IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 30, 2019, accepted June 14, 2019, date of publication June 19, 2019, date of current version July 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2923660

A Two-Stage Fault Tolerance Method for Large-Scale Manufacturing Network

YINAN WU[1], GONGZHUANG PENG[2], (Member, IEEE), HONGWEI WANG[3], AND HEMING ZHANG[1]

[1]Department of Automation, Tsinghua University, Beijing 100084, China
[2]Engineering Research Institute, University of Science and Technology Beijing, Beijing 100083, China
[3]ZJU-UIUC Institute, Zhejiang University, Hangzhou 310058, China

Corresponding authors: Hongwei Wang (hongweiwang@zju.edu.cn) and Heming Zhang (hmz@tsinghua.edu.cn)

This work was supported in part by the Zhejiang University/University of Illinois at Urbana–Champaign Institute, led by Principal Supervisor Prof. H. Wang, in part by the National Key R&D Program of China under Grant 2018YFB1701602, and in part by the State Key Laboratory of Intelligent Manufacturing System Technology under Grant QYYE1601.

ABSTRACT Nowadays, networked manufacturing paradigm, such as cloud manufacturing, has brought new challenges for fault tolerance methods. Failures and errors of manufacturing services are unavoidable in the large-scale network. Appropriate fault tolerance methods need to be adopted to improve the reliability of the whole manufacturing network. Aiming at solving this problem, a two-stage fault tolerance method is, thus, proposed. In the off-line stage, a LeaderRank-based manufacturing nodes ranking (LNR) algorithm is put forward to rank the manufacturing services according to their significance in fault tolerance. Replication strategies are employed only for critical services to achieve the tradeoff between fault tolerance effect and fault tolerance loss. In the online stage, an A*-based heuristic alternative path searching (AHPS) algorithm is proposed to find suitable replacement schemes for composite services. The experimental results illustrate that the two-stage fault tolerance method can improve the reliability of large-scale manufacturing networks both effectively and efficiently.

INDEX TERMS Fault tolerance, manufacturing network, nodal importance ranking, A∗ search.

I. INTRODUCTION

In an increasingly competitive market environment, manufacturing industries tend to enhance their own superior resources and collaborate with others. With the development of the Internet of Things (IoT), cloud computing and communication technologies, networked and service-oriented manufacturing paradigms such as manufacturing gird [1] and cloud manufacturing [2]–[5] are more and more widely applied. In these paradigms, a variety of manufacturing resources are encapsulated into services and flexibly composed to meet the demands of customers with lower cost and better performance. In the process of manufacturing service execution, service failure will inevitably occur due to hardware malfunction, excessive load, communication blockage or human factors. Without proper fault tolerance method, multiple manufacturing tasks will fail which may result in great losses. Therefore, how to build a high-reliability manufacturing service networks with effective methods has become an essential and challenging research topic.

The reliability engineering of web service has been studied systematically and deeply. To summarize, there are four types of approaches to improve reliability, which are fault prevention, fault forecasting, fault removal and fault tolerance [6]. However, it is almost impossible to build fault-free systems by using fault prevention and fault removal because of the inaccessible of the underlying structure of web services. Therefore, employing fault tolerance methods become an optimal choice to improve the reliability by masking faults instead of preventing or removing faults. One manufacturing service cannot be shared by multiple manufacturing tasks at the same time, so employing functionally equivalent yet independently designed manufacturing services to tolerant faults is an efficient method, which has been widely applied in the field of software fault tolerance. However, it is difficult and expensive to provide alternative functionally equivalent resource for all manufacturing services. A balanced method is to deploy fault tolerance for critical manufacturing

The associate editor coordinating the review of this manuscript and approving it for publication was Bohui Wang.

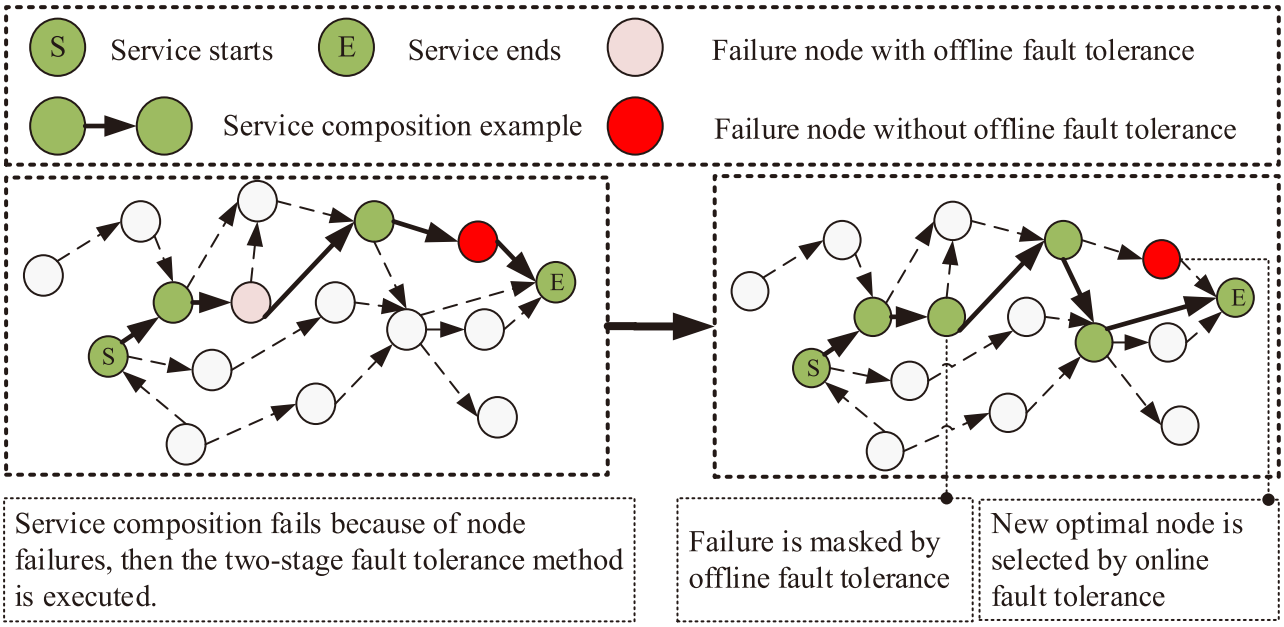

FIGURE 1. A motivating example of two-stage fault tolerance.

services with the redundant resources in the manufacturing network. Thus, how to identify critical services becomes an important issue to be addressed. Traditional approaches identify critical services in web service networks according to the network structure or the reliability properties of service nodes [7], [8], which is not comprehensive in the field of networked manufacturing. The static and dynamic reliability properties and the interactions between related manufacturing services should also be taken into consideration. On the basis above, a LeaderRank based manufacturing node ranking algorithm (LNR) is designed to identify critical services from the complex manufacturing network. After that, the fault tolerance strategy is deployed on the critical services to achieve the trade-off between fault tolerance effect and fault tolerance loss during the offline stage.

In the actual production environment, there is still the possibility of failures after the offline fault tolerance because of the neglect of non-critical services. However, few studies have focused on the online fault tolerance method aiming at masking faults of non-critical services. Therefore, another major aim of our work is to design an effective method to find an alternative scheme for failure services during the online stage to further improve the reliability of the manufacturing network. From the perspective of service composition in manufacturing networks, the problem is to find an alternative sequence from the precursor node to the end node of the failure service, which should satisfy the functional and non-functional (Quality of Service, QoS) constraints at the same time. This problem can be modeled as a Multi-constrained optimal path selection (MCOP) problem [9]. An A* based heuristic alternative path searching (AHPS) algorithm is proposed in our work to solve the MCOP problem in the manufacturing network.

The motivation of the two-stage fault tolerance is to balance the fault tolerance effect and cost. By backing up key service nodes in the offline stage, the reliability of critical services is ensured. The fault migration delay can be reduced at the same time by deploying the critical service and the backup in the same location. Through constructing alternative manufacturing path from the shared resource pool in the online stage, resource utilization is increased. Therefore, the offline method is characterized by excellent fault tolerance effect, low delay but low resource utilization. In contrast, the online fault tolerance method has high resource utilization, but the fault migration delay may be large. In order to take advantage of offline fault tolerance and online fault tolerance, the two-stage fault tolerance framework deploys different fault tolerance strategies for different manufacturing services in order to achieve the trade-off between resource utilization and fault-tolerant delay.

From what has been discussed above, a two-stage fault tolerance method is proposed, which is illustrated in figure 1. This method combines the offline fault-tolerant stage and the online fault-tolerant stage, aiming to further improve the overall reliability of the manufacturing network. The rest of the paper describes the method in detail, which is organized as follows: Section II is a literature review of related work. Section III addresses formal descriptions and mathematical model of this problem. Section IV proposes a novel two-stage fault tolerance method to improve the reliability and robustness of manufacturing networks. Multiple experiments are performed and discussed in Section V to verify the

effectiveness of the proposed method. Section VI provides a conclusion and indicates some directions for further research.

II. RELATED WORK

In recent decades, numerous fault tolerance methods are studied in the field of web service, which can be divided into reactive methods and proactive methods [10]. The two-stage fault tolerance method belongs to reactive methods, which contains checking points [11], [12], replication [7], [8], [13]–[15], retry [16], task resubmission, N-version [17] and so on. Unlike the characteristics of web service, the nodes represent physical manufacturing units in complex manufacturing networks; the intermediates during the manufacturing process are also physical products. Moreover, faults in manufacturing networks are usually caused by hardware malfunction or other factors. So reactive methods which are widely applied in web service are inappropriate in the manufacturing environment. This is not only because the hardware malfunction cannot be recovered through re-execution, but also that the above methods will scrap the half-finished products. Therefore, it is appropriate to employ replication strategy based on redundant resources in manufacturing networks.

In order to achieve a better fault tolerance effect with less loss, it is practicable to deploy replication methods only for critical manufacturing services. A new approach is therefore needed for identifying critical services in the manufacturing network. Much research has been done on node ranking in cloud computing networks [7], [8], [18]–[21] in order to select the important nodes to deploy offline fault tolerance. Inspired by PageRank and SPARS-J [22], two node ranking methods named FTCloud and FTCLoud2 are proposed in [7] which take into account the network structure and the importance of the web services. Qiu et al. improved the node ranking model based on FTCloud and proposed ROCloud [8]. Wu et al. proposed the FSCRank framework to evaluate the importance of cloud applications, which introduced the buffer node to accelerate algorithm convergence [23]. Aiming at solving node ranking problem in manufacturing networks, Wu et al. proposed a fault tolerance framework LIVMA based on LeaderRank algorithm [21]. All these work plays an important role in the node ranking problem in complex networks. However, unlike the regular network, the reliability problem of the manufacturing network is characterized by the following features: (1) the static reliability of the manufacturing service itself (reliability of the equipment, reliability of the process and so on) need to be extracted and modeled. (2) The importance degree between different nodes varies with the indicators such as the cooperation relationship and the logistic cost. The work mentioned above only focuses on the failure rate and input, output degree of the node when designing the importance transfer matrix. In the field of manufacturing, the importance of the services involves more factors, such as the static reliability, online time and so on. Thus, how to extract and model the reliability characteristics of complex manufacturing networks, and how to implement the ranking of manufacturing services based on node ranking algorithm in complex networks still remain to be addressed.

The method of online fault tolerance can be viewed as the dynamic replacement of service composition. Many scholars and experts have made efforts in finding optimal service compositions by meta-heuristic algorithms, such as genetic algorithms (GA) [24]–[26], artificial bee colony algorithms (ABC) [27], [28] and particle swarm optimization (PSO) [29]. These studies only consider the characteristics of the services, but do not take into account the relationship between services when building the online fault-tolerance service composition model. Other experts try to find the optimal solution by solving the MCOP problem [9], [30]. Heuristic Optimal Social Trust Path (H_OSTP) algorithm [31] and Multiple Foreseen Path-Based Heuristic Optimal Social Trust Path (MFPB_HOSTP) algorithm [32] are two of the best algorithms among them in both solution quality and algorithm efficiency [32]. However, the H_OSTP method suffers from the imbalance problem [32], the solutions found by the method are not near optimal sometimes. Moreover, the complexity of the MFPB_HOSTP method is related to the dimensions of QoS [32]. Therefore, when the network scale or the QoS dimension is large, the complexity of MFPB_HOSTP becomes unbearable. Aiming at solving these problems, an improved path selection algorithm (AHPS) is developed to complete the online fault tolerance process.

III. MODELING OF RELIABILITY IN THE MANUFACTURING NETWORK

A. MANUFACTURING SERVICE NETWORK MODEL

In the manufacturing network, the service providers share the idle manufacturing capacities and encapsulate them into multi-functional manufacturing services through platform middleware. A large number of manufacturing services form a complex manufacturing service network. Nodes in complex networks represent manufacturing services with functional attributes and non-functional attributes. There are different non-functional indexes among manufacturing service nodes, such as cooperation tendency, logistics cost, and other factors. If there is a cooperative relationship between two manufacturing service nodes, the two nodes satisfy the constraints such as process constraints, logistics cost constraints, and so on. In this case, a directed arrow is used to connect the two nodes. When the manufacturing task comes, a single manufacturing service node or the service composition composed of several service nodes is invoked to meet the functional and non-functional requirements of the task.

