Hindawi
Security and Communication Networks
Volume 2022, Article ID 5019584, 18 pages
https://doi.org/10.1155/2022/5019584

Research Article

Towards Optimal Resources Allocation in Cloud Manufacturing: New Task Decomposition Strategy and Service Composition Model

Zhou Fang, Qilin Wu, and Dashuai Guan

College of Information Engineering, Chaohu University, Chaohu/238024, China

Correspondence should be addressed to Qilin Wu; qlw@chu.edu.cn

Received 2 November 2021; Accepted 4 January 2022; Published 15 February 2022

Academic Editor: Xiaolong Xu

Service composition optimization is one of the core issues in cloud manufacturing research. However, all current studies of service composition in cloud manufacturing assume that tasks have been decomposed into subtasks, so they can be directly mapped to existing services. However, due to the complexity, diversity, and multilevel of services in cloud manufacturing, services have different granularity. Therefore, the matching between tasks and services does not always occur at the lowest level. For solving the problem of discontinuity between task decomposition and service composition, this paper considers the characteristics of existing services in the cloud pool and proposes a task decomposition strategy based on task/service matching on the basis of refining the description model of tasks and services. Then, for the decomposed subtask set, the E-CARGO model is used to model the optimal composition process of services, and CPLEX is used to solve the model. Practical cases show that the proposed task decomposition strategy can solve the problem of discontinuity between task decomposition and service composition without relying on more expert systems. In addition, the proposed service composition model is more flexible, can easily model more variable factors, and CPLEX can solve the model more quickly and stably.

1. Introduction

Recently, with the promotion of emerging technologies such as cloud computing, Internet of things (IOT), network physical system (CPS), big data analysis, and artificial intelligence, a new industrial manufacturing mode with the core characteristics of globalization, personalization, digitization, cloud computing, collaboration and integration, namely cloud manufacturing, has been proposed [1–7]. As an emerging manufacturing model, cloud manufacturing can effectively solve the problems of shortage and idleness of manufacturing resources, deficiency, and excess of manufacturing capacity in China's manufacturing industry, and realize a manufacturing-oriented service model characterized by sharing, collaboration and on-demand use, which can provide a new impetus for the transformation and upgrading of manufacturing industry [8, 9].

Up to now, cloud manufacturing has attracted a large number of scholars around the world, and many problems that need to be solved in the future are proposed and discussed, such as the architecture of cloud manufacturing service platform, business model, description of manufacturing resources, optimal allocation of resources, etc. [9, 10]. Among these, optimal allocation of manufacturing resources is the core function of cloud manufacturing, mainly including two core processes: task decomposition and service composition.

Task decomposition is the basis and premise of service composition. Its goal is to obtain a highly cohesive task sequence to ensure that different service providers can cooperate to fulfill the customers' requirements. However, in the current research, task decomposition and service composition are usually carried out as two independent and irrelevant procedures. On the one hand, task decomposition heavily relies on industry expert systems and lacks consideration of the existing services' status in the cloud pool, which makes it difficult to keep up with the changing market demand; on the other hand, the existing service composition research lacks the modeling of collaboration between services, which has a certain impact on the overall execution

efficiency of manufacturing tasks. Therefore, how to integrate task decomposition and service composition together is the core issue of this paper.

As an intermediate procedure between task decomposition and service composition, service matching is mainly used to connect manufacturing tasks and services to maximize the satisfaction of both supply and demand. How to effectively use service matching to solve the problem of discontinuity between task decomposition and service composition is one core problem to be solved in this paper. In addition, the E-CARGO model, which was proposed by Professor H. B. Zhu in 2006, is used to describe a role-based collaboration system, and how to use it to model the collaboration between services to improve the practicability of the service composition model is another problem to be solved in this paper.

The main contributions of this paper are as follows:

(1) According to the characteristics of candidate service sets obtained after task service matching, specific computable formulas are given for the internal competition within candidate service sets and the collaboration and dependence between candidate service sets.

(2) Considering the service state, a new task decomposition algorithm based on task/service matching is proposed, which can combine the two processes of task decomposition and task/service matching and reduce the dependence on the expert system.

(3) A new cloud manufacturing service composition model based on role collaboration is constructed, where the mapping relationship between the service composition problem and the E-CARGO model is established. This can not only solve the problem that the heuristic algorithm is easy to fall into local optimization but also facilitate the introduction of multiple variable factors to expand and optimize the model.

The remainder of this paper is organized as follows. Related work is described in Section 2. Section 3 describes the core problems related to the optimal allocation of manufacturing resources. Section 4 gives the description model of tasks and services, introduces the task/service matching based task decomposition algorithm, and describes the subtask reorganization algorithm combined with service characteristics. Section 5 formalizes the service composition approach by utilizing the E-CARGO model and the corresponding calculation methods. Section 6 analyzes and verifies the proposed methods through application case and performance analysis. Finally, the conclusions appear in Section 7.

2. Related Work

As the core part of implementing cloud manufacturing, manufacturing resource optimized allocation aims to provide a set of capabilities/services for satisfying personalized manufacturing demands through a process of resource composition and optimal selection. Efficient shared manufacturing resource allocation can not only achieve rapid response to diverse manufacturing demands but also facilitate full-scale sharing of enterprises' resources.

Task decomposition is one of the most important preprocessing stages of manufacturing resource optimized allocation while is a challenging task, not only because they are characterized by cross-industry heterogeneity, but also because the decomposition process need considering the state of service (such as service granularity and relevance).

In the existing research on task decomposition, some methods used a design structure matrix (DSM) to express the interaction information and correlation degree between tasks [11–13]. Kherbachi et al. used DSM to cluster the tasks in product development and matched the corresponding development tasks to the appropriate research group and development group [14]. Liu and Zhou proposed a method of task decomposition and reorganization based on DSM combined with adjustable task granularity [15]. However, DSM has shortcomings in both the quantitative analysis ability of uncertain information and the decomposition ability of complex tasks. In [16], Shriyam et al. proposed an approach based on a dynamic grid for decomposing exploration tasks among multiple Unmanned Surface Vehicles (USVs) in port regions. In other ways, the task is decomposed into subtasks with appropriate granularity according to the task hierarchy and task correlation. In [17], Zhang et al. constructed a global manufacturing business process network (GMBPN) according to the input and output relationships of manufacturing business activities. Based on the GMBPN, they further presented a two-phase decomposition algorithm. Hu et al. divided the complex manufacturing tasks into multiple stages according to the attributes and characteristics of the production process and proposed a novel hybrid method combining depth first search, fast modular, and artificial bee colony to optimize multistage production processes [18]. To solve complex parts machining problems in CMfg, Guo et al. presented a machining task decomposition strategy that uses features of the complex part as task granularity [19]. Liu et al. proposed an ordered task decomposition method (task decomposition method based on hierarchical task network) considering task granularity, cohesion, and correlation [20].

The above literature studies the task decomposition strategy from different levels and perspectives, but they only use the characteristics of the task itself or the correlation between tasks to decompose the task and take no consideration of the service state, which results in the problem that the process of task decomposition is divorced from that of matching and assignment of task and service. To solve the above problems, Yi et al. [21] decomposed the manufacturing task into atomic tasks according to the predefined decomposition rules and then reorganized these atomic tasks using clustering algorithms by considering task correlation, matching degree of tasks and services, and competition between services. Although the algorithm considers the state of the service, it inverts the dependency between the task and the service. On the one hand, the decomposition process of atomic tasks will rely heavily on

expert systems, and then it is difficult to realize a comprehensive expert system for massive heterogeneous tasks in different industries and categories. On the other hand, even if atomic subtasks can be successfully decomposed, there is a high probability that atomic subtasks cannot match the appropriate candidate service set.

Service composition is another core function in the process of resource optimized allocation. Its main purpose is to select a group of optimal service combinations from the candidate services of each subtask to complete the demander's manufacturing task. In essence, this process is the process of combining multiple services (atomic services or composite services) into value-added services to complete one or a group of tasks. As a typical multiobjective and multiconstraint NP-hard problem, a large number of metaheuristic algorithms, such as genetic algorithm, particle swarm optimization algorithm, and ant colony algorithm, have been proposed to find the optimal or nearly optimal combination scheme in a reasonable time [22–31]. The typical processes of these algorithms are (1) propose heuristic algorithms for fixed models and (2) repeatedly test and adjust the heuristic algorithms to obtain the required performance. If some aspects of the service (such as service availability or service quality) change, this process must usually be repeated. It can be seen that these algorithms have a complex design process and are usually for specific problems. Therefore, they are not adaptive to the dynamic environment, which means that when the environment changes, they may need to be redesigned.

