entropy

MDPI

Article

Robustness of Cloud Manufacturing System Based on Complex Network and Multi-Agent Simulation

Xin Zheng and Xiaodong Zhang *

School of Economics and Management, University of Science and Technology Beijing, Beijing 100083, China
* Correspondence: xdzhang@manage.ustb.edu.cn

Abstract: Cloud manufacturing systems (CMSs) are networked, distributed and loosely coupled, so they face great uncertainty and risk. This paper combines the complex network model with multi-agent simulation in a novel approach to the robustness analysis of CMSs. Different evaluation metrics are chosen for the two models, and three different robustness attack strategies are proposed. To verify the effectiveness of the proposed method, a case study is then conducted on a cloud manufacturing project of a new energy vehicle. The results show that both the structural and process-based robustness of the system are lowest under the betweenness-based failure mode, indicating that resource nodes with large betweenness are most important to the robustness of the project. Therefore, the cloud manufacturing platform should focus on monitoring and managing these resources so that they can provide stable services. Under the individual server failure mode, system robustness varies greatly depending on the failure behavior of the service provider: Among the five service providers (S1–S5) given in the experimental group, the failure of Server 1 leads to a sharp decline in robustness, while the failure of Server 2 has little impact. This indicates that the CMS can protect its robustness by identifying key servers and strengthening its supervision of them to prevent them from exiting the platform.

Keywords: cloud manufacturing; robustness; complex network; multi-agent simulation

Citation: Zheng, X.; Zhang, X. Robustness of Cloud Manufacturing System Based on Complex Network and Multi-Agent Simulation. *Entropy* **2023**, *25*, 45. https://doi.org/10.3390/e25010045

Academic Editor: Adam Lipowski

Received: 23 November 2022
Revised: 20 December 2022
Accepted: 24 December 2022
Published: 27 December 2022

1. Introduction

In the era of Industry 4.0, advanced information technology such as cloud computing and the Internet of Things has brought profound changes to the manufacturing industry. Li et al. [1] conceptualized a new service-oriented networked manufacturing model known as "cloud manufacturing", which aggregates manufacturing resources and capabilities into the cloud platform, and fully realizes the sharing of manufacturing resources and capabilities through service integration [2].

This concept has received much attention from academia and enterprises, and a great amount of research has already been carried out on various aspects of cloud manufacturing, including its hierarchical structure [3,4], typical features [5], key technologies [5–12], operation modes [13–16] and service portfolio scheduling [17–20].

The cloud manufacturing system (CMS) is networked, distributed and loosely coupled; this creates great uncertainty and interference [21], which is an important issue that the CMS must face and solve. Zhang et al. [22] and Zhu et al. [23] argued that the development of cloud manufacturing is restricted by a lack of trust and security, and blockchain technology provides new ideas for overcoming such restrictions due to its reliability, tamper-proof nature, traceability and high transparency. Further, Laili et al. [24] stated that orders of different tasks affect the CMS. As such, the allocation and scheduling capability of the CMS when facing multiple tasks [25–27] is an important component when considering the robustness of the system. Wang et al. [28] studied the impact of service anomalies on the CMS, proposing a dynamic service composition reconfiguration model when anomalies occur. Liang et al. [29] stated that the complex demands of consumers and the changes in the

dynamic environment pose challenges to common CMS-based scheduling algorithms. They proposed deep reinforcement learning to enable the system to overcome these challenges through continuous training and learning. In their study on the typical problems of cloud manufacturing, Tao et al. [6] proposed that uncertain factors exist within the cloud manufacturing life cycle, which can be classified into five categories: uncertainty of tasks, uncertainty of resource services, uncertainty of quality of service (QoS), uncertainty of the correlation between resource services and uncertainty of other factors. The CMS faces complex and diverse types of uncertain interference. This includes not only common interference found in the traditional manufacturing field, but also unique, uncertain factors found in the cloud manufacturing field [30].

It is of great practical significance for the implementation and deployment of cloud manufacturing projects to (a) accurately identify the impact of uncertain factors on cloud manufacturing, (b) explore the robustness level of the cloud manufacturing process (CMP) under different disturbances and attacks, and (c) further improve the stability and invulnerability of the system. However, existing research on the robustness of the CMS is currently lacking. The complex network approach is commonly used in the study of network robustness, where the entities in the system are abstracted as "nodes", the relationships between entities are "connected edges", and relevant topological parameters (e.g., degree, betweenness, agglomeration coefficient) reflect the structural characteristics of the system as a whole. Commonly used robustness measures include the relative size of connected subgraphs, average distance and network efficiency, among others. To explore the robustness and vulnerability problems of traditional production processes, Li et al. [31] constructed a complex multi-task directed weighted network using (a) nodes to represent equipment, personnel and departments in the production plant, and (b) connected edges to represent logistical and process-based relationships. They took network connectivity and the maximum connected subgraph as the robustness measurement indexes, comparing the robustness of the workshop network under both random and selective interference. Shi et al. [32] and Shi et al. [33] used sensitivity analysis to explore the effects of factors such as the number of nodes, number of interconnected links, interconnection mode, scaling index and load capacity on the interdependent supply network robustness under random and intentional disruptions. The robustness index was selected with consideration given to the heterogeneity of different nodes, and the connected sub-network with the largest size including all kinds of nodes (LACS) is proposed to replace the traditional connected sub-network with the largest size (LCS). Fan et al. [34] proposed a two-layer maintenance support service network composed of an undirected network layer and a directed entity layer, comparing the changes in network robustness under three different partnership construction strategies. Moghaddam and Deshmukh [35] studied the robustness of cyber-physical production systems using a complex network method for the scenarios of cascading and non-cascading disruptions. Based on the characteristics of these two disruption modes, different formulae for measuring robustness metrics were given.

The above research shows that the complex network method can satisfactorily reflect the structural characteristics of the CMS. It also has a wide range of applications (e.g., in manufacturing networks and supply chain networks) because it can be used to obtain the change of robustness index under different network attack strategies. However, because this method abstracts the entities and inter-entity relationships structurally, it is then difficult to reflect the logical judgments and dynamic temporal relationships between entities.

In contrast, simulation models can accurately describe the behavioral interactions and temporal relationships of the entities of manufacturing systems. Simulation research on the CMS is also a popular topic in current research. For example, Zhao et al. [2] designed and implemented an agent-based cloud manufacturing simulation platform. They provided a high-level encapsulation of services in the cloud platform, including a five-layer (i.e., data, low tool, management, upper tool and application) architecture of the cloud platform. In Zhang et al. [36], typical intelligent manufacturing simulation technologies were analyzed from three aspects: manufacturing unit simulation, manufacturing integration simulation

and manufacturing intelligent simulation. Various other perspectives have also been studied, such as cloud service entity packaging [37,38], selection and scheduling [39–41], and trust and security issues [23]. However, despite its importance, simulation-based research on the robustness of the CMS remains rare.

This study proposes a robustness analysis method that combines complex networks with multi-agent simulation to analyze the robustness of the CMS from two perspectives: a static structure and dynamic process. The remainder of the paper is organized as follows. In Section 2, a multi-agent simulation model of the CMP is constructed. Here, the behavioral characteristics and models of several key agents in the CMP are given, and QoS is proposed as a robustness measure. Section 3 explores the robustness of the static topology of the cloud manufacturing network (CMN). Here, the complex network model of cloud manufacturing resources is established through both the order–task relationship and the task–resource relationship, and network efficiency and the largest connected subgraph are proposed as robustness measures. In Section 4, the robustness attack strategies are designed, where a degree-based resource failure mode (ID), betweenness-based resource failure mode (IB) and individual server-based resource failure mode are proposed. In Section 5, a case study of a cloud manufacturing project is presented, and its robustness is studied under different failure modes by combining the multi-agent simulation software Anylogic and Python 3.0 tools. Section 6 provides the conclusions and prospects of this paper.

