Hindawi
Complexity
Volume 2019, Article ID 7819523, 20 pages
https://doi.org/10.1155/2019/7819523

Research Article

A Heuristic Algorithm for Optimal Service Composition in Complex Manufacturing Networks

Yinan Wu,[1] **Gongzhuang Peng**,[2] **Hongwei Wang**,[3] **and Heming Zhang**[1]

[1] *Department of Automation, Tsinghua University, Beijing 100084, China*
[2] *Engineering Research Institute, University of Science and Technology Beijing, Beijing 100083, China*
[3] *ZJU-UIUC Institute, Zhejiang University, 314400 Haining, China*

Correspondence should be addressed to Hongwei Wang; hongweiwang@zju.edu.cn

Received 28 November 2018; Revised 28 February 2019; Accepted 18 March 2019; Published 1 April 2019

Academic Editor: Michele Scarpiniti

Service composition in a Cloud Manufacturing environment involves the adaptive and optimal assembly of manufacturing services to achieve quick responses to varied manufacturing needs. It is challenged by the inherent heterogeneity and complexity of these services in terms of their diverse and complex functions, qualities of service, execution paths, etc. In this paper, a manufacturing network is constructed to explicitly identify and describe the relationships between individual services based on their attributes. On this basis, the service composition problem can be modeled as a multiple-constrained optimal path (MCOP) selection problem by taking into account different types of composition, namely, sequence, parallel, selection, and cycle. A novel Dual Heuristic Functions based Optimal Service Composition Path algorithm (DHA_OSCP) is proposed to solve the NP-Complete MCOP problem, which involves exploiting the backward search procedure with different search targets to obtain two heuristic functions for the forward search procedure. The proposed algorithm is evaluated through a set of computational experiments in which the proposed algorithm and other popular algorithms such as MFPB_HOSTP are applied to the same dataset, and the results obtained show that DHA_OSCP can efficiently find the optimal service composition path with better Quality of Service (QoS). The viability of DHA_OSCP is further proved in a case study of services composition on a Cloud Manufacturing platform.

1. Introduction

The rapid development of information technologies such as the Internet of things (IoT) and Cloud Computing has led to substantial changes in the manufacturing industry. To quickly respond to fast-changing market needs and satisfy customers' diverse requirements in terms of product performance and cost, companies begin to seek ways of sharing resources and fulfill needs through collaboration in a manufacturing network. Among various advanced manufacturing modes, Cloud Manufacturing (CMfg) is one of the most comprehensive and widely applied [1–4]. In CMfg, different kinds of manufacturing resources are interconnected, virtualized, and encapsulated as Cloud-based services which are managed by intelligent CMfg platforms. To deal with complex manufacturing tasks, resources with varied functions and different QoS levels are selected and composed on these platforms. Thus, service composition has become a key enabling technology in the technical framework for CMfg. Due to the rapid development of CMfg, research work on manufacturing service composition is rapidly increasing [5]. In the process of service composition, functional and nonfunctional attributes are both taken into account to search for the optimal compositions of services for meeting the functional requirements of complex manufacturing tasks with improved satisfaction of the services. Nonfunctional attributes of manufacturing services are usually relative to QoS, such as time latency, price, reliability, availability, and so on. When multiple manufacturing services with the same functional attributes are able to meet the requirements of a manufacturing task, QoS attributes play an important role in resource selection and service composition.

Service composition problem (SCP) is one of the most primary and essential problems for optimally manufacturing resources allocating [6]. SCP in CMfg has the characteristics of high heterogeneity and large scale. Traditional service

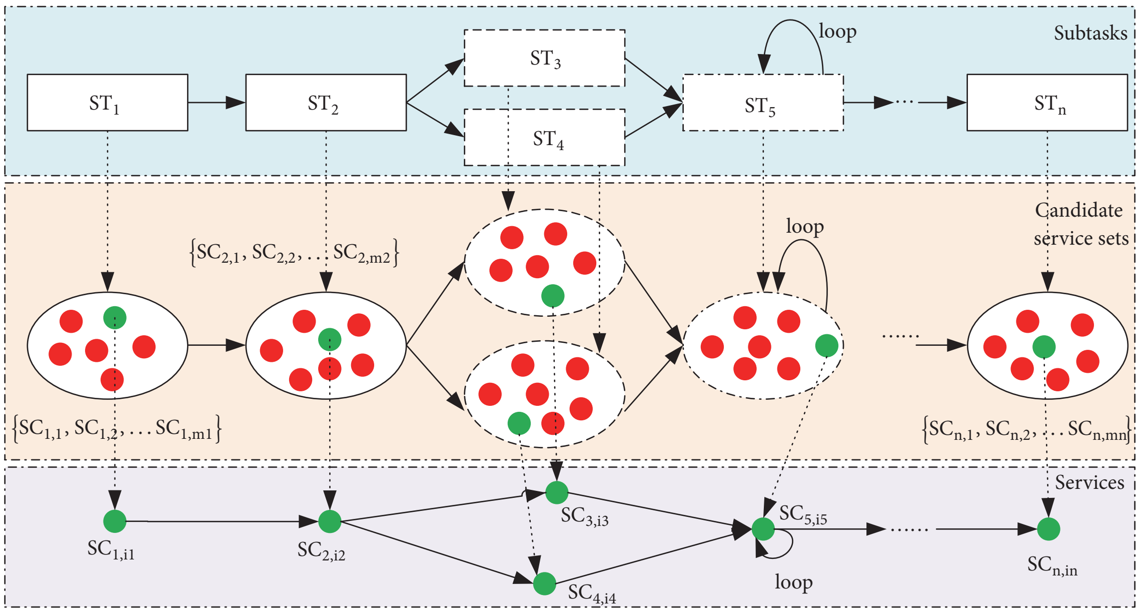


Figure 1: Framework of a traditional service composition process.

composition approaches in CMfg usually use the service composition model of Cloud Computing, which are set up based on the QoS values of service nodes. A traditional service composition framework is shown in Figure 1 where $\{ST_1, ST_2, \ldots, ST_n\}$ is the set of subtasks of manufacturing task T. The service candidate set that are able to execute subtask ST_i can be represented as $\{SC_{i1}, SC_{i2}, \ldots, SC_{im}\}$. Traditional service composition methods focus on selecting appropriate service nodes from each service candidate set to obtain the optimal service composition. This idea is based on the assumption that all manufacturing services in the adjacent service candidate sets can be composed. This assumption is reasonable in the case of composition problem in web services because web services follow the standard access protocol (i.e., SOAP) and registration specifications (i.e., UDDI). This ensures good compatibility between different web services. Moreover, the cost of transportation between different web services is negligible compared to the costs of these web services. However, the composition of adjacent manufacturing services may not be possible in the field of manufacturing, due to the process constraints, partnership constraints, and other factors. If two manufacturing services are geographically far apart, the logistics costs between them cannot be ignored at all. So the manufacturing service composition model should be improved based on Figure 1 and take into account the cooperation relationships between different manufacturing services. Moreover, traditional service composition methods have some limitations when the business process is not well predefined [7]. As a consequence, the addressing of these problems requires a more accurate model and a more efficient algorithm.

By taking into account the cooperation relationship between different service nodes, a manufacturing service network model is constructed in this paper. In the process of service composition, the constraint of each QoS attribute imposed by a service user is considered and the appropriate manufacturing services are selected to achieve the best composition performance in terms of QoS. To address this issue, the processing method for the complex manufacturing network needs to be put forward first. Compared with traditional processing methods for complex structures, the processing method proposed in this paper takes into account not only the characteristics of a manufacturing service but also the relationship between different manufacturing services. After the preprocessing of complex structures of the manufacturing network such as parallel structure and cyclic structure, the complex manufacturing network is converted into a simple directed graph, and then the problem of manufacturing service composition can be abstracted as a Multi-Constraint Optimal Path (MCOP) selection problem, which is NP-Complete [8]. Aiming at solving the MCOP selection problem, some algorithms have been designed. However, the solution quality and time efficiency of these algorithms can be optimized by designing a better heuristic method. This paper precisely aims to put forward a more accurate model and solving the MCOP selection problem in a more efficient way. The main contributions of this work include the following.

(1) A more accurate model for manufacturing service composition problem is proposed to effectively describe both the service nodes and their cooperation relationship based on network architecture.

(2) A processing method for complex process structures (e.g., parallel structure and cyclic structure) based on QoS aggregation, which can model a service composition problem as a standard MCOP formulation.

(3) A novel Dual Heuristic Functions based Optimal Service Composition Path algorithm (DHA_OSCP) for solving the MCOP problem with better solution quality and higher time efficiency.

The rest of this paper is organized as follows. Section 2 gives a literature review of related work. Section 3 details the formulation of the mathematical model for the service composition problems in which the cooperation relationships between nodes are taken into account. Section 4 describes a novel improved algorithm for MCOP selection problems based on the service composition model. Section 5 discusses the experimental results of the DHA-OSCP in solving SCP and compares its performance with those of the popular algorithms including the H_OSTP algorithm and the MFPB_HOSTP algorithm. Section 6 further demonstrates the viability of the proposed model and algorithm with a detailed application example of Cloud Manufacturing. Finally, the conclusions of this work are discussed in Section 7 as well as potential future work.

2. Related Work

2.1. Composition Using QoS. Much research has been done on the service composition problems and other related problems in networked manufacturing such as service provider selection methods, service network models, service composition algorithms, and manufacturing resource allocation [6, 9]. For example, Wu et al. [10] proposed a fuzzy-based decision-making method to find the best service provider; Ho et al. [11] made a review of multicriteria decision making approaches to the supplier evaluation and selection problem. In these methods, the supplier selection problem is described as a form of resource optimal allocation which is restricted to small scale resource scheduling problems without consideration of the QoS indexes of resources (e.g., cost, reliability, and so on). These models and algorithms are not suitable for dynamic and flexible on-demand service composition optimization problems. Although the construction of a Cloud Manufacturing service composition model can refer to Cloud Computing models, many changes need to be made by taking account of the particular characteristics of manufacturing processes. The previous research on service composition is mostly focused on the framework, indexes, and the optimization of composition algorithms. According to the specific means of problem-solving, the service composition algorithms can be divided into two types, namely, the local optimization approach and the global optimization approach. Specifically, the former chooses services for each subtask, obtains the candidate solution sets, and finally gets the local optimal result through greedy selection. However, it has some limitations such as the satisfaction of the overall QoS constraint, processing of non-linear QoS index problems, and so on. The latter considers the QoS attributes not only for a single service but also for the composite service so that they can get the global optimal solution. Global optimization algorithms can be divided into three categories: Non-heuristic (Exact) Algorithms, Heuristic Algorithms, and Meta-Heuristic Algorithms. Every optimization problem can be solved by exhaustive search if the time consumption and search space are ignored. The Non-heuristic (Exact) algorithm is an optimized version of the exhaustive search, which can reduce the time consumption of the algorithm to a large extent. Yu et al. [12] modeled the service composition problem as a multichoice knapsack problem which is multidimensional and multiobjective and obtained the best utility function solution when the global QoS constraints are satisfied. Grabrel et al. [13] proposed an algorithm using the dependency graph and 0-1 linear programming to solve the optimal composition problem for transactional web services. Yang et al. [14] solved the dynamic composition problem of web services by proposing a Greedy Quick-hull algorithm.