In addition to devising these algorithms, another important problem is how to establish an appropriate service composition model. As an important index to determine the quality of service composition and an important factor to be considered in the process of service composition, Quality of Service (QoS) is widely used in the modeling of service composition. Therefore, Que et al. [30] proposed the method of using the user model (M2U) to solve service composition and optimization selection and established the corresponding mathematical evaluation model by comprehensively considering the four QoS evaluation indexes (time, cost, reliability, and capability). Li et al. proposed an extended Gale-Shapley (GS) algorithm for service composition that allows the generation of multiple service composition solutions effectively, where the requirements with different constraints have been considered [31]. Considering the one-to-one mapping between basic services and subtasks, Liu and Zhang [32] freely combined multiple basic services with equivalent functions into a cooperative service group (SESG) to complete each subtask together. At the same time, the optimized structure of SESG was introduced into the QoS evaluation model, and the corresponding QoS evaluation formula was given. In [33], Jin et al. proposed a correlation-based service description model to describe the QoS dependence of a single service on other related services and then introduced a service correlation mapping model to automatically obtain the value of QoS correlation between services. Afterward, Laili et al. [34] studied the multistage integrated scheduling problem of hybrid tasks in a cloud manufacturing environment to maximize production efficiency while balancing different production task orders. The experimental results show that this method reduces the production cost and shortens the production time. In [23], Yuan et al. proposed six basic QoS indexes including time, composability, quality, availability, reliability, and cost, and determined the weight of each index value in the QoS model by using the improved fuzzy comprehensive evaluation method.

Because services and tasks in cloud manufacturing have different granularity, service composition is essentially a dynamic matching process between multigranularity tasks and services. However, most of the current research on the service combination focuses on QoS modeling and rarely considers how to add more practical constraints in practical applications to easily expand the existing models, such as cooperation and competition between related services.

3. Problem Description

Cloud manufacturing software service platform needs to solve the problems of low sharing rate of manufacturing resources, poor collaboration among enterprises, and low customization level of manufacturing solutions in the manufacturing process. Figure 1 shows its operation principle, and it mainly includes three types of user roles: resource providers, resource demanders, and platform operators. Resource providers describe and publish the manufacturing resources (which will be encapsulated as services) in the manufacturing process in a unified model; resource demanders submit their manufacturing requirements (which can also be called tasks) and access various manufacturing resources on demand with the support of the platform; platform operators mainly audit and manage the resources and requirements in the platform, release or update various templates of resources and requirements in time, and monitor the transactions between the suppliers and the demanders.

Optimal allocation of manufacturing resources is an important procedure in cloud manufacturing, and its basic workflow is shown in Figure 2:

(1) Decomposing the total tasks submitted by resource demanders into subtasks for collaborative completion.

(2) According to the task-service matching method, the cloud manufacturing software service platform searches and matches the candidate service sets that can complete each subtask of the initial total tasks.

(3) Considering the influence factors of QoS, such as time, cost, quality, reliability, and so on, the cloud manufacturing software service platform searches for the best services in the candidate service set of each subtask to form an optimal execution plan for the initial total manufacturing task.

(4) Executing and supervising the completion process of tasks according to the optimal execution plan.

Task decomposition is the most basic initialization step of optimal allocation of manufacturing resources, and its

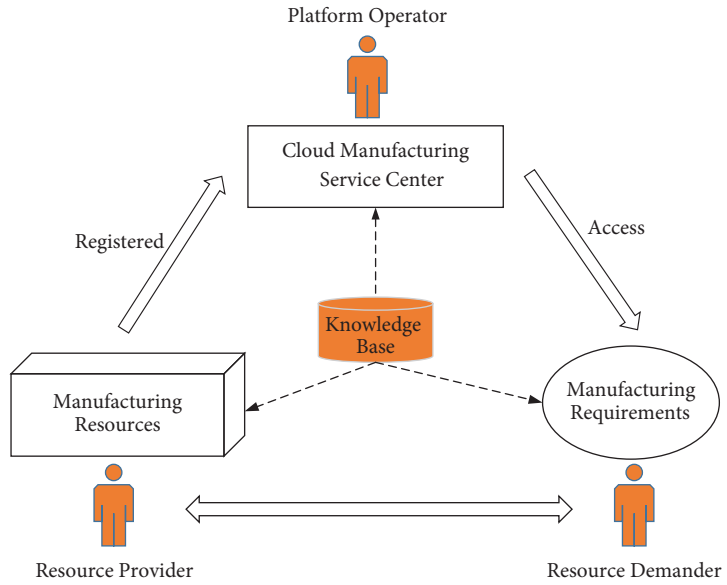

Figure 1: Operation principle of cloud manufacturing software service platform.

purpose is to decompose the overall manufacturing task into subtasks that should be executable and have low relevance between each other so that the cloud manufacturing software service platform can match the appropriate service to complete the user's demand. It is clear that the task decomposition results are closely related to the processing quality of the subsequent procedure of resource optimization.

In the process of task decomposition, the scale of subtasks is characterized by task granularity, which will directly affect the quality and progress of task collaboration. Specifically, the larger the granularity, the higher the task integrity, but the implementation is complex, which is not conducive to multienterprise cooperation. On the other hand, the finer the granularity, the more interaction between tasks, but the coordination is difficult, and the logistics cost and information interaction cost are prominent, which affect the quality, progress, and cost of task completion. Therefore, how to combine the characteristics of the services in the cloud pool and complete the decomposition of tasks at an appropriate granularity is of great significance in resource optimization.

After matching the manufacturing service set for each manufacturing subtask under the functional constraints, some suitable services need to be selected from each candidate set and assembled into composite services in a certain order to collaboratively complete the user's manufacturing requirements. How to build a more flexible combination model, which can easily model more variable factors to adapt to the open and dynamic manufacturing environment, is one of the urgent problems to be solved.

4. Task Decomposition Strategy

In this paper, task decomposition is divided into two stages: preliminary decomposition and reorganization. In the preliminary task decomposition stage, the total task is decomposed into executable atomic subtasks based on task/service matching. In the task reorganization stage, subtasks with the small granularity are merged into subtasks with appropriate granularity by considering the internal competition of candidate service setting, cooperation, and dependence between candidate service sets.

4.1. Description Model of Tasks and Services. Cloud manufacturing users come from different enterprise and engineering application fields, and their needs are more diversified and personalized. In addition, manufacturing resources are widely distributed in various forms and types. The description of requirements and resources by manufacturing enterprises is often unclear, incomplete, and inconsistent, which is difficult to realize the dynamic cooperation between users and resources in the cloud manufacturing environment. Therefore, a unified formal description is inevitable. Here, the requirements are modeled as manufacturing tasks, and the combination of several resources is modeled as manufacturing services to complete the specified tasks. Based on the formal description of the existing manufacturing services and manufacturing tasks (cloudservice = {ID, TypeInfo, BaseInfo, ResourceInfo, FuncInfo, AssessInfo, StatuInfo}) [35], we further extend the description of the function information FunInfo = {FunProfile, InputParam, OutputParams}, where we have the following:

(1) Funprofile: function summary, which briefly describes the functions provided by the service.

(2) Inputparams: the input information of the function, indicating that the demand side needs to provide necessary information or materials during the implementation of the service. For example, for a service of manufacturing a special wrench, the demand side may need to provide corresponding design drawings.