2. CMP Model and Robustness Measurement Index Based on Multi-Agent Simulation

2.1. Multi-Agent Simulation Model Construction

The CMS includes the cloud platform, cloud task, cloud resource, cloud message, order and other types of subjects, as well as two types of user role: cloud service providers and cloud demanders [3]. As shown in Figure 1, the CMP [1] broadly includes the following:

(1) Cloud service providers unify various types of manufacturing equipment resources and manufacturing capability resources into the cloud platform, depositing them into the cloud resource pool through information transformation, resource sensing, resource access, unified modeling of cloud services and other technology. This bypasses the limitations of space and distance by enabling resources that are originally distributed across the world to be centrally managed and shared.
(2) Cloud demanders submit service requirements (i.e., orders) to the cloud platform through terminal devices. Orders from multiple cloud demanders are uniformly stored in the cloud demand set, waiting to be processed.
(3) According to the service route of the order to be processed, the cloud platform integrates and adapts different cloud tasks to form orderly and stable cloud task sequences.
(4) When the cloud demand set is not empty, the platform imports each order into the corresponding cloud task sequence, in turn, to carry out cloud manufacturing services. When the cloud tasks are being processed, the corresponding resources are requested from the resource pool according to the task type. Resources in an idle state change to a busy state after being requested. After the task is completed, the resource is released and returned to an idle state.

The CMS contains multiple entity types, and various forms of information transmission and behavior interaction occur between the same entities and different entities. Therefore, the CMS model can be expressed as follows:

$$CMS = \{PA, DA, SA, TA, RA, OA, MA, E\} \quad (1)$$

where PA is the cloud platform agent; DA is the cloud demander agent; SA is the cloud service agent; TA is the cloud task agent; RA is the cloud resource agent; OA is the order agent issued by the DA; MA is the message agent sent to the SA when the TA requests or releases resources; and E is the external environment of information transmission and behavior interaction among entities.

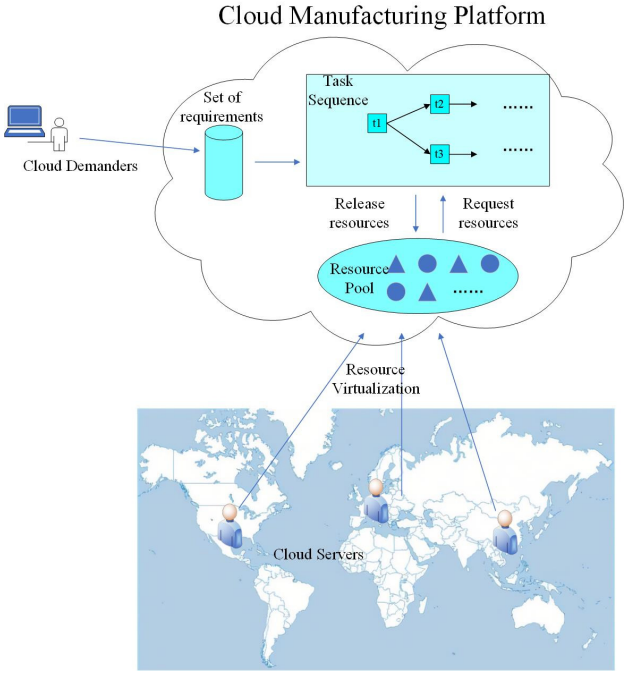


Figure 1. Schematic diagram of cloud manufacturing process.

2.1.1. Cloud Platform Modeling

The cloud platform is the center of the CMS. The cloud demander sends orders to the cloud platform, and the platform assigns these orders to the corresponding tasks. Throughout the CMP, the platform records any processing successes and failures, and at the end of the service cycle, relevant performance indicators (e.g., service time, cost, reliability, order completion rate) are calculated. Additionally, the platform carries out a variety of roles, such as model parameter initialization and experimental parameter adjustment. The cloud platform agent can be expressed as follows:

$$\begin{aligned} PA =< & tobeProcessedOrderList, finishedOrderList, \\ & failedOrderList, attackNum, Func_{ini}, Func_{allocationOrders}, Func_{settingFaultStatus}, \\ & Func_{calcuQos}, Func_{outputNetwork} > \end{aligned} \tag{2}$$

where $tobeProcessedOrderList$ stores orders sent by each cloud demander that are to be processed; $finishedOrderList$ records successfully completed orders; $failedOrderList$ records failed orders; $attackNum$ is the number of failed resources; $Func_{ini}$ is used to initialize model parameters; $Func_{allocationOrders}$ assigns orders to their respective corresponding cloud tasks for processing; $Func_{settingFaultStatus}$ sets the specified resource of the specified server to the failure state according to the node failure mode; $Func_{calcuQos}$ counts data related to service time, cost, reliability and the order completion rate at the end of simulation, then integrates these to calculate the QoS index; and $Func_{outputNetwork}$ sorts the order–task relationship and task–resource relationship of the processed orders into a node-list form and outputs it, to be used to construct the complex network model.

2.1.2. Cloud Resource Modeling

Cloud resources are the virtual resources formed by integrating the manufacturing equipment resources and manufacturing capability resources of service providers into the cloud resource pool through information transformation, resource access, cloud service unified modeling and other technology. The main function of the cloud resource agent is to cooperate with cloud tasks to complete the processing of cloud orders:

$$RA =< ID, produceLevel, busy, broken, owner, price > \tag{3}$$

where ID is the unique identifier number of the resource; *produceLevel* is the productivity level of the resource, which is defined as an integer from 1–10; *busy* indicates whether the resource is in a busy state; *broken* indicates whether the resource is faulty; *owner* specifies which cloud server the resource belongs to; and *price* represents the cost of the resource, which is randomly generated with normal distribution during model initialization.

2.1.3. Cloud Task Modeling

The construction of the cloud task agent is key to cloud manufacturing simulation modeling. It covers not only the processing path of all order types (e.g., serial, parallel, hybrid path) but also the behavior interaction and information transfer between the cloud server agent and the cloud resource agent. In addition, the cloud task agent formulates (a) the selection mechanism of the optimal service provider, (b) various statistical data, such as service cycle and cost, and (c) cloud task and cloud resource node information. To achieve this, existing process modeling library components are adapted accordingly. The cloud task agent can be expressed as follows:

$$\begin{aligned} TA = < ID, ownerOrders, pretaskList, aftertaskList, requeresourceList, basicWorkingTime, \\ currentOrder, Func_{selectBestServer}, Func_{selectBestResource}, Func_{recordRouteStamp}, Func_{recordTaskTime}, \\ Func_{recordTaskCost}, Func_{recordTaskReliability} > \end{aligned} \tag{4}$$

where ID is the unique identification number of the task; *ownerOrders* specifies which type of order processing path the task belongs to; *pretaskList* and $aftertaskList$ specify the pre-order task and post-order task, respectively; *requeresourceList* specifies the type of resource requested by the task; *basicWorkingTime* specifies the standard time for completing the task; *currentOrder* represents the order currently being processed; $Func_{selectBestServer}$ determines the optimal server based on resource price, logistics, distance and other factors; $Func_{selectBestResource}$ determines the optimal resource; $Func_{recordRouteStamp}$ records the order–task relationship and task–resource relationship for completed orders; and $Func_{recordTaskTime}$, $Func_{recordTaskCost}$ and $Func_{recordTaskReliability}$ record the service time, service cost and service reliability of the current task, respectively.

Figure 2 shows the detailed CMP simulation inside the cloud task agent, which is realized by editing and adapting the existing component codes from Anylogic's process modeling library. The details of this process are as follows:

(1) The order is imported into the internal process of the cloud task through the *enter* component. If the current task is first in the task sequence, the order is directly assigned by the cloud platform; otherwise, the order is assigned by the preceding task after its completion (e.g., task 2 orders are assigned from task 1 once task 1 has been completed).