The objective function is designed to guarantee that the search direction is a sufficient descent direction per round of iteration and the optimal solution can be obtained in a relatively short period of time. Compared with Exact algorithms, the heuristic function has great advantages in reducing time cost and search space. Klein et al. [15] put forward the hill-climbing algorithm and proved that it had a lower time complexity compared with the linear integer programming. Luo et al. [16] proposed a heuristic HCE algorithm for web service composition optimization which also satisfied the end to end QoS constraints. Rodrigues et al. [17] presented an A$*$ algorithm which solves the problem of semantic input-output message structure matching for web service composition. In order to satisfy the overall QoS constraints and reduce the time complexity, several heuristic algorithms [18–20] have been proposed to find a near optimal solution for service composition. Heuristic function design is critical to a heuristic search algorithm, which has been extensively researched in the area of heuristic algorithm development and selection. In this paper, a novel dual heuristic functions algorithm is designed to solve the service composition problem based on an improved A$*$ algorithm in which two different heuristic functions are employed by considering both feasibility and quality of the composition results at the same time.

By generating or selecting a heuristic method, the meta-heuristic algorithm is designed to provide a sufficiently good solution to a specific optimization problem, which is applicable to a broad range of problems. Some commonly used meta-heuristic algorithms have been improved and adjusted to solve the problem of service composition such as particle swarm optimization (PSO) [21, 22], simulated annealing (SA) [23], genetic algorithms (GA) [24–27], ant colony optimization (ACO) [28], and bee colony algorithms (BA) [29, 30]. The service composition problem can be formulated as a multiobjective problem and near-optimal solutions can be obtained by employing the meta-heuristic algorithm. In addition, some other methods have also been adopted to solve the problem of service composition. For instance, Bekkouche et al. [31] described a novel approach based on a Harmony Search algorithm that addressed functional requirements and nonfunctional requirements simultaneously through a fitness function, to select the optimal or near-optimal solution in semantic web service composition; Jatoth et al. [32] proposed a novel MapReduce-based Evolutionary Algorithm with Guided Mutation that lead to a better Big service composition with better solution and execution time; Labbaci et al. [33] put forward a deep learning approach for dynamic QoS based service composition which got promising results compared with existing QoS prediction techniques. These pieces of work have shown that the service composition problem is an essential part of the current research on intelligent manufacturing.

Since the Cloud Manufacturing paradigm entails the provision of cloud services by physical manufacturing resources and the transportation of between these resources, the cooperation relationship and logistics cost between these manufacturing resources need to be considered. As such, the network model proposed in this paper is more appropriate to describe the manufacturing service composition problem compared with those proposed in the existing studies. So far, the research of correlation-aware service composition has been focused by only a few studies [34]. Unfortunately, the solving methods of these studies are limited to PSO and other meta-heuristic algorithms, which are not intuitive and comprehensive. By preprocessing the complex manufacturing network, the service composition problem based on the improved model is converted to an MCOP problem, which can be solved by pathfinding methods. On this basis, a heuristic pathfinding algorithm for addressing the service composition problems in manufacturing networks is developed by taking into account both service attributes and the relationships between services.

2.2. Multiple-Constrained Path Selection Method. Existing work has shown that QoS-based service composition can be modeled as a multiple-constrained optimal path selection problem, which has been proved to be NP-Complete [9]. This model takes into account not only the QoS indexes of manufacturing service nodes but also the cooperative relationships between different nodes; this makes it more suitable for the actual manufacturing environment. Korkmaz et al. [8] developed a heuristic H_MCOP algorithm for solving the multiple-constrained optimal path selection problem in service invocation. Based on this method, Liu et al. proposed the Heuristic Optimal Social Trust Path (H_OSTP) algorithm [35] and the Multiple Foreseen Path-Based Heuristic Optimal Social Trust Path (MFPB_HOSTP) algorithm [36], which made a two-way search in the trust network based on Dijkstra's shortest path algorithm [37] to get a near optimal solution. These three algorithms will be described in detail in Section 3 of this paper. Before H_OSTP, H-MCOP was one of the most promising algorithms for the multiple-constrained optimal path selection problem due to its outstanding performance in terms of both solution quality and algorithm efficiency. The H_OSTP algorithm inherited the advantages of the H_MCOP algorithm and can achieve better search results and faster search speed by using a better search strategy. Yet, there is a problem of QoS imbalance in H_OSTP algorithm. MFPB_HOSTP algorithm can solve this problem by calculating the intermediate path, but it brings unbearable time consumption when the imbalance problem occurs frequently. H_OSTP and MFPB_HOSTP concentrate on the trust network in social networks, which unfortunately ignore the characteristics of manufacturing processes. For a practical manufacturing environment, the DHA-HOSCP algorithm is proposed in this paper, which designs two heuristic functions according to the characteristics of the manufacturing network. It achieves better results and lower time consumption compared with the MFPB_HOSTP algorithm [36].

3. Modeling of Manufacturing Networks

3.1. Problem Description. In the execution process of manufacturing services on a CMfg platform, the free manufacturing resources are encapsulated as services with different levels of information granularity relative to their manufacturing capabilities. For instance, machining has a higher level of granularity than cutting and milling. Among these services, the coarse-grained manufacturing services are composed of fine-grained manufacturing services, and there are some atomic services which cannot be further decomposed. Manufacturing service nodes and their relationships will form a complex network of CMfg services. Atomic services are represented as nodes in the network, which have functional attributes and nonfunctional attributes. The cooperation relationships between manufacturing service nodes are represented as the edges of the network, including the degree of cooperation intimacy, logistics cost, and other factors. To better illustrate the idea, the improved model is shown in Figure 2. In this model, not all adjacent manufacturing services can be composed due to uncertain relationships between different services.

The major difference between the manufacturing network model and traditional service composition models is in whether the edges (cooperation, logistics, etc.) between manufacturing service providers are considered. As opposed to a computer network in which the communication time and cost between nodes are negligible compared with the node itself, a manufacturing network involves considerable logistics cost and cooperation relationship between manufacturing service providers. Therefore, both the service providers and the relationships between them should be taken into consideration in model construction and algorithm development.

To accomplish a complex manufacturing task, the demand for orchestrating manufacturing services is decomposed into several atomic tasks, each of which can be completed by a specific atomic service from the candidate set. The information about alternative services and their relationships is retrieved from the Cloud Manufacturing service network and is constructed as a subgraph. Thus the service composition problem is transformed into the problem of selecting a path with the highest global utility value under the condition that the subgraph is covered by this path and the QoS constraints are satisfied. The process of optimal path selection is illustrated in Figure 3, which can be divided into three stages.

(1) Manufacturing Demand Analysis and Task Decomposition. When a CMfg platform receives a complex task demand (denoted as T), it first decomposes the complex manufacturing task into a combination of atomic tasks according to the specific requirements and manufacturing process constraints. The service composition is usually divided into four types: sequence, parallelism, selection, and cycle [38]. The overall QoS requirements for a manufacturing task can be denoted

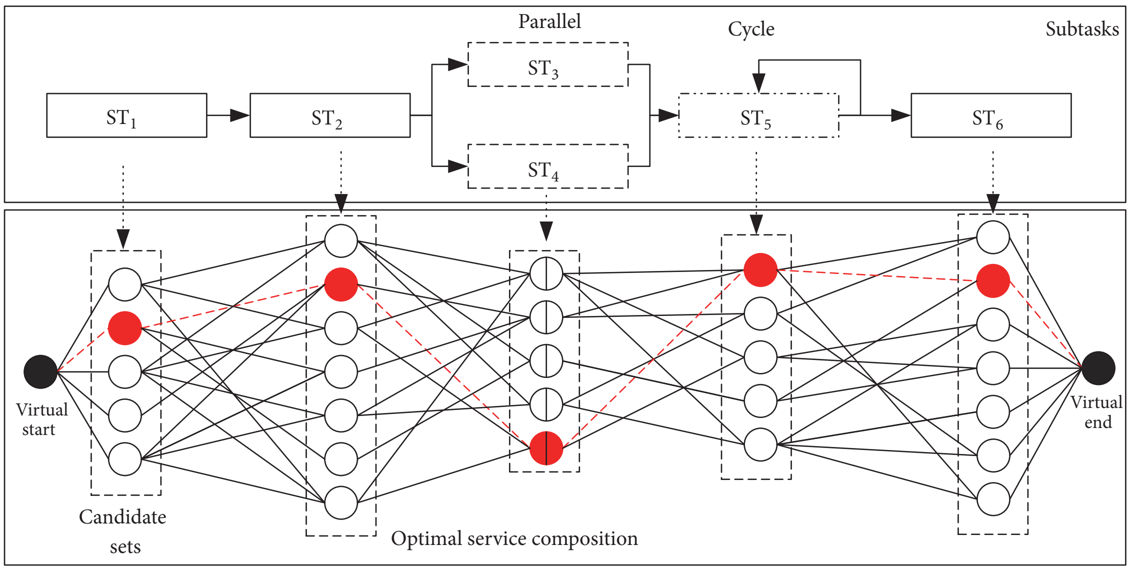


FIGURE 2: Manufacturing service composition model.