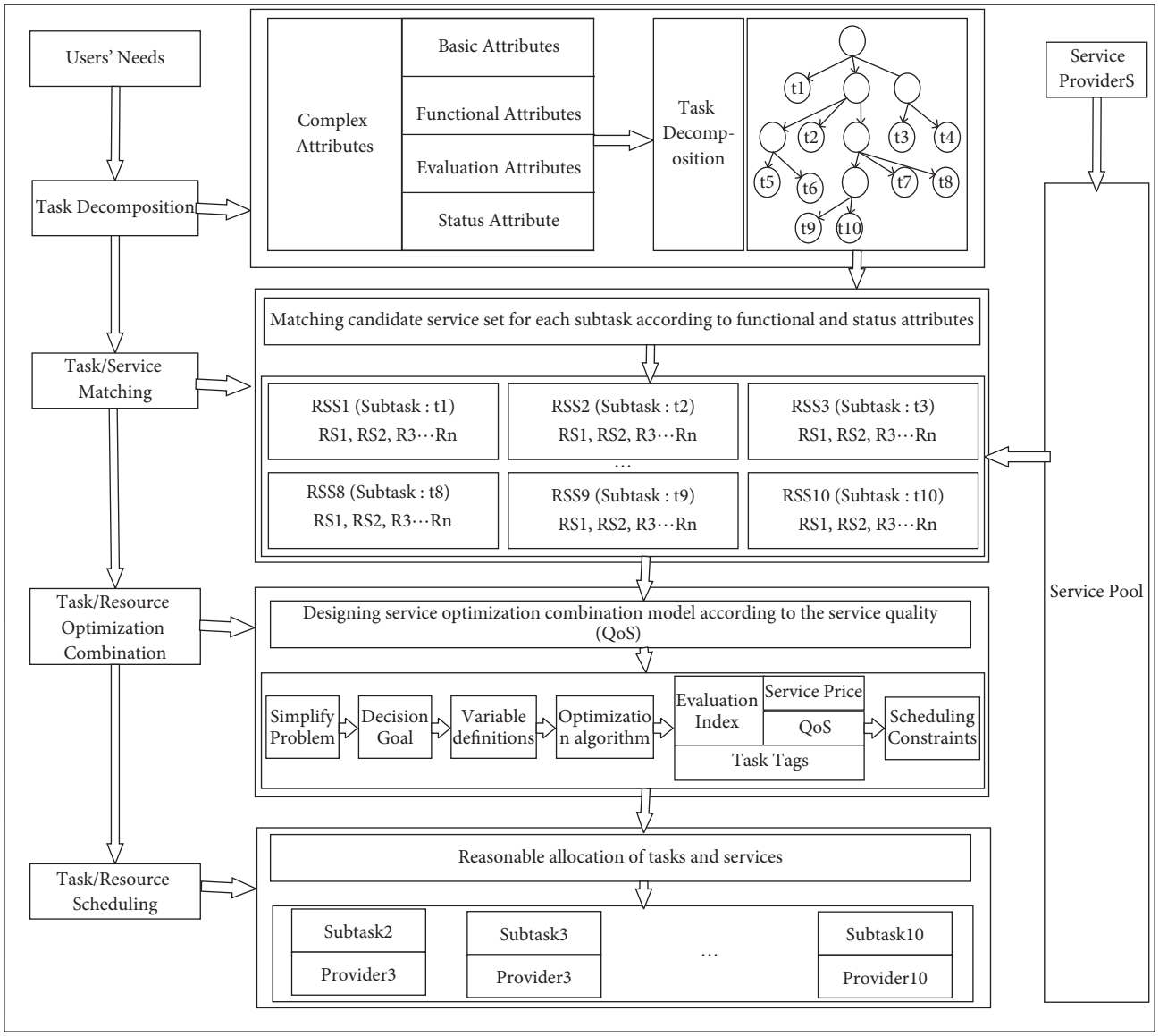

Figure 2: Basic process of resource optimization in cloud manufacturing software service platform.

(3) Outputparams: the output content of the function, which indicates the service results to be provided to the demander after the service is executed. As for a software development service, the output content may be executable source code or a one-year technical support service.

4.2. Preliminary Decomposition Method. To avoid relying too much on the industry knowledge and prevent the problem that the decomposed subtasks cannot match any appropriate services, we integrate the task decomposition and task-service matching as a whole, where we use the existing services in the cloud pool to adaptively complete the preliminary task decomposition. The basic flow of the method is shown in Algorithm 1.

Step 1. Initialize the variable T_k^i, where k represents the k-th subtask waiting to be decomposed, and i is the i-th subtask of the k-th subtask. In particular, when $k=0$, T_k^i means the original task T, and when $i=0$, T_k^i means the k-th original subtask which has not been decomposed. RSS_k^i is the corresponding candidate service set of T_k^i.

Step 2. Match the service set that meets the output requirements of T_k^i in the cloud pool. If the service set can be matched, proceed to Step 3, otherwise, proceed to Step 5.

The matching methods are as follows:

(1) Use TF-IDF algorithm to find out the keywords of task T_k^i and service j $(0 \le j \le \mathrm{N})$ respectively, where N represents the number of all services in the cloud pool that have not yet participated in the matching.

(2) Select several keywords from task T_k^i and service j respectively, and merge them into set D.

(3) Calculate the word frequency of task T_k^i and service j relative to all words in set D in turn.

Input: task waiting to be decomposed T
Output: sub-tasks T_k^i the corresponding candidate service set RSS_k^i
(1) initialize k to 0, i to 0, n to 0
(2) label T as T_k^i
(3) match the service set RSS_k^i in the cloud pool that meets the output requirements of T_k^i
(4) **if** RSS_k^i is not empty
(5) save T_k^i and RSS_k^i
(6) calculate the count n of the intersection of inputs of RSS_k^i, and label it as InSect
(7) set the input of the T_k^i to InSect
(8) **if** n is not equal to 0
(9) set $k+1$ to k
(10) for epoch = 1, 2, . . ., n do
(11) set $i+1$ to i
(12) create a new subtask T_k^i whose output of Fun-Info is the i-th value of InSect
(13) **goto 3**
(14) **end for**
(15) set i to 0
(16) **else:**
(17) **goto 11**
(18) **end if-else**
(19) **else**:
(20) **if** k equal to 0 and i equal to 0
(21) end algorithm
(22) **else**:
(23) **goto 11**
(24) **end if-else**
(25) **end if-else**

Algorithm 1: Preliminary decomposition algorithm.

(4) Calculate and save the cosine similarity $S_{i,j}$ between task T_k^i and service j.
(5) Sort all $S_{i,j}$ and calculate the average value of $S_{i,j}$ ranking in top M. If the average value is greater than the threshold C, the matching is regarded as successful, and the top m services are the corresponding service candidate set; otherwise, the matching is regarded as failed, where C and M are constants.

Step 3. Save task T_k^i and the corresponding service set RSS_k^i and calculate the intersection of inputs of services in RSS_k^i, which can be labeled as InSect, and then set the InSect to be the input of T_k^i.

Step 4. Create a new subtask using the content of the InSect to be the output, and then repeat Steps 2 to 4 to obtain subtask T_{k+1}^i and the corresponding service candidate set RSS_{k+1}^i, where $i\varepsilon[1,n]$ and n is the number of InSect.

Step 5. Judge T_k^i is the original task T; if it is, the algorithm is terminated; otherwise, proceed to Step 4.

4.3. Subtask Reorganization Algorithm. In the preliminary task decomposition stage, only the function matching of the service is considered, and the original task is decomposed into executable subtasks with smaller granularity, without considering the competition within the candidate set, the dependence between candidate sets and the difficulty of collaboration, which will be detrimental to the collaborative work between the final services.

Definition 1. Internal competitiveness of candidate service sets. It refers to the relative number N of services available in the service candidate set corresponding to each subtask after preliminary decomposition. To preserve the competitiveness between services, formula (1) can be used for calculation in the actual calculation process:

$$N = \frac{\sum_s \mathrm{similarity}_i}{S}, \tag{1}$$

where S is the number of services in the candidate service set (that is, in the preliminary task decomposition stage, the service candidate set with the matching degree in the top s), $\mathrm{similarity}_i$ indicates the matching degree between the ith service in the candidate set and the corresponding subtask. The higher the N, the more competitive the candidate service set.

It should be noted that in the cold start stage of the system, the number of services in the cloud pool is less, so the value of S can be set smaller. As the number increases, the value of S can be gradually increased, but the increased value should also be weighed against the efficiency of the algorithm.

The competitiveness of a candidate service set is to ensure that when a resource is selected, we can quickly find a substitute when it is unable to provide services due to unexpected circumstances. From a long-term perspective,

reserving competition space will make the pricing given by manufacturing resource owners more reasonable and urge them to actively improve service quality so as to promote the healthy development of the cloud manufacturing platform [21].

Definition 2. The degree of dependency between services R, which can be expressed by the correlation between services. Considering that the correlation is mainly determined by the logistics correlation and information exchange correlation between them, to reduce the complexity of calculation, the number, and type of inputs of task dependencies can be used as parameters to calculate the dependence between service candidate sets, as shown in

$$R = \sum_{i}^{M} w_i \times Q_i, \tag{2}$$

where M is the number of categories of inputs, Q_i is the number of categories i, w_i $(0 \le w_i \le 1)$ is the correlation coefficient of category i. The actual value of w_i is determined by the expert evaluation method and will be modified according to the operation results during the daily operation of the platform. The smaller the granularity of task decomposition, the greater the dependency.

Definition 3. Coordination difficulty between services. It refers to the difference between the longest and shortest service time (i.e., the execution waiting time of dependent tasks), which can be calculated by

$$T_i = \sum_{N} \left(t_{\max} - t_{\min}\right), \tag{3}$$

where T_i is the coordination difficulty of the i-th task, N is the number of layers of all subtasks of the i-th task, $t_{\max}$ and $t_{\min}$ represent the maximum execution time and minimum execution time of subtask in the specified level, respectively. Obviously, the larger the granularity of task decomposition, the more difficult it is to cooperate.

Definition 4. The granularity G of the candidate service set, which indicates the suitability of the granularity of the candidate service set to complete the specified task. According to the comprehensive analysis formulas (2) and (3), the higher the interdependence of the candidate service sets, the more unfavorable it is to complete the tasks in collaboration. That is, with the increase of R value, the G value should be appropriately increased (i.e., subtasks should be merged). However, the larger the decomposition granularity, the more waiting time for other parallel tasks, which is more unfavorable for the system to complete the task, that is, with the increase of T value, the g value should be appropriately reduced (that is, the task should be decomposed), so it can be expressed by

$$G = N \times R - (1 - N) \times T. \tag{4}$$

It should be noted that when we calculate G using formula (4), the values of N, R, and T need to be regularized.

Definition 5. The state tree of the candidate service set, which is used to simplify the description of the task reorganization process. The node in the tree is the meta-task obtained after the preliminary decomposition of the task. The value of the node represents the internal competitiveness of the service candidate set corresponding to the node, and the weight of edges represents the granularity of the candidate set relative to the parent node.

With the purpose of increasing the internal competitiveness of the candidate service set, reducing the degree of dependency between candidate service sets, and reducing the waiting time of the parallel services in candidate service sets, we design the pruning algorithm as shown below to realize task reorganization (Algorithm 2).

Step 6. Calculate the internal competition of each candidate service set, the dependencies between each candidate service set and others, and the coordination difficulty of each candidate service set, so as to construct the state tree of the candidate resource set using formula (4).

Step 7. Traverse all nodes in the m-th layer of the candidate service set state tree to determine whether the subnodes under the node need to be merged.