(2) The *queue* component temporarily stores the current order while the following judgments are made: (a) if the current task is first in the task sequence or there is only one task in the preceding task sequence, the *hold* and *hold1* components are simultaneously opened and the current order is entered into *queue2* for subsequent processing; or (b) if there is more than one task in the preceding task sequence, the current order must wait until the orders of all preceding tasks have completed before entering *queue2* for subsequent processing

(3) The *queue1* component merges the information of several branch orders, and the *hold2* component ensures that only one order is entered for subsequent processing at a time. When the current order is completed and exits through *exit*, *hold2* opens again and continues to serve the next order.

(4) The order enters *queue3*, where the task agent selects the optimal server and sends "resource request" information to it. When the optimal service provider accepts the request, the busy attribute corresponding to the optimal resource changes to "true", and the *hold3* component opens. The order flows through the *delay* component to

simulate the cloud manufacturing service. After a certain delay time, the service is completed.

(5) The order enters *queue4* and continues to send the "release resource" message to the optimal server. When the optimal server accepts the message, the busy attribute corresponding to the optimal resource changes to "false", and the *hold4* component opens. The order flows through the *delay1* component. After a certain delay time, the release of the resource is completed.

(6) The order flows through the *exit* component to complete all its service processes in this task. It then imports the post-order task sequence of this task: (a) if there is only one post-order task, it is directly imported into the *enter* component of the post-order task; (b) if there are multiple post-order tasks, the information of the current order is copied and imported into the *enter* component of the respective post-order tasks; and (c) if there is no post-order task, this signifies that the task is already the final task in the task sequence. As such, the order is added to the set of completed orders, and information such as the service cycle, service cost and route record are counted and output.

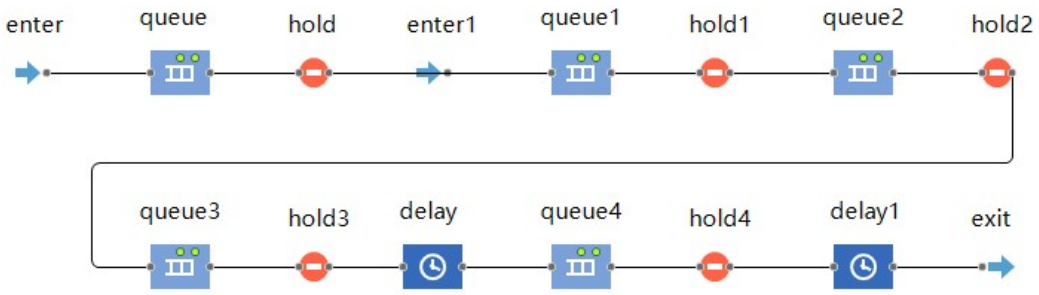


Figure 2. Internal process of cloud task agent.

2.1.4. Cloud Order Modeling

The orders are submitted to the cloud platform by the cloud demanders through terminal devices. They are then imported to the corresponding cloud tasks according to their respective task routes to complete service processing. The cloud order agent is represented as follows:

$$OA =< ID, owner, taskList, routeStamp, cost1Accum, cost2Accum, cost3Accum, reliabilityAccum, startTime, finishTime > \quad (5)$$

where ID is the order's unique identification number; *owner* specifies which demander the order is issued by; *taskList* specifies the complete task path corresponding to the order; *routeStamp* records the order–task relationship and task–resource relationship corresponding to the order when the order is completed; $cost1Accum$, $cost2Accum$ and $cost3Accum$ record the request resource cost, logistics cost and release resource cost of the order, respectively; $reliabilityAccum$ records the completion reliability of orders; and $startTime$ and $finishTime$ record the start processing time and completion processing time of the order, respectively.

2.1.5. Cloud Message Modeling

When processing orders, cloud tasks need to send "request resource" information to the cloud server. After the processing is complete, a "release resource" message is sent to the cloud server. Since Anylogic's built-in message agent cannot carry extra information, this paper encapsulates a messages agent type, which can be expressed as follows:

$$MA = \langle msg, resourceList, owner \rangle \quad (6)$$

where *msg* is the content of the message (i.e., when a resource is requested, the msg's value is "request", and when a resource is released, the msg's value is "release"); *resourceList* specifies which resources are to be requested and released; and *owner* indicates which cloud task sent the message.

2.1.6. Cloud Demander Modeling

The cloud demander issues demand orders to the cloud platform to drive the operation of the model. The cloud demander agent can be represented as follows:

$$DA = \langle ID, location, orderList, Func_{sendOrders} \rangle \tag{7}$$

where ID is the cloud demander's unique identification number; *location* is the latitude and longitude coordinates of the demander, which is used to initialize the location of the demander in the GIS map; *orderList* is used to initialize the orders issued by the demander; and $Func_{sendOrders}$ sends the orders of cloud demanders to the cloud platform, where the cloud platform schedules and allocates the orders uniformly.

2.1.7. Cloud Server Modeling

Cloud servers mainly transfer information and interact with cloud tasks. All types of cloud resources are stored in the resource pool of each cloud server. When receiving the "request resource" message, the server finds the corresponding resource in its resource pool and allocates it to the cloud task. When receiving the "release resource" message, the server releases the corresponding cloud resource and puts it back into the cloud resource pool. The cloud server agent can be represented as:

$$\begin{aligned} SA =< ID, location, resourcePool, dScore, \\ pScore, totalScore, Func_{configureResource} > \end{aligned} \tag{8}$$

where ID is the unique identification number of the cloud service provider; *location* is the latitude and longitude coordinates of the server, which is used to initialize the location of the server in the GIS map; *resourcePool* is used to store the respective virtual resources of the cloud server; *dScore*, *pScore* and *totalScore* are the respective distance score, price score and total score when the cloud task selects the optimal cloud server; and $Func_{configureResource}$ is used to execute and allocate resources when receiving information about cloud tasks. If the message's content is "release resource", the server finds the corresponding resource and changes its busy attribute to "false". If the message's content is "request resource", the corresponding resource is judged as follows: (a) if its value is "true", this indicates that the resource is faulty and the cloud task cannot be completed; as such, the order requested for processing is classified as failed; or (b) if the value is "false", this indicates that the resource is not faulty. Here, its busy attribute is judged as follows: (i) if the busy attribute is "false", the processing of the cloud order can be started, and (ii) if the busy attribute is "true", this indicates that the resource is being invoked by other tasks and it needs to wait until the other task is completed.

2.2. Robustness Measurement Index Based on Multi-Agent Simulation

Based on the multi-agent model and order task sequence stated in Section 2.1, the dynamic simulation of the CMP can be realized. At the end of the simulation, the order completion time, logistics transportation distance, resource occupation and other data can be output to evaluate the performance. QoS is commonly used to evaluate the CMP. As such, based on the combination of relevant literature and the simulation output data, this paper comprehensively evaluates the QoS value from four aspects: service time, service cost, service reliability and order completion rate [20,42,43]. The calculation formulae of these four indicators are as follows:

(1) Service time

This is the sum of the completion times of all orders within the simulation cycle, which can be expressed as

$$T = \sum_{j=1}^{m} t_j \tag{9}$$

where m is the total number of orders; j = 1, 2, . . . , m is the jth order in the order sequence; and t_j is the completion time of the jth order, which can be obtained by the simulation results.

(2) Service cost

This total cloud service cost is calculated from three aspects: the cloud resource service fee, logistics service fee and cloud resource release fee, which can be expressed as

$$C = \sum_{j=1}^{m}\sum_{i=1}^{n_j} t_{i,j}^{serving} * p_{i,j}^{resource} + \sum_{j=1}^{m}\sum_{i=1}^{n_j} d_{i,j} * c^{logistic} + \sum_{j=1}^{m}\sum_{i=1}^{n_j} t_{i,j}^{releasing} * p^{release} \tag{10}$$

where m is the total number of orders; n is the number of tasks corresponding to each order; j = 1, 2, . . . m represents the jth order in the order sequence; i = 1,2, . . . n represents the ith task in the task sequence; $t_{i,j}^{serving}$ is the cloud service time of the ith task in the jth order; $p_{i,j}^{resource}$ is the service cost per unit time of the resource corresponding to the task; $d_{i,j}$ is the logistics distance corresponding to the task; $c^{logistic}$ is the logistics cost per unit distance; $t_{i,j}^{releasing}$ is the release time of the cloud resources for the task; and $p^{release}$ is the cost per unit time of releasing resources.