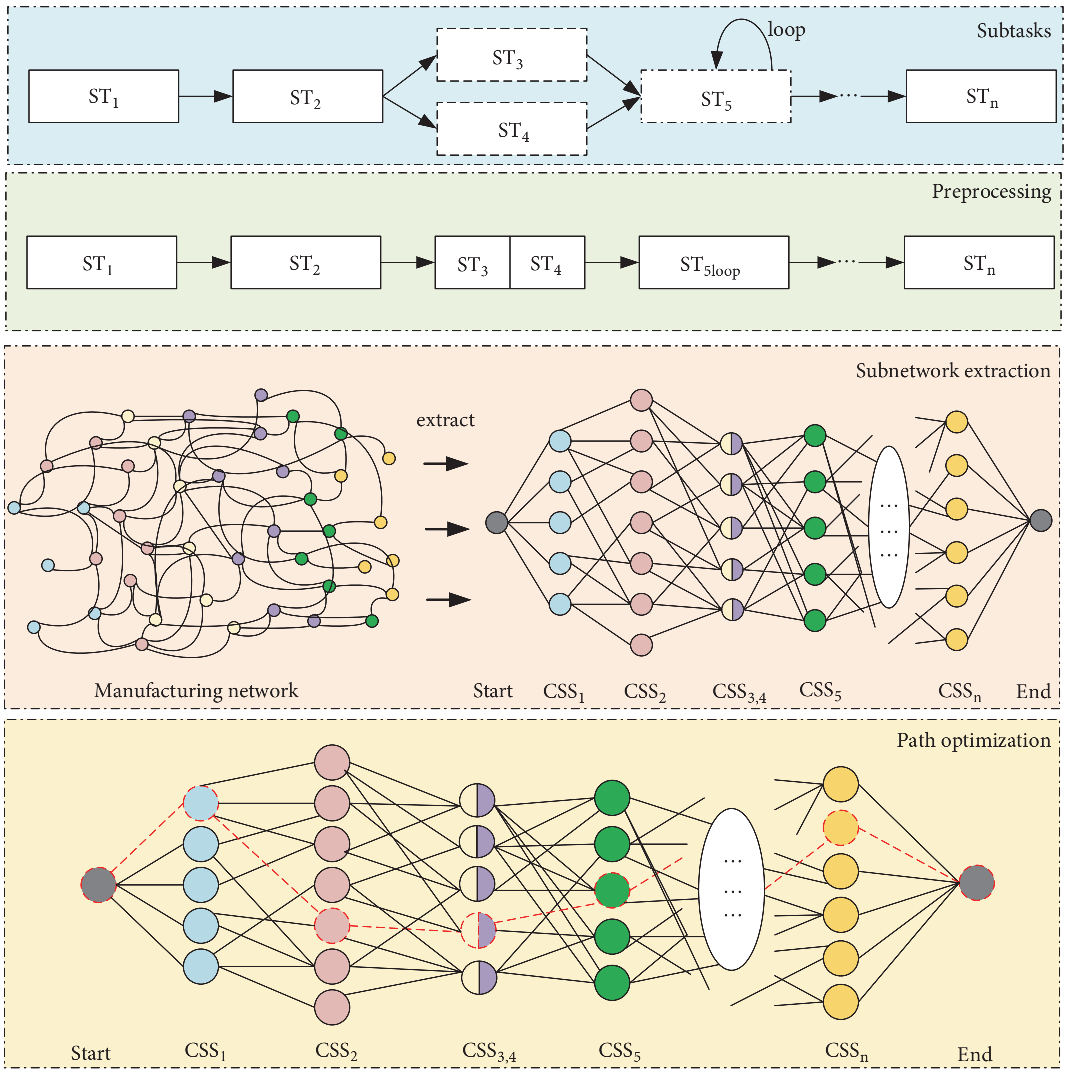


FIGURE 3: Process of optimal path selection.

as $Q(T) = \{q_1(T), q_2(T), \ldots q_n(T)\}$, where $q_i(T)$ is the ith QoS requirement and n is the number of QoS indexes.

(2) Extraction of Candidate Service Subnetwork. The CMfg platform generates a candidate manufacturing service set for each atomic manufacturing task by manufacturing resource selection according to the functional requirements of the atomic manufacturing task. The candidate service set can be denoted as

$$\begin{gathered} CS(T) = \{cs(t_1), cs(t_2), \ldots cs(t_m)\} \\ cs(t_i) = \{s_1(t_i), s_2(t_i), \ldots s_{n_i}(t_i)\} \end{gathered} \tag{1}$$

In the above equation, m is the number of atomic manufacturing tasks and n_i is the number of candidate services for atomic manufacturing task i. Candidate manufacturing services for the same atomic manufacturing task have the same functional properties but their QoS attributes can be different. These candidate service nodes and their relationships are then extracted from the manufacturing network as a subgraph.

(3) Selection of the Optimal Service Path. After the previous steps are completed, each of the atomic manufacturing tasks has got a set of services that can potentially meet its specific requirements. The optimal service path selection is then used to find out the optimal service execution path that satisfies the QoS constraints of manufacturing tasks and has the best overall utility function. This is an MCOP problem which has been proved to be NP-complete. In the past, the solving method of the manufacturing service composition is based on the idea of web service composition, which does not consider the relationship between different service nodes. According to the location constraints, cooperation constraints, process constraints, and QoS constraints, both the attributes of the candidate manufacturing service nodes and their relationships are taken into consideration so that the actual manufacturing scenario can be better addressed. This paper specifically focuses on the effective model and algorithm for this step.

3.2. Mathematical Model. The optimal service composition problem can be modeled as an optimal path selection problem with specific QoS constraints. Before the algorithm is detailed, the methods for QoS aggregation, network structure preprocessing, and mathematical model formulation need to be given first.

(1) QoS Attributes of Manufacturing Network. The QoS attributes in Cloud Computing network usually includes response time, bandwidth, computing overhead, and so on. However, in a manufacturing network, the QoS attributes that should be particularly considered are different from those of Cloud Computing. In a manufacturing network, the time and cost it takes to complete the entire manufacturing process are the essential attributes of substantial importance. Moreover, the reliability, usability, credibility, and sustainability are also important QoS attributes in a manufacturing network. In this paper, time, cost, and reliability are applied in the DHA_OSCP algorithm as three representative QoS attributes.

(2) QoS Attribute Aggregation. Time. Time refers to the execution period from the time when the demands are submitted to the platform to the final completion time of the manufacturing service. It consists of the time of the nodes and the time of the edges. For a manufacturing service node, Time is the sum of online time and offline time (e.g., resource configuration time, computing time, queue time, and execution time). For the edge of the network, time refers to the logistics time of raw materials and semi-finished products as well as to the delivery time. The time attribute needs to be corrected by the correction factor, so as to modify the evaluation value to indicate the time attribute more accurately. The calculation method of time is given by

$$\begin{aligned} Time &= T_{corr} \times \left(T_{node} + T_{edge}\right) \\ T_{node} &= T_{res} + T_{computing} + T_{queue} + T_{execution} \\ T_{edge} &= T_{logistic} + T_{delivery} \end{aligned} \tag{2}$$

Costs. Costs also includes both the cost of the node and the cost of the edge. For a manufacturing service node, Costs is the sum of software costs and hardware costs. Software costs are composed of computing costs, transmission costs, and access costs. Hardware costs include management costs, material costs, personnel costs, and execution costs. For the edge of the network, costs refer to the logistics costs of raw materials and semifinished products as well as to the delivery costs. The calculation of costs is given by

$$\begin{aligned} Costs &= C_{node} + C_{edge} \\ C_{node} &= C_{software} + C_{hardware} \\ C_{software} &= C_{computing} + C_{transmission} + C_{access} \\ C_{hardware} &= C_{management} + C_{material} + C_{personnel} \\ &\quad + C_{execution} \\ C_{edge} &= C_{logistic} + C_{delivery} \end{aligned} \tag{3}$$

Reliability. Reliability represents the possibility that the manufacturing service can successfully complete the manufacturing task under certain QoS constraints. Reliability can be expressed by the ratio of the number of manufacturing tasks successfully executed to the total number of tasks received by the service node. F_k indicates the number of manufacturing tasks that the service node k failed in the process of service execution. The calculation of reliability is given by

$$Reliability = \frac{F_k}{TotalTask^k} \tag{4}$$

Based on the QoS index aggregation method, time and costs are cumulative indexes while reliability is a multiplicative index. Assume that the QoS model contains m manufacturing service nodes and n QoS indexes. The model can be expressed

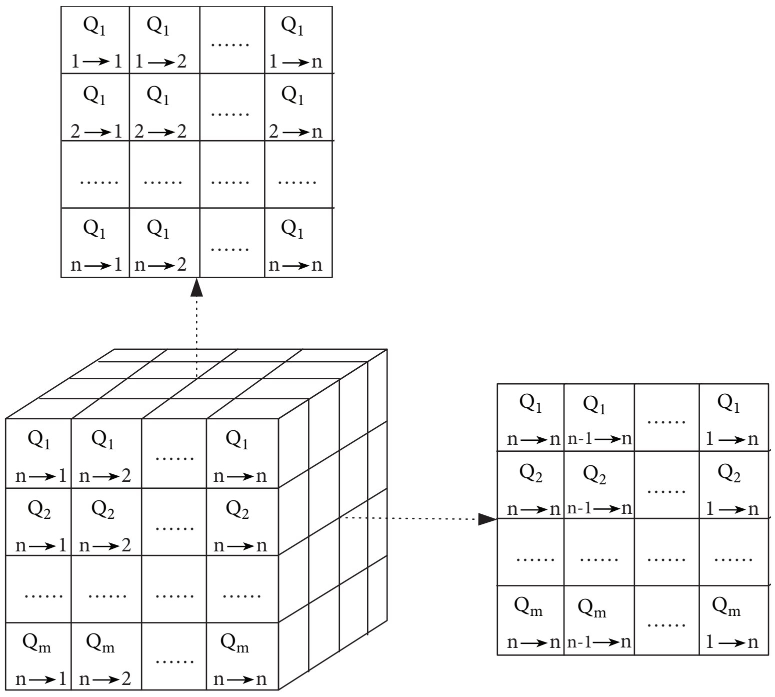


Figure 4: Cube Model of QoS attributes.

by a three-dimensional matrix. Each "layer" of the three dimensional matrix represents the adjacency matrix of a single QoS index of the manufacturing service network. The matrix element a_{ii} represents the QoS attributes of node i and $a_{i,j}$ represent the QoS of the edge from node i to node j. The QoS 3D model of the manufacturing service network is shown in Figure 4.