(1) Set $k = k + 1$ and repeat steps (1)–(3) if the k-th node in the m-th layer is a leaf node; otherwise, go to step (2).

(2) Get the value nodeVal of the k-th node in the m-th layer; if the value is less than zero, set k to $k + 1$ and return back to step (1); otherwise, go to step (3).

(3) Crop all child nodes under the k-th node, recalculate the node values in the new state tree, set $k = k + 1$, and repeat steps (1)–(3).

Step 8. Set $m = m + 1$ and go back to Step 7.

5. Service Composition Model

5.1. Problem Description. For the original manufacturing task T submitted by the customer, after task decomposition, the subtask list is obtained as shown in Table 1, where the "number of acceptable services" represents the number of service providers that can be accepted by the demander, In the specific implementation process, considering that some subtasks may be complex, multiple services are required to complete them.

Considering that there may be a large number of manufacturing tasks of the same category with the same or similar input and output in the cloud manufacturing software service platform, it is necessary to consider the composition quality of all tasks of the same category while building the optimal composition model of services. Let subtask T_k belong to category TC_k. Therefore, the service candidate set RSS_k matching task T_k also can match all other tasks under category TC_k. Afterward, according to the given QoS evaluation model, we can calculate the competency of each service in the candidate set RSS_k for each task under

Input: atomic subtasks and the corresponding candidate service sets
Output: subtasks with suitable granularity and the corresponding candidate service sets
(1) initialize k to 0, m to 0, nodeVal to −1
(2) construct the state tree of candidate resource set
(3) set m to the count of the original subtasks
(4) **for** epoch = 1, 2, . . ., m do
(5) set k to the count of the sub-tasks of the m-th original subtasks
(6) **for** epoch = 1, 2, . . ., k do
(7) **if** sub-task T_m^k is not a leaf node
(8) get the value of nodeVal
(9) **if** nodeVal is greater than 0
(10) crop and recalculate the state tree
(11) **end if**
(12) **end if**
(13) **end for**
(14) **end for**

Algorithm 2: Subtask reorganization algorithm.

Table 1: List of subtasks of task T.

Subtasks	T_1	T_2	...	T_k	...
Number of acceptable services	3	2	...	1	...

category TC_k, as shown in Table 2. The combination target is to find a service combination with the highest competency, which means the sum of competencies of each subtask of the manufacturing task T is the largest, and ensure all other tasks under each category TC_k have the highest competency at the same time.

It should be noted that the competency value in Table 2 needs to be calculated under a given QoS model according to the evaluation data of the specific evaluation system in the cloud manufacturing service platform. As each attribute of QoS has different measurement methods and dissimilar units, the aggregated QoS values for each attribute should be normalized before evaluating the global QoS of the cloud manufacturing service. Each attribute is either a positive or a negative factor (for a negative factor, the smaller the value of the index, the better for the service requesters and vice versa). This can be obtained using the following equations:

$$F_n = \begin{cases} \dfrac{f_n - \min f_n}{\max f_n - \min f}, & \min f_n \neq \max f_n, \\ 1, & \min f_n = \max f_n, \end{cases} \tag{5}$$

$$F_n = \begin{cases} \dfrac{\max f_n - f_n}{\max f_n - \min f}, & \min f_n \neq \max f_n, \\ 1, & \min f_n = \max f_n. \end{cases} \tag{6}$$

Formulas (5) and (6) are associated with the normalization of positive QoS indices (such as availability and reliability) and negative QoS attributes (such as time and cost), respectively. In this paper, for the convenience of expression, the aggregated QoS values are generated randomly between 0 and 1.

Collaboration is a typical feature of service composition,while as a typical model using to describe role-based collaborative system, E-CARGO [36–41], which is proposed by Professor Zhu in 2006, is applied to the service composition in cloud manufacturing for the clearly description of the collaboration of service composition in this paper. In E-CARGO, a role-based system is described as nine-tuples. $\Sigma := (C, O, A, M, R, E, G, S0, H)$, where C is a set of classes, O is a set of objects, A is a set of agents who are representatives of human users, M is a set of messages, R is a set of roles, E is a set of environments, G is a set of groups, $s0$ is the initial state of a collaborative system, and H is a set of users.

5.2. The Proposed Service Composition Model. To use the E-CARGO model for service composition in cloud manufacture, we map the related concepts involved in service composition to the corresponding tuples, and the specific process is shown in Figure 3.

Here, we introduce some necessary parameters to simplify the description of the model.

N is a nonnegative integer

$m(=|A|)$ is the number of agents, which is mapped to the number of services in one service candidate set

$n(=|R|)$ is the number of roles, which is mapped to the number of all tasks whose category is the same as the specified sub-task in the current total task

$q(=|Q|)$ is the number of all sub-tasks of the current original task after decomposition

Next, we give the following definitions combined with the above parameters.

Table 2: Competency of each service in RSS_k for each task under category TC_k.

	T_K^1	T_K^2	T_K^3	...	T_K^i	...
RSS_K^1	0.43	0.73	0.74	0.59	0.69	0.73
RSS_K^2	0.54	0.84	0.78	0.63	0.72	0.94
RSS_K^3	0.65	0.95	0.83	0.75	0.89	0.85
...				...		
RSS_K^j	0.95	0.35	0.87	0.83	0.92	0.65
...				...		
RSS_K^m	0.78	0.48	0.28	0.89	0.02	0.48

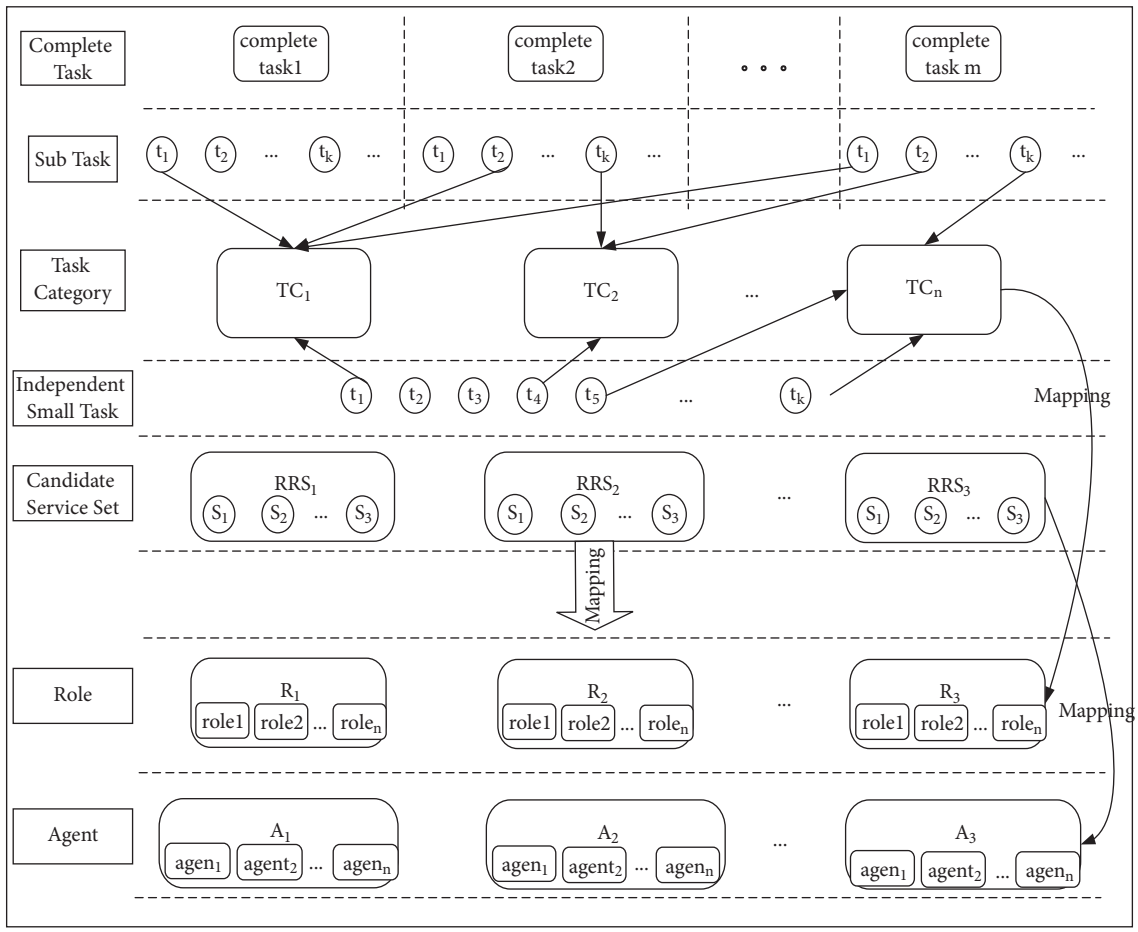

Figure 3: Mapping relationship between service composition and E-CARGO model.

Definition 1: in group G, $<i, j>$ is used to indicate that role j is assigned to agent i. It means that the manufacturing task j is assigned to service i in the cloud manufacturing environment.

Definition 2: in group G, $L(j) \in \in n$ $(1 \le j \le n)$ expresses that the role j needs to be assigned at least L $[j]$ agents. As for the cloud manufacturing environment, $L(c, j) \in n$ $(1 \le c \le q, 1 \le j \le n)$ expresses that subtask j of category c needs at least $L(c, j)$ services to complete cooperatively.