(3) Service reliability

Service reliability is a multiplicative index [42], which can be expressed as

$$rel = \frac{\sum_{j=1}^{m}\prod_{i=1}^{n_j} r_{i,j}}{m} \tag{11}$$

where $r_{i,j}$ is the service reliability of the i-th task in the j-th order, which is given in the *reliabilityAccum* attribute of the order agent.

(4) Order completion rate

The order completion rate is the ratio of the number of completed orders within the simulation cycle to the total number of orders planned to be completed:

$$ofr = \frac{N_1}{N_1 + N_2} \tag{12}$$

where N_1 is the number of orders completed within the simulation cycle and N_2 is the number of orders that failed to be completed.

In addition, the index values need to be standardized to consider the different index dimensions. A series of robustness experiments will be carried out later in this paper, so range standardization is carried out with the index values of each experiment as samples. The calculation formulae are as follows:

$$nt = \frac{T_k - T_{min}}{T_{max} - T_{min}} \tag{13}$$

$$nc = \frac{C_k - C_{min}}{C_{max} - C_{min}} \tag{14}$$

where T_k and C_k are the respective service time and service cost of the k-th experiment; T_{min} and T_{max} are the respective minimum and maximum values of service time; and C_{min}

and C_{max} are the respective minimum and maximum values of service cost. The reliability and order completion rate indicators are originally in the range of [0, 1], so there is no need for standardization.

The QoS value can be evaluated by synthesizing the above four dimensions:

$$QoS = \omega_1 * nt + \omega_2 * nc + \omega_3 * rel + \omega_4 * ofr \tag{15}$$

where ω_1, ω_2, ω_3 and ω_4 are the respective weight coefficients of the four indicators, and $\sum_{i=1}^{4} \omega_i = 1$.

3. Cloud Manufacturing Network Model and Robustness Measurement Index-Based on Complex Network

3.1. Construction of Cloud Manufacturing Complex Network Model

The cloud manufacturing network (CMN) is composed of cloud service resources and the connections between resources. Due to the large number of resources and complex connection relationships, the network can be analyzed using the complex network model. Figure 3a shows the processing task paths of Order-A and Order-B, the resources used by each task in these paths, and the corresponding relationships between resources and servers. If two tasks are connected on a path, the respective resources used by the two tasks are also considered to be connected. Figure 3b shows how the CMN is formed by taking all the resources as network nodes and the connections between resources as connected edges.

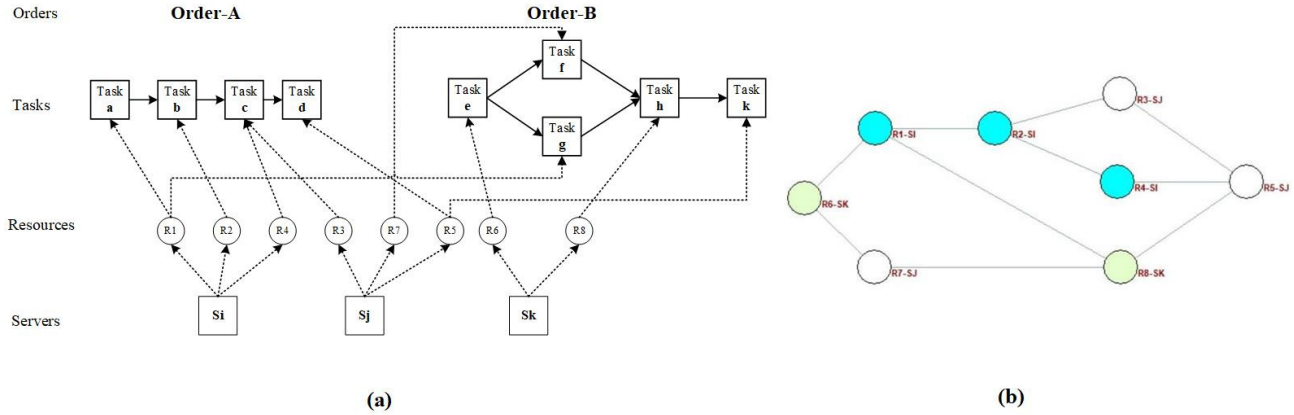


Figure 3. Schematic diagram of (**a**) internal relationship of the cloud manufacturing system (CMS) and (**b**) cloud resource network.

3.2. Robustness Measurement Index Based on Static Network Topology

It is generally considered that network robustness refers to the degree of network performance retention after the failure of network nodes or edges [44], and the change of the maximum connected subgraph after network node failure can reflect the degree of retention of the network's structural integrity. As such, the change rate of the maximum connected subgraph's node number is selected as one of the robustness evaluation indexes in this study:

$$S = \frac{N'}{N} \tag{16}$$

where N' is the number of nodes in the maximum connected subgraph after the network is attacked, and N is the total number of nodes in the original network. In particular, $S = 0$ indicates that the network is in an unconnected state; and $S = 1$ indicates that the network is fully connected, and there is no isolated node.

Additionally, the connections between the nodes change when a network node fails. This, in turn, affects the efficiency of information dissemination in the network. Therefore, network efficiency is used to evaluate the robustness of the network transfer efficiency

when nodes are lost. The shorter the distance between two nodes in a network, the faster information can be transferred from one node to another. Based on this, the formula of network efficiency can be defined as:

$$\Phi_G = \frac{1}{N(N-1)} \sum_{i \neq j} \frac{1}{d_{ij}} \tag{17}$$

where N represents the total number of nodes in the network and d_{ij} represents the shortest path between node i and node j. In particular, $G = 0$ indicates the efficiency of the network is the worst, where the whole network contains isolated nodes, and $G = 1$ indicates that the efficiency of the network is the best, where the information exchange between nodes is smooth.

4. Failure Mode Design for Robustness Analysis

The design of failure modes is key to robustness analysis. Based on the characteristics of cloud manufacturing, this paper proposes a topology-based resource failure mode and a server-based resource failure mode.

Topology-based resource failures are further divided into degree-based and betweenness-based resource failures, where (a) node degree (i.e., how closely a resource node is connected to other resource nodes in the CMN) is commonly used to measure a node's importance, and (b) node betweenness reflects the structural importance of the node [44,45], with a node with high betweenness having greater control over logistics and information flow in the network. The specific topology-based resource failure mode designs are shown in Table 1.

Table 1. Design of topology-based resource failure modes.

	Failure Mode Description	**Failure Mode Calculation Process**
Topology-based resource failure modes	Initial node degree loss (ID)	Sort the resource nodes in the initial network by degree, from largest to smallest. Remove one node at a time, and repeat n times until all nodes in the network are removed.
	Initial node betweenness loss (IB)	Sort the resource nodes in the initial network by betweenness, from largest to smallest. Remove one node at a time, and repeat n times until all nodes in the network are removed.

Note: The removal of nodes is performed differently in the complex network model and the multi-agent model: (a) the complex network model is performed by deleting the corresponding resource nodes and all connected edges on the nodes, and (b) the multi-agent model is represented by setting the corresponding resource agent to a "fault" state, that is, where the resource cannot provide services.

Server-based resource failures fully consider the realistic scenario of cloud manufacturing, where resource nodes involved in cloud manufacturing belong to different cloud servers. The successive failure of resource nodes of the same cloud server can simulate the scenario where the cloud server gradually exits the platform and no longer provides resources. Key cloud servers can be identified by comparing the robustness indexes of different cloud servers after the loss of resources, and focused monitoring and management of these key servers can effectively ensure the robustness of the CMN. The specific server-based resource failure mode designs are shown in Table 2.