(3) Preprocessing of Network Structure. In different manufacturing processes, the completion of a manufacturing task may involve four kinds of process structures, namely, sequence, parallelism, selection, and cycle. In order to apply the heuristic multiconstrained optimal path search method to solve the manufacturing service composition problem, the parallel structure and the cyclic structure need to be preprocessed in the manufacturing service network ascribed to the characteristics of Dijkstra algorithm. Finally, the extracted manufacturing service subnetwork is processed into a weighted directed acyclic graph from the start point to the end point, and the multiconstraint service composition problem is transformed into a multiconstrained optimal path selection problem. Since the service composition model proposed in this paper considers not only the manufacturing service itself but also the relationship between manufacturing services, the preprocessing of the network structure also needs to consider the attributes both of nodes and of edges. The aggregation functions for QoS attributes of different services composition types are illustrated in Table 1 according to the recursive characteristics [39] of manufacturing services.

(4) The Method of QoS Attributes Aggregation. During the search process, the aggregation method of QoS attributes needs to be determined to support calculation in the algorithm. For time and cost, the algorithm needs to add them up to get the total time and cost of the existing composition. For reliability, the product of different nodes (edges) is the reliability of the composition service. Therefore, the aggregation methods of different nodes and edges in the manufacturing network are shown below.

Time. Time is a cumulative index. Suppose that an optimal manufacturing path has n candidate; then the aggregated time can be calculated using (5), which includes the aggregation of service nodes and the aggregation of edges.

$$A\left(T_{v_s \longrightarrow v_t}\right) = \sum_{i=2}^{n-1} T_{node}^{i} + \sum_{j=2}^{n-2} T_{edge}^{j,j+1} \qquad (5)$$

Cost. Cost is a cumulative index. The aggregated Costs can be calculated using

$$A\left(C_{v_s \longrightarrow v_t}\right) = \sum_{i=2}^{n-1} C_{node}^{i} + \sum_{j=2}^{n-2} C_{edge}^{j,j+1} \qquad (6)$$

Reliability. Reliability is a multiplicative index. Suppose that optimal manufacturing path has n candidate; then the aggregated reliability can be calculated by

$$A\left(R_{v_s \longrightarrow v_t}\right) = \prod_{i=2}^{n-1} R_{node}^{i} \times \prod_{j=2}^{n-1} R_{edge}^{j,j+1} \qquad (7)$$

Table 1: Aggregation Method of Different Process Structures.

Structure	Time	Cost	Reliability
Parallel	$q_{par}^{time} = \max\left(q_i\,(node)\right) + \max\left(q_j\,(edge)\right)$	$q_{par}^{cost} = \sum_{i=1}^{n} q_i\,(node) + \sum_{i=1}^{m} q_j\,(edge)$	$q_{par}^{reli} = \prod_{i=1}^{n} q_i\,(node) \times \prod_{j=1}^{m} q_j\,(edge)$
Selective	$q_{sel}^{time} = \sum_{i=1}^{n} q_i\,(node) \times \omega_i + \sum_{j=1}^{m} q_j\,(edge) \times \omega_j$	$q_{sel}^{cost} = \sum_{i=1}^{n} q_i\,(node) \times \omega_i + \sum_{j=1}^{m} q_j\,(edge) \times \omega_j$	$q_{sel}^{reli} = \prod_{i=1}^{n} q_i\,(node) \times \omega_i \times \prod_{j=1}^{m} q_j\,(edge) \times \omega_j$
Cyclic	$q_{sel}^{time} = \theta \times \left(\sum_{i=1}^{n} q_i\,(node) + \sum_{j=1}^{m} q_j\,(edge)\right)$	$q_{sel}^{time} = \theta \times \left(\sum_{i=1}^{n} q_i\,(node) + \sum_{j=1}^{m} q_j\,(edge)\right)$	$q_{par}^{reli} = \theta \times \left(\prod_{i=1}^{n} q_i\,(node) \times \prod_{j=1}^{m} q_j\,(edge)\right)$

(5) Utility Function. In this model, the utility function describes the overall performance of the optimal service composition path in different QoS aspects. Since different QoS indexes have different scales, the QoS indexes are normalized in the utility function.

The QoS indexes are divided into two kinds: positive indexes and negative indexes. Reliability is a positive multiplicative index, while time and costs are negative cumulative indexes. The utility function is calculated by

$$U(T,C,R) = w_T \times \frac{A\left(T_{v_s \longrightarrow v_t}\right)}{Tconstrains} + w_C \times \frac{A\left(C_{v_s \longrightarrow v_t}\right)}{Cconstrains} + w_R \times \frac{\ln\left(A\left(R_{v_s \longrightarrow v_t}\right)\right)}{\ln(Rconstrains)} \tag{8}$$

In the equation, w_T, w_C, w_R are the weights of T, C, R, respectively, with the conditions of $w_T + w_C + w_R = 1$ and $0 < w_T, w_C, w_R < 1$.

(6) Traditional Service Composition Model. In the existing studies, the manufacturing service composition problem can be described as a 0-1 integer constrained multiobjective optimization problem based on the model in Figure 1. The traditional service composition model is formulated as

$$\begin{gathered} \min U(T,C,R) \\ 0 \le A\left(T_{v_s \longrightarrow v_t}\right) \le Tconstrains \\ 0 \le A\left(C_{v_s \longrightarrow v_t}\right) \le Cconstrains \\ 0 \le A\left(R_{v_s \longrightarrow v_t}\right) \le Rconstrains \end{gathered} \tag{9}$$

In these studies, this problem is usually solved using metaheuristic algorithms such as genetic algorithms (GA), particle swarm optimization (PSO), ant colony optimization (ACO), and bee colony algorithms (BA). These methods usually work pretty well.

(7) Multiple Constrained Optimization Path Selection Model. In the improved model (Figure 2), there are more constraints to ensure the existence of edges between the selected manufacturing service nodes. In summary, the problem of manufacturing service composition can be abstracted as a multiconstrained optimization path selection problem and formulated as (10). In (10), m is the number of subtasks; d_i is the number of candidate services in candidate service set j; and $x_{i,j}$ is the decision variable. $x_{i,j}$ equals 1 if the ith manufacturing service in service candidate j is chosen and otherwise it equals 0.

$$\begin{aligned} \min \quad & U(T,C,R) \\ \text{s.t.} \quad & 0 \le A\left(T_{v_s \longrightarrow v_t}\right) \le Tconstrains \\ & 0 \le A\left(C_{v_s \longrightarrow v_t}\right) \le Cconstrains \\ & 0 \le A\left(R_{v_s \longrightarrow v_t}\right) \le Rconstrains \\ & \sum_{i=1}^{d_1} x_{i,1} = 1 \\ & \sum_{i=1}^{d_2} x_{i,2} = 1 \\ & \cdots \\ & \sum_{i=1}^{d_m} x_{i,m} = 1 \\ & x_{i,j} = 0 \ or \ 1 \end{aligned} \tag{10}$$

The meta-heuristic algorithms can also solve the service selection problem with multi-QoS constraints and decision variable constraints. However, [40] proved that they have low efficiencies in finding a near-optimal solution in large-scale networks. Taking GA as an example, with the increase of node scale, the algorithm takes a very long time to obtain the near optimal solution, and it is often unable to obtain the feasible solution within the set upper limit of iteration times. Compared with meta-heuristic algorithms, the proposed DHA_OSCP is a more direct and effective algorithm to solve this problem. This is ascribed to the advantage that the DHA_OSCP algorithm can always satisfy the constraints of decision variables in the process of execution.

4. Service Composition Path Selection Algorithms

In this section, some existing approximation algorithms for the MCOP selection problem are firstly introduced, including H_MCOP, H_OSTP, and MFPB_HOSTP. Then a novel Dual Heuristic Functions based Optimal Service Composition Path algorithm (DHA_OSCP) is described in detail.

4.1. Existing Algorithms

(1) H_MCOP. H_MCOP [8] is a heuristic algorithm proposed by Korkmaz and Krunz for the multiple–constrained optimal path selection problem. This algorithm first proposed a method of QoS aggregation which is also the target of the reverse search, as shown in

$$g_\lambda(p) \triangleq \left(\frac{q_1(p)}{Q^1_{v_s,v_t}}\right)^\lambda + \left(\frac{q_2(p)}{Q^2_{v_s,v_t}}\right)^\lambda + \cdots + \left(\frac{q_m(p)}{Q^m_{v_s,v_t}}\right)^\lambda \tag{11}$$

H_MCOP first adopts Dijkstra's shortest path algorithm to find the path with the minimum $g_\lambda(p)$ and investigates whether there exists a feasible path satisfying all QoS constraints. If the $g_\lambda(p)$ of an intermediated node v_k is greater than m, it is proved that there is no feasible path from v_k to v_t.

If there exists at least one feasible solution, then this algorithm will search the network from v_s to v_t in order to identify a feasible path with the minimal cost of services.

Before the H_OSTP algorithm [35] was proposed in 2010, H_MCOP was one of the most promising algorithms for the MCOP selection problem. It proved to outperform prior existing algorithms in terms of both efficiency and solution quality.

(2) H_OSTP. In H_OSTP, Liu et al. first proposed the objective function given in (12) and adopted the backward search algorithm to identify whether there was a feasible path with the minimal δ from v_t to v_s. If a feasible solution exists, H_OSTP then adopts the forward search algorithm to deliver a near-optimal solution.

$$\delta(p) \triangleq \max\left\{\frac{1-T_p}{1-Q^T_{v_s,v_t}}, \frac{1-T_p}{1-Q^T_{v_s,v_t}}, \frac{1-T_p}{1-Q^T_{v_s,v_t}}\right\} \tag{12}$$

H_OSTP designed a target function δ(p) which is better than H_MCOP for reverse search, and this algorithm can provide an optimal service composition path that is no worse than H_MCOP. In addition, through the reverse search strategy, H_OSTP can calculate whether the foreseen path is feasible in advance which reduces the search space of the algorithm and improves the efficiency of the algorithm. However, this algorithm has some shortcomings in balancing different QoS attributes.