Definition 3: in group G, $L^a(i) \in m$ $(1 \le i \le m)$ means that agent i can only be assigned to $L^a(i)$ roles at most. As for the cloud manufacturing environment, $L^a(c, i) \in m$ $(1 \le c \le q, 1 \le i \le m)$ means that service i in the c-th candidate service set can only serve $L^a(i)$ tasks at the same time.

Definition 4: in group G, the matrix $Q[i, j] \in [0, 1]$ $(1 \le i \le m, 1 \le j \le n)$ represents the competence degree of agent I for role j, where 0 is the lowest competence degree, and 1 is the highest. In the cloud manufacturing environment, we introduce the variable c representing the category to expand the matrix Q. The expanded matrix Q $[c, i, j] \in [0, 1]$ (where $1 \le c \le q$, $1 \le i \le = m$, $1 \le j \le n$) describes the ability of service i in the c-th candidate service set to complete the j-th subtask of

category c. The values of competence degree are usually calculated according to the corresponding QoS model, which generally assigns different weights to different QoS criteria (e.g., time, cost, etc.).

Definition 5: in group G, the assignment matrix $T[i, j] = \{0, 1\}$ (where $1 \le i \le m$, $1 \le j \le n$) indicates whether role j is assigned to agent i. $T[i, j] = 1$ means role j is assigned to agent i, and $T[i, j] = 0$ means not assigned. In the cloud manufacturing environment, $T[c, i, j] \in [0, 1]$ (where $1 \le c \le q$, $1 \le i \le m$, $1 \le j \le n$) is used to indicate whether the j-th task of category c is assigned to the i-th candidate service set.

Definition 6: in group G, the assignment efficiency σ represents the total competency degree of all agents assigned roles, and it can be calculated by

$$\sigma = \sum_{1}^{m} \sum_{1}^{n} T(i, j) * Q(i, j). \tag{7}$$

In the cloud manufacturing environment, σ can be calculated by

$$\sigma = \sum_{c=1}^{q} \sum_{i=1}^{m} \sum_{j=1}^{n} T(c, i, j) * Q(c, i, j). \tag{8}$$

Definition 7: in group G, role j is workable when there are enough agents that can compete for it, and there is

$$\sum_{i=1}^{m} T[i, j] = L(j), \quad (1 \le j \le n). \tag{9}$$

In the cloud manufacturing environment, a manufacturing task j can be effectively assigned when all of its subtasks are effectively assigned, and formula (10) should be satisfied:

$$\sum_{i=1}^{m} T[c, i, j] = L(j), \quad (1 \le c \le q, 1 \le j \le n). \tag{10}$$

Definition 8: in group G, agent i is workable when the number of roles assigned to agent i does not exceed its workload, and there is

$$\sum_{i=1}^{n} T[i, j] = L^a(i) \quad (1 \le i \le m). \tag{11}$$

In the cloud manufacturing environment, the assignment result for service i is effective if it satisfies the constraint condition in

$$\sum_{c=1}^{q} \sum_{j=1}^{n} T[c, i, j] = L^a(i) \quad (1 \le c \le q, 1 \le j \le n). \tag{12}$$

Definition 9: in group G, the assignment matrix T is workable if each role and each agent is workable. If T is workable, then group G is workable. In the cloud manufacturing environment, if all subtasks decomposed from an original task can be assigned to enough services that meet the workload requirements, we can say there is effective service composition.

Finally, the process of role collaboration-based service composition is to find the optimal assignment scheme T, where $A(|A| = m)$, $R(|R| = n)$, $Q(|Q| = q)$, L and L^a are given. Namely, we need to solve the maximum value of the target object σ shown as formula (13) under the specified constraints.

$$\max \sum_{c=1}^{q} \sum_{i=1}^{m} \sum_{j=1}^{n} T(c, i, j) * Q(c, i, j). \tag{13}$$

subject to

$$\begin{aligned} &\sum_{i=1}^{m} T[c, i, j] = L(c, j), \quad (1 \le c \le q, 1 \le j \le n), \\ &\sum_{c=1}^{q} \sum_{j=1}^{n} T[c, i, j] = L^a(i), \quad (1 \le c \le q, 1 \le j \le n), \\ &T[c, i, j] \in \{0, 1\}, \quad (1 \le c \le q, 1 \le i \le m, 1 \le j \le n). \end{aligned} \tag{14}$$

5.3. Cplex-Based Solving Method. For obtaining higher execution efficiency, this paper bypasses the compilation process of the IBM ILOG CPLEX development environment and uses the method of directly referencing the ILOG development package in Java project to solve the above model. The specific steps are as follows:

(1) Find the mapping relationship: It is necessary to map the relevant elements involved in the service combination model to the four basic elements (objective function, function variable, variable coefficient, constraint condition) of the linear programming problem in ILOG, where the objective function is σ, the variables of the objective function correspond to the assignment matrix T, the variable coefficients correspond to the quality of service (QoS), and the constraint conditions are related to L and La.

(2) Add objective function: When using ILOG to solve linear programming problems, we need to transform the matrices Q and T into one-dimensional vectors and form the final objective function. Then the optimization target is added in the Java code by calling the following method of ILOG:

IloIntVar[] X = cplex.intVarArray($q * m * n$, 0, 1);
cplex.addMaximize(cplex.scalProd(X, V));
where, $X[c * m * n + i * n + j] = T[c, i, j]$, $V[c * m * n + i * n + j] = Q[c, i, j]$ ($1 \le c \le q$, $1 \le i < m$, $1 \le j \le n$).

(3) Add constraints:

First, declare the expression object of the constraints:

6. Experimental Analysis and Verification

Considering the types of open-source toolkits in different development languages and the characteristics of task decomposition algorithm and service composition model in this paper, Python language, and its mainstream scientific computing library are used to implement the task decomposition algorithm, and Java language and IBM ILOG CPLEX library are used to implement the service composition algorithm. The specific hardware and software experimental environment are shown in Table 3.

6.1. Case Analysis. The design and production process of military electric vehicles is extremely complex, and it is difficult to rely on a single service provider to complete the manufacturing task. Therefore, after receiving the task, the platform needs to decompose the task into executable and appropriate fine-grained subtasks and assign them to different service providers to complete the task cooperatively. After an in-depth understanding of the design and production process of military electric vehicles, we combine them in detail to form a more specific process as shown in Table 4, and a logical relationship between the processes is shown in Figure 4, where Serial Number 0 represents the completed vehicle [39].

The implementation of the task decomposition method proposed in this paper relies on the existing services. Therefore, the first step is to collect sufficient service data sets. We crawl some related service sets from the network and expand them through reproduce, mirroring, local adjustment and other methods according to the characteristics of the crawled network data in a reasonable range, where the service times of each service are randomly generated in a specified range according to the complexity of the service, and meantime, the dependency degree of the specified input type is set to a fixed value initially, for simplifying the difficulty of the solution, we only consider two input types: logistics and communication.

For verifying the effectiveness of the task preliminary decompose algorithm, we take the design and production of the drive motor system as input and obtain the result shown in Figure 5, where the values of nodes is the corresponding serial number in Table 5. Next, we reorganize the result using the reorganization algorithm proposed in this paper, but the result has not changed.

Then, the design and production of the electric drive system is taken as input for the proposed algorithm. The result of the preliminary decomposition shown in Figure 6(a) and that of the reorganization shown in 6(b) are obtained, respectively.

Without any expert system in the industry, this proposed decomposition method can decompose complex manufacturing tasks into subtasks with enough candidate service sets according to the characteristics of the existing service in the platform, which provides the necessary premise for the realization of collaborative manufacturing.

After the subtasks and the corresponding candidate service sets are obtained by task decomposition, an optimal assignment scheme is needed to lay the foundation for the subsequent task scheduling. To simplify the description process, this paper takes the task described as the design and production of the electric drive system as an example to verify the effectiveness of the proposed service composition model.

Known from Figure 6, the task named design and production of electric drive system can be decomposed into two first-level subtasks, design and production of drive motor system and design and production of power system, which can come from task decomposition or direct release by other users. Assuming that the number of them is n1 and n2, respectively, and that of the corresponding candidate service sets are m1 and m2, respectively. At the same time, the related data involved in this experiment are initialized as the following: n1, n2, m1, and m2 are random numbers between 3 and 5; the number of services can be accepted by a single task L and the workload of single service L^a are all random numbers between 1 and 3; the service quality of each service in the candidate set (in real application scenarios, it is calculated based on its historical evaluation data) is a random number between 0 and 1. Finally, the initial situation is shown in Table 6.

According to the initialization data of the above model shown in Table 6, we use the Cplex-based method to solve the model and get the assignment scheme shown in Table 5, which takes 204.02 ms and the sum of the QoS in this scheme is 7.13. It should be noted that the above randomly generated data may not be able to find the assignment scheme, which is a common scenario in practical applications, and the specific processing measures are not within the scope of this paper.

6.2. Performance Analysis. On the one hand, the current research on task decomposition is relatively few, and there is no unified evaluation standard. On the other hand, considering that the main advantage of the proposed decomposition strategy in this paper is to reduce the dependence on industry expert system and solve the problem of disconnection between task decomposition and service composition process, which has been reflected in the specific implementation process. Therefore, we take no in-depth comparative analysis of that.