B. RELIABILITY MODELING

1) STATIC RELIABILITY MODELING

Static reliability refers to the influence of equipment attributes, manufacturing capability and other indicators of the service itself on its ability to complete manufacturing tasks. The static reliability of the manufacturing service includes the equipment reliability and the process reliability.

a: RELIABILITY OF THE MANUFACTURING EQUIPMENT

The reliability of manufacturing equipment refers to the ability of production equipment to successfully execute manufacturing tasks. The reliability of manufacturing equipment can be modeled by the degradation index, which is shown in (1).

$$D_{EG} = 1 - e^{-\gamma \times (T_{MA} + T_{ON})} \tag{1}$$

γ is the degradation parameter of manufacturing equipment. T_{MA} is the time interval between the last maintenance and the current time. T_{ON} is the online time span of the equipment.D_{EG} refers to the degree of machine degradation, which is negatively correlated with machine reliability. The derivation of D_{EG} is based on the state-based machine maintenance model [33], which is a popular model in related studies. The model estimates the remaining state of the machine by the equation $h = I \times e^{\gamma_1 \times x_1 + \gamma_2 \times x_2 + \cdots + \gamma_n \times x_n}$. I is the initial state. γ_i is the coefficient and x_i is the influence factor. The negative exponential relationship between the machine degeneration degree and time is also illustrated in [34]. Taking the influence of time (T_{MA} and T_{ON}) into consideration, $D_{EG} = \frac{1}{I} \times \left(I - I \times e^{-\gamma \times (T_{MA} + T_{ON})}\right) = 1 - e^{-\gamma \times (T_{MA} + T_{ON})}$

b: RELIABILITY OF THE PROCESS

The process reliability refers to the possibility that a manufacturing service composition can produce a qualified product under the constraints of various processes. The process reliability is modeled by (2).

$$P_R = \frac{A_{cc}}{P_n \times S_n} \times e^{-\lambda t} \tag{2}$$

The process reliability model is designed based on the machine degeneration model [33].A_{cc} is the process accuracy constraint. When the constraint is strict (A_{cc} is small), the possibility of failure increases. So A_{cc} is positively correlated with the reliability. P_n is the number of links in the manufacturing process. S_n is the number of types of process structures. Both of them represent the complexity of the process. Complex processes lead to higher failure rates. Therefore, the two parameters are negatively correlated with process reliability. $e^{-\lambda t}$ is the time weighted index and λ is the coefficient of time. It represents the degradation degree of process reliability with time, and the derivation is similar to equation (1).

2) DYNAMIC RELIABILITY MODELING

The actual failure data of manufacturing services can be collected in the process of manufacturing network execution. Dynamic reliability can be modeled by failure probability and failure effect based on the collected failure data.

a: FAILURE PROBABILITY

The failure probability represents the possibility that the failure will occur when the manufacturing service executes a manufacturing task with a duration of t. The failure probability of the manufacturing service needs to be determined before the critical node in the complex networks is identified. The failure probability $F_i(t)$ of node i during the period of t is given by (3) and (4).

$$F_i(t) = 1 - e^{-\delta_i \times t} \tag{3}$$

$$\delta_i = \frac{-n \times ln\left(1 - \sum_{j=1}^{n} f\left(t_{i,j}\right) / \left(n \times e_{exec}\left(t_{i,j}\right)\right)\right)}{\sum_{j=1}^{n} t_{i,j}} \tag{4}$$

The time-dependent function of failure probability is introduced based on the research [33]. The failure probability is represented as (3). δ_i is the failure intensity of node i. It can be derived from the average failure probability of the node during n time periods. $f\left(t_{i,j}\right)$ represents the number of failures of node i during the period of $t_{i,j}$. $e_{exec}\left(t_{i,j}\right)$ represents the number of times node i has been executed during the period of $t_{i,j}$. The detailed derivation process is $\frac{1}{n} \times \sum_{j=1}^{n} \frac{f\left(t_{i,j}\right)}{e_{exec}\left(t_{i,j}\right)} = 1 - e^{-\delta_i \times \frac{\sum_{j=1}^{n} t_{i,j}}{n}}$. Then Equation (4) can be obtained from the equation above.

b: FAILURE EFFECT

In manufacturing networks, different manufacturing services have different degrees of importance due to their different functional and non-functional attributes. When a more critical manufacturing service fails, more service compositions may fail because of it. Equation (5) gives the failure effect of manufacturing nodes.

$$E_{ff}^{i} = \frac{\sum_{j=1}^{m_i} B\left(\frac{f_{i,j}}{f_{i,j} + s_{i,j}} > Threshold\right)}{m_i} \tag{5}$$

E_{ff}^{i} is a measure of the impact on the manufacturing network when node i fails. Given a **Threshold**, when node i fails, if the proportion of the failed service compositions ($f_{i,j}$) in the total service compositions ($f_{i,j} + s_{i,j}$) is greater than the **Threshold**, then the node failure is recorded ($B\left(\frac{f_{i,j}}{f_{i,j}+s_{i,j}} > Threshold\right) = 1$), otherwise, it is not recorded ($B\left(\frac{f_{i,j}}{f_{i,j}+s_{i,j}} > Threshold\right) = 0$). When node i fails for m_i times, the ratio of the number of failures recorded to m_i can represent the failure effect of node i. $B\left(\frac{f_{i,j}}{f_{i,j}+s_{i,j}} > Threshold\right)$ is a Boolean function, which is detailed in (6).

$$B\left(\frac{f_{i,j}}{f_{i,j} + s_{i,j}} > Threshold\right) = \begin{cases} 1 & when \ \frac{f_{i,j}}{f_{i,j} + s_{i,j}} > Threshold \ \ is \ \ true \\ 0 & when \ \frac{f_{i,j}}{f_{i,j} + s_{i,j}} > Threshold \ \ is \ false \end{cases} \tag{6}$$

c: IMPORTANCE TRANSMISSION MODELING

In the process of importance transferring of a node, the close connection between two adjacent manufacturing nodes is an important factor affecting the weight of the transfer. The weight calculation method of the directed edge between node i and j is given by (7).

$$w_{i,j}^{edge} = \frac{C_{i,j}}{L_{i,j}} \tag{7}$$

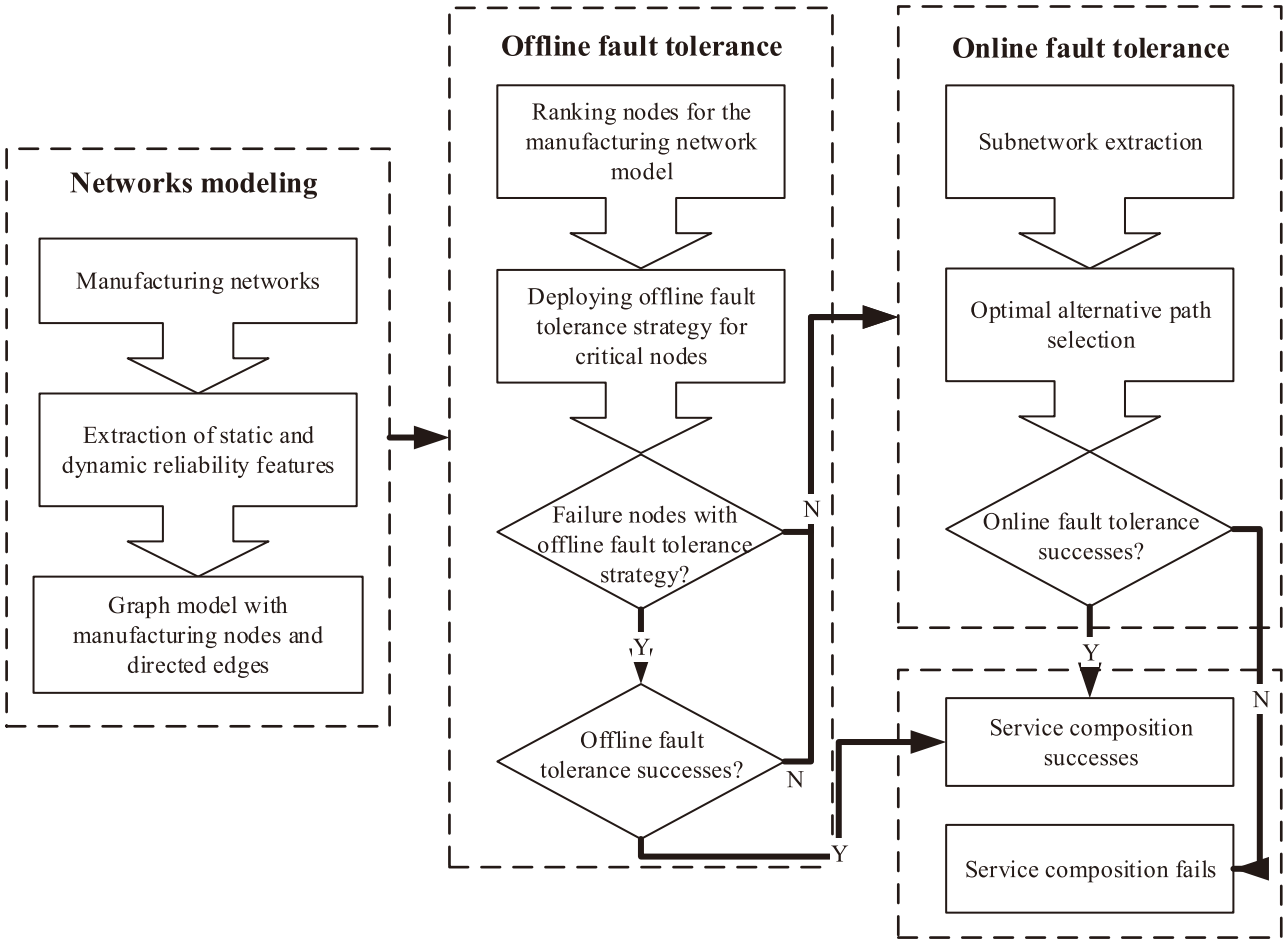

FIGURE 2. Two-stage fault tolerance framework.

$C_{i,j}$ is the number of invocations from node i to node j. $L_{i,j}$ represents the logistics cost from node i to node j. The tightness of the relationship between nodes is positively correlated with the number of invocations between nodes and negatively correlated with the logistics cost between nodes. Therefore, the tightness of the relationship between nodes can be expressed as the ratio of $C_{i,j}$ and $L_{i,j}$.

IV. TWO-STAGE FAULT TOLERANCE METHOD

A. FAULT TOLERANCE FRAMEWORK

The two-stage fault tolerance framework consists of offline fault tolerance and online fault tolerance stage, which is shown in figure 2. The whole process of the framework consists of three steps.

1) FAILURE DETECTION

In the process of manufacturing service execution, if the status, consistency and QoS constraints of the nodes are not satisfied, the fault-tolerant strategy will be executed. Many fault detection and diagnosis technologies have been studied deeply to identify the failure nodes in the manufacturing process. Among these data-driven process monitoring methods are effective methods to detect failure nodes. Partial Least-Square (PLS) [35], [36] and Principal Component Analysis (PCA) [37] can be used to detect faults for Gaussian process data. In addition, the data of many industrial processes are non-Gaussian, and Independent Component Analysis (ICA) [38] can be employed to detect failure nodes in these processes. Data of actual industrial processes may contain non-linear components, kernel tricks can handle the problem well, such as Kernel PLS (KPLS) [39], Kernel PCA (KPCA) [39] and Kernel ICA(KICA) [40]. If the industrial process has dynamic characteristics, dynamic PCA [41] performs well in detecting the failure nodes. Therefore, the deployment of appropriate and effective fault detection methods for different manufacturing services can detect fault nodes in a timely manner and lay a good foundation for the two-stage fault-tolerant process.

2) OFFLINE FAULT TOLERANCE

In the offline stage, a complete fault processing process is arranged for the manufacturing services according to the functional and non-functional attributes of the manufacturing service. According to node importance ranking method,

critical manufacturing nodes with high invocation frequency, low execution success rate or large impact of failure are identified. The republication method is deployed for critical nodes to achieve the trade-off of fault tolerance effect and fault tolerance loss.

3) ONLINE FAULT TOLERANCE

Online fault tolerance method is needed to further enhance the reliability of manufacturing networks during the execution of the manufacturing process because some non-critical nodes without offline fault tolerance strategy may also fail. By extracting the sub-network from the fault node to the end node of the service, the online fault tolerance process looks for the optimal alternative plan to complete the manufacturing task according to the QoS constraints.

B. OFFLINE FAULT TOLERANCE METHOD

The offline stage is designed to deploy a stable, efficient and reliable fault tolerance scheme for critical services with a high failure rate through redundant resources. That is, manufacturing services that provide offline fault tolerance for critical nodes will always be in the standby state, and will not participate in other service compositions. The resource sharing strategy is not applicable at the offline stage because faults occur frequently or have a large impact on the network. It is more adequate to deploy low-latency backups to tolerant faults. The offline method reduces the fault tolerance delay and improves the fault tolerance effect at the cost of reducing resource utilization. Therefore, this method is only suitable for deployment to important nodes in the manufacturing network.