(3) MFPB_HOSTP. In order to solve the imbalance problem of H_OPTP, Liu et al. proposed the MFPB_HOSTP algorithm [36] in 2013. In addition to selecting δ(p) as the target to search a network, the algorithm also identifies the optimal paths with the searching target T_{min}, C_{min}, and R_{max}. When the imbalance problem occurs, MFPB-HOSTP determines the QoS index (for example, T) that does not satisfy the constraint. Then the algorithm selects the node of the T_{min} path to replace the node with the best δ(p) path as the starting point for the next searching stage. At the same time, the algorithm defines the CBLP path sets to prevent the new imbalance problem. By this method, the algorithm can get a near-optimal path that is no worse than H_OSTP.

The MFPB_HOSTP algorithm inherits a lot of advantages from H_OSTP and uses the CBLP path sets to solve the imbalance problem that best δ(p) searching may bring. However, this algorithm will incur a large amount of computation when there are lots of imbalance problems or complex processes in the service network.

4.2. The DHA_OSCP Algorithm

(1) Overview. First of all, some definitions are introduced.

Definition 1 (feasible heuristic path). In a subnetwork from v_s to v_t, a Feasible Heuristic Path (FHP) is the path from v_t to the intermediate node v_k, identified by the Delta Backward Search with (13) as the target.

$$\delta\left(p^{backward}_{v_k\longrightarrow v_t}\right) = \min\left(\max\left\{\frac{T^{backward}_{v_k\longrightarrow v_t}}{T^{constraint}_{v_s\longrightarrow v_t}}, \frac{C^{backward}_{v_k\longrightarrow v_t}}{C^{constraint}_{v_s\longrightarrow v_t}}, \frac{\ln\left(R^{backward}_{v_k\longrightarrow v_t}\right)}{\ln\left(R^{constraint}_{v_s\longrightarrow v_t}\right)}\right\}\right) \tag{13}$$

Definition 2 (utility heuristic path). In a subnetwork from vs to v_t, a Utility Heuristic Path (UHP) is the path from v_t to the intermediate node v_k, identified by the utility backward search with (14) as the target.

$$uti\left(p^{backwarduti}_{v_k\longrightarrow v_t}\right) = w_T \times \frac{T^{backwarduti}_{v_k\longrightarrow v_t}}{T^{constraint}_{v_s\longrightarrow v_t}} + w_C \times \frac{C^{backwarduti}_{v_k\longrightarrow v_t}}{C^{constraint}_{v_s\longrightarrow v_t}} + w_R \times \frac{\ln\left(R^{backwarduti}_{v_k\longrightarrow v_t}\right)}{\ln\left(R^{constraint}_{v_s\longrightarrow v_t}\right)} \tag{14}$$

Definition 3 (forward path). In a subnetwork from v_s to v_t, a Forward Path (FP) is the path from v_s node v_t, identified by the forward search. The forward path is no worse than the FHP and UHP.

Based on the definitions above, a novel Dual Heuristic Functions based Optimal Service Composition Path algorithm (DHA_OSCP) is proposed in this paper, which adopts two heuristic functions and the pruning searching strategy to obtain feasible heuristic path (FHP) and utility heuristic path (UHP).

The DHA_OSCP algorithm is divided into two parts, namely, the backward search process determining two heuristic functions and the forward search process confirming the best utility path. In the backward search procedure, the algorithm determines the feasible heuristic path (FHP) from v_k to v_t (denoted by $p^{backward}_{v_k\longrightarrow v_t}$) by using (12) as the search target. Then the algorithm determines the utility heuristic path (UHP) from v_k to v_t (denoted by $p^{backwarduti}_{v_k\longrightarrow v_t}$) with (13) as the search target. The overview of DHA_OSCP algorithm is shown in Figure 5.

(2) Description. A more detailed description of the proposed DHA_OSCP algorithm is given below. The algorithm is divided into four main steps: the pruning of the manufacturing service network, the feasible backward search, the utility backward search, and the A∗ forward search. These parts will be introduced in turn. The pseudocode of the proposed DHA_OSCP is given in Appendix A.

Step 1 (network pruning). Designing appropriate pruning strategies can effectively reduce the search space and the time complexity of the DHA_OSCP algorithm without affecting the quality of the solution. The following two strategies are used to prune the manufacturing service network.

The first strategy is pruning based on the process flow. According to the decomposition of the manufacturing service task, the nodes and the edges of the manufacturing service are extracted so that the manufacturing service network suitable

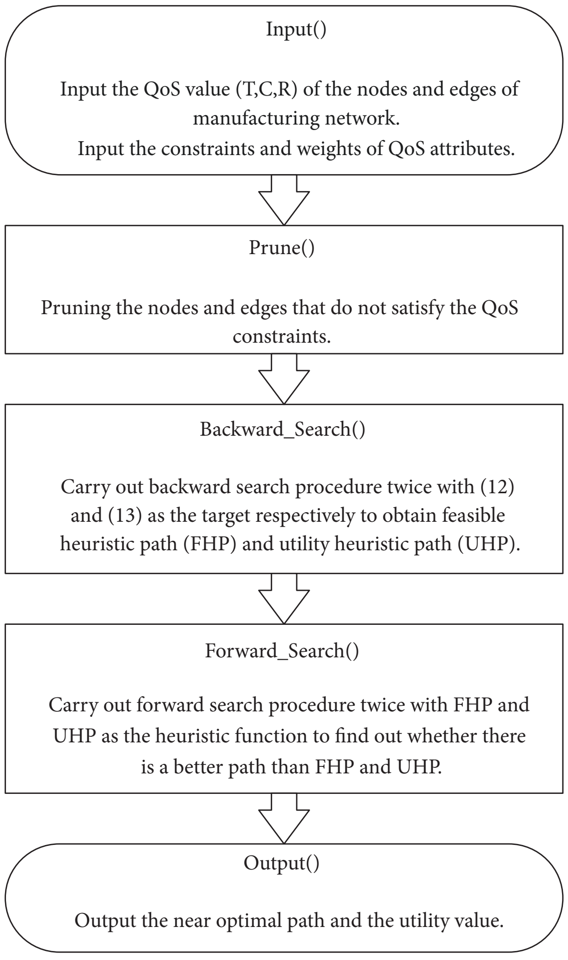

Input()
Input the QoS value (T,C,R) of the nodes and edges of manufacturing network.
Input the constraints and weights of QoS attributes.
Prune()
Pruning the nodes and edges that do not satisfy the QoS constraints.
Backward_Search()
Carry out backward search procedure twice with (12) and (13) as the target respectively to obtain feasible heuristic path (FHP) and utility heuristic path (UHP).
Forward_Search()
Carry out forward search procedure twice with FHP and UHP as the heuristic function to find out whether there is a better path than FHP and UHP.
Output()
Output the near optimal path and the utility value.

Figure 5: Overview of DHA_OSCP algorithm.

for the task is obtained. In the manufacturing network, the edges caused by other process flows are removed by the pruning strategy.

The second strategy is pruning based on QoS constraints. Nodes and edges that cannot satisfy QoS constraints are removed from the network according to (15) where γ represents the threshold.

$$\min\left(\frac{T_{node/edge}}{T^{constraint}_{v_s\longrightarrow v_t}},\frac{C_{node/edge}}{C^{constraint}_{v_s\longrightarrow v_t}},\frac{\ln\left(R_{node/edge}\right)}{\ln\left(R^{constraint}_{v_s\longrightarrow v_t}\right)}\right)>\gamma \qquad (15)$$

Step 2 (identification of a feasible backward path with minimal delta). In feasible backward search procedure, the DHA_OSCP algorithm searches the service network from $\mathrm{v_t}$ to intermediate $\mathrm{v_k}$, to investigate whether there exists a feasible solution in the manufacturing network. Feasible backward search can define the FHP of each intermediate node $\mathrm{v_k}$, which is one of the heuristic functions to support the A$*$ forward search procedure. The pseudocode of Delta Backward Search is given in Appendix B.

Theorem 4. *In the feasible backward search procedure, the search process can successfully find a feasible solution if there is at least one feasible solution that exists in the service network.*

The proof of Theorem 4 is given in Appendix C.

Step 3 (identification of a backward path with the best utility). In the utility backward search procedure, the DHA_OSCP algorithm searches the service network from v_t to each intermediate node v_k and identifies the best weighted mean path using a greedy strategy. $p^{backwarduti}_{v_k\longrightarrow v_t}$ participates in the A$*$ forward search procedure as another heuristic function. The utility backward search procedure needs to be employed to handle the imbalance problems which may occur in the forward search procedure. For example, the QoS constraints of the service network are $(\mathrm{T}\le 1,\mathrm{C}\le 1,\mathrm{R}\ge 0.4)$. The QoS values of the paths from v_t to two intermediate nodes v_i, v_j are $QoS(p^{backward}_{v_i\longrightarrow v_t})=(0.5,0.1,0.9)$ and $QoS(p^{backward}_{v_j\longrightarrow v_t})=(0.45,0.45,0.45)$. Then the feasible backward search procedure will select v_j as the next node while the utility backward search procedure will select v_i as the next node. Obviously, the node v_i has a larger QoS value to spare so that v_i is easier to avoid the imbalance problem. Therefore, choosing the dual heuristic function strategy of FHP and UHP can effectively avoid the occurrence of the imbalance problem and generate a chance to deliver a better solution. The pseudocode of utility backward search is given in Appendix D.

Step 4 (identification the near-optimal path based on FHP and UHP). The forward search procedure will investigate whether there is a path $p^{forward}_{v_s\longrightarrow v_t}$ that is better in quality than both $p^{backward}_{v_s\longrightarrow v_t}$ and $p^{backwarduti}_{v_s\longrightarrow v_t}$. The pseudocode of forward search is given in Appendix E. In this procedure, DHA_OSCP searches the path with the best utility from v_s to v_t based on A$*$ algorithm. Assume that v_{kf} and v_{ku} are selected as the intermediate nodes based on FHP and UHP; then two foreseen paths $p^{forward+backward}_{v_s\longrightarrow v_{kf}\longrightarrow v_t}$ and $p^{forward+backwarduti}_{v_s\longrightarrow v_{ku}\longrightarrow v_t}$ are formed. According to the feasibility of these two paths, DHA-OSCP uses the following strategies to identify the optimal path.

Case 1 (both $p^{forward+backward}_{v_s\longrightarrow v_{kf}\longrightarrow v_t}$ and $p^{forward+backwarduti}_{v_s\longrightarrow v_{ku}\longrightarrow v_t}$ are feasible). The DHA-OSCP algorithm calculates the unity values of the two foreseen paths and then continues to search the network.