The proposed service composition model can smoothly transition from the task decomposition procedure to make full use of the existing service state in the cloud pool, and it can be easily introduced with a variety of variable factors to expand and optimize itself. For example, all the same, or similar independent tasks want to be assigned the best services. Therefore, the demander's acceptance matrix M can be introduced to expand the model (13) to obtain the optimization model (14); all interdependent services need a solid foundation for cooperation with each other. Therefore, the provider's acceptance matrix N can be introduced to expand the model (14) to (15); in addition, with the rapid development of social networks, the influence of peer effect in the evaluation system of the platform becomes increasingly important, which can directly or indirectly affect the choice of users themselves or other users. Therefore, the

Table 3: Hardware and software experimental environment.

System environment	
CPU	Intel(R) Core(TM) i7-6500U CPU @2.50 GHz 2.50 GHz
Memory	8 G (7.87 G可用)
Operating system	Windows 7 ultimate
Java development environment	
Development tool	Eclipse version: Luna Release (4.4.0)
JDK	jdk8u241
Third-party library	Cplex
Python development environment	
Development tool	PyCharm Community Edition 2021.1.1+ Anaconda3
Python	Python 3.7.3
Third-party library	Jieba 0.42.1+ gensim 3.8.3+ numpy 1.18.3

Table 4: Specific design activity units chart.

Serial number	Design and production name
1	Electric drive system
2	Power system
3	Power battery
4	Battery management system
5	Car charger
6	Auxiliary power source
7	Drive motor system
8	Electronic controller
9	Power converter
10	Drive motor
11	Mechanical transmission
12	Clutch
13	Transmission
14	Transmission shaft and other universal transmissions
15	Axle (main reducer, differential axle housing, etc.)
16	Wheels
17	Vehicle controller
18	Motor controller
19	Current sensor
20	Voltage sensor
21	Temperature sensor
22	Body
23	Body-in-white
24	Body safety guard
25	Auxiliary system
26	Automotive instrumentation, lighting and accessories
27	Power steering
28	Steering mechanism
29	Steering gear
30	Steering transmission mechanism
31	Front and rear suspension
32	Braking system

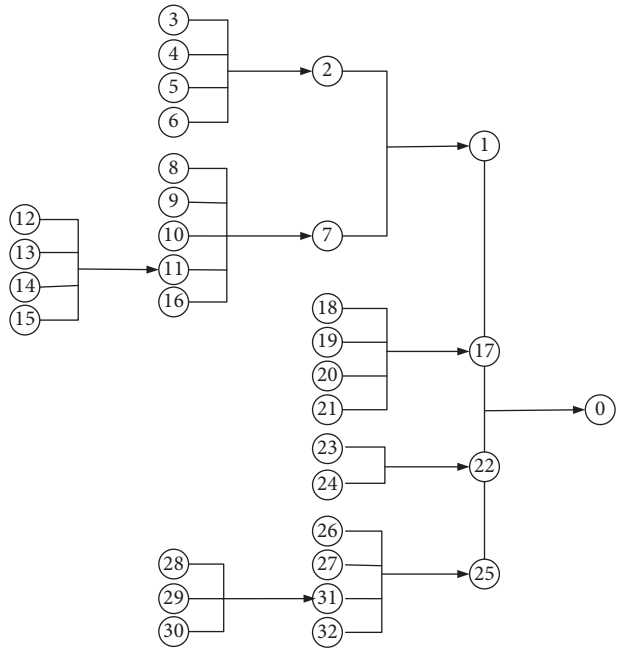

Figure 4: Logical relationship between the processes.

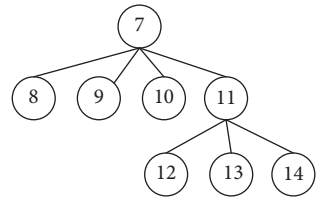

Figure 5: Decomposition result of "design and production of drive motor system."

Table 5: Current tasks waiting to be assigned, corresponding candidate service sets, and their constraints.

Design and production of power system				
Services(m1)	Tasks(n1)			
	Task1/L [1] = 1	Task2/L [2] = 1	Task3/L [3] = 2	Task4/L [4] = 2
Service1/L^a [1] = 2	0.88	0.70	0.04	0.52
Service2/L^a [2] = 2	0.94	0.47	0.87	0.74
Service3/L^a [3] = 2	0.72	0.46	0.94	0.24
Design and production of drive motor system				
Services(m2)	Tasks(n2)			
	Task1/L [1] = 1	Task2/L [2] = 1	Task3/L [3] = 1	Task4/L [4] = 2
Service1/L^a [1] = 2	0.27	0.50	0.71	0.09
Service2/L^a [2] = 1	0.22	0.06	0.89	0.01
Service3/L^a [3] = 1	0.58	0.16	0.00	0.49
Service4/L^a [4] = 2	0.65	0.52	0.53	0.01

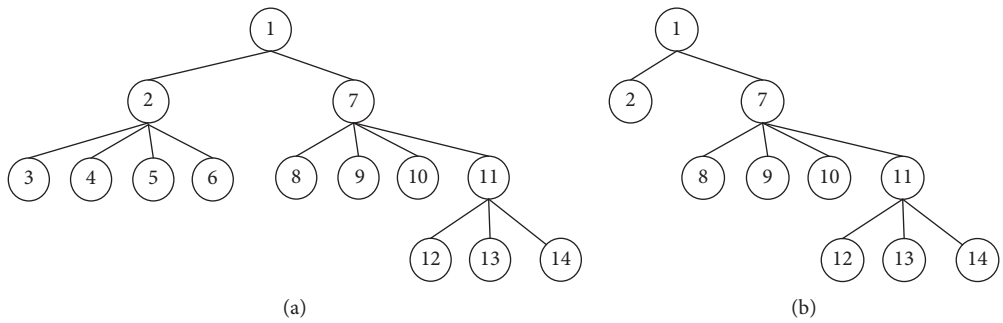

Figure 6: Decomposition result of "design and production of the electric drive system." (a) Result of decomposition. (b) Result of reorganization.

Table 6: Signment results based on the proposed model.

Design and production of power system				
Services(m1)	Tasks(n1)			
	Task1/L [1] = 1	Task2/L [2] = 1	Task3/L [3] = 2	Task4/L [4] = 2
Service1/L^a [1] = 2	0	1	0	1
Service2/L^a [2] = 2	0	0	1	1
Service3/L^a [3] = 2	1	0	1	0
Design and production of drive motor system				
Services(m2)	Tasks(n2)			
	Task1/L [1] = 1	Task2/L [2] = 1	Task3/L [3] = 1	Task4/L [4] = 2
Service1/L^a [1] = 2	0	0	0	1
Service2/L^a [2] = 1	0	0	1	0
Service3/L^a [3] = 1	0	0	0	1
Service4/L^a [4] = 2	1	1	0	0

user's social relationship S can be introduced to optimize the model and obtain the optimization model (17). However, considering the limited space of this paper, the modeling of M, N, and S is not described in detail. In the next experimental verification, we only take the proposed service composition model as the basic model and verify its effectiveness and adaptability by comparing it with the improved Hungarian algorithm [37, 42] (we call it KMB in the following description):

$$\max \sum_{l=1}^{q} \sum_{i=1}^{m} \sum_{j=1}^{n} T(l,i,j) * Q(l,i,j) * M(l,i,j), \tag{15}$$

$$\max \sum_{l=1}^{q} \sum_{i=1}^{m} \sum_{j=1}^{n} T(l,i,j) * Q(l,i,j) * M(l,i,j) * N(l,i,j), \tag{16}$$

$$\max \sum_{l=1}^{q} \sum_{i=1}^{m} \sum_{j=1}^{n} T(l,i,j) * Q(l,i,j) * M(l,i,j) * N(l,i,j)\delta * S(l,i,j). \tag{17}$$

Firstly, the validity is verified, and the experimental parameters are set as follows: q, n, m, $L[j]$, and L^a[i] are all random integers, where, for improving the success rate of assignment, the values of m are usually set to an integer multiple of values of n (the number of services is generally required to be more than the number of tasks), $L^a[i]$ is set between 1 and 3 and $L[j]$ is set between 1 and $2m/n$. The experimental results are shown in Table 7. It can be seen that the proposed method and the KBM algorithm can get comparable results at most times; in addition, KMB algorithm is more efficient when the number of tasks n is less than 20. However, when the number of tasks n is greater than 20, the KMB algorithm's efficiency is significantly lower than that of the proposed method. Considering that in the real business environment, the number of subtasks and their corresponding service candidate sets is usually much more than 20, so the proposed composition model and the corresponding solution algorithm can better meet the engineering requirements.

Table 7: Comparison of validity of the proposed composition model, the corresponding solution algorithm and km.