The most critical step of offline fault tolerance is to identify the critical nodes in the manufacturing networks. In this section, the nodes in the manufacturing network are abstracted and analyzed, and the ontological features, location features, interaction features and other feature information of the nodes are extracted. The manufacturing nodes ranking algorithm (LNR) is proposed based on the LeaderRank algorithm. According to the LNR algorithm, the critical nodes which have a great impact on the overall reliability of the manufacturing networks are identified and selected to deploy the fault tolerance strategy. According to the static and dynamic reliability indexes, the LNR algorithm is designed based on LeaderRank algorithm. The LeaderRank based manufacturing nodes ranking algorithm is given by Algorithm 1.

The critical step of LNR algorithm is to design the iterative matrix and the background node, which determines the effect and convergence of the algorithm. The design of iterative matrix has been studied in [7], [8]and [23]. In FTCloud [7], the iterative matrix only takes the network structure into consideration. At the same time, the convergence of the algorithm is guaranteed by the relaxation coefficient d based on the PageRank algorithm [42]. In ROCloud [8], the failure rates of the nodes are taken into consideration, which is more accurate and effective. In FSCRank [23], the buffer node is introduced to accelerate the convergence of the algorithm. However, the iterative matrix does not consider static reliability and transmission weights. Compared with these methods, the iterative matrix proposed in this paper considers more relative factors. The background node can not only ensure the convergence of the LNR algorithm but also stabilize the importance of critical nodes.

Algorithm 1 The LNR Algorithm

Input: Iterative matrix M^{iter}.

Output: Nodes' importance $\mathrm{V}_i(s)$.

1: Assign the random initial importance value $\mathrm{V}_i(0)$ (i=1,2, ...n+1)
2: **Loop:**
3: Update: $V_i(t) = \sum_{j=1}^{n+1} \frac{M_{j,i}^{iter}}{\sum_{k=1}^{n+1} M_{j,k}^{iter}} \times V_i(t-1)$
4: **Until**
5: For a very small e, $|V_i(t) - V_i(t-1)| < e$
6: Assign the importance value of $\mathrm{V}_{n+1}(s)$ to other nodes according to:
7: $V_i(s) = V_i(s) + \frac{M_{n+1,i}^{iter}}{\sum_{k=1}^{n} M_{n+1,k}} \times V_{n+1}(s)$
8: **return** $V_i(s)$, i=1,2...n+1

1) ITERATIVE MATRIX

The static and dynamic reliability of manufacturing nodes should be considered when designing the iterative matrix. Moreover, the transfer of reliability between nodes should also consider the transfer weight of the directed edge between nodes. Assuming that there are n nodes in the manufacturing networks, the iterative matrix M^{iter} is a square matrix with n dimensions. The iterative matrix M^{iter} is given by (8).

$$M_{i,j}^{iter} = w_{i,j}^{edge} \times (w_1 \times \frac{D_{EG}^i}{P_R^i} + w_2 \times \sum_{k=1}^{m} \frac{F_i(t_k)}{k} \times E_{ff}^i) \tag{8}$$

M^{iter} is the transition probability matrix. $M_{i,j}^{iter}$ represents the probability and proportion of importance transfer from node i to node j. Therefore, it is negatively correlated with the reliability of node i and positively correlated with the cooperation trend between node i and j. When designing the iterative matrix, $M_{i,j}^{iter}$ is proportional to $w_{i,j}^{edge}$. D_{EG}^i refers to the degree of machine degradation, which is negatively correlated with machine reliability and positively correlated with $M_{i,j}^{iter}$. Similarly, the process reliability P_R^i is negatively correlated with $M_{i,j}^{iter}$. $\sum_{k=1}^{m} \frac{F_i(t_k)}{k}$ represents the average failure rate of the manufacturing service in k manufacturing processes. E_{ff}^i represents the influence range on the manufacturing network when node i fails. Both of them are positively correlated with $M_{i,j}^{iter}$. Therefore, the importance of a single manufacturing service can be represented as (8).

2) BACKGROUND NODE

The convergence condition of the LNR algorithm is that the manufacturing networks have strong connectivity.

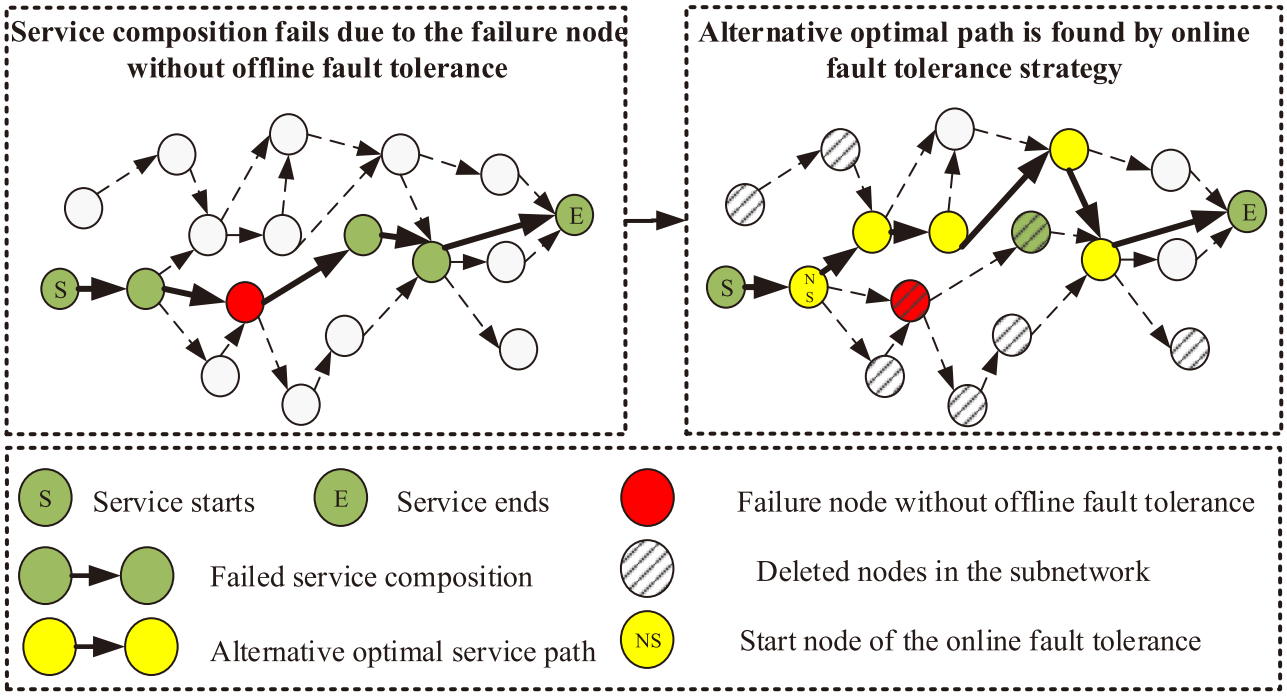

FIGURE 3. The process of online fault tolerance.

A background node n+1 is needed to ensure the convergence of the algorithm. It collects and redistributes the importance indicators according to the node attributes and edge attributes of each manufacturing node to prevent the malicious accumulation of importance of nodes with output degree of zero. All nodes have the same probability to visit the background node, and the nodes with high iteration coefficient in the manufacturing node have a higher probability to be visited by the background node, so the iteration coefficient of the background node is given by (9).

$$M_{i,n+1}^{iter} = \frac{1}{n}, \quad M_{n+1,i}^{iter} = \frac{\sum_{i=1}^{n} M_{i,j}^{iter}}{\sum_{i=1}^{n} B(M_{i,j}^{iter} \neq 0)} \tag{9}$$

The design of the background node is also an essential part of the LNR algorithm [43], [44]. It extends the iterative matrix from n dimension to n+1 dimension. The background node guarantees the strong connectivity of the manufacturing network so that it ensures the convergence of the LNR algorithm. The proof of convergence is discussed in detail in Section IV, part D. The design of the background node also influences the effectiveness of the LNR algorithm. The design method follows two strategies. The strategies can further consolidate the importance of important nodes through the background node.

- The importance of all nodes is transmitted to the background nodes with equal proportion.
- The background node transfers a greater proportion of importance to nodes that are more important.

C. ONLINE FAULT TOLERANCE METHOD

Deployment of offline fault tolerance strategy for important manufacturing nodes in manufacturing networks can improve the overall reliability of the manufacturing network, and also achieve the trade-off of fault tolerance effect and fault tolerance loss. However, in the actual production environment, there is still the possibility of failures after offline fault tolerance deployment, which can not be ignored. Therefore, the online fault-tolerant strategy is adopted to find alternative service compositions for the manufacturing task and meet the demand of the manufacturing task and QoS constraints in order to achieve reliability improvement. This method completes the fault tolerance process by sharing the functionally equivalent yet independently designed manufacturing services, which can improve resource utilization and avoid causing the redundancy of the manufacturing network. The online fault tolerance process is shown in figure 3.

The key problem to find a replacement service composition is to find another path in the subnetwork that can satisfy the QoS constraints of the manufacturing task. Therefore, the search for alternative paths is a multiple constraint optimization path (MCOP) selection problem, which is NP-complete [32].

In Cloud Computing, the QoS model, which contains the response time, network load, bandwidth and so on, are widely studied in recent researches. In the Cloud Manufacturing environment, the QoS attributes that should be particularly concerned are very different from those in Cloud Computing. In the process of online fault tolerance, the system delay,

manufacturing time, service cost and service availability are significant QoS attributes, which play important roles in selecting the alternative scheme. In addition, the reputation, sustainability, environmental protection degree and other attributes are also important QoS indexes that can be taken into account. Without loss of generality, the migration delay, manufacturing time, service cost and service availability are chosen as the representative QoS attributes, which are applied to the AHPS algorithm after modeling. The modeling method for the QoS metrics is described in detail below.

1) SYSTEM DELAY

For the failed manufacturing service, system delay includes the fault detection delay (D_{fd}) and fault migration delay (D_{fm}). For a single manufacturing service i, the system delay includes software response delay (D_i^{sw}) and hardware response delay (D_i^{hw}). In addition, there is also the network communication delay ($D_{j,j+1}^{com}$) between manufacturing services. Assuming that a service composition contains n services and the calculation method of delay is shown in (10).

$$D_{1\to n} = D_{fd} + D_{fm} + \sum_{i=1}^{n}(D_i^{sw} + D_i^{hw}) + \sum_{j=1}^{n-1} D_{j,j+1}^{com} \tag{10}$$

2) MANUFACTURING TIME

Manufacturing time is the period from the time when the manufacturing task is migrated to the alternative service composition to the time when the manufacturing process ends. This attribute is aggregated by addition between different services. For the manufacturing service (T_i^{sv}), manufacturing time includes process planning time (T_i^{pro}), resource allocation time (T_i^{res}) and manufacturing execution time (T_i^{exe}). For different manufacturing services, there are logistics time and queue time between them. The calculation formula of manufacturing time is shown in (11)

$$\begin{cases} T_{1\to n} = \sum_{i=1}^{n} T_i^{sv} + \sum j = 1^{n-1} T_{j,j+1}^{edge} \\ T_i^{sv} = T_i^{pro} + T_i^{res} + T_i^{exe} \\ T_{j,j+1}^{edge} = T_{j,j+1}^{log} + T_{j,j+1}^{que} \end{cases} \tag{11}$$

3) SERVICE COST

Service cost is the cost required to complete the fault migration process and employ the alternative service composition to complete the manufacturing task. It consists of fault migration cost(C_{fm}), computing cost (C_i^{cpt}), material cost (C_i^{mat}), labor cost (C_i^{lab}) and depreciation cost (C_i^{dep}). Moreover, the manufacturing task involves the transportation of semi-finished products ($C_{j,j+1}^{log}$) and the delivery of finished products ($C_{j,j+1}^{del}$). Therefore, there are logistic cost and delivery cost. To summarize, the calculation method of service cost is given in (12).

$$\begin{cases} C_{1\to n} = C_{fm} + \sum_{i=1}^{n} C_i^{sv} + \sum_{j=1}^{n-1} C_{j,j+1}^{edge} \\ C_i^{sv} = C_i^{cpt} + C_i^{mat} + C_i^{lab} + C_i^{dep} \\ C_{j,j+1}^{edge} = C_{j,j+1}^{log} + C_{j,j+1}^{del} \end{cases} \tag{12}$$

4) SERVICE AVAILABILITY

Service availability refers to the probability that the manufacturing service can be executed normally at a certain inspection time. Since the original service composition has failed, the manufacturing products may not be delivered on schedule if the alternative service composition becomes unavailable. Therefore, service availability is a critical attribute in the online fault tolerance stage. The exponentially weighted moving average (EWMA) [45] method can be adopted to calculate the availability of a single service in the current state based on the historically available data obtained during the service execution. The availability of a service composition can be expressed as a product of the availability of single services. Assuming that the data of n services in m periods are collected. U_i^j represents the number of times that the service i is available in the jth period. E_i^j represents the number of times that service i executes in the jth period. The available of the service composition can be calculated by (13)

$$\begin{cases} A_{1\to n} = \prod_{i=1}^{n} A_i^{m+1} \\ A_i^{j+1} = \frac{\beta \times A_i^j + (1-\beta)\times a_i^{j+1}}{1-\beta^j} \\ A_i^1 = 0, \quad a_i^j = \frac{U_i^j}{E_i^j} \end{cases} \tag{13}$$

β is usually greater than 0.9. Equation (13) starts from j = 1 and iterates until A_i^{m+1} is calculated.