Case 2 ($p^{forward+backward}_{v_s\longrightarrow v_{kf}\longrightarrow v_t}$ is feasible and $p^{forward+backwarduti}_{v_s\longrightarrow v_{ku}\longrightarrow v_t}$ is infeasible). The forward search procedure based on FHP can continue to search. The forward search procedure based on UHP will search another neighborhood node of v_s with the best utility and form a new foreseen path $p^{forward+backwarduti'}_{v_s\longrightarrow v'_{ku}\longrightarrow v_t}$. Then the algorithm forms another hybrid foreseen path $p^{forward+backward}_{v_s\longrightarrow v_{ku}\longrightarrow v_t}$ based on FHP. The the algorithm moves forward to determine whether these two foreseen paths are feasible and choose the feasible path with higher utility to continue searching. If both of the two paths are infeasible, then the algorithm starts searching the path from v_s in the subnetwork without taking link $v_s\longrightarrow v_{ku}$ into consideration.

Case 3 ($p^{forward+backward}_{v_s\longrightarrow v_{kf}\longrightarrow v_t}$ is infeasible and $p^{forward+backwarduti}_{v_s\longrightarrow v_{ku}\longrightarrow v_t}$ is feasible). The forward search procedure based on UHP can

Table 2: Complexity analysis of the three algorithms.

Algorithm	Complexity
H_OSTP	O(2×(NlogN+E)) [35]
MFPB_HOSTP	O(6×(NlogN+E)+KE) [36]
DHA_OSCP	O(4×(NlogN+E))

continue to search. The forward search procedure based on FHP will search another neighborhood node of v_s with the minimal δ and form a new foreseen path $p_{v_s \longrightarrow v'_{kf} \longrightarrow v_t}^{forward+backward'}$. Then the algorithm forms another hybrid foreseen path $p_{v_s \longrightarrow v_{ku} \longrightarrow v_t}^{forward+backwarduti}$ based on UHP. Next the algorithm moves forward to determine whether these two foreseen paths are feasible and choose the feasible path with higher utility to continue searching. If both of the two paths are infeasible, then the algorithm starts searching the path from v_s in the subnetwork without taking link $v_s \longrightarrow v_{kf}$ into consideration.

Case 4 (both $p_{v_s \longrightarrow v_{kf} \longrightarrow v_t}^{forward+backward}$ and $p_{v_s \longrightarrow v_{ku} \longrightarrow v_t}^{forward+backwarduti}$ are infeasible). The algorithm adopts the two processing methods described in Cases 2 and 3, respectively, when the corresponding path is infeasible. Then four new foreseen paths $p_{v_s \longrightarrow v'_{ku} \longrightarrow v_t}^{forward+backwarduti'}$, $p_{v_s \longrightarrow v_{ku} \longrightarrow v_t}^{forward+backward}$, $p_{v_s \longrightarrow v'_{kf} \longrightarrow v_t}^{forward+backward'}$, and $p_{v_s \longrightarrow v_{ku} \longrightarrow v_t}^{forward+backwarduti}$ are formed. Then the algorithm moves forward to determine whether these foreseen paths are feasible and choose the feasible path with higher utility to continue searching. If all of these paths are infeasible, then the algorithm starts searching the path from v_s in the subnetwork without taking link $v_s \longrightarrow v_{ku}$ and $v_s \longrightarrow v_{kf}$ into consideration.

(3) Algorithm Complexity Analysis. Assume that only three QoS attributes (i.e., T, C, and R) are taken into consideration; the time complexity of these three algorithms is shown in Table 2 where N is the number of nodes in the subnetwork between v_s and v_t, E is the number of the edges in the manufacturing network, and K is the hops between v_s and v_t.

Below the detailed analysis of these representations is given. Specifically, H_OSTP takes twice as much time as Dijkstra's shortest path algorithm. As such, the time complexity of this algorithm is $O(2\times(N\log N + E))$ which is equal to $O(N\log N + E)$. MFPB_HOSTP adopts Dijkstra's shortest path algorithm four times in the Backward_Search procedure with the time complexity of $O(4\times(N\log N + E))$. In the Forward_Search, MFPB_HOSTP takes twice as much time as Dijkstra's shortest path algorithm, which means the time complexity for this part is $O(2\times(N\log N + E))$. Besides, the time complexity of finding the feasible foreseen paths is O(KE). So the time complexity of MFPB_HOSTP is $O(N\log N{+}KE)$. DHA_OSCP adopts Dijkstra's shortest path algorithm twice both in the Backward_Search procedure and in Forward_Search procedure, so its time complexity is $O(4\times(N\log N + E))$ which is equal to $O(N\log N + E)$.

The analysis above is based on the assumption that only three QoS attributes are taken into account. If M QoS attributes are taken into consideration, the time complexity of MFPB_HOSTP will turn into $O(M\times(N\log N + E){+}KE)$ while the time complexity of DHA_OSCP is still $O(N\log N + E)$, meaning that the proposed algorithm is better for large-scale problems.

5. Computational Experiments and Discussion

In order to evaluate the performance of the proposed DHA_OSCP algorithm, computational experiments are conducted to make various comparisons between DHA_OSCP and popular H_OSTP and MFPB_HOSTP algorithms that have been proved to be effective for service composition problems.

The execution time and the quality of the solution are influenced by the scale and structure of the network and the QoS constraints. In order to study the performance of the proposed heuristic algorithm under varied conditions, the service networks with different scales and structures are randomly generated. Then the qualities of the solutions obtained by the three algorithms are compared in detail. Figure 6 shows the comparisons of results under varied conditions. There are 20 groups of manufacturing networks in the experiments and 20 sets of QoS constraints in each group. Therefore, the results of 400 groups of experiments are shown in each subgraph of Figure 6. QoS preferences regarding the task are chosen as $w_T = 0.5, w_C = 0.6, w_R = 0.8$ and the initial QoS constraint values are chosen as $T_{\text{constraints}} = 0.9, C_{\text{constraints}} = 0.9$ $R_{\text{constraints}} = 0.2$. Then the T, C are increased by 0.01 and the R is reduced by 0.01.

Figure 6 shows that DHA_OSCP algorithm has a larger possibility to find a better solution under different network scale and QoS constraints. The X-axis and the Y-axis represent different network scales and QoS constraints, respectively. The Z-axis represents the utility values obtained by the three algorithms. Each point in the figure represents the utility value of the optimal path under certain network scale and QoS constraints. As shown in the figure, if there are no feasible solutions in the network, all of the three algorithms can determine the infeasibility. Moreover, DHA_OSCP manages to find feasible solutions for all cases without a utility value worse than that of H_OSTP and obtains better results than H_OSTP in 12.375% of the total experiments (i.e., 198 of 1600). DHA_OSCP also obtains better results than MFPB_HOSTP in 10.875% of the total experiments (i.e., 174 of 1600). This validates the effectiveness of the DHA_OSCP algorithm in finding a near-optimal solution. The execution times of the algorithms are influenced by the network scale and the number of hops in the network. An additional experiment with details below is then designed to compare the utility values and execution times of the three algorithms.

Seven groups of networks are generated with a node size of 100, 150, 200, 250, 300, 350, and 400, respectively. For each group of networks, 20 sets of QoS constraints are taken and 20 networks are generated for each set of the QoS constraints to carry out the experiment. The sum of utility values and

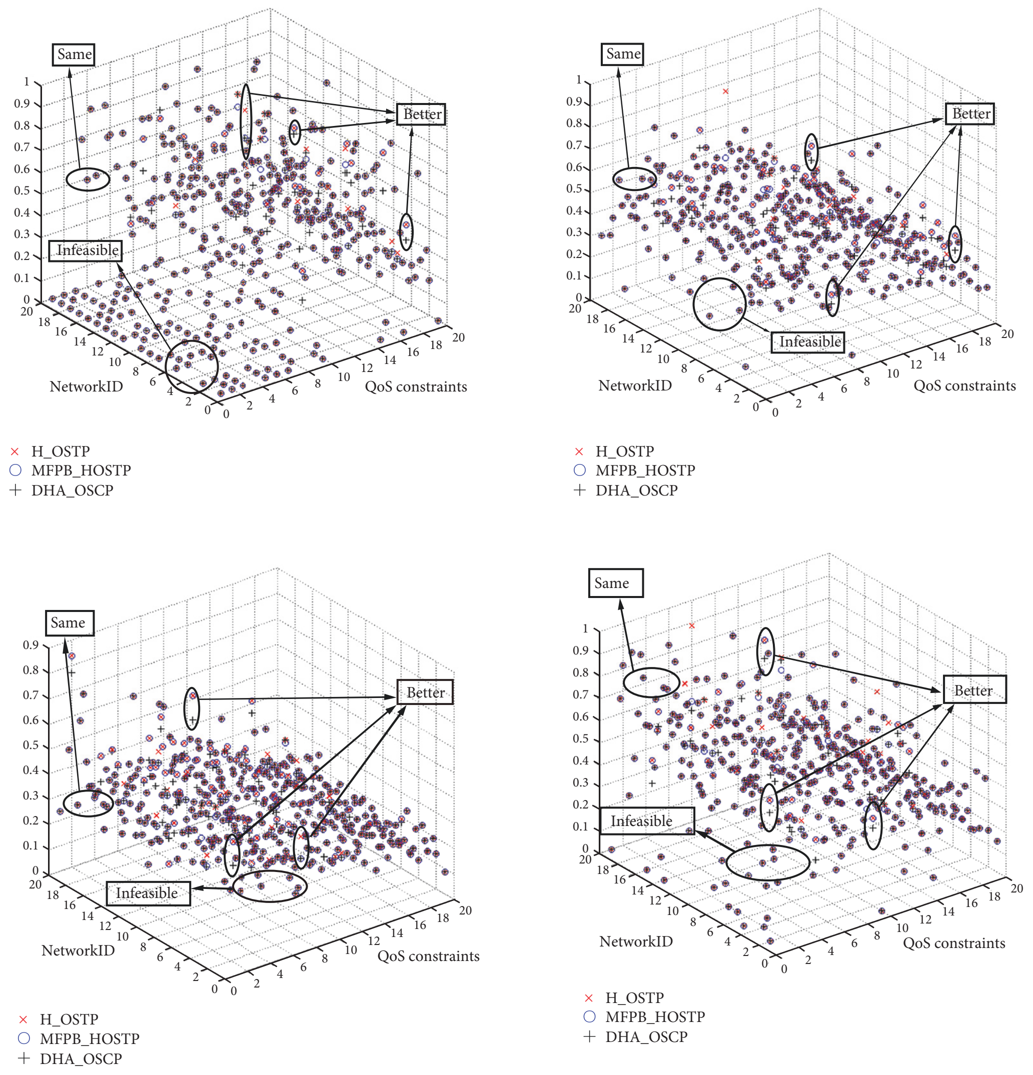


Figure 6: Solution quality with different scales of service network.

execution time of the 400 experiments are compared for each node size. The experiment results are shown in Tables 3–5.