Table 2. Design of server-based resource failure modes.

	Failure Mode Description	Failure Mode Calculation Process
Server-based resource failure modes	Successive failure of Server-1′s node (group S1)	Select the resource nodes belonging to server S1 in the CMN. Remove one node at a time, and repeat n times until all resource nodes belonging to server S1 in the network are removed.
	Successive failure of Server-2′s node (group S2)	Select the resource nodes belonging to server S2 in the CMN. Remove one node at a time, and repeat n times until all resource nodes belonging to server S2 are removed.
	.....	.....
	Successive failure of Server-n′s node (group Sn)	Select the resource nodes belonging to server Sn in the CMN. Remove one node at a time, and repeat n times until all resource nodes belonging to server Sn are removed.

5. Case Study

5.1. Description of Model Parameters

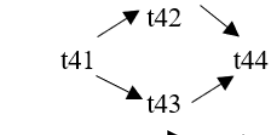
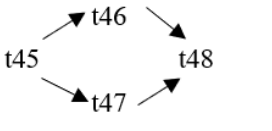
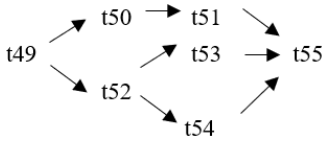
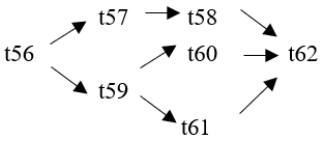
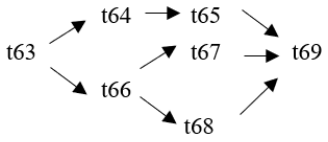
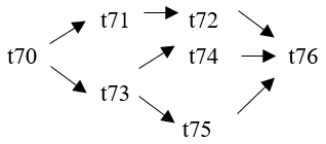
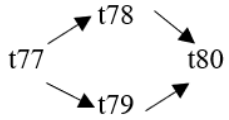
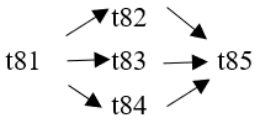
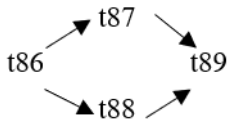
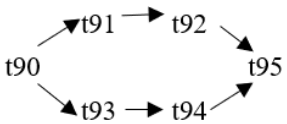
A case study is carried out using the cloud manufacturing project of a new energy vehicle. This project provides life-cycle cloud manufacturing services for new energy vehicles, where the vehicles served are equipped with technology such as electrification and autonomous driving.

The cloud manufacturing project includes 24 order types, 95 cloud tasks (t1–t95) and 72 resource types (r1–r72). The corresponding resource types for each cloud task are shown in Table 3, and the cloud task routes of each order type are shown in Table 4.

Table 3. Correspondence between tasks and resources.

Task	Resource	Task	Resource	Task	Resource	Task	Resource
t1	(r1)	t2	(r3)	t3	(r2)	t4	(r4)
t5	(r11, r30)	t6	(r5)	t7	(r12, r29)	t8	(r6)
t9	(r12, r29)	t10	(r31)	t11	(r11, r30)	t12	(r67)
t13	(r32)	t14	(r1)	t15	(r7)	t16	(r1)
t17	(r2)	t18	(r8)	t19	(r2)	t20	(r5)
t21	(r5)	t22	(r41)	t23	(r6)	t24	(r6)
t25	(r71)	t26	(r42)	t27	(r33)	t28	(r9)
t29	(r9)	t30	(r34)	t31	(r10)	t32	(r10)
t33	(r9)	t34	(r7)	t35	(r9)	t36	(r7)
t37	(r10)	t38	(r8)	t39	(r10)	t40	(r8)
t41	(r13, r14)	t42	(r61)	t43	(r21, r23)	t44	(r51, r52)
t45	(r15, r16)	t46	(r62)	t47	(r22, r24)	t48	(r53, r54)
t49	(r21, r23)	t50	(r47, r48)	t51	(r47, r48)	t52	(r33)
t53	(r35, r37)	t54	(r63)	t55	(r43, r45)	t56	(r22, r24)
t57	(r49, r50)	t58	(r49, r50)	t59	(r34)	t60	(r36, r38)
t61	(r64)	t62	(r44, r46)	t63	(r35, r37)	t64	(r41)
t65	(r41)	t66	(r39)	t67	(r25, r27)	t68	(r65)
t69	(r13, r14)	t70	(r36, r38)	t71	(r42)	t72	(r42)
t73	(r40)	t74	(r26, r28)	t75	(r66)	t76	(r15, r16)
t77	(r47, r48)	t78	(r51, r52)	t79	(r57, r58)	t80	(r47, r48)
t81	(r49, r50)	t82	(r53, r54)	t83	(r68)	t84	(r59, r60)
t85	(r49, r50)	t86	(r57, r58)	t87	(r17, r19)	t88	(r63)
t89	(r55)	t90	(r59, r60)	t91	(r18, r20)	t92	(r69)
t93	(r64)	t94	(r70)	t95	(r56)		

Table 4. Correspondence between orders and cloud service routes.

Orde Type	Cloud Service Route	Order Type	Cloud Service Route
Order 11	t1 → t2	Order 12	t3 → t4
Order 13	t5 → t6	Order 14	t7 → t8
Order 15	t9 → t10	Order 16	t11 → t12 → t13
Order 21	t14 → t15 → t16	Order 22	t17 → t18 → t19
Order 23	t20 → t21 → t22	Order 24	t23 → t24 → t25 → t26
Order 25	t27 → t28 → t29	Order 26	t30 → t31 → t32
Order 31	t33 → t34 → t35 → t36	Order 32	t37 → t38 → t39 → t40
Order 33		Order 34	
Order 41		Order 42	
Order 43		Order 44	
Order 51		Order 52	
Order 53		Order 54	

A total of 5 cloud servers (S1–S5) participate in the project, with each server providing 72 types of cloud resources. The cloud servers have different pricing of resources, and they are at different distances from the cloud demanders, so they compete for different orders. To distinguish between them, resource r1 of servers S1–S5 are labeled r1-S1, r1-S2, r1-S3, r1-S4 and r1-S5, and so on.

In addition, there are 14 cloud demanders (d1–d14). Each cloud demander submits 24 orders, and the number of orders of each type is 1 (i.e., 1 of each of the 24 order types). The basic information of each cloud service provider and cloud demander is externally imported from Excel, as shown in Table 5.

Table 5. Attributes of cloud servers and cloud demanders.

ID	City	Location (Latitude, Longitude)	ID	City	Location (Latitude, Longitude)
S1	Beijing	(39.91, 116.41)	d5	Jinan	(36.4, 117)
S2	Shanghai	(31.21, 121.43)	d6	Lanzhou	(36.03, 103.73)
S3	Chengdu	(30.66, 104.06)	d7	Wulumuqi	(43.76, 87.68)
S4	Hangzhou	(30.26, 120.2)	d8	Changsha	(28.21, 113)
S5	Shenzhen	(22.61, 114.06)	d9	Nanchang	(28.68, 115.9)
			d10	Fuzhou	(26.08, 119.3)
d1	HaErbin	(45.75, 126.63)	d11	Nanning	(22.48, 108.19)
d2	ShenYang	(41.8, 123.38)	d12	Lasa	(29.6, 91)
d3	Baotou	(40.39, 109.49)	d13	Lianyungang	(34.36, 119.1)
d4	Tianjin	(39.13, 117.2)	d14	Hefei	(31.52, 117.17)

In this paper, the weight coefficient of QoS is set as $\omega_1 = 0.35$, $\omega_2 = 0.35$, $\omega_3 = 0.1$, $\omega_4 = 0.2$.