The target of the online fault tolerance is to find an optimal service composition to replace the failed one and satisfy the functional and non-functional (QoS) constraints at the same time. In the existing studies, the problem can be abstracted as a 0-1 integer constrained multi-objective optimization problem, which is formulated in (14).

$$\begin{cases} \min D_{1\to n}, \quad T_{1\to n}, \ C_{1\to n} \max A_{1\to n} \\ 0 < D_{1\to n} < D_{cons} \\ 0 < T_{1\to n} < T_{cons} \\ 0 < C_{1\to n} < C_{cons} \\ A_{cons} < A_{1\to n} < 1 \end{cases} \tag{14}$$

D_{cons}, T_{cons}, C_{cons} and A_{cons} represents the QoS constraints of D, T, C and A respectively. This problem can be solved by meta-heuristic algorithms, such as GA, PSO, ACO, etc. However, the search process for the alternative service composition is conducted on the manufacturing network. There are more constraints to ensure the existence of edges between the manufacturing services. The problem is converted into an MCOP problem. The additional constraints of the problem is formulated in (15).

$$\begin{cases} \sum_{i=1}^{scs_j} x_{i,j} = 1, \quad j = 1, 2 \ldots m \\ \sum_{i=1}^{scs_j} \sum_{k=1}^{scs_{j+1}} x_{i,j} A_{i,k} x_{k,j+1} = 1 \\ x_{i,j} = 0 \text{ or } 1 \end{cases} \tag{15}$$

In (15), scs_j represents the number of services in the service candidate set j and m is the number of remaining subtasks. The decision variable x_i^j equals 1 if the ith manufacturing

Algorithm 2 A* Based Heuristic Alternative Path Searching (AHPS)

Input: D_{fd}, D_{fm}, D_i^{sw}, D_i^{hw}, $D_{j,j+1}^{com}$, T_i^{sv}, $T_{j,j+1}^{edge}$, C_i^{sv}, $C_{j,j+1}^{edge}$, A_i^{m+1}, $(i = 1, 2, \ldots n,\ j = 1, 2, \ldots n-1)$, D_{cons},T_{cons},C_{cons}, A_{cons}.

Output: Weighted optimal path $P_{s\to e}$, $R(P_{s\to e})$.

1: Assign the initial value $R(P_{s\to e})$=0
2: Find the optimal path from end to node s with the greedy function $G(P_{s\to e})$ (16) as the target
3: **if** $G(P_{s\to e})$>1 **then**
4: **return** *msg*(*Online fault tolerance failed*)
5: **else**
6: Find the optimal path from end to node s with the relaxed function $R(P_{s\to e})$ (17) as the target.
7: Execute the **Forward Search** algorithm to find the near optimal solution $P_{s\to e}$
8: **end if**
9: **return** $P_{s\to e}$ and $R(P_{s\to e})$

service in set j is chosen for the alternative service composition and otherwise it equals 0. A is the adjacency matrix of the manufacturing network. This problem can also be solved by meta-heuristic algorithms. However, as the scale of the manufacturing network increases (scs_j increases), the decision variable constraints become more and more. This leads to the low efficiency of the meta-heuristic algorithms in finding the near-optimal solution, which has been proved in [9]. In order to solve this problem, an A* based heuristic alternative path searching (AHPS) algorithm is proposed in this section. The AHPS algorithm is a more effective algorithm to solve the MCOP problem due to that it can always satisfy the constraints in (15) in the process of execution. In AHPS, the multi-objective optimization problem is transformed into a single-objective optimization problem by linear weighting method, and then the greedy function and relaxation function are solved as heuristic functions to find the near-optimal path. The pseudo-code of the algorithm is shown in Algorithm 2.

The algorithm first conducts two reverse searches on sub-networks with the Dijkstra method. The search targets are the greedy target (16) and the relaxed target (17). After two reverse searches, each node has its own greedy function value and relaxation function value to the end node. Taking these two values as heuristic functions, the forward search is conducted on the sub-network. The pseudo-code of the forward search procedure is shown in Appendix A.

$$G\left(P_{i\to j}\right) = \min\left\{\max\left(\frac{D_{i\to j}}{D_{cons}}, \frac{T_{i\to j}}{T_{cons}}, \frac{C_{i\to j}}{C_{cons}}, \frac{ln(A_{i\to j})}{ln(A_{cons})}\right)\right\} \quad (16)$$

$$R\left(P_{i\to j}\right) = \left[\frac{D_{i\to j}}{D_{cons}}, \frac{T_{i\to j}}{T_{cons}}, \frac{C_{i\to j}}{C_{cons}}, \frac{ln(A_{i\to j})}{lnA_{cons})}\right] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \quad (17)$$

The A* based heuristic alternative path searching (AHPS) algorithm tries to find a near-optimal path that satisfies the QoS constraints. If the path exists, the alternative path replaces the original failure path, online fault tolerance succeeds. Otherwise, online fault tolerance fails. The online fault tolerance strategy can not only find a replacement for failed nodes but also ensure that the replacement is near optimal. Part D in this section proves the effectiveness and complexity of the AHPS algorithm. The experiment in Section VI compares the QoS values of alternative paths selected by different methods and further proves the effectiveness of the algorithm.

D. THEORETICAL PROOFS OF LNR AND AHPS ALGORITHMS

1) EFFECTIVENESS PROOF

The effectiveness proof of the LNR algorithm and the AHPS algorithm is given in Appendix B.

2) COMPLEXITY PROOF

Assuming that there are N nodes and E edges in the manufacturing network. The complexity of the LNR algorithm and the AHPS algorithm is analyzed as follows.

a: LNR ALGORITHM: $O(KN^2)$

The iteration process of the LNR algorithm is to calculate the stationary distribution of the Markov chain. Each iteration process requires a matrix-vector multiplication. The complexity of each iteration is no worse than $O(N^2)$. Then the complexity of the LNR algorithm is $O(KN^2)$, where K is the number of iterations when the algorithm converges. The complexity of the LNR algorithm is independent of the number of edges in the manufacturing network.

b: AHPS ALGORITHM $O(NlogN + E)$

The complexity of the Dijkstra shortest path algorithm is $O(NlogN + E)$ after heap optimization. In the backward search procedure, the AHPS algorithm adopts Dijkstra shortest path algorithm twice to find the greedy heuristic path and relax heuristic path. Therefore, the complexity of the backward search procedure is $O(2NlogN + 2E)$. In the forward search procedure, the AHPS algorithm adopts the Dijkstra shortest path algorithm twice to find whether there is a better solution. So the complexity of the forward search is $O(2NlogN + 2E)$, too. In conclusion, the complexity of the AHPS algorithm is $O(4NlogN + 4E)$, which is equal to $O(NlogN + E)$.

V. EXPERIMENT

To verify the effectiveness of the two-stage fault tolerance method, four experiments were designed in this section. The first experiment proves the effect of LNR algorithm on reliability improvement. Then the second experiment verifies that the two-stage fault tolerance method can further improve the reliability of the manufacturing networks and outperforms the existing algorithms. The third experiment compares the performance of the offline method, the online method and the two-stage method on resource utilization and fault

TABLE 1. Parameter distributions of manufacturing networks.

Parameter description	Parameter distribution	Unit
Aging parameter	N(3000,500)	Null
Maintenance interval	N(90,10)	d
Online time	N(8,2)	h
Process accuracy	N(0.2,0.1)	mm
Process links	Randn int (4,20)	Null
Process types	Randn int (2,9)	Null
Initial success rate	Randn (0.7,1)	Null
Invocation number	Randn int (0, 62)	Null
Logistics cost	Randn (6, 80)	$

migration delay. Finally, the fourth one tests the QoS metrics (Delay, Time, Cost and Availability) of the alternative scheme selected by different algorithms and proves that the alternative path chosen by AHPS algorithm is near-optimal. The computational experiments are carried out in Matlab R2018a in Windows 10 system. Several functions are designed to construct the experimental environment and implement the algorithms proposed in this paper. The function logic diagram is shown in Appendix C.

A. EXPERIMENT SETUP

Pajek [46] is used to generate a directed network. The generated networks contain 150 nodes and 2147 directed edges. Based on the characteristics of real data, experimental data sets are generated on the simulation platform, which are shown in Table 1. Then 1000 sets of service compositions were obtained through the random walk in the generated networks. The following process was carried out before the fault tolerance experiment.

- Test the reliability of the service composition. Since the service composition is generated by the random walk, the reliability of the service composition need to be tested in order to prevent the impact of extreme cases on the experimental results. If the reliability of the service composition does not reach the threshold (ReF), it needs to be refactored. The threshold ReF is also a parameter when comparing the effect of different algorithms.
- Execute the service compositions 500 times to calculate the failure probability and failure effect of the nodes in the networks.
- Assign the parameters in Table 1 to the nodes and the edges of the generated networks. Set the **threshold** in Equation (5) to 0.3.

B. EXPERIMENT ON FAILURE RATE OF OFFLINE METHODS

To study the performance of LNR algorithm, six approaches are compared as follows.

1) NO FAULT TOLERANCE (NFT)

No fault tolerance strategies are employed for the manufacturing nodes. This set of experiments is designed to calculate the failure rate of the manufacturing networks.

2) RANDOM FAULT TOLERANCE (RFT)

Fault tolerance strategies are employed for the K percent nodes. The nodes are randomly selected.

3) FTCLOUD [7] (FCFT)

Fault tolerance strategies are employed for the K percent critical nodes. The nodes are ranked by employing the algorithm given in [7].

4) ROCLOUD [8] (TCFT)

Fault tolerance strategies are employed for the K percent critical nodes. The nodes are ranked by employing the algorithm given in [8].

5) LNR ALGORITHM (LNRFT)

Fault tolerance strategies are employed for the K percent critical nodes. The nodes are ranked by employing the LNR algorithm proposed in this paper.

6) ALL FAULT TOLERANCE (AFT)

Fault tolerance strategies are employed for all the nodes in the manufacturing networks.

In order to compare the effect of these fault tolerance algorithm, two variables are set which are the fault tolerance ratio (Top-K) and the refactoring threshold (ReF). Failure rates of service compositions are important indicators to evaluate the effectiveness of different fault tolerance method [7], [8]. In this experiment, 1000 sets of service compositions are generated in the manufacturing network. The proportions of failed service compositions under different parameters and different fault tolerance methods are counted. The mean values of results obtained from 100 independent tests are recorded, which are shown in Table 2 and Figure 4, 5

7) PERFORMANCE COMPARISON

- Among these approaches, the failure rate of the manufacturing networks is the highest when no fault tolerance

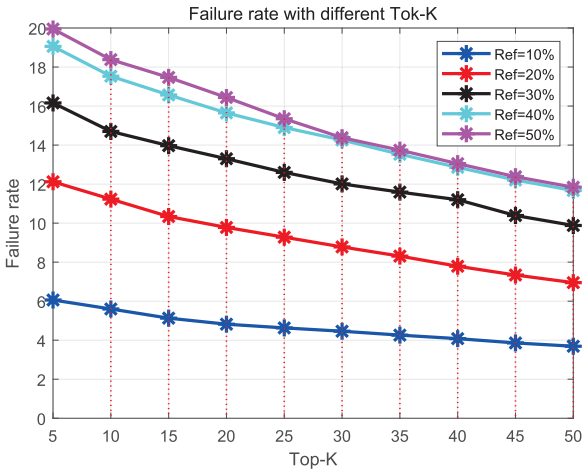

FIGURE 4. Failure rate with different Top-K.

TABLE 2. Result comparison of different Ref and different Top-K.