Though the computational experiments were conducted on networks with different scales, structures, and constraints, we can draw the conclusion that on average DHA_OSCP can get the path with better utility value than that of MFPB_HOSTP at a lower time cost. In terms of mean execution time, DHA_OSCP can achieve reduction of time cost by 43.7%, 50.3%, and 40.7%, respectively, for the examples shown in Tables 3–5. Since both DHA_OSCP and MFPB_HOSTP have measures of addressing the imbalance problem at the cost of execution time, they achieve much better utility values than those of H_OSTP but more execution time is incurred. When both quality of result and time cost are considered, DHA_OSCP achieves better overall performance compared with MFPB_HOSTP and H_OSTP, which is important for working with manufacturing service networks.

Additionally, compared with H_OSTP, the DHA_OSCP algorithm adopts two heuristic functions in the forward search procedure; this turns to ensure that, no matter whether imbalanced problems of QoS attributes exist, it can always get a near-optimal path that is no worse than the one obtained using H_OSTP. Although the execution time of DHA_OSCP is increased due to taking additional measure for addressing imbalance problems, it achieves the same level of time complexity as that of H_OSTP. In summary, DHA_OSCP can obtain solutions that are much better than H_OSTP at a reasonable time cost.

Both MFPB_HOSTP and DHA_OSCP can solve the imbalance problem to a great extent. Nonetheless, the time complexity of the DHA_OSCP algorithm is lower. The time complexity of MFPB_HOSTP would become intolerable if the number of hops in the manufacturing network increased to a large level. Moreover, the DHA_OSCP algorithm can get a solution with a much better utility value than that of MFPB_HOSTP. This is because the forward search procedure adopts UHP as a heuristic function which takes into account the coefficients of the utility function. This is a better search strategy compared with MFPB_HOSTP

Table 3: Results of Different Node Size.

Node size	H_OSTP		MFPB_HOSTP		DHA_OSCP	
	Utility value	Execution time	Utility value	Execution time	Utility value	Execution time
100	151.0454	12.1380	149.5541	30.1070	148.0524	23.4170
150	170.6428	25.1980	168.8756	77.0790	165.4677	52.9100
200	148.9640	43.8350	147.1278	94.9910	144.7855	138.0930
250	135.5056	67.2910	134.4681	218.9750	132.5991	147.5550
300	123.8503	98.1380	123.0763	321.9790	120.7582	214.1920
350	113.3137	133.9490	112.7778	466.6080	110.4149	292.3780
400	105.3334	177.5580	104.6502	592.8810	103.1810	385.6320

Table 4: Results of Different QoS Constraints.

QoS constraints			H_OSTP		MFPB_HOSTP		DHA_OSCP	
T	C	R	Utility value	Execution time	Utility value	Execution time	Utility value	Execution time
0.90	0.90	0.2	214.9406	23.5230	214.8814	64.4120	209.3961	44.3610
0.94	0.94	0.18	195.6596	22.9250	195.6285	59.9620	188.4452	42.0600
0.98	0.98	0.16	179.0496	23.1040	179.0496	66.1650	177.2467	43.8820
1.02	1.02	0.14	170.1983	24.4250	170.1769	72.3250	169.8541	46.5420
1.04	1.04	0.12	175.3381	24.5360	171.7547	69.6620	170.1082	46.6200
1.08	1.08	0.10	199.5259	24.7680	198.3515	79.2850	196.7191	49.3220
1.10	1.10	0.08	133.8181	24.1440	133.3709	74.9020	132.6810	52.3690
1.12	1.12	0.06	153.5065	24.3210	153.0866	79.9630	149.5530	51.9410

Table 5: Results of Different Coefficient.

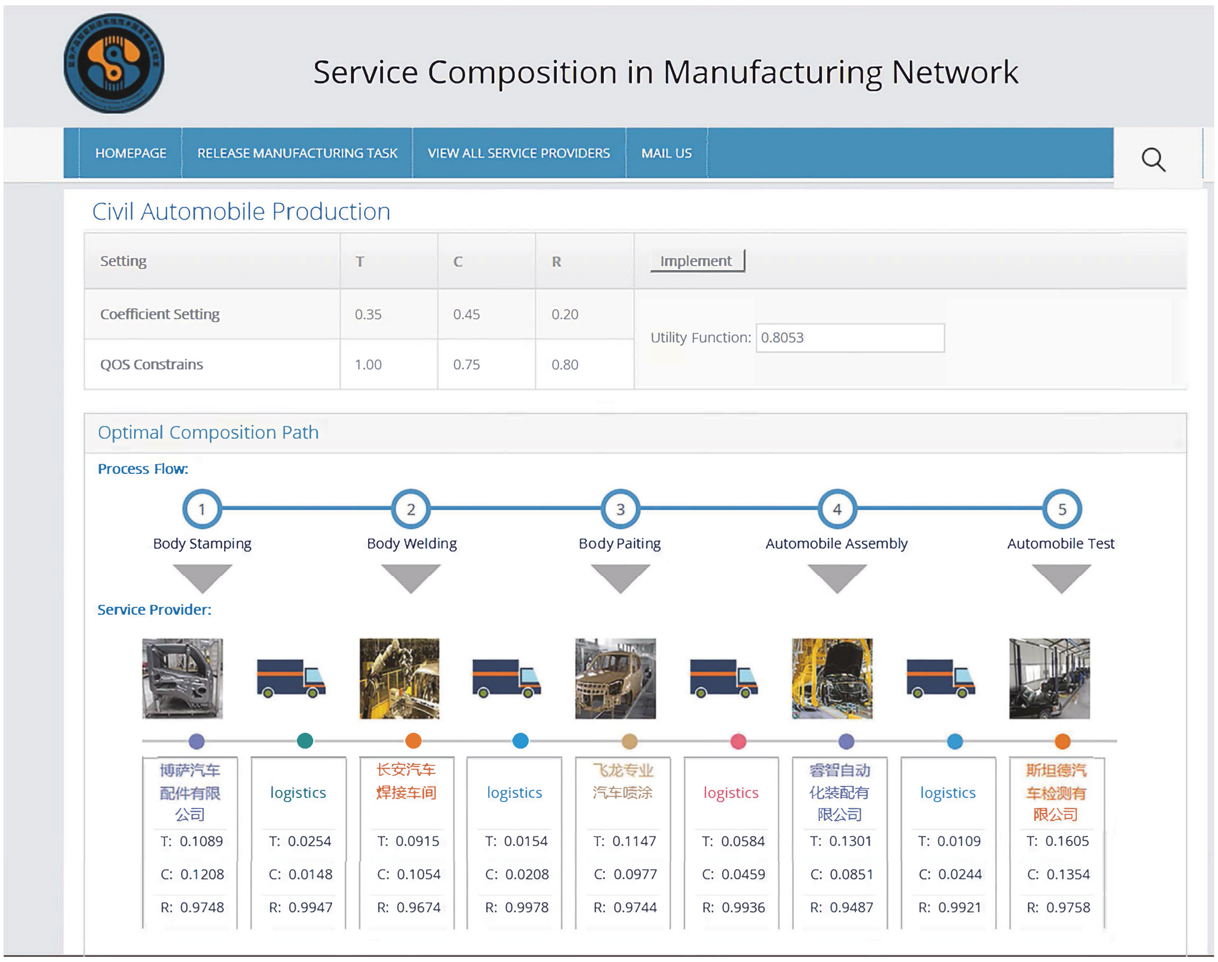
Coefficient			H_OSTP		MFPB_HOSTP		DHA_OSCP	
w_T	w_C	w_R	Utility value	Execution time	Utility value	Execution time	Utility value	Execution time
0.1	0.1	0.8	163.6718	27.3650	161.2995	76.7250	158.5643	55.7790
0.1	0.8	0.1	193.8172	25.1740	192.4318	75.6600	190.5009	52.1430
0.8	0.1	0.1	189.3298	25.6050	187.6882	76.2140	185.6349	54.6890
0.3	0.3	0.4	210.3357	25.0490	209.9200	73.9410	208.8499	52.9790
0.2	0.6	0.2	211.6633	24.9890	210.9084	74.4460	208.6980	53.8810
0.2	0.2	0.6	189.5776	24.8760	188.8923	74.0830	187.6530	51.3050
0.1	0.4	0.5	189.8427	24.5940	189.1458	73.4170	188.0153	52.3260
0.5	0.4	0.1	214.8865	25.1090	214.3177	74.2740	212.8045	52.6320

which uses CBLPs to solve the imbalance problem. With the above discussions, it can be concluded that DHA_OSCP outperforms MFPB_HOSTP in terms of solution quality and time complexity.

6. Application to Automobile Manufacturing Service Composition

The CASICloud (www.casicloud.com) is a CMfg platform for registering and accessing manufacturing services, which has a large number of service providers and service users. In spite of being a popular platform, it can only recommend a single manufacturing service provider for a manufacturing task and the capability of conducting manufacturing service composition for complex tasks is missing. To verify the viability of the proposed model and algorithm, they are implemented and integrated into the platform as an add-on function and are tested using data from service providers obtained from the CASICloud. In this case, an automobile manufacturing process is chosen as an illustrative example.