5.2. Structural Robustness Analysis

The cloud resource network is obtained according to the network model construction method stated in Section 3.1, as shown in Figure 4. Matlab-2020a software is used to perform data statistics on the network, and both the relevant network topology parameters and the degree distribution are obtained, as shown in Table 6 and Figure 5, respectively.

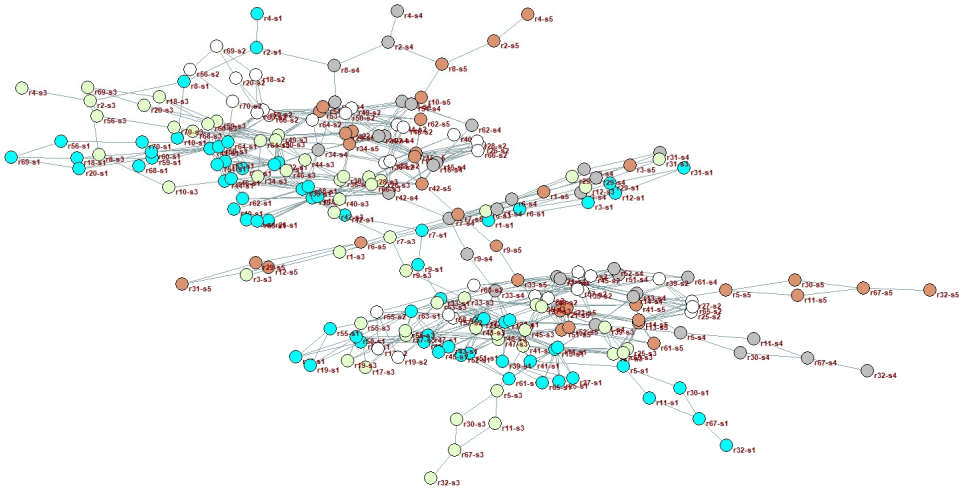


Figure 4. Spatial layout of cloud resource network.

Table 6. Topological parameters of cloud resource network.

Topological Parameter	Number of Nodes	Average Degree	Density	Average Path Length
Cloud resource network	231	21.208	0.087	4.231

Notes: (1) The average degree is the average of the degree of all nodes in the network. (2) The density of a network can be expressed as the ratio of the actual number of edges to the maximum possible number of edges in the network. (3) The average path length is the average of the distance between any two nodes in the network.

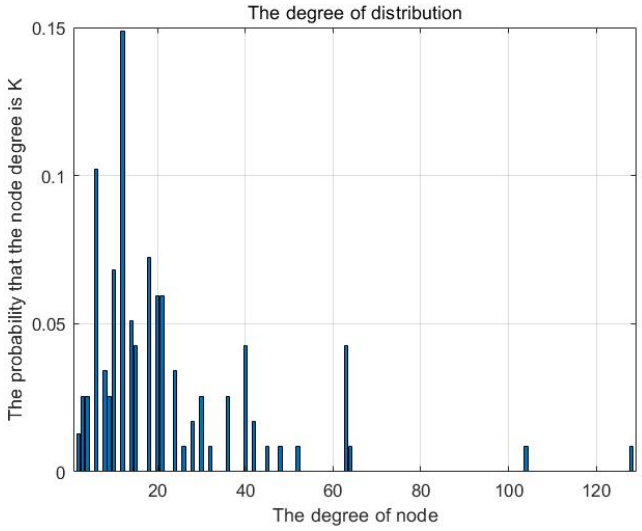


Figure 5. Degree distribution of cloud resource network.

There are 231 resource nodes in the network, and the distribution of node degrees is seriously uneven. A few nodes occupy the majority of connected edges, indicating that

the network has the typical characteristics of a scale-free network. The small density of the network indicates that it is a sparse network, with the nodes with higher degree values tending to connect the nodes with lower degree values.

Next, based on the failure modes designed in Section 4, Python 3.0 is used to simulate and calculate the changes of structural robustness indexes under the different failure modes.

The calculation results based on the topology-based failure mode are shown in Figure 6. This shows that (a) both network efficiency and the largest connected subgraph gradually decrease with the increase of the node failure ratio, and (b) the index value in the IB mode is always lower than that in the ID mode, which indicates that the robustness of the CMN is more fragile in the IB mode. Further, in the IB mode, when the failure ratio is around 0.05, there is a precipitous decline in robustness. In contrast, the overall decline trend of robustness in the ID mode is relatively stable. When the failure ratio reaches 0.4, the maximum connected subgraph in the IB mode drops below 50, and the network efficiency drops below 0.05, which indicates that the network is in a state of collapse. This shows that from the perspective of complex networks, resource nodes with large betweenness are more important to the maintenance of the structural robustness of CMNs. As such, the focus should first be on protecting those nodes with larger betweenness, followed by those nodes with a larger degree.

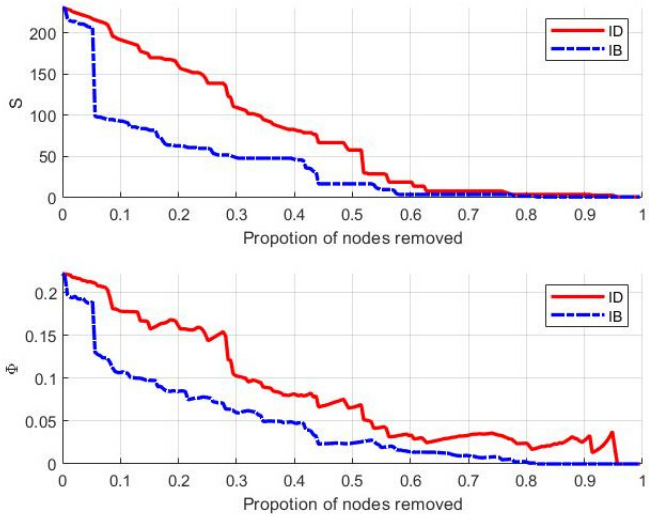


Figure 6. Trend of S and Φ based on topology-based failure mode.

The robustness analysis results under the cloud server failure mode are shown in Figure 7. This shows that (a) the network efficiency value shows a fluctuating trend during the failure of cloud server nodes, and there is no significant decline; however, the maximum connected subgraph decreases rapidly with the increase of the node failure ratio, which indicates that the maximum connected subgraph is more sensitive to the failure of the cloud server; and (b) the maximum connected subgraph of S1 not only decreases the most out of the 5 servers (from 231 to 161), but it also shows a significant decline when the node failure ratio is in the [0.05, 0.1] interval. Further, the maximum connected subgraph of S3 decreases from 231 to 175, and the respective maximum connected subgraphs of S2, S4 and S5 are all above 195 after being attacked. From the perspective of the complex network, the key cloud servers are selected as S1 followed by S3. These servers should be monitored and managed to protect the structural robustness of the CMN.

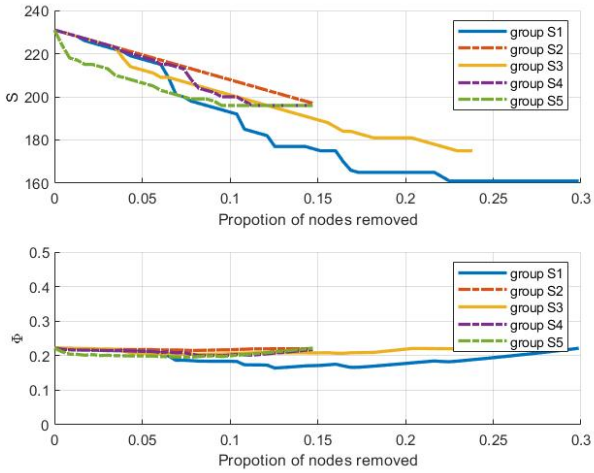


Figure 7. Trend of S and Φ based on cloud server failure mode.

5.3. Process Robustness Analysis

In addition to the process robustness measures and failure modes, this paper uses multi-agent simulation software Anylogic and Python 3.0 to explore the variations of QoS under the different failure modes, as shown in Figure 8.