Ref	Top-K	NFT	RFT	FCFT	RCFT	LNRFT	AFT
10%	5%	6.6177%	6.3745%	6.3342%	6.2352%	6.0591%	0.9282%
	10%	6.6177%	6.0109%	5.7827%	5.6872%	5.5995%	0.9559%
	15%	6.6177%	5.6827%	5.3586%	5.2441%	5.1332%	0.9391%
	20%	6.6177%	5.4077%	5.1897%	5.1035%	4.8111%	0.9395%
	25%	6.6177%	5.2932%	4.8325%	4.7122%	4.6264%	0.9191%
20%	5%	12.9000%	12.4650%	12.3050%	12.2428%	12.1191%	1.6805%
	10%	12.9000%	11.7055%	11.5218%	11.4177%	11.2341%	1.7064%
	15%	12.9000%	11.5895%	10.6105%	10.5285%	10.3364%	1.7345%
	20%	12.9000%	10.9436%	10.1836%	9.9841%	9.7836%	1.2614%
	25%	12.9000%	10.6041%	9.7568%	9.5142%	9.2782%	1.3095%
30%	5%	17.4515%	16.7835%	16.3025%	16.2731%	16.1590%	1.9620%
	10%	17.4515%	15.3195%	15.2255%	15.0931%	14.6960%	1.9535%
	15%	17.4515%	15.9350%	14.3265%	14.2877%	13.9690%	1.9670%
	20%	17.4515%	14.4505%	13.5300%	13.5244%	13.2985%	2.0035%
	25%	17.4515%	13.0000%	12.9760%	12.8344%	12.5988%	1.9330%

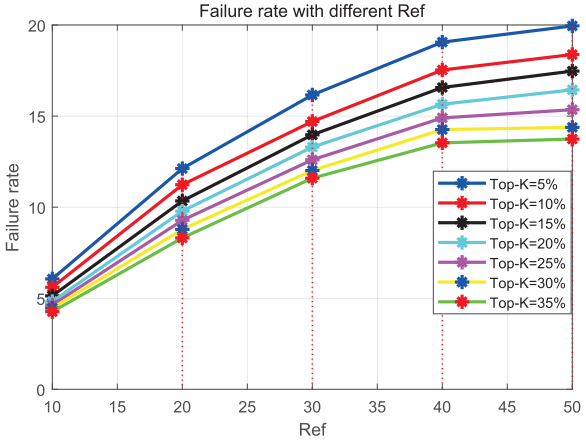

FIGURE 5. Failure rate with different Ref.

strategy is employed. The failure rate is the lowest when all the nodes in the network employ fault tolerance strategy, but it also means a huge cost.

- Compared with RFT, FCFT, RCFT, and LNRFT obtain better performance in all experimental settings. This result indicates that selecting critical nodes and making targeted fault tolerance can achieve better reliability than tolerating faults of randomly selected nodes. This is because the critical nodes may have higher failure rates or greater failure effects. This result also shows the effectiveness of the algorithm.
- Compared with FCFT and RCFT, LNRFT obtains better results in all experimental settings. This result indicates that the LNR algorithm can define the importance of the nodes more accurately. FCFT considers the structural characteristics of the manufacturing network and RCFT considers the characteristics of the node itself. However, the LNR algorithm not only models the reliability of the nodes more accurately but also considers the transfer model of importance between related nodes. Therefore, the LNR algorithm can give a more convincing order when the importance of some nodes are close to each other.
- With the increase of Top-K from 5 percent to 25 percent, the failure probabilities performance of LNR algorithm decreased monotonically. The result indicates that the reliability of manufacturing networks can be improved by employing the LNR algorithm when more nodes are deployed with fault tolerance strategies.
- With the increase of Ref from 10 percent to 30 percent, the failure rate of the manufacturing networks increased monotonically. This is because the manufacturing service can be re-factored only if the failure rate is higher than Ref. High Ref will retain the manufacturing service with a higher failure rate in the networks, which can lead to a decrease in the overall reliability of the manufacturing networks.
- Figure 4 shows that when Ref is low, the change of Top has little influence on the improvement of reliability; when Ref is high, the change of Top has more influence on the improvement of reliability. This is because When Ref is low, the reliability of the service is relatively high, and the number of nodes that have a large impact on overall reliability is very small, so the proportion of nodes deploying fault tolerance strategy has little impact on overall reliability. When the Ref is high, the reliability of the service nodes is relatively poor, and more nodes depend on the fault-tolerance strategy to improve the overall reliability.
- Figure 5 shows that when top-k remains unchanged, the impact of Ref on overall reliability tends to be flat. This is because the manufacturing service itself has been well designed and implemented with good reliability. When Ref is high, its further enlargement does not necessarily lead to more service re-factoring, so its impact on reliability is getting smaller and smaller.

C. EXPERIMENT ON FAILURE RATE OF TWO STAGE METHOD

To study the performance of two-stage fault tolerance method, one more approach is compared as follows.

1) TWO-STAGE FAULT TOLERANCE METHOD (TSFT)

In the offline stage, fault tolerance strategies are employed for the K percent critical nodes. The nodes are ranked by employing the LNR algorithm proposed in this paper. In the

TABLE 3. Comparison of TSFT and LNRFT.

Ref=10%				Ref=20%			
Methods	Failure rate	Top-K	Failure rate reduction	Methods	Failure rate	Top-K	Failure rate reduction
NFT	6.6177%	0%	0.00%	NFT	12.9000%	0%	0.00%
AFT	0.9364%	100%	85.85%	AFT	1.5390%	100%	88.07%
LNRFT	6.0591%	5%	8.47%	LNRFT	12.1191%	5%	6.05%
TSFT	4.5432%	5%	31.37%	TSFT	9.0464%	5%	29.87%
LNRFT	5.5995%	10%	15.42%	LNRFT	11.2341%	10%	12.91%
TSFT	4.1655%	10%	37.08%	TSFT	8.3305%	10%	35.42%
LNRFT	5.1332%	15%	22.46%	LNRFT	10.3364%	15%	19.87%
TSFT	3.9759%	15%	39.94%	TSFT	7.7768%	15%	39.71%
LNRFT	4.8111%	20%	27.30%	LNRFT	9.7836%	20%	24.16%
TSFT	3.8245%	20%	42.23%	TSFT	7.3909%	20%	42.71%
LNRFT	4.6264%	25%	30.11%	LNRFT	9.2782%	25%	28.08%
TSFT	3.6936%	25%	44.21%	TSFT	7.1473%	25%	44.59%

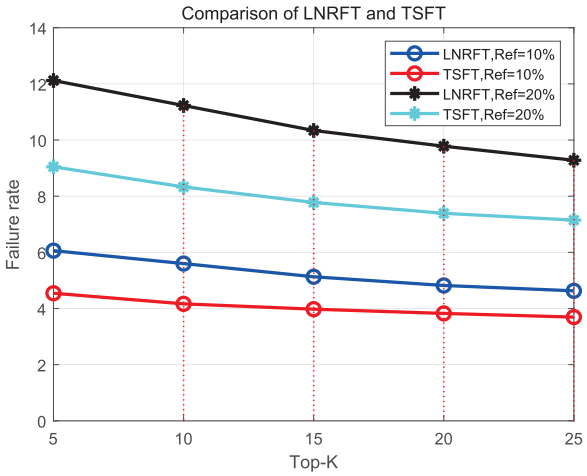

FIGURE 6. Comparison of LNRFT and TSFT.

online stage, AHPS algorithm is adopted to find alternative solutions for failure nodes.

Experiment 1 has proved that LNR algorithm has the best effect on overall reliability improvement among the six approaches. Therefore, this part compares the effect of two-stage fault tolerance method and LNR algorithm. In this experiment, 1000 sets of service compositions are generated in the manufacturing network. The proportions of failed service compositions under different parameters and different fault tolerance methods are counted. The mean values of results obtained from 100 independent tests are recorded. The results are shown in Table 3 and figure 6.

2) PERFORMANCE COMPARISON

- Compared with LNRFT and other algorithms in experiment 1, TSFT obtains better results in all experimental settings. This result indicates that the two-stage fault tolerance algorithm has a better effect on improving the overall reliability of the manufacturing networks. This is because this method not only uses the LNR algorithm for offline fault tolerance deployment of the network but also find alternative schemes for failure nodes by AHPS algorithm in the execution stage of the service.
- Compared with the LNR algorithm, the two-stage fault tolerance method has a greater influence on reliability when Ref is large. This is because if the Ref is large, the reliability of the service nodes is relatively poor. More nodes without offline fault tolerance strategy may fail during execution. Therefore, the online fault tolerance strategy based on APHS algorithm plays a more important role.
- Figure 6 shows that when Ref remains unchanged, the curve of failure rate becomes smooth with the increase of Top-k. This is because if there are fewer nodes with offline fault tolerance strategy, more nodes depend on the online fault tolerance strategy to improve reliability. So the two-stage fault tolerance algorithm is more effective. The effect of AHPS algorithm becomes less obvious when more nodes deploy the offline fault tolerance strategy.

D. EXPERIMENT ON RESOURCE UTILIZATION AND FAULT MIGRATION DELAY

The two-stage fault tolerance method achieves the trade-off between fault tolerance effect and fault tolerance loss by configuring different fault tolerance strategies for different manufacturing services. The loss in fault tolerance can be measured by resource utilization and fault migration delay. The offline method can deploy the backup service and the failed service in the same place, so its fault migration delay is small, namely the local migration delay. The online method needs to re-select the appropriate manufacturing service in the network, so its migration delay is relatively large, namely the remote migration delay. In this experiment, the local migration delay and the remote migration delay are set to N(100, 10) and N(200, 40), respectively. The resource utilization of the manufacturing network is represented as the ratio of the total number of executions of the services to the number of service nodes. One thousand sets of service compositions are generated in the manufacturing network. The mean values of results obtained from 100 independent tests are recorded. Table 4 shows the resource utilization of different fault tolerance methods. Figure 7 shows the number of offline fault tolerance nodes in the network when

TABLE 4. Resource utilization of different algorithms.

Ref	Top-K	RFT	FCFT	RCFT	LNRFT	AHPS	TSFT	AFT
10%	5%	31.95541	32.35032	32.47134	32.54777	35.08667	33.03185	18.15667
	10%	30.53939	31.32121	31.41818	31.58182	35.06	31.94545	18.15667
	15%	29.37791	30.23256	30.31977	30.6686	35.09333	30.9593	18.15667
	20%	28.22778	28.96667	29.13333	29.45556	35.02667	29.76667	18.15667
	25%	27.56684	28.17112	28.59893	28.85561	35.16	29.09626	18.15667
20%	5%	32.07006	32.51592	32.61783	32.79618	36.70667	33.93631	19.57
	10%	30.71515	31.50303	31.80606	32.21212	36.67333	33.13333	19.57
	15%	30.13953	31.0814	31.21512	31.62209	36.75333	32.36047	19.57
	20%	28.8	30.06667	30.23333	30.51667	36.72	31.1	19.57
	25%	28.69519	29.11765	29.34225	29.57754	36.66667	30.03209	19.57
30%	5%	32.12102	33.10828	33.18471	33.43312	38.56	35.03822	21.29333
	10%	30.74545	32.5697	32.98788	33.6303	38.52	34.75758	21.29333
	15%	29.87209	32.05233	32.48256	32.9186	38.54	33.82558	21.29333
	20%	28.70556	31.27222	31.65556	31.93333	38.46667	32.78333	21.29333
	25%	28.45455	30.36364	30.77005	31.11765	38.57333	31.91979	21.29333

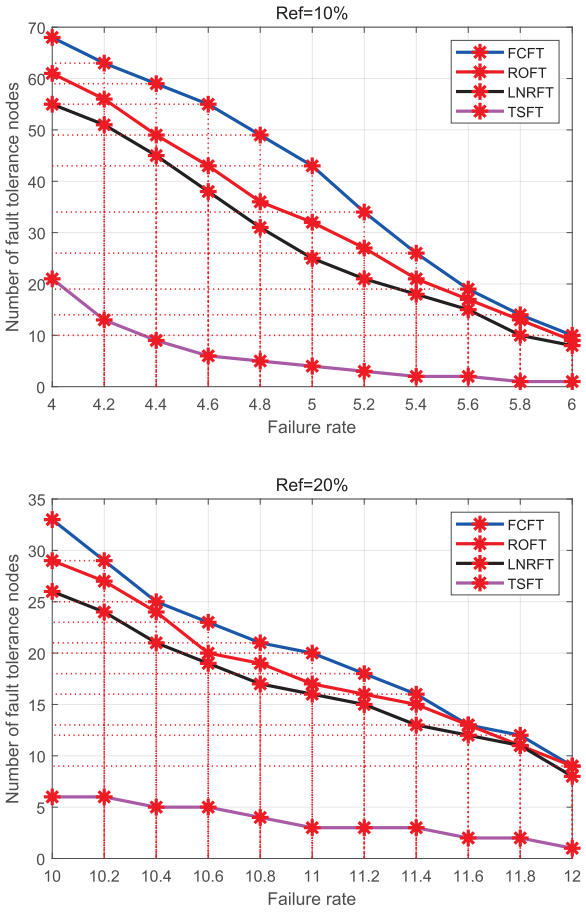

FIGURE 7. Additional fault tolerance nodes of different algorithms.

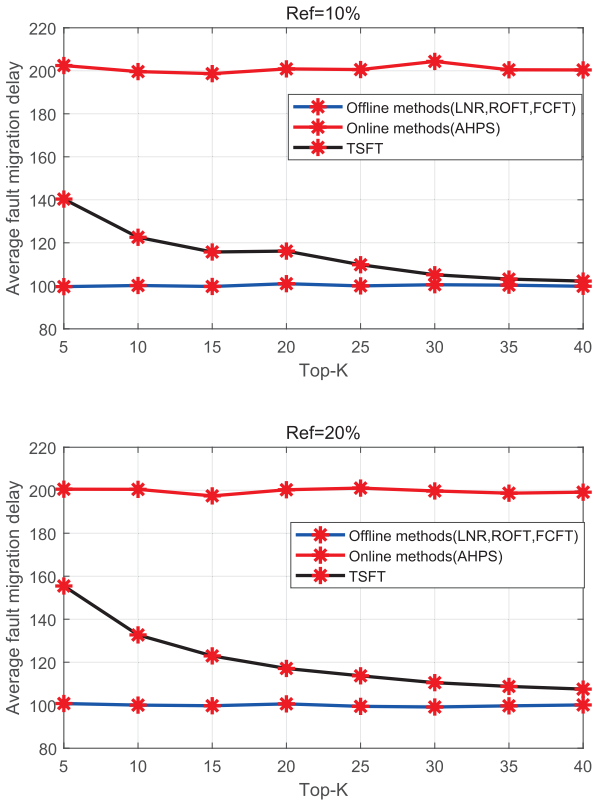

FIGURE 8. Fault migration delay of different algorithms.

the manufacturing network reaches the same failure rate by different fault tolerance methods. Figure 8 shows the average fault migration delay of different fault tolerance methods.