Numbers	q	m	n	Cplex execution time (s)			km execution time (s)			Sum of QoS (times) (Cplex ? km)		
				Avg	Max	Min	Avg	Max	Min	>	=	<
100	[1–5]	$2 * n$	[5–10]	0.0233	0.188	0.005	0.0010	0.004	0.0	51	0	49
100	[1–5]	$5 * n$	[5–10]	0.0357	0.206	0.007	0.0026	0.009	0.0	54	0	46
100	[5–10]	$2 * n$	[20–50]	0.4283	0.774	0.116	0.7094	1.52	0.078	57	0	43
100	[5–10]	$5 * n$	[20–50]	0.8664	1.838	0.236	4.0076	8.924	0.414	49	0	51
100	[1–5]	$2 * n$	[100–150]	1.5421	3.644	0.287	10.7309	26.428	1.257	57	0	43
100	[5–10]	$5 * n$	[100–150]	4.0185	9.091	0.745	66.5779	161.536	8.317	55	0	44

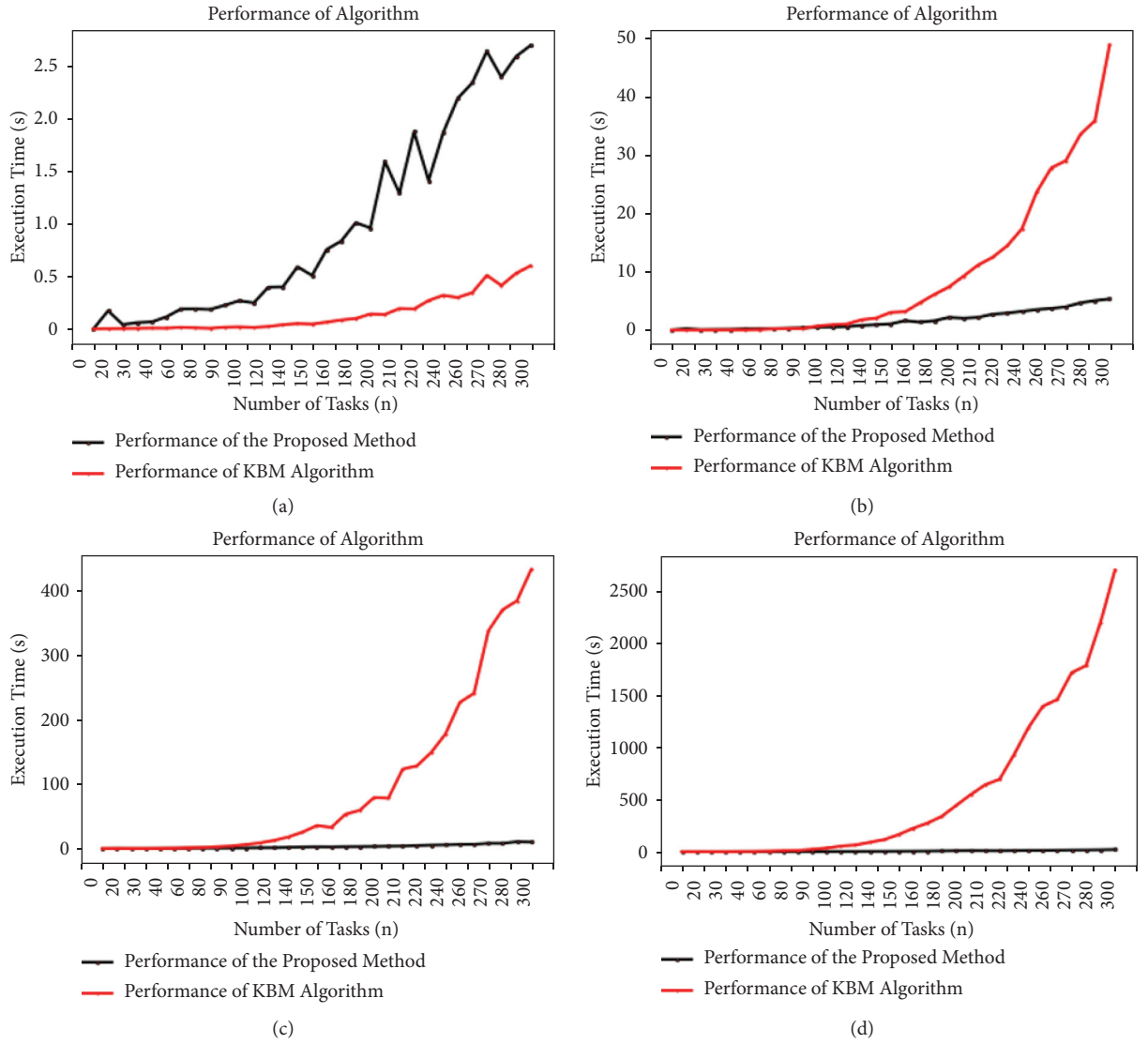

Figure 7: Performance comparison of our method and KBM algorithm. (a) $m = n$, $L^a[i] \in [1, 2]$. (b) $m = 2 * n$, $L^a[i] \in [1, 3]$. (c) $m = 3 * n$, $L^a[i] \in [1, 4]$. (d) $m = 4 * n$, $L^a[i] \in [1, 5]$.

Secondly, to verify the overall performance of the proposed composition model and the corresponding solution algorithm, we use different scales of random integers for experimental analysis, where q is a fixed value of 2, n increases from 10 to 300 with the pace of 10 each time, m is set to $1 * n$, $2 * n$, $3 * n$, $4 * n$ in turn, $L[j]$ and $L^a[i]$ are random numbers from 1 to m/n. At the same time, for avoiding the randomness of the experimental results, each data pair(q, m, n) is randomly tested for 100 times, and the average value is collected as the final experimental data, and the final performance trend chart as shown in Figure 7 is formed.

It can be seen that the KMB algorithm's execution efficiency decreases sharply with the increase of $L^a[i]$, and its complexity is $O\sum L^a[i]^3$. Compared with the KMB algorithm, the execution efficiency of the proposed method is more stable, and when $L^a[i]$ is greater than or equal to 2, it is far better than the KMB algorithm; at the same time, we notice that both KMB algorithm and our solution method have some fluctuations in the execution process. When $m = n$, more exceptions exist in the proposed composition model and the corresponding solution method than that of the KBM algorithm, which is caused by the unreachable assignment scheme under the existing conditions (the number of services and tasks are the same, but some tasks need multiple services). And when $m > n$, the exception of our method does not exist. However, the KMB algorithm still has some anomalies in some cases. In most cases, the above two solutions can meet the practical needs, but compared with the KMB algorithm, our composition model and solution method is obviously better than KMB algorithm in both efficiency and stability.

7. Conclusions and Future Work

Optimal manufacturing resource allocation is a core problem in the cloud manufacturing mode. In the specific implementation process, we should solve the problems of resource virtualization, task decomposition, task service matching, service composition, and scheduling in turn. This paper mainly proposes the corresponding solutions for task decomposition and service composition model. As for task decomposition, we refine the description model of task and service and propose the preliminary decomposition scheme based on task service matching and the reorganization strategy based on the characteristics of the service candidate set. As a result, the problem of discontinuity between task decomposition and service composition is solved. For solving the problem of task and service assignment, we design the service composition model using E-CARGO and solve the model using Cplex, which not only greatly reduces the problem of falling into local optimum of heuristic algorithms but also provides the necessary foundation for the introduction of more variable factors(such as cooperation, conflict and other constraints). Finally, the practicability of decomposition strategy and service composition model is proved by the experimental analysis.

The future work may follow several aspects:

(1) Task scheduling of manufacturing resource optimized allocation. Compared with the management of independent resources of enterprises, the management of shared resources is more dynamic (e.g., devices can join or withdraw from sharing services at any time). Hence, the difficulty in task scheduling will be greatly increased.

(2) Personalized recommendation applied in manufacturing resource optimized allocation. To improve the user-friendliness and convenience of online platforms, the personalized service recommendation for different customer requirements is an effective means. However, since manufacturing services usually appear in the form of composite services, existing Web service-based personalized recommendation technologies are difficult to be applied effectively.

(3) Real-time response of large-scale cases in a dynamic environment should be investigated to prove the superiority of Blockchain technology-based method over the centralized optimization methods. And more comparisons with other methods (e.g., PSO, game theory) should be made [43].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Key Research and Development Program of Anhui Province of China (T04a05020094, 904a05020091, 04a05020091, 04a05020092, and 04a05020093), the Scientific Research Project of Chaohu University (XLZ-202107, XLZ-201807), the Teaching and Research Project of Chaohu University (ch18jxyj18), and the Applied Curriculum Project of Chaohu University (ch18yygc13).

References

[1] B. H. Li, L. Zhang, S. L. Wang, F. Tao, and X. D. Chai, ""Cloud manufacturing: a new service-oriented networked manufacturing model," [In Chinese]," *Computer Integrated Manufacturing Systems*, vol. 16, no. 1, pp. 1–7, 2010.

[2] B. H. Li, L. Zhang, L. Ren, X. D. Chai, F. Tao, and Y. L. Luo, ""Further discussion on cloud manufacturing," [in Chinese]," *Computer Integrated Manufacturing Systems*, vol. 17, no. 3, pp. 449–457, 2011.

[3] Y. Liu and X. Xu, "Industry 4.0 and cloud manufacturing: a comparative analysis," *Journal of Manufacturing Science and Engineering*, vol. 139, no. 3, 2017.

[4] Y. Zhang, K. Wang, Q. He et al., "Covering-based Web service quality prediction via neighborhood-aware matrix factorization," *IEEE Transactions on Services Computing*, vol. 14, no. 5, pp. 1333–1344, 2021.