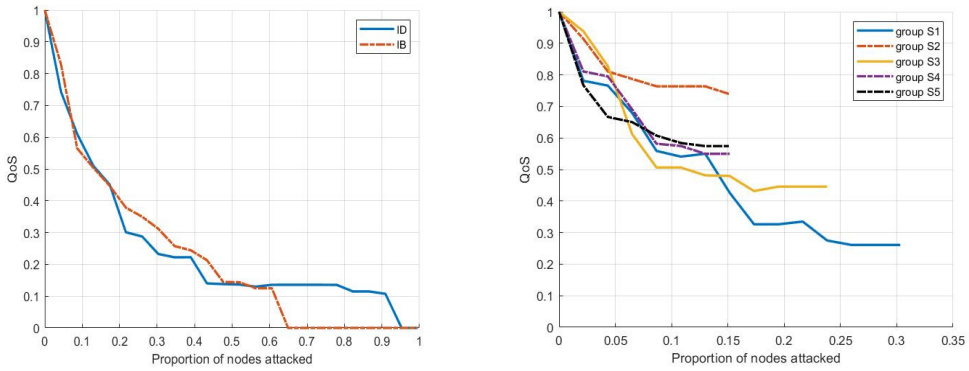


Figure 8. Trend of QoS change based on (**a**) topology-based failure mode and (**b**) cloud server failure mode.

As shown in Figure 8a, QoS rapidly decreases with the increase of the node failure ratio in both the IB and ID modes. For both modes, the QoS values drop below 0.25 when the node failure ratio is about 0.4. However, in the IB mode, all cloud orders fail to be processed when the node failure ratio is about 0.65, whereas in the ID mode, this occurs only when the node failure ratio is about 0.95. This shows that the robustness of the CMP is more fragile in the IB mode. From the perspective of multi-agent simulation, nodes with large betweenness are more important for maintaining the robustness of the dynamic processes of cloud manufacturing.

As shown in Figure 8b, the QoS index of S1 decreases the most (from 1 to less than 0.3), followed by S3 (from 1 to 0.44). Further, the QoS indexes of S2, S4 and S5 are all above 0.5 after being attacked, which indicates that the resource failure of these servers has little impact on the robustness of the CMP. From the perspective of multi-agent simulation, S1 and S3 are the key cloud servers, which is consistent with the evaluation results from the complex network perspective stated in Section 5.2.

6. Conclusions

This paper proposes a novel approach for the robustness analysis of the CMS, combining complex network analysis with multi-agent simulation, which extends the robustness

analysis object from the CMN to the CMP. First, a multi-agent simulation model is constructed. The behavioral characteristics and models of several key agents in the CMP are detailed, and QoS is proposed as a robustness measure. Second, a complex network model of cloud manufacturing resources is established through both the order–task relationship and the task–resource relationship to investigate the robustness of the static topology of the CMN. For this, network efficiency and the maximum connected subgraph are proposed as robustness measures. Three attack strategies are designed, which are resource failure modes based on the degree, betweenness and individual server. To verify the feasibility and effectiveness of the proposed method, a case study is then conducted on a cloud manufacturing project of a new energy vehicle. The results show that the robustness of the system (both for the CMN and the CMP) is lowest under the betweenness-based resource failure mode. This indicates that resource nodes with large betweenness are most important to the structural robustness and process robustness of the project. As such, the CMP should focus on monitoring and managing these cloud manufacturing resources so that they can provide stable services. Under the server-based failure mode, the robustness of the system varies greatly depending on the failure behavior of the service provider (e.g., the failure of S1 leads to a sharp decline in robustness, but the failure of S2 has little impact). This indicates that the CMS can protect its robustness by identifying key servers and strengthening its supervision of them to prevent them from exiting the platform.

This paper primarily focuses on how various failure modes affect the performance of the CMS, and it proposes related robustness analysis methods and protection measures. Future research based on this established complex network and multi-agent simulation model will involve the design of corresponding recovery strategies and elasticity measures of the CMS. Simulation research will also be carried out to provide a quantitative and dynamic decision basis for the improvement of the robustness of the cloud manufacturing platform.

Author Contributions: Conceptualization, X.Z. (Xin Zheng); methodology, X.Z. (Xiaodong Zhang); software, X.Z. (Xin Zheng); formal analysis, X.Z. (Xin Zheng); investigation, X.Z. (Xin Zheng); resources, X.Z. (Xin Zheng); data curation, X.Z. (Xin Zheng); writing—original draft preparation, X.Z. (Xin Zheng); writing—review and editing, X.Z. (Xin Zheng), X.Z. (Xiaodong Zhang); visualization, X.Z. (Xin Zheng); supervision, X.Z. (Xiaodong Zhang); project administration, X.Z. (Xiaodong Zhang). All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China (No. 71871018).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The used and analyzed datasets during the present study are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Li, B.; Zhang, L.; Wang, S.; Tao, F.; Cao, J.; Jiang, X.; Song, X.; Chai, X. Cloud manufacturing: A new service-oriented networked manufacturing model. *Comput. Integr. Manuf. Syst.* **2010**, *16*, 1–7.
2. Zhao, C.; Zhang, L.; Liu, Y.; Zhang, Z.; Yang, G.; Li, B.H. Agent-based simulation platform for cloud manufacturing. *Int. J. Model. Simul. Sci. Comput.* **2017**, *8*, 1742001. [CrossRef]
3. Li, B.; Zhang, L.; Ren, L.; Chai, X.; Tao, F.; Luo, Y.; Wang, Y.; Yin, C.; Huang, G.; Zhao, X. Further discussion on cloud manufacturing. *Comput. Integr. Manuf. Syst.* **2011**, *17*, 449–457.
4. Li, B.; Zhang, L.; Chai, X. Introduction to Cloud Manufacturing. *ZTE Technol. J.* **2010**, *16*, 5–8.
5. Li, B.; Zhang, L.; Ren, L.; Chai, X.; Tao, F.; Wang, Y.; Yin, C.; Huang, P.; Zhao, X.; Zhou, Z. Typical characteristics, technologies and applications of cloud manufacturing. *Comput. Integr. Manuf. Syst.* **2012**, *18*, 1345–1356.
6. Tao, F.; Zhang, L.; Guo, H.; Luo, Y.; Ren, L. Typical characteristics of cloud manufacturing and several key issues of cloud service composition. *Comput. Integr. Manuf. Syst.* **2011**, *17*, 477–486.

