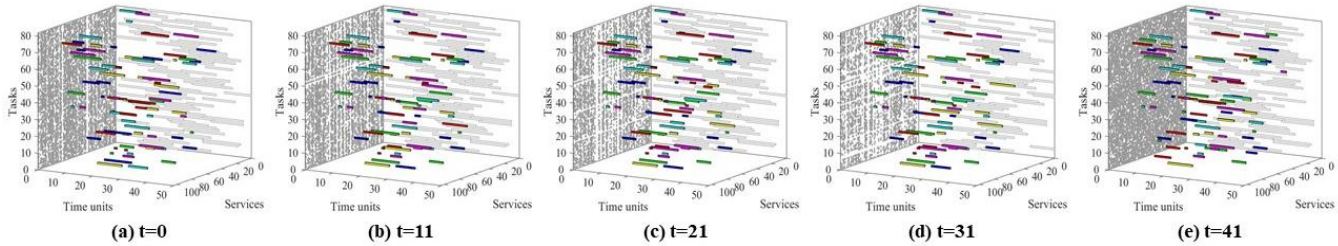
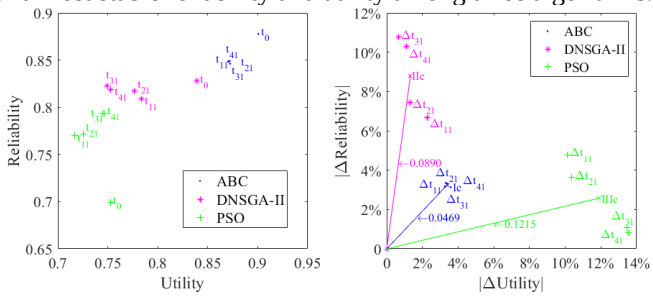


Fig. 9. Scheduling schemes of pre-scheduling and subsequent re-scheduling by ABC algorithm.

In order to analyze the clustering of those four scatters of re-scheduling, the central point of each four scatters for any group of experiments is introduced. For example, the four rates of change by ABC algorithm are marked as Δt_{11}~Δt_{41} in blue and their central point is marked as I_c in Fig.10(b). This central point is calculated by $I_c = (\Delta t_{11} + \Delta t_{21} + \Delta t_{31} + \Delta t_{41})/4$. To discuss the stability of the proposed approach, the distances between the central points of re-scheduling and the origin of pre-scheduling by ABC, DNSGA-II and PSO algorithms are 0.0469, 0.0890 and 0.1215. The distances reflect the deviation of dynamic-event-driven re-scheduling results from the scheme of pre-scheduling. The smaller the distance, the lower the deviation from pre-scheduling to re-scheduling. It reveals that, no matter what dynamic events occur, the proposed approach always has the effectiveness to cope with dynamic uncertainties, and ABC algorithm gives a dynamic response with the highest and most stable reliability and utility among three algorithms.



(a) (b)

Fig. 10. Analysis on solutions from pre-scheduling to re-scheduling.

3) Comparison on dynamic re-scheduling approaches

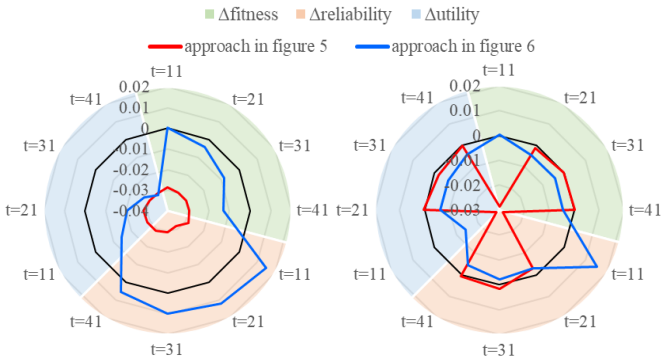
To specifically compare the re-scheduling approaches in Fig. 5 and 6, two sets of re-scheduling experiments are conducted based on the pre-scheduling result obtained by ABC algorithm. The results of pre-scheduling and re-scheduling are collected in Table 6. The values obtained by the approach in Fig. 6 are almost larger than the approach in Fig. 5. For further analyzing these two approaches, the difference of fitness, reliability and utility between each re-scheduling and pre-scheduling, as well as the difference between each re-scheduling and its last scheduling, are displayed in Fig. 11(a) and (b), respectively.

As shown in Fig. 11(a), comparing the values of each re-scheduling by two approaches with the value of pre-scheduling, the fitness values decrease with the occurrences of dynamic events. In particular, the approach in Fig. 6 can obtain re-scheduling schemes that have lower deviation from pre-scheduling by sorting optional services with lower utility attenuation. As to the reliability value, it is interesting that re-scheduling results generated by the approach in Fig. 6 are larger than the value of pre-scheduling. The utility values of both approaches are much smaller than pre-scheduling. One reason is that the utility attenuation of some tasks that have been executed before the influenced time t. However, observed from Fig. 11(b), the approach in Fig. 5 can obtain the values that have lower deviation from each last scheduling except at $t = 11$, no matter the value is fitness, reliability or utility. That is to say, the algorithm-based re-scheduling approach in Fig. 5 performs with better dynamic adaptability.