1) PERFORMANCE COMPARISON

- Table 4 shows that the LNRFT method obtains better resource utilization than other offline methods (RFT, FCFT and ROFT) in all experiment settings. This indicates that the services selected by the LNR algorithm have more times of offline fault tolerance. It also proves that services selected by the LNR algorithm are more likely to fail or have a greater influence on the network.
- Compared with the LNRFT method, the TSFT method has higher resource utilization. This is because that the TSFT method not only deploy the backup services to complete the offline tolerance but also execute the online fault tolerance with the help of the original services in the network after faults occur. Therefore, the TSFT method has higher resource utilization than offline methods.

- From Table 2 and 4, the conclusion can be drawn that although deploying offline fault tolerance for all services can greatly reduce the failure rate, it will also greatly reduce resource utilization. However, if only randomly select some services to deploy offline fault tolerance, both the failure rate and resource utilization are very poor. Therefore, it is meaningful to select important services to deploy offline fault tolerance.
- When Ref remains unchanged, the resource utilization decreases with the increase of Top-K. This is because that the backups of high-ranked services have a higher probability of being employed to complete the fault tolerance. This also illustrates that the LNR algorithm can find out the services that are more important.
- Figure 7 shows that the TSFT method requires only a small number of services to be backed up to achieve the same failure rate as other offline methods. This is because the online stage can efficiently solve the low-frequency faults so that it can reduce the failure rate of the network. Therefore, the TSFT method has much higher resource utilization than other offline methods.
- Figure 8 shows that the fault migration delay of the TSFT method is higher than that of the offline methods but obviously lower than that of the online methods. This is because that the TSFT method solves most faults by offline methods so that the delay is relatively low. As Top-K increases, the delay of TSFT decreases monotonically. In practical application, the ratio of services that deploy offline methods (Top-K) can be adjusted according to actual demands to achieve the trade-off between resource utilization and fault migration delay.

E. EXPERIMENT ON QOS METRICS OF ONLINE METHODS

The experiments above demonstrate the effectiveness of the two-stage fault tolerance algorithm from three aspects: failure rate, resource utilization and fault migration delay. In the online fault tolerance stage, the QoSmetric is an important index to judge whether the alternative scheme is good or not. To verify the QoS metric of the alternative scheme selected by the AHPS algorithm is near optimal, the experiment is designed in this part. Without loss of generality, the experiment selected four QoS indexes: delay (D), time (T), cost (C) and availability (A), and compared the results of the following three algorithms under different node sizes, different QoS constraints and different linear weights.

1) H_OSTP [31]

Alternative schemes are found by A* algorithm under one heuristic path. The heuristic path is defined by H_OSTP algorithm.

2) MFPB_HOSTP [32]

Alternative schemes are found by improved A* algorithm under multiple heuristic paths. The paths are defined by MFPB_HOSTP algorithm.

TABLE 5. QoS values of different node scale.

Node Scale	Algorithm	F/400	Delay	Time	Cost	Availability
200	H_OSTP	331	0.6214	0.5822	0.5236	0.3201
	MFPB	331	0.6155	0.582	0.5226	0.321
	AHPS	331	0.5972	0.5805	0.5083	0.3264
250	H_OSTP	368	0.5985	0.5388	0.4918	0.3513
	MFPB	368	0.5984	0.5385	0.492	0.3515
	AHPS	368	0.5978	0.5383	0.4859	0.3551
300	H_OSTP	389	0.5849	0.5317	0.4586	0.3594
	MFPB	389	0.5813	0.5312	0.4585	0.3595
	AHPS	389	0.5779	0.5281	0.4541	0.363
350	H_OSTP	397	0.5478	0.5201	0.4331	0.3963
	MFPB	397	0.5469	0.52	0.4322	0.3965
	AHPS	397	0.5346	0.5197	0.4251	0.4087
w=[0.2,0.1,0.4,0.3], Constraints=[0.8,0.8,0.8,0.15]						

TABLE 6. QoS values of different Constraints.

Constraints	Algorithm	F/400	Delay	Time	Cost	Availability
[0.7,0.7,0.7,0.2]	H_OSTP	202	0.5879	0.536	0.5042	0.3454
	MFPB	202	0.5846	0.5316	0.504	0.3462
	AHPS	202	0.5835	0.5308	0.4979	0.3475
[0.8,0.8,0.8,0.15]	H_OSTP	331	0.6214	0.5822	0.5236	0.3201
	MFPB	331	0.6155	0.582	0.5226	0.321
	AHPS	331	0.5972	0.5805	0.5083	0.3264
[0.9,0.9,0.9,0.1]	H_OSTP	357	0.6579	0.6038	0.5297	0.3159
	MFPB	357	0.6518	0.6037	0.5295	0.3163
	AHPS	357	0.6499	0.603	0.523	0.3175
w=[0.2,0.1,0.4,0.3], Node=200						

TABLE 7. QoS values of different Weights.

Weights	Algorithm	F/400	Delay	Time	Cost	Availability
[0.1,0.7,0.1,0.1]	H_OSTP	334	0.6878	0.4557	0.662	0.2449
	MFPB	334	0.6881	0.4509	0.6628	0.2448
	AHPS	334	0.6875	0.4457	0.6621	0.2448
[0.2,0.1,0.4,0.3]	H_OSTP	331	0.6214	0.5822	0.5236	0.3201
	MFPB	331	0.6155	0.582	0.5226	0.321
	AHPS	331	0.5972	0.5805	0.5083	0.3264
[0.1,0.4,0.4,0.1]	H_OSTP	325	0.7487	0.525	0.5081	0.2603
	MFPB	325	0.7501	0.5243	0.5073	0.2602
	AHPS	325	0.7488	0.5208	0.5058	0.2558
Constraints=[0.8,0.8,0.8,0.15], Node=200						

3) AHPS

Alternative schemes are found by improved A* algorithm under greedy heuristic function and relaxed heuristic function. These functions are proposed in(16) and (17).

The mean values of results obtained from 100 independent tests are recorded. Table 5, 6 and 7 are normalized QoS metrics under different experiment settings. In these tables, MFPB is used to represent the MFPB_HOSTP algorithm. F/400 is the number of feasible solutions in 400 experiments. Equation (17) aggregates the four QoS metrics through the linear weighting method, so it can be used to measure the overall QoS value. Figure 9 shows the aggregation values of the three methods that are calculated through (17) in 400 independent experiments with different network scale and QoS constraints. Figure 10 shows the execution time of the three methods in 400 independent experiments with different network scale.

4) PERFORMANCE COMPARISON

- If the alternative solution exists, all of these three algorithms can find the solution that satisfies the

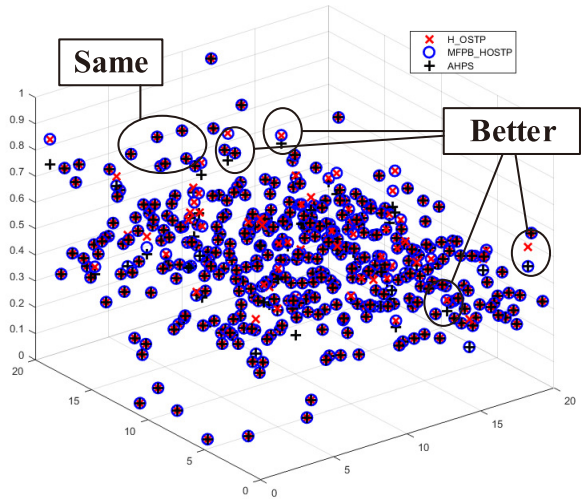

FIGURE 9. Aggregation QoS values of different algorithms.

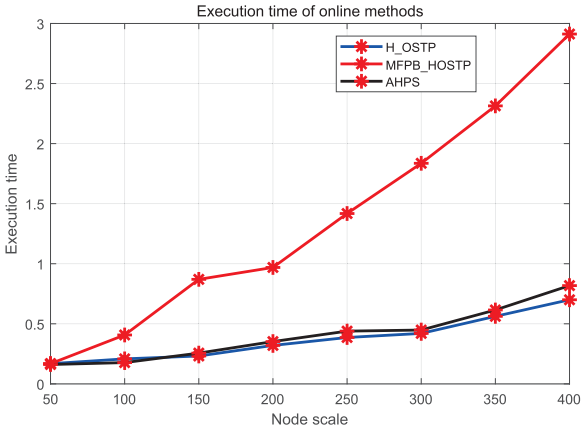

FIGURE 10. Execution time of different algorithms.

QoS constraints. So all these three algorithms can be used to deploy the online fault tolerance strategy.

- Figure 9 illustrates that the AHPS method can find the alternative solution with better weighted average QoS value sometimes under different QoS constraints and network scale. The X-axis and Y-axis are different experiment settings (20 groups of network scale and 20 groups of QoS constraints). The AHPS algorithm obtains better results than H_OSTP in 7.75% of all experiments (31 of 400). The AHPS also find better solutions in 7.25% of all experiments (29 in 400). This validates the effectiveness of the AHPS algorithm in finding the alternative solutions that are near optimal.
- From Table 5, 6 and 7 the conclusion can be drawn that compared with H_OSTP and MFPB_HOSTP, the AHPS algorithm obtains better average results in all experimental settings. This result indicates that the AHPS algorithm can find a more optimal alternative scheme than other algorithms. This is because the AHPS algorithm uses the greedy function and the relaxed function as heuristic functions, and the optimal path is more likely to be found when QoS constraints are satisfied.
- Table 5 shows that with the increase of the network scale, the QoS metrics become better. This is because if there are more services in the network, there is a larger probability to choose a better solution through more candidate services.
- Table 6 illustrates that when the QoS constraints become stricter, the feasible solutions in the network become less but the average quality of the solutions increases. This is because that some feasible solutions with low quality become unfeasible under stricter constraints. Therefore, in the 400 independent experiments, there are fewer experiments with feasible solutions but these feasible solutions have higher quality because of the stricter QoS constraints.
- Table 7 shows that the AHPS is more inclined to optimize the QoS metrics with higher weights. This is determined by the search target of the algorithm. In the search process, the QoS metric with higher weight has a greater impact on the search result. Therefore, the metric with higher weight needs to be optimized more. In the application, users can set weights according to their actual demands to achieve the customized optimization of QoS metrics.
- Figure 10 shows that the execution time of AHPS is almost similar to that of H_OSTP. This is because that the two algorithms have the same time complexity, which is equal to $O(N\log N + E)$. The execution time of AHPS is much shorter than that of the MFBP_HOSTP algorithm. This is because the MFPB_HOSTP algorithm is more complex due to the multiple foreseen path [32]. Therefore, the execution time of the MFPB_HOSTP algorithm increases rapidly with the increase of the network scale.

VI. CONCLUSION

This paper proposes a two-stage fault tolerance framework for complex manufacturing networks. In the proposed framework, the LNR algorithm and the AHPS algorithm are put forward to complete the offline fault tolerance stage and the online fault tolerance stage. In the LNR algorithm, the significance values of the manufacturing nodes are calculated by the static attributes and dynamic attributes. By deploying the republication strategy to critical nodes in the manufacturing networks, the tradeoff of fault tolerance effect and fault tolerance loss is achieved in the offline stage. During the online stage, the AHPS algorithm is used to find the near-optimal alternative schemes for the nodes without the offline fault tolerance strategy. The experimental results show that the LNR algorithm outperforms other approaches. The two-stage fault tolerance method can further improve the reliability of the manufacturing networks. The AHPS algorithm can find a near-optimal alternative path for the non-critical nodes that fail during the online stage.

Our current two-stage fault tolerance method can be employed to tolerate faults in the computing networks or manufacturing networks. In the future, we will investigate to build more accurate models for manufacturing networks. More node ranking algorithms will be designed to form a ranking algorithm library. Moreover, accurate cost constraint will be considered to be an important factor in critical node selection. The two-stage fault tolerance method will also be deployed in an actual avionics assembly network to improve the reliability of the network.