[5] Y. Zhang, G. Cui, S. Deng, F. Chen, Y. Wang, and Q. He, "Efficient query of quality correlation for service composition," *IEEE Transactions on Services Computing*, vol. 14, no. 3, pp. 695–709, 2021.

[6] Y. Zhang, J. Pan, L. Qi, and Q. He, "Privacy-preserving quality prediction for edge-based IoT services," *Future Generation Computer Systems*, vol. 114, no. 2021, pp. 336–348, 2021.

[7] Y. Zhang, C. Yin, Q. Wu, Q. He, and H. Zhu, "Location-Aware deep collaborative filtering for service recommendation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 6, pp. 3796–3807, 2021.

[8] F. Tao, L. Zhang, H. Guo, Y. L. Luo, and L. Ren, ""Typical characteristics of cloud manufacturing and several key issues of cloud service composition," [In Chinese]," *Computer Integrated Manufacturing Systems*, vol. 17, no. 3, pp. 477–486, 2011.

[9] Y. Liu, L. Wang, X. V. Wang, X. Xu, and P. Xu, "Cloud manufacturing: key issues and future perspectives," *International Journal of Computer Integrated Manufacturing*, vol. 32, no. 9, pp. 858–874, 2019.

[10] D. Wu, M. J. Greer, D. W. Rosen, and D. Schaefer, "Cloud manufacturing: strategic vision and state-of-the-art," *Journal of Manufacturing Systems*, vol. 32, no. 4, pp. 564–579, 2013.

[11] S. Son, J. Kim, J. Lee, and J. Ahn, "Improving supply chain management process using design structure matrix based cross-functional analysis," *Systems Engineering*, vol. 22, no. 4, pp. 313–329, 2019.

[12] H. Son, Y. Kwon, S. C. Park, and S. Lee, "Using a design structure matrix to support technology roadmapping for product-service systems," *Technology Analysis & Strategic Management*, vol. 30, no. 3, pp. 337–350, 2018.

[13] G. E. da Cunha Barbosa and G. F. M. de Souza, "A risk-based framework with Design Structure Matrix to select alternatives of product modernisation," *Journal of Engineering Design*, vol. 28, no. 1, pp. 23–46, 2017.

[14] S. Kherbachi, Q. Yang, and L. Yang, "Multi-domain integration of team-product-function and organization clustering in product development project," *Systems Engineering*, vol. 38, no. 6, pp. 1557–1565, 2017.

[15] D. T. Liu and D. J. Zhou, ""Task decomposition and recombination design structure matrix based on interval number," [In Chinese]," *Machine Design and Reasearch*, vol. 25, no. 6, pp. 7–9, 2009.

[16] S. Shriyam, B. C. Shah, and S. K. Gupta, "Decomposition of collaborative surveillance tasks for execution in marine environments by a team of unmanned Surface vehicles," *Journal of Mechanisms and Robotics*, vol. 10, no. 2, 2018.

[17] Z. Zhang, Y. Zhang, J. Lu, F. Gao, and G. Xiao, "A novel complex manufacturing business process decomposition approach in cloud manufacturing," *Computers & Industrial Engineering*, vol. 144, Article ID 106442, 2020.

[18] Y. Hu, Z. Zhang, J. Wang, Z. Wang, and H. Liu, "Task decomposition based on cloud manufacturing platform," *Symmetry*, vol. 13, no. 8, p. 1311, 2021.

[19] L. Guo, Y. Xu, W. He, and Y. Cheng, "Optimization of complex part-machining services based on feature decomposition in cloud manufacturing," *International Journal of Computer Integrated Manufacturing*, vol. 33, no. 12, pp. 1227–1244, 2020.

[20] M. Z. Liu, Q. Wang, and L. Lin, "Cloud manufacturing task decomposition method based on HTN," *China Mechanical Engineering*, vol. 28, no. 8, pp. 924–930, 2017.

[21] S. P. Yi, M. Z. Tan, Z. L. Guo, W. P. Han, and J. Zhou, "Manufacturing task decomposition optimization in cloud manufacturing service platform," *[In Chinese] Computer Integrated Manufacturing Systems*, vol. 21, no. 8, pp. 2201–2212, 2015.

[22] Y. Liu, L. Wang, X. V. Wang, X. Xu, and L. Zhang, "Scheduling in cloud manufacturing: state-of-the-art and research Challenges," *International Journal of Production Research*, vol. 57, no. 15-16, pp. 4854–4879, 2019.

[23] M. Yuan, Z. Zhou, X. Cai, C. Sun, and W. Gu, "Service composition model and method in cloud manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 61, Article ID 101840, 2020.

[24] E. Aghamohammadzadeh and O. Fatahi Valilai, "A novel cloud manufacturing service composition platform enabled by Blockchain technology," *International Journal of Production Research*, vol. 58, no. 17, pp. 5280–5298, 2020.

[25] J. Zhou and X. Yao, "Multi-objective hybrid artificial bee colony algorithm enhanced with Lévy flight and self-adaption for cloud manufacturing service composition," *Applied Intelligence*, vol. 47, no. 3, pp. 721–742, 2017.

[26] L. Zhu, P. Li, G. Shen, and Z. Liu, "A novel service composition algorithm for cloud-based manufacturing environment," *IEEE Access*, vol. 8, pp. 39148–39164, 2020.

[27] J. Zhou, X. Yao, Y. Lin, F. T. S. Chan, and Y. Li, "An adaptive multi-population differential artificial bee colony algorithm for many-objective service composition in cloud manufacturing," *Information Sciences*, vol. 456, no. 5, pp. 50–82, 2018.

[28] C. Li, J. Guan, T. Liu, N. Ma, and J. Zhang, "An autonomy-oriented method for service composition and optimal selection in cloud manufacturing," *International Journal of Advanced Manufacturing Technology*, vol. 96, no. 5-8, pp. 2583–2604, 2018.

[29] B. Xu, J. Qi, X. Hu, K.-S. Leung, Y. Sun, and Y. Xue, "Self-adaptive bat algorithm for large scale cloud manufacturing service composition," *Peer-to-Peer Networking and Applications*, vol. 11, no. 5, pp. 1115–1128, 2018.

[30] 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," *International Journal of Advanced Manufacturing Technology*, vol. 96, no. 9-12, pp. 4455–4465, 2018.

[31] F. Li, L. Zhang, Y. Liu, and Y. Laili, "QoS-Aware service composition in cloud manufacturing: a gale-shapley algorithm-based approach," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 7, pp. 2386–2397, 2020.

[32] B. Liu and Z. Zhang, "QoS-aware service composition for cloud manufacturing based on the optimal construction of synergistic elementary service groups," *International Journal of Advanced Manufacturing Technology*, vol. 88, no. 9-12, pp. 2757–2771, 2017.

[33] H. Jin, X. Yao, and Y. Chen, "Correlation-aware QoS modeling and manufacturing cloud service composition," *Journal of Intelligent Manufacturing*, vol. 28, no. 8, pp. 1947–1960, 2017.

[34] Y. Laili, S. Lin, and D. Tang, "Multi-phase integrated scheduling of hybrid tasks in cloud manufacturing environment," *Robotics and Computer-Integrated Manufacturing*, vol. 61, pp. 101850.1–101850.18, 2020.

[35] H. Huang, Z. Wang, Y. J. Ji, and Y. Yan, ""Analysis on distributed networked cloud manufacturing mode," [In Chinese]," *Modern Manufacturing Engineering*, vol. 11, pp. 42–48+65, 2017.

[36] H. B. MengChu Zhou and M. C. Zhou, "Role-based collaboration and its kernel mechanisms," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 36, no. 4, pp. 578–589, 2006.

[37] H. Zhu, M. Zhou, and R. Alkins, "Group role assignment via a kuhn-munkres algorithm-based solution," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 42, no. 3, pp. 739–750, 2012.

[38] Y. Sheng, H. Zhu, X. Zhou, and W. Hu, "Effective approaches to adaptive collaboration via dynamic role assignment," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 1, pp. 76–92, 2016.

[39] H. Zhu, "Avoiding conflicts by group role assignment," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 4, pp. 535–547, 2016.

[40] H. Zhu, Y. Sheng, X. Zhou, and Y. Zhu, "Group role assignment with cooperation and conflict factors," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 6, pp. 851–863, 2018.

[41] D. Liu, Q. Jiang, H. Zhu, and B. Huang, "Distributing UAVs as wireless repeaters in disaster relief via group role assignment," *International Journal of Cooperative Information Systems*, vol. 29, no. 01n02, Article ID 2040002, 2020.

[42] H. Zhu, D. Liu, S. Zhang, Y. Zhu, L. Teng, and S. Teng, "Solving the Many to Many assignment problem by improving the Kuhn-Munkres algorithm with backtracking," *Theoretical Computer Science*, vol. 618, pp. 30–41, 2016.

[43] X. Xiangqian, Y. Kewei, D. Yajie, Z. Zhou, and T. Yuejin, "High-end equipment development task decomposition and scheme selection method," *Journal of Systems Engineering and Electronics*, vol. 32, no. 1, pp. 118–135, 2021.