Based on the above analysis, ABC algorithm is most effective among the three algorithms. The pre-scheduling result by ABC algorithm has the largest fitness value; and the fitness values of dynamic re-scheduling obtained by ABC algorithm

TABLE VI
RESULTS BY DIFFERENT RE-SCHEDULING APPROACHES BASED ON THE PRE-SCHEDULING SOLUTION BY ABC ALGORITHM

Approaches in Figures 5 & 6	Fitness		Reliability		Utility	
t=0	0.8891		0.8777		0.9006	
t=11	0.8606	0.8895	0.8492	0.8929	0.8720	0.8862
t=21	0.8593	0.8849	0.8462	0.8898	0.8724	0.8800
t=31	0.8593	0.8808	0.8480	0.8877	0.8707	0.8738
t=41	0.8595	0.8763	0.8487	0.8831	0.8704	0.8695



(a) (b)

Fig. 11. Analysis on fitness, reliability and utility change by different dynamic re-scheduling approaches.

are less than the priority ordering-based approach, as the algorithm-based re-scheduling approach may fall into local optimal solution. However, the algorithm-based approach is still more effective to handle dynamic-event-driven re-scheduling because of the smaller deviation between each re-scheduling and its last scheme. This is an interesting inference, so that we plan to (a) conduct more experiments in future to prove this inference sufficiently, and (b) further try to find whether there are some specific rules of supply-demand dual dynamic uncertainties for operation of industrial Internet platforms by employing the approach proposed in this article.

REFERENCES

[1] M. Aazam, S Zeadally, and K. A. Harras, "Deploying fog computing in Industrial Internet of Things and Industry 4.0," *IEEE Trans. Ind. Inform.*, vol. 14, no. 10, pp. 4674-4682, 2018.

[2] Y. Cheng, Y. F. Xie, D. Zhao, P. Ji, and F. Tao, "Scalable hypernetwork-based manufacturing services supply demand matching toward industrial Internet platforms," *IEEE Trans. Syst., Man, Cybern., Syst.*, 2019, DOI: 10.1109/TSMC.2019.2944524.

[3] Y. F. Zhang, Z. G. Guo, J. X. Lv, and Y. Liu, "A framework for smart production-logistics systems based on CPS and Industrial IoT," *IEEE Trans. Ind. Inform.*, vol. 14, no. 9, pp. 4019-4032, 2017.

[4] F. Tao, J. F. Cheng, and Q. L. Qi, "IIHub: An Industrial Internet-of-Things Hub toward smart manufacturing based on cyber-physical system," *IEEE Trans. Ind. Inform.*, vol. 14, no. 5, pp. 2271-2280, 2018.

[5] H. Akbaripour, M Houshmand, T. van Woensel, and N. Mutlu, "Cloud manufacturing service selection optimization and scheduling with transportation considerations: mixed-integer programming models," *Int. J. Adv. Manuf. Tech.*, vol. 95, no. 1-4, pp. 43-70, 2018.

[6] Y. K. Liu, X. Xu, L. Zhang, L. Wang, and R. Y. Zhong, "Workload-based multi-task scheduling in cloud manufacturing," *Robot. Com-Int. Manuf.*, vol. 45, pp. 3-20, 2017.

[7] L. F. Zhou, L. Zhang, Y. J. Laili, C. Zhao, and Y. Y. Xiao, "Multi-task scheduling of distributed 3D printing services in cloud manufacturing," *Int. J. Adv. Manuf. Tech.*, vol. 96, no. 9-12, pp. 3003-3017, 2018.

[8] Y. Cao, S. L. Wang, L. Kang, and Y. Gao, "A TQCS-based service selection and scheduling strategy in cloud manufacturing," *Int. J. Adv. Manuf. Tech.*, vol. 82, no. 1-4, pp. 1-17, 2016.

[9] Y. Cheng, F. Tao, Y. L. Liu, D. Zhao, L Zhang, and L. D. Xu, "Energy-aware resource service scheduling based on utility evaluation in cloud manufacturing," *P. I. Mech. Eng. B-J. Eng.*, vol. 227, no. 12, pp. 1901-1915, 2013.

[10] E. C. Balta, Y. K. Lin, K. Barton, D. M. Tilbury, and Z. M. Mao, "Production as a Service: A digital manufacturing framework for optimizing utilization," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 4, pp. 1483-1493, 2018.

[11] Q. Duan, J. Zeng, K. Chakrabarty, and G. Dispoto, "Real-time production scheduler for digital-print-service providers based on a dynamic incremental evolutionary algorithm," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 701-715, 2015.

[12] D. Ouelhadj, and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *J. Scheduling*, vol. 12, no. 4, pp. 417-431, 2009.

[13] L. Liu, H. Y. Gu, and Y. G. Xi, "Robust and stable scheduling of a single machine with random machine breakdowns," International Journal of Advanced Manufacturing Technology, *Int. J. Adv. Manuf. Tech.*, vol. 31, no. 7-8, pp. 645-654, 2007.