APPENDIX A
FORWARD SEARCH ALGORITHM

Algorithm 3 Path Update(P_1,P_2,P_3)

Input: P_1,P_2,P_3.
Output: Null.
1: **if** $R(P_1+P_2) < R(P_3)$ **then**
2: $P_3 = P_1 + P_2$
3: **end if**

APPENDIX B
EFFECTIVENESS PROOF OF THE LNR ALGORITHM AND THE AHPS ALGORITHM
LNR ALGORITHM

The LNR algorithm is an algorithm to calculate the importance of manufacturing services in manufacturing networks. To meet the demands of complex manufacturing tasks, the services need to be composed. Therefore, the importance of manufacturing services is related not only to their own attributes but also to the services with which they accomplish the manufacturing tasks together. That is, the importance of manufacturing services can be transmitted to adjacent services. In the LNR algorithm, the initial importance value $V_i(0)$ of the node is randomly assigned. The importance values of manufacturing services are iteratively updated by $V_i(t) = \sum_{j=1}^{n+1} \frac{M_{j,i}^{iter}}{\sum_{k=1}^{n+1} M_{j,k}^{iter}} \times V_i(t-1)$. To prove the effectiveness of the LNR algorithm, it need to be proved that $lim_{t\to\infty} V_i(t)$ exists and it is independent of the value of $V_i(t_0)$. The matrix form of the iterative process is shown in (18) and (19).

$$V(t) = \begin{bmatrix} \widetilde{M_{1,1}^{iter}} & \cdots & \widetilde{M_{1,n+1}^{iter}} \\ \vdots & \ddots & \vdots \\ \widetilde{M_{n+1,1}^{iter}} & \cdots & \widetilde{M_{n+1,n+1}^{iter}} \end{bmatrix}^T \times \begin{bmatrix} V_1(t-1)) \\ \vdots \\ V_{n+1}(t-1)) \end{bmatrix} \tag{18}$$

$$\widetilde{M_{i,j}^{iter}} = M_{i,j}^{iter} / \sum_{j=1}^{n+1} M_{i,j}^{iter} \tag{19}$$

The process of calculating the importance of manufacturing services is a Markov process according to (18) and (19). V(t) is a state of the Markov chain and $\widetilde{M^{iter}}^T$ is the state transition matrix. The importance value of the services is the stationary distribution of the Markov chain. If the Markov process converges, then $\lim_{t\to\infty} V(t)$ exists and it is independent of the value of $V(0)$. The convergence of Markov processes requires the following three conditions. (A) $\widetilde{M^{iter}}$ is a stochastic matrix. (B) $\widetilde{M^{iter}}$ is an irreducible matrix. (C) $\widetilde{M^{iter}}$ is an aperiodic matrix.

The proof of (A): According to (19), $\forall i = 1, 2, 3 \ldots n+1$, $\sum_{j=1}^{n+1} M_{i,j}^{iter} = \sum_{j=1}^{n+1} \left(M_{ij}^{iter} / \sum_{k=1}^{n+1} M_{i,k}^{iter} \right) = 1$. Therefore, $\widetilde{M^{iter}}$ is a stochastic matrix.

Algorithm 4 Forward Search Algorithm

Input: D_{fd}, D_{fm}, D_i^{sw}, D_i^{hw}, $D_{j,j+1}^{com}$, T_i^{sv}, $T_{j,j+1}^{edge}$, C_i^{sv}, $C_{j,j+1}^{edge}$, A_i^{m+1}, $(i = 1, 2, \ldots n,\ j = 1, 2, \ldots n-1)$, D_{cons},T_{cons},C_{cons}, A_{cons},P_{e-k}^G,P_{e-k}^R,$G(P_{e-k}^G)$,$R(P_{e-k}^R)$.
Output: Weighted optimal path $P_{s\to e}$, $R(P_{s\to e})$.
1: Set VS_1=VS_2=$\varnothing$, $R(P_{s-k}^1)$=$R(P_{s-k}^2)$=∞
2: Move s into VS_1 and VS_2
3: **while** $VS_1 \neq \varnothing$ and $VS_2 \neq \varnothing$ **do**
4: Find $R(P_{s-a})$=min(all nodes in VS_1)
5: Find $R(P_{s-b})$=min(all nodes in VS_2)
6: **if** a=b and $R(P_{s-a})$=$R(P_{s-b})$ **then**
7: **while** edge(k,a) exists **do**
8: **case 1:** both P_{s-a}^1+P_{a-e}^G and P_{s-b}^2+P_{b-e}^R satisfy the QoS constraints
9: **Path update**(P_{s-a}^1,P_{a-e}^G,P_{s-e}^1)
10: **Path update**(P_{s-b}^2,P_{b-e}^R,P_{s-e}^2)
11: **case 2:** P_{s-a}^1+P_{a-e}^G satisfy the constraints and P_{s-b}^2+P_{b-e}^R doesn't
12: **Path update**(P_{s-a}^1,P_{a-e}^G,P_{s-e}^1)
13: **Path update**(P_{s-b}^2,P_{b-e}^G,P_{s-e}^2)
14: **case 3:** P_{s-a}^1+P_{a-e}^G doesn't satisfy the constraints and satisfies
15: **Path update**(P_{s-a}^1,P_{a-e}^R,P_{s-e}^1)
16: **Path update**(P_{s-b}^2,P_{b-e}^R,P_{s-e}^2)
17: **end while**
18: **else**
19: **while** edge(k,a) exists **do**
20: **case 1:** P_{s-a}^1+P_{a-e}^G satisfies the QoS constraints
21: **Path update**(P_{s-a}^1,P_{a-e}^G,P_{s-e}^1)
22: **case 2:** P_{s-a}^1+P_{a-e}^R satisfies the QoS constraints
23: **Path update**(P_{s-a}^1,P_{a-e}^R,P_{s-e}^1)
24: **end while**
25: **while** edge(k,b) exists **do**
26: **case 1:** P_{s-b}^2+P_{b-e}^G satisfies the QoS constraints
27: **Path update**(P_{s-b}^2,P_{a-e}^G,P_{s-e}^2)
28: **case 2:** P_{s-b}^2+P_{b-e}^R satisfies the QoS constraints
29: **Path update**(P_{s-b}^2,P_{b-e}^R,P_{s-e}^2)
30: **end while**
31: **end if**
32: Remove a from VS_1 and b from VS_2
33: **end while**
34: **return** $P_{s\to e}$=min(P_{s-e}^1,P_{s-e}^2) and $R(P_{s\to e})$

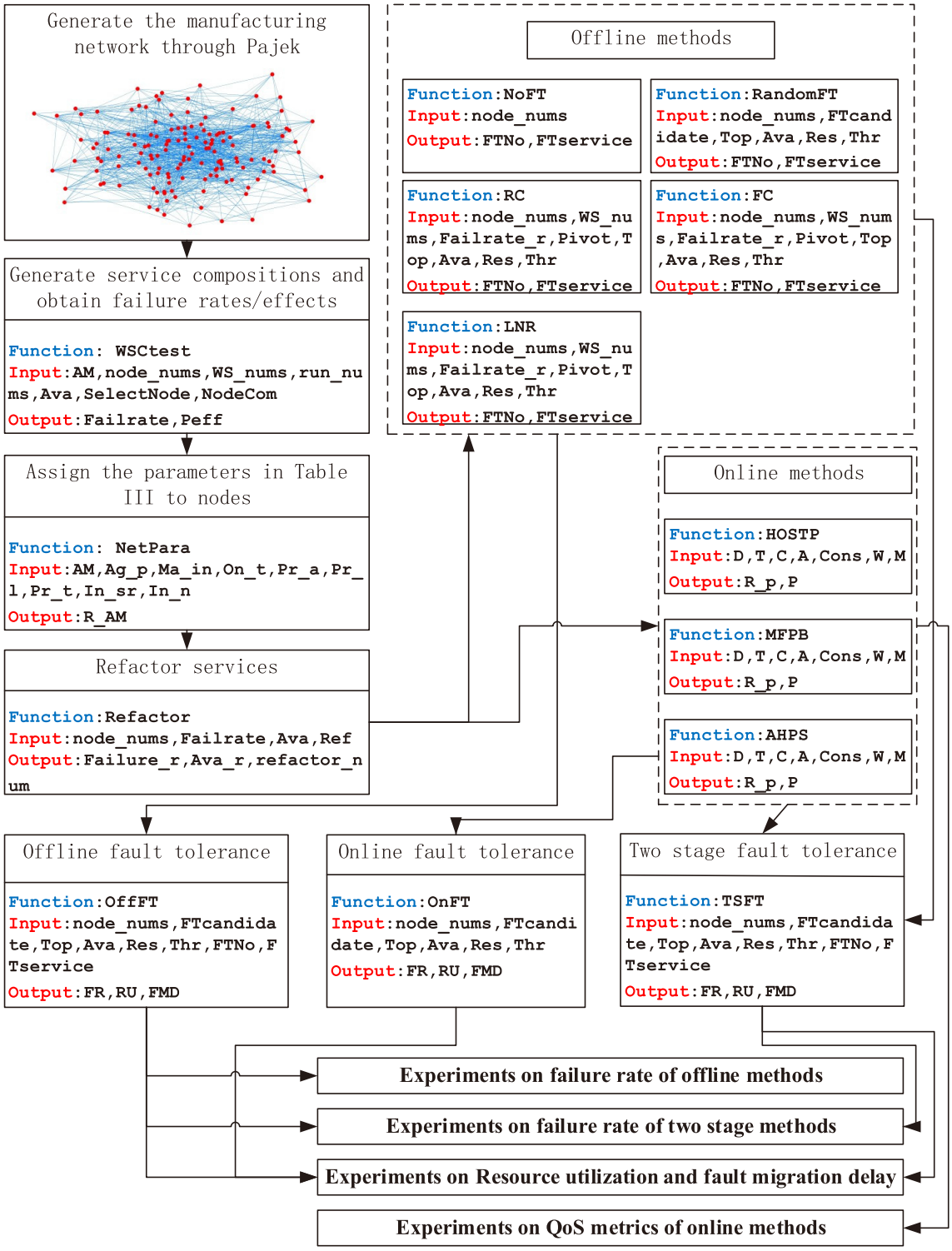

FIGURE 11. Function logic diagram of the computational experiments.

The proof of (B): According to (9), there is a background node that is connected to all the other nodes. Therefore, the graph corresponding to $\widetilde{M^{iter}}$ is a strongly-connected graph. $\widetilde{M^{iter}}$ is an irreducible matrix.

The proof of (C): According to (8) and (19), $\widetilde{M^{iter}}$ is a primitive matrix. Therefore, $\widetilde{M^{iter}}$ is not periodic.

From what has been discussed above, the conclusion can be drawn that $\lim_{t\to\infty} V(t)$ exists and it is independent of

the value of V(0). The effectiveness of the LNR algorithm is proved.

AHPS ALGORITHM

The AHPS algorithm is designed to find a near-optimal service composition to replace the failed one and meet the QoS constraints of the manufacturing task at the same time. It consists of two backward searches and one forward search. In order to prove the effectiveness of AHPS algorithm, the following three points need to be proved. (A) If there is a feasible service composition in the network, the backward search with (16) as the target can guarantee to find the feasible solution. (B) If the QoS constraints are not taken into consideration, the backward search with (17) as the target can find the optimal $\mathrm{R}(P_{e\to i})$ paths from each node to the end node. (C) Assuming that p_G is defined by the backward search with (16) as the target, p_R is defined by the backward search with (17) as the target and p_F is defined by the forward search. If p_G is feasible, p_F is feasible and $\mathrm{R}(p_R) \leq \mathrm{R}(p_F) \leq \mathrm{R}(p_G)$.

The Proof of (A): Assuming that p_G is the solution found by the backward search and p_G is infeasible. p'_G is the feasible path in the network. Since p_G is the path found by the Dijkstra algorithm with (16) as the target. Therefore, $G(p_G) \leq G(p'_G)$. According to (16), p_G is infeasible, then $G(p_G) > 1$, p'_G is feasible, then $G(p'_G) \leq 1$ and $G(p_G) > G(p'_G)$. This contradicts $G(p_G) \leq G(p'_G)$. Therefore, p_G is the feasible solution.

The Proof of (B): Since the Dijkstra algorithm is used for the backward process with (17) as the target. The paths found by the procedure have minimal $\mathrm{R}(P_{e\to i})$. However, the search process does not take into account the QoS constraints, the paths may not be feasible.

The proof of (C): The proof of (C). The purpose of the forward search is to find a near-optimal path under the premise of satisfying QoS constraints. Equation (17) is the search target of the forward search process. p_R and p_G are two heuristic paths to accelerate the search process. Assuming that $\mathcal{V}_s$ is the start node, $\mathcal{V}_e$ is the end node, the forward search process reaches the node $\mathcal{V}_i$ and $\mathcal{V}_j$ is the neighbor node of $\mathcal{V}_i$. Two conditions need to be satisfied if $\mathcal{V}_j$ is selected to compose the path p_F. The first condition is that the path $p_{s-i-j-e}$ must satisfy the QoS constraints. The second one is that $\mathrm{R}(p_{s-i}) + \mathrm{R}(p_{i-e}) < \mathrm{R}(p_{s-e})$. If $\mathcal{V}_i$ is in p_G, condition 1 is satisfied and condition 2 may not be satisfied. When condition 2 is not satisfied, there may be another feasible path with better R value. That is, $\mathrm{R}(p_G)$ is the supremum of $\mathrm{R}(p_F)$, $\mathrm{R}(p_F) \leq \mathrm{R}(p_G)$. If $\mathcal{V}_j$ is in p_R, condition 2 is satisfied and condition 1 may not be satisfied. When condition 1 is not satisfied, there may be another feasible path. However, the R-value of the feasible path is larger than $\mathrm{R}(p_R)$ according to the proof of (B). That is, $\mathrm{R}(p_R)$ is the infimum of $\mathrm{R}(p_F)$, $\mathrm{R}(p_R) \leq \mathrm{R}(p_F)$. Since (17) is the search target, the forward search is a process of approximating the infimum as far as possible under the premise of satisfying QoS constraints. Therefore, the service composition found by the AHPS algorithm is proved to be near optimal.