7. Zhang, L.; Luo, Y.; Tao, F.; Ren, L.; Guo, H. Key technologies for the construction of manufacturing cloud. *Comput. Integr. Manuf. Syst.* **2010**, *16*, 2510–2520.
8. Yin, C.; Huang, B.; Liu, F.; Wen, L.; Wang, Z.; Li, X.; Yang, S.; Ye, D.; Liu, X. Common key technology system of cloud manufacturing service platform for small and medium enterprises. *Comput. Integr. Manuf. Syst.* **2011**, *17*, 495–503.
9. Yao, X.; Yu, M.; Chen, Y.; Xiang, Z. Connotation, architecture and key technologies of Internet of manufacturing things. *Comput. Integr. Manuf. Syst.* **2014**, *20*, 1–10.
10. Luo, Y.; Zhang, L.; Tao, F.; Zhang, X.; Ren, L. Key technologies of manufacturing capability modeling in cloud manufacturing mode. *Comput. Integr. Manuf. Syst.* **2012**, *18*, 1357–1367.
11. Li, W.; Lin, H.; Mo, T.; Chu, W. The technologies in cloud manufacturing. *Manuf. Autom.* **2011**, *33*, 7–10.
12. Li, C.; Shang, Y.; Hu, C. Research of Structure and Key Technologies for Cloud Manufacturing. *Modul. Mach. Tool Autom. Manuf. Tech.* **2011**, *7*, 104–107.
13. Zhang, L.; Luo, Y.; Fan, W.; Tao, F.; Ren, L. Analyses of cloud manufacturing and related advanced manufacturing models. *Comput. Integr. Manuf. Syst.* **2011**, *17*, 458–468.
14. Zhan, D.; Zhao, X.; Wang, S.; Cheng, Z.; Zhou, X.; Nie, L.; Xu, X. Cloud manufacturing service platform for group enterprises oriented to manufacturing and management. *Comput. Integr. Manuf. Syst.* **2011**, *17*, 487–494.
15. Ren, L.; Zhang, L.; Zhang, Y.; Tao, F.; Luo, Y. Resource virtualization in cloud manufacturing. *Comput. Integr. Manuf. Syst.* **2011**, *17*, 511–518.
16. Yao, X.; Jin, H.; Xu, C.; Zhu, J. Virtualization and servitization of cloud manufacturing resources. *J. South China Univ. Technol. Nat. Sci.* **2013**, *41*, 1–7.
17. Li, X.; Fang, Z.; Yin, C. A machine tool matching method in cloud manufacturing using Markov Decision Process and cross-entropy. *Robot Cim.-Int. Manuf.* **2020**, *65*, 101968. [CrossRef]
18. Lim, M.K.; Xiong, W.; Wang, Y. A three-tier programming model for service composition and optimal selection in cloud manufacturing. *Comput. Ind. Eng.* **2022**, *167*, 108006. [CrossRef]
19. Delaram, J.; Houshamand, M.; Ashtiani, F.; Fatahi Valilai, O. A utility-based matching mechanism for stable and optimal resource allocation in cloud manufacturing platforms using deferred acceptance algorithm. *J. Manuf. Syst.* **2021**, *60*, 569–584. [CrossRef]
20. Wang, Y.; Wang, S.; Gao, S.; Guo, X.; Yang, B. Adaptive multi-objective service composition reconfiguration approach considering dynamic practical constraints in cloud manufacturing. *Knowl.-Based Syst.* **2021**, *234*, 107607. [CrossRef]
21. Zeng, Z.; Wu, Q.; Yan, Y. Analysis of Influencing Factors on Cloud Service Composition Flexibility: Based on Perspective of Cloud Manufacturing Service Platform. *Sci. Technol. Manag. Res.* **2019**, *39*, 234–239.
22. Zhang, Y.; Zhang, L.; Liu, Y.; Luo, X. Proof of service power: A blockchain consensus for cloud manufacturing. *J. Manuf. Syst.* **2021**, *59*, 1–11. [CrossRef]
23. Zhu, X.; Shi, J.; Huang, S.; Zhang, B. Consensus-oriented cloud manufacturing based on blockchain technology: An exploratory study. *Pervasive Mob. Comput.* **2020**, *62*, 101113. [CrossRef]
24. Laili, Y.; Lin, S.; Tang, D. Multi-phase integrated scheduling of hybrid tasks in cloud manufacturing environment. *Robot Cim.-Int. Manuf.* **2020**, *61*, 101850. [CrossRef]
25. Liu, S.; Zhang, L.; Zhang, W.; Shen, W. Game theory based multi-task scheduling of decentralized 3D printing services in cloud manufacturing. *Neurocomputing* **2021**, *446*, 74–85. [CrossRef]
26. Li, F.; Liao, T.W.; Zhang, L. Two-level multi-task scheduling in a cloud manufacturing environment. *Robot Cim.-Int. Manuf.* **2019**, *56*, 127–139. [CrossRef]
27. Elgendy, A.; Yan, J.; Zhang, M. Integrated Strategies to an Improved Genetic Algorithm for Allocating and Scheduling Multi-Task in Cloud Manufacturing Environment. *Procedia Manuf.* **2019**, *39*, 1872–1879. [CrossRef]
28. Wang, Y.; Wang, S.; Kang, L.; Wang, S. An effective dynamic service composition reconfiguration approach when service exceptions occur in real-life cloud manufacturing. *Robot. Cim.-Int. Manuf.* **2021**, *71*, 102143. [CrossRef]
29. Liang, H.; Wen, X.; Liu, Y.; Zhang, H.; Zhang, L.; Wang, L. Logistics-involved QoS-aware service composition in cloud manufacturing with deep reinforcement learning. *Robot. Cim.-Int. Manuf.* **2021**, *67*, 101991. [CrossRef]
30. Toro, R.; Correa, J.E.; Ferreira, P.M. A Cloud-Monitoring Service for Manufacturing Environments. *Procedia Manuf.* **2018**, *26*, 1330–1339. [CrossRef]
31. Li, H.; Yuan, Y.; Li, X.; Feng, H. Robustness Analysis of Production Workshop Network Based on Complex Networks. *Modul. Mach. Tool Autom. Manuf. Tech.* **2016**, *9*, 149–152.
32. Shi, X.; Deng, D.; Long, W.; Li, Y.; Yu, X. Research on the robustness of interdependent supply networks with tunable parameters. *Comput. Ind. Eng.* **2021**, *158*, 107431. [CrossRef]
33. Shi, X.; Long, W.; Li, Y.; Deng, D. Robustness of interdependent supply chain networks against both functional and structural cascading failures. *Phys. A Stat. Mech. Appl.* **2022**, *586*, 126518. [CrossRef]
34. Fan, D.; Lin, J.; Cai, B.; Liu, B. Robustness of maintenance support service networks: Attributes, evaluation and improvement. *Reliab. Eng. Syst. Saf.* **2021**, *210*, 107526. [CrossRef]
35. Moghaddam, M.; Deshmukh, A. Resilience of cyber-physical manufacturing control systems. *Manuf. Lett.* **2019**, *20*, 40–44. [CrossRef]
36. Zhang, L.; Zhou, L.; Ren, L.; Laili, Y. Modeling and simulation in intelligent manufacturing. *Comput. Ind.* **2019**, *112*, 103123. [CrossRef]

37. Zhao, C.; Luo, X.; Zhang, L. Modeling of service agents for simulation in cloud manufacturing. *Robot. Cim.-Int. Manuf.* **2020**, *64*, 101910. [CrossRef]
38. Zhao, C.; Wang, L.; Zhang, X. Service agent networks in cloud manufacturing: Modeling and evaluation based on set-pair analysis. *Robot. Cim.-Int. Manuf.* **2020**, *65*, 101970. [CrossRef]
39. Eisa, M.; Younas, M.; Basu, K.; Awan, I. Modelling and Simulation of QoS-Aware Service Selection in Cloud Computing. *Simul. Model Pract. Theory* **2020**, *103*, 102108. [CrossRef]
40. Zhou, L.; Zhang, L.; Ren, L. Modelling and simulation of logistics service selection in cloud manufacturing. *Procedia CIRP* **2018**, *72*, 916–921. [CrossRef]
41. Vespoli, S.; Grassi, A.; Guizzi, G.; Santillo, L.C. Evaluating the advantages of a novel decentralised scheduling approach in the Industry 4.0 and Cloud Manufacturing era. *IFAC-Pap. OnLine* **2019**, *52*, 2170–2176. [CrossRef]
42. Yang, B.; Wang, S.; Cheng, Q.; Jin, T. Scheduling of field service resources in cloud manufacturing based on multi-population competitive-cooperative GWO. *Comput. Ind. Eng.* **2021**, *154*, 107104. [CrossRef]
43. Huang, B.; Li, C.; Tao, F. A chaos control optimal algorithm for QoS-based service composition selection in cloud manufacturing system. *Enterp. Inf. Syst.* **2014**, *8*, 445–463. [CrossRef]
44. Zhou, H. Research on Dynamic Robustness of Knowledge Collaboration Network of Open Source Product Community. Ph.D. Thesis, University of Science and Technology Beijing, Beijing, China, 2018.
45. Lei, S. Robustness of Open-Source Community Knowledge Collaboration Network Based on Multiple Failure Modes. Ph.D. Thesis, University of Science and Technology Beijing, Beijing, China, 2022.