[14] J. Branke, and D. Mattfeld, "Anticipation and flexibility in dynamic scheduling," *Int. J. Prod. Res.*, vol. 43, no. 15, pp. 3103-3129, 2005.

[15] D. B. Tang, M. Dai, M. A. Salido, and A. Giret, "Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization," *Comput. Ind.*, vol. 81, pp.82-95, 2016.

[16] C. Tarantilils, C. Kiranoudis, and N. Theodorakopoulos, "A web-based ERP system for business services and supply chain management: application to real-world process scheduling," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 1310-1326, 2008.

[17] S. Jain, and W. Foley, "Dispatching strategies for managing uncertainties in automated manufacturing systems," *Eur. J. Oper. Res.*, vol. 248, no. 1, pp. 328-341, 2016.

[18] R. Zhong, G. Huang, Q. Dai, and T. Zhang, "Mining SOTs and dispatching rules from RFID-enabled real-time shopfloor production data," *J. Intell. Manuf.*, vol. 25, no. 4, pp. 825-843, 2014.

[19] L. Li, and Z. Jiang, "Self-adaptive dynamic scheduling of virtual production systems," *Int. J. Prod. Res.*, vol. 45, no. 9, pp. 1937-1951, 2007.

[20] S. C. Zhang, and T. N. Wong, "Flexible job-shop scheduling/rescheduling in dynamic environment: a hybrid MAS/ACO approach," *Int. J. Prod. Res.*, vol. 55, no. 11, pp. 3173-3196, 2017.

[21] L. F. Zhou, L. Zhang, B. R. Sarker, Y. J. Laili, and L. Ren, "An event-triggered dynamic scheduling method for randomly arriving tasks in cloud manufacturing," *Int. J. Comput. Integ. M.*, vol. 31, no. 3, pp. 318-333, 2018.

[22] Y. Cheng, F. Tao, D. Zhao, and L. Zhang, "Modeling of manufacturing service supply-demand matching hypernetwork in service-oriented manufacturing systems," *Robot. Com-Int. Manuf.*, vol. 45, pp. 59-72, 2017.

[23] C. Zhao, J. S. Li, N. J. Huang, and G. DeCroix, "Optimal planning of plant flexibility: Problem formulation and performance analysis," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 2, pp. 718-731, 2017.

[24] H. Jiang, J. J. Yi, S. L. Chen, and X. M. Zhu, "A multi-objective algorithm for task scheduling and resource allocation in cloud-based disassembly," *J. Manuf. Syst.*, vol. 41, pp. 239-255, 2016.

[25] C. A. Silva, J. M. C. Sousa, and T. A. Runkler, "Rescheduling and optimization of logistic processes using GA and ACO," *Eng. Appl. Artif. Intel.*, vol. 21, no. 3, pp. 343-352, 2008.

[26] Y. H. Jia, W. N. Chen, T. L. Gu, H. X. Zhang, H. Q. Yuan, Y. Lin, W. J. Yu, and J. Zhang, "A dynamic logistic dispatching system with set-based particle swarm optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 9, pp. 1607-1621, 2018.

[27] D. Karaboga, and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459-471, 2007.

[28] S. Y Jiang, and S. X. Yang, "Evolutionary dynamic multiobjective optimization: benchmarks and algorithm comparisons," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 198-211, 2017.

Ying Cheng received the B.S. degree in Mechanical Engineering from Wuhan University of Technology, Wuhan, China in 2010, and received the Ph.D. degree in Control Science and Engineering from Beihang University, Beijing, China in 2016.

She is currently an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. Her research interests include service-oriented smart manufacturing, manufacturing services supply-demand matching and collaboration. She has authored over 30 journal or conference papers in the above areas.

Yifan Xie received the B.S. degree in Automation from Beihang University, Beijing, China in 2019, and is currently pursuing the M.S. degree in Department of Automation, Shanghai Jiao Tong University, Shanghai, China.

Her research interests include manufacturing service supply-demand optimization, and supervisor control of discrete-event system.

Dongxu Wang received the B.S. degree in Aircraft quality and reliability from Beihang University, Beijing, China in 2019, and is currently pursuing the M.S. degree in School of Automation Science and Electrical Engineering, Beihang University, Beijing, China.

His research interests include manufacturing service collaboration optimization and dataspace.

Fei Tao (SM'17) received the B.S. and Ph.D. degrees in Mechanical Engineering from Wuhan University of Technology, Wuhan, China, in 2003 and 2008.

He is currently a Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include service-oriented smart manufacturing, manufacturing service management, and digital twin driven product design/manufacturing/service. He has authored five monographs and over 100 journal papers in the above areas.

Ping Ji received the M.S. degree in Manufacturing Engineering from Beihang University, Beijing, China, and the Ph.D. degree from West Virginia University, Morgantown, West Virginia, USA. He went to National University of Singapore in 1992, and came to The Hong Kong Polytechnic University in 1996.

He is currently a Professor with the Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong, China. His research includes operations management, applied optimization, and information systems.