APPENDIX C
FUNCTION LOGIC DIAGRAM OF THE COMPUTATIONAL EXPERIMENTS

See Figure 11.

REFERENCES

[1] F. Tao, L. Zhang, and A. Y. C. Nee, "A review of the application of grid technology in manufacturing," *Int. J. Prod. Res.*, vol. 49, no. 13, pp. 4119–4155, Jul. 2011.

[2] C. Xu-Dong, L. Bo-Hu, Z. Lin, W. Shi-Long, T. Fei, C. Jun-Wei, J. Xiao-Dan, and S. Xiao, "Cloud manufacturing:A new service-oriented networked manufacturing model," *Comput. Integr. Manuf. Syst.*, vol. 16, no. 1, pp. 1–17, Jan. 2010.

[3] X. Xu, "From cloud computing to cloud manufacturing," *Robot. Comput.-Integr. Manuf.*, vol. 28, no. 1, pp. 75–86, Feb. 2012.

[4] W. He and L. Xu, "A state-of-the-art survey of cloud manufacturing," *Int. J. Comput. Integr. Manuf.*, vol. 28, no. 33, pp. 239–250, 2015.

[5] L. Ren, L. Zhang, L. Wang, F. Tao, and X. Chai, "Cloud manufacturing: Key characteristics and applications," *Int. J. Comput. Integr. Manuf.*, vol. 30, no. 6, pp. 501–515, 2017.

[6] Z. Zheng and M. R. Lyu, "Selecting an optimal fault tolerance strategy for reliable service-oriented systems with local and global constraints," *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 219–232, Jan. 2015.

[7] Z. Zheng, T. C. Zhou, M. R. Lyu, and I. King, "Component ranking for fault-tolerant cloud applications," *IEEE Trans. Services Comput.*, vol. 5, no. 4, pp. 540–550, 4th Quart., 2012.

[8] W. Qiu, Z. Zheng, X. Wang, X. Yang, and M. R. Lyu, "Reliability-based design optimization for cloud migration," *IEEE Trans. Services Comput.*, vol. 7, no. 2, pp. 223–236, Apr. 2014.

[9] T. Korkmaz and M. Krunz, "Multi-constrained optimal path selection," in *Proc. IEEE INFOCOM Conf. Comput. Commun., 20th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 2, Apr. 2001, pp. 834–843.

[10] M. A. Mukwevho and T. Celik, "Toward a smart cloud: A review of fault-tolerance methods in cloud systems," *IEEE Trans. Services Comput.*, to be published.

[11] B. Mohammed, M. Kiran, K. M. Maiyama, M. M. Kamala, and I.-U. Awan, "Failover strategy for fault tolerance in cloud computing environment," *Softw., Pract. Exper.*, vol. 47, no. 9, pp. 1243–1274, 2017.

[12] I. Jangjaimon and N.-F. Tzeng, "Effective cost reduction for elastic clouds under spot instance pricing through adaptive checkpointing," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 396–409, Feb. 2015.

[13] B. Balasubramanian and V. K. Garg, "Fault tolerance in distributed systems using fused data structures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 4, pp. 701–715, Apr. 2013.

[14] A. Zhou, S. Wang, C.-H. Hsu, M. H. Kim, and K.-S. Wong, "Network failure–aware redundant virtual machine placement in a cloud data center," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 24, Dec. 2017, Art. no. e4290.

[15] R. Jhawar, V. Piuri, and M. Santambrogio, "Fault tolerance management in cloud computing: A system-level perspective," *IEEE Syst. J.*, vol. 7, no. 2, pp. 288–297, Jun. 2013.

[16] C. Wang, L. Xing, H. Wang, Z. Zhang, and Y. Dai, "Processing time analysis of cloud services with retrying fault-tolerance technique," in *Proc. 1st IEEE Int. Conf. Commun. China (ICCC)*, Aug. 2012, pp. 63–67.

[17] A. Avizienis, "The N-version approach to fault-tolerant software," *IEEE Trans. Softw. Eng.*, vol. SE-11, no. 12, pp. 1491–1501, Dec. 1985.

[18] Z. Zheng, Y. Zhang, and M. R. Lyu, "CloudRank: A QoS-driven component ranking framework for cloud computing," in *Proc. 29th IEEE Symp. Reliable Distrib. Syst.*, Oct./Nov. 2010, pp. 184–193.

[19] Z. Liu, C. Jiang, J. Wang, and H. Yu, "The node importance in actual complex networks based on a multi-attribute ranking method," *Knowl. Based Syst.*, vol. 84, pp. 56–66, Aug. 2015.

[20] J. Liu, Q. Xiong, W. Shi, X. Shi, and K. Wang, "Evaluating the importance of nodes in complex networks," *Phys. A, Statist. Mech. Appl.*, vol. 452, pp. 209–219, Feb. 2016.

[21] Y. Wu and H. Zhang, "A reliability optimization method for collaborative manufacturing service network based on mining of important nodes," in *Proc. IEEE 21st Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, Apr. 2017, pp. 402–407.
[22] K. Inoue, R. Yokomori, T. Yamamoto, M. Matsushita, and S. Kusumoto, "Ranking significance of software components based on use relations," *IEEE Trans. Softw. Eng.*, vol. 31, no. 3, pp. 213–225, Mar. 2005.
[23] N. Wu, D. Zuo, Z. Zhang, P. Zhou, and Y. Zhao, "FSCRank: A failure-sensitive structure-based component ranking approach for cloud applications," *IEICE Trans. Inf. Syst.*, vol. 102, no. 2, pp. 307–318, Feb. 2019.
[24] Z. Ding, J. Liu, Y. Sun, C. Jiang, and M. Zhou, "A transaction and QoS-aware service selection approach based on genetic algorithm," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 7, pp. 1035–1046, Jul. 2015.
[25] T. Li, T. He, Z. Wang, and Y. Zhang, "An approach to IoT service optimal composition for mass customization on cloud manufacturing," *IEEE Access*, vol. 6, pp. 50572–50586, 2018.
[26] Y. Que, W. Zhong, H. Chen, X. Chen, and X. Ji, "Improved adaptive immune genetic algorithm for optimal QoS-aware service composition selection in cloud manufacturing," *Int. J. Adv. Manuf. Technol.*, vol. 96, nos. 9–12, pp. 4455–4465, Jun. 2018.
[27] X. Min, X. Xu, Z. Liu, D. Chu, and Z. Wang, "An approach to resource and QoS-aware services optimal composition in the big service and Internet of Things," *IEEE Access*, vol. 6, pp. 39895–39906, 2018.
[28] J. Zhou and X. Yao, "A hybrid artificial bee colony algorithm for optimal selection of QoS-based cloud manufacturing service composition," *Int. J. Adv. Manuf. Technol.*, vol. 88, nos. 9–12, pp. 3371–3387, 2017.
[29] X. Zhao, B. Song, P. Huang, Z. Wen, J. Weng, and Y. Fan, "An improved discrete immune optimization algorithm based on PSO for QoS-driven Web service composition," *Appl. Soft Comput.*, vol. 12, no. 8, pp. 2208–2216, Aug. 2012.
[30] Y. Wu, G. Peng, H. Wang, and H. Zhang, "A heuristic algorithm for optimal service composition in complex manufacturing networks," *Complexity*, vol. 2019, Apr. 2019, Art. no. 7819523.
[31] G. Liu, Y. Wang, M. A. Orgun, and E.-P. Lim, "A heuristic algorithm for trust-oriented service provider selection in complex social networks," in *Proc. IEEE Int. Conf. Services Comput.*, Jul. 2010, pp. 130–137.
[32] G. Liu, Y. Wang, M. A. Orgun, and E.-P. Lim, "Finding the optimal social trust path for the selection of trustworthy service providers in complex social networks," *IEEE Trans. Services Comput.*, vol. 6, no. 2, pp. 152–167, Apr./Jun. 2013.
[33] A. K. S. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mech. Syst. Signal Process.*, vol. 20, no. 7, pp. 1483–1510, 2006.
[34] L. Li, M. You, and J. Ni, "Reliability-based dynamic maintenance threshold for failure prevention of continuously monitored degrading systems," *J. Manuf. Sci. Eng.*, vol. 131, no. 3, May 2009, Art. no. 031010.
[35] G. Wang and S. Yin, "Quality-related fault detection approach based on orthogonal signal correction and modified PLS," *IEEE Trans. Ind. Informat.*, vol. 11, no. 2, pp. 398–405, Apr. 2015.
[36] R. Muradore and P. Fiorini, "A PLS-based statistical approach for fault detection and isolation of robotic manipulators," *IEEE Trans. Ind. Electron.*, vol. 59, no. 8, pp. 3167–3175, Aug. 2012.
[37] M. Z. Sheriff, M. Mansouri, M. N. Karim, H. Nounou, and M. Nounou, "Fault detection using multiscale PCA-based moving window GLRT," *J. Process Control*, vol. 54, pp. 47–64, Jun. 2017.
[38] S. Li, X. Zhou, F. Pan, H. Shi, K. Li, and Z. Wang, "Correlated and weakly correlated fault detection based on variable division and ICA," *Comput. Ind. Eng.*, vol. 112, pp. 320–335, Oct. 2017.
[39] C. Botre, M. Mansouri, M. Nounou, H. Nounou, and M. N. Karim, "Kernel PLS-based GLRT method for fault detection of chemical processes," *J. Loss Prevention Process Industries*, vol. 43, pp. 212–224, Sep. 2016.
[40] J. Yu, J. Yoo, J. Jang, J. H. Park, and S. Kim, "A novel hybrid of auto-associative kernel regression and dynamic independent component analysis for fault detection in nonlinear multimode processes," *J. Process Control*, vol. 68, pp. 129–144, Aug. 2018.
[41] F. Zhou, J. H. Park, and Y. Liu, "Differential feature based hierarchical PCA fault detection method for dynamic fault," *Neurocomputing*, vol. 202, pp. 27–35, Aug. 2016.
[42] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the Web," Stanford InfoLab, Stanford Univ., Palo Alto, CA, USA, Tech. Rep. 422, 1999. [Online]. Available: http://ilpubs.stanford.edu:8090/422/
[43] L. Lü, Y.-C. Zhang, C. H. Yeung, and T. Zhou, "Leaders in social networks, the delicious case," *PloS One*, vol. 6, no. 6, Jun. 2011, Art. no. e21202.
[44] S. Xu and P. Wang, "Identifying important nodes by adaptive LeaderRank," *Phys. A, Stat. Mech. Appl.*, vol. 469, pp. 654–664, Mar. 2017.
[45] H. J. Stuart, "The exponentially weighted moving average," *J. Quality Technol.*, vol. 18, no. 4, pp. 203–210, 1986.
[46] V. Batagelj and A. Mrvar, "Pajek—A program for large network analysis," *Connections*, vol. 21, no. 2, pp. 47–57, Feb. 1998.

YINAN WU received the bachelor's degree from the Department of Automation, Tsinghua University, Beijing, China, in 2014, where he is currently pursuing the Ph.D. degree with the National Engineering Research Centre of Computer Integrated Manufacturing System (CIMS-ERC). His research interests include manufacturing service composition and service reliability.

GONGZHUANG PENG received the bachelor's degree from the Department of Automation, Beihang University, in 2012, and the Ph.D. degree from the Department of Automation, Tsinghua University, Beijing, China, in 2018. He is currently an Assistant Researcher with the Engineering Research Institute, University of Science and Technology Beijing. His research interests include knowledge management and manufacturing service.

HONGWEI WANG received the bachelor's degree from the College of Biomedical Engineering and Instrumentation Science and the Chukezhen Honors College, Zhejiang University, the M.Sc. degree from the National CIMS Engineering Research Center, Tsinghua University, and the Ph.D. degree from the Engineering Design Center, Cambridge University, in 2010. He is currently a tenured Associate Professor and a Ph.D. Advisor with ZJUI, Zhejiang University, with the honor of specially invited expert by Zhejiang Province. His main research interests include the application of information and computing technologies to the design, analysis, manufacture, and maintenance of complex engineering systems. Specifically, he was involved in complex system design, knowledge engineering, collaborative modeling and simulation, and the condition monitoring and fault diagnosis of power systems.

HEMING ZHANG received the Ph.D. degree in mechanical engineering from Zhejiang University, China, in 1995. He joined the faculty of the Department of Automation, Tsinghua University, China, after completing a two-year Postdoctoral Fellowship at the National Engineering Research Center of Computer Integrated Manufacturing System (CIMS-ERC), where he is currently a Professor with CIMS-ERC. His research interests include multidisciplinary modeling and collaborative simulation, service-oriented modeling and simulation, and concurrent and collaborative design.