Figure 7 shows the graphical interface of the additional function of service composition, which also gives detailed information about the service composition task for automobile manufacturing being solved using the proposed model and the DHA_OSCP algorithm. Specifically, the task of automobile manufacturing can be decomposed into five subtasks, namely, body stamping, body welding, body painting, automobile assembly, and automobile test. Each subtask has a set of candidate manufacturing services that can be selected to meet the requirements of the subtask. Each candidate service provider can be abstracted as a node in the manufacturing network. The logistics relationship between service providers is recorded as an edge in the manufacturing network. Using the DHA_OSCP algorithm, a near-optimal solution can be found from the starting node of the manufacturing network to the end one. The choice



Service Composition in Manufacturing Network

Civil Automobile Production

Setting	T	C	R
Coefficient Setting	0.35	0.45	0.20
QOS Constrains	1.00	0.75	0.80

Utility Function: 0.8053

Optimal Composition Path

Process Flow:

1 Body Stamping — 2 Body Welding — 3 Body Paiting — 4 Automobile Assembly — 5 Automobile Test

Service Provider:

博萨汽车配件有限公司	logistics	长安汽车焊接车间	logistics	飞龙专业汽车喷涂	logistics	睿智自动化装配有限公司	logistics	斯坦德汽车检测有限公司
T: 0.1089	T: 0.0254	T: 0.0915	T: 0.0154	T: 0.1147	T: 0.0584	T: 0.1301	T: 0.0109	T: 0.1605
C: 0.1208	C: 0.0148	C: 0.1054	C: 0.0208	C: 0.0977	C: 0.0459	C: 0.0851	C: 0.0244	C: 0.1354
R: 0.9748	R: 0.9947	R: 0.9674	R: 0.9978	R: 0.9744	R: 0.9936	R: 0.9487	R: 0.9921	R: 0.9758

FIGURE 7: Selection of automobile manufacturing service provider.

of the path takes into account both the QoS attributes of the manufacturing service provider and the logistics QoS between the service providers. In this example of application, a user of the system can specify both the QoS constraints (i.e., T, C, and R) and the coefficients of these constraints. After the values are specified, the algorithm can execute the searching process detailed in the previous sections and deliver a near-optimal path. At the same time, the QoS of each selected service provider and the logistics QoS between them are also available to the user. The utility function is calculated using (8) with values generated according to the specific result chosen by the user.

This application illustrates the effectiveness of the proposed algorithm in the field of automobile manufacturing service composition. In this application, both the QoS attributes of the service providers and logistics are taken into consideration to choose the optimal service composition, which is more accurate and credible in actual manufacturing scenario than many other algorithms.

7. Conclusion

In this paper, a new model of manufacturing service composition based on network architecture is proposed. This model not only inherits the advantages of the model based on Cloud Computing but also takes the cooperation relationship between services into account; this makes the service composition process more accurate and closer to actual manufacturing scenarios. To solve the NP-Complete problem of selecting the optimal service composition path with end-to-end QoS constraints in the service network, the advantages and disadvantages of the existing efficient H_OSTP and MFPB_HOSTP algorithms are first analyzed in detail. On this basis, a novel DHA_OSCP algorithm is developed to solve the imbalance problem with less cost in terms of computation time. Computational experiments are conducted using various datasets to evaluate the performance of the proposed DHA_OSCP algorithm, and the results obtained demonstrate that it outperforms existing methods including MFPB_HOSTP and H_OSTP in optimal service path selection. Its application to a real-world application of automobile manufacturing further shows that the proposed algorithm is viable in searching for optimal service composition path with excellent performance in terms of both solution quality and execution efficiency.

Based on the work detailed in this paper, a service composition search engine, which can work with more

```
Data: $\mathrm{SN}_{v_s\longrightarrow v_t}(T,C,R), T^{constraint}_{v_s\longrightarrow v_t}, C^{constraint}_{v_s\longrightarrow v_t}, R^{constraint}_{v_s\longrightarrow v_t}$
Result: $p^{forward}_{v_s\longrightarrow v_t}, U(p^{forward}_{v_s\longrightarrow v_t})$
1  begin
2  $p^{forward}_{v_s\longrightarrow v_t} = \emptyset, p^{forward}_{v_s\longrightarrow v_t} = \emptyset$
3  Delta_Backward_Search ($\mathrm{SN}_{v_s\longrightarrow v_t}(T,C,R), T^{constraint}_{v_s\longrightarrow v_t}, C^{constraint}_{v_s\longrightarrow v_t}, R^{constraint}_{v_s\longrightarrow v_t}$)
4  if $\delta(p^{backward}_{v_s\longrightarrow v_t}) > 1$
5        Return no feasible solution
6  else
7        Utility_Backward_Search ($\mathrm{SN}_{v_s\longrightarrow v_t}(T,C,R), T^{constraint}_{v_s\longrightarrow v_t}, C^{constraint}_{v_s\longrightarrow v_t}, R^{constraint}_{v_s\longrightarrow v_t}$)
8        Forward_Search ($\mathrm{SN}_{v_s\longrightarrow v_t}(T,C,R), T^{constraint}_{v_s\longrightarrow v_t}, C^{constraint}_{v_s\longrightarrow v_t}, R^{constraint}_{v_s\longrightarrow v_t}, \delta(p^{backward}_{v_k\longrightarrow v_t})$,
         $uti(p^{backwarduti}_{v_k\longrightarrow v_t}), deltaback^{\mu}(p^{backward}_{v_k\longrightarrow v_t}), utilityback^{\mu}(p^{backwarduti}_{v_k\longrightarrow v_t}), \mu \in \{\mathrm{T,C,R}\}$)
9        Return $p^{forward}_{v_s\longrightarrow v_t}$ and $U(p^{forward}_{v_s\longrightarrow v_t})$
10  end if (line 4)
11  end
```

Algorithm 1: DHA_OSCP.

```
Data: $\mathrm{SN}_{v_s\longrightarrow v_t}(T,C,R), T^{constraint}_{v_s\longrightarrow v_t}, C^{constraint}_{v_s\longrightarrow v_t}, R^{constraint}_{v_s\longrightarrow v_t}$
Result: $p^{backward}_{v_k\longrightarrow v_t}, \delta(p^{backward}_{v_k\longrightarrow v_t}), deltaback^{\mu}(p^{backward}_{v_k\longrightarrow v_t}), \mu \in \{\mathrm{T,C,R}\}$
1   begin
2    set $\mathrm{VS} = \emptyset, \delta(p^{backward}_{v_k\longrightarrow v_t}) = \infty\ (v_k \neq v_t), \delta(p^{backward}_{v_t\longrightarrow v_t}) = 0,\ p^{backward}_{v_t\longrightarrow v_t} = v_t$
3    move $v_t$ into VS
4    while $\mathrm{VS} \neq \emptyset$ then
5      find $v_a(value) = \min(v^*(value))\ v^* \in \mathrm{VS}$
6      if edge$(v_k, v_a)$ exists
7      for each $v_k$
8      $\delta\left(v_k \longrightarrow v_a + p^{backward}_{v_a\longrightarrow v_t}\right) = \max\left(\frac{deltaback^{T}\left(p^{backward}_{v_a\longrightarrow v_t}\right) + \mathrm{SN}_{v_s\longrightarrow v_t}(T)[a][k]}{T^{constraint}_{v_s\longrightarrow v_t}}, \frac{deltaback^{C}\left(p^{backward}_{v_a\longrightarrow v_t}\right) + \mathrm{SN}_{v_s\longrightarrow v_t}(C)[a][k]}{C^{constraint}_{v_s\longrightarrow v_t}}, \frac{\ln\left(deltaback^{R}\left(p^{backward}_{v_a\longrightarrow v_t}\right) \times \mathrm{SN}_{v_s\longrightarrow v_t}(R)[a][k]\right)}{\ln\left(R^{constraint}_{v_s\longrightarrow v_t}\right)}\right)$
9      if $\delta(v_k \longrightarrow v_a + p^{backward}_{v_a\longrightarrow v_t}) < \delta(p^{backward}_{v_k\longrightarrow v_t})$
10      $\delta(p^{backward}_{v_k\longrightarrow v_t}) = \delta(v_k \longrightarrow v_a + p^{backward}_{v_a\longrightarrow v_t})$
11      $deltaback^{\mu}(p^{backward}_{v_k\longrightarrow v_t}) = deltaback^{\mu}(p^{backward}_{v_a\longrightarrow v_t}) + \mathrm{SN}_{v_s\longrightarrow v_t}(\mu)[a][k], \mu \in \{\mathrm{T,C}\}$
12      $deltaback^{R}(p^{backward}_{v_k\longrightarrow v_t}) = deltaback^{R}(p^{backward}_{v_a\longrightarrow v_t}) \times \mathrm{SN}_{v_s\longrightarrow v_t}(R)[a][k]$
13      $p^{backward}_{v_k\longrightarrow v_t} = v_k \longrightarrow v_a + p^{backward}_{v_a\longrightarrow v_t}$
14      end if (line 9)
15      end (line 7)
16      end if (line 6)
17      Remove $v_a$ from VS
18    end (line 4)
19    return $p^{backward}_{v_k\longrightarrow v_t}, \delta(p^{backward}_{v_k\longrightarrow v_t}), deltaback^{\mu}(p^{backward}_{v_k\longrightarrow v_t}), \mu \in \{\mathrm{T,C,R}\}$
20    end
```

Algorithm 2: Delta_Backward_Search.

manufacturing applications, will be developed in the next step. In this system, the proposed model and algorithm will be extended to help a user select a better manufacturing service composition path to meet the demand of various end to end QoS constraints. Additionally, prediction and service composition for dynamic QoS is also an essential problem worth studying. The algorithms suitable for this scenario will also be studied and developed in our future work.

Appendix

A.

See Algorithm 1.

B.

See Algorithm 2.

```
Data: $SN_{v_s\longrightarrow v_t}(T,C,R), T^{constraint}_{v_s\longrightarrow v_t}, C^{constraint}_{v_s\longrightarrow v_t}, R^{constraint}_{v_s\longrightarrow v_t}$
Result: $p^{backwarduti}_{v_k\longrightarrow v_t}, uti(p^{backwarduti}_{v_k\longrightarrow v_t}), utilityback^{\mu}(p^{backwarduti}_{v_k\longrightarrow v_t}), \mu \in \{T, C, R\}$
1  begin
2   set $VS = \emptyset, uti(p^{backwarduti}_{v_k\longrightarrow v_t}) = \infty\ (v_k \neq v_t), uti(p^{backwarduti}_{v_t\longrightarrow v_t}) = 0,\ p^{backwarduti}_{v_t\longrightarrow v_t} = v_t$
3   move $v_t$ into VS
4   while $VS \neq \emptyset$ then
5     find $v_a(value) = \min(v^*(value))\ v^* \in VS$
6     if edge$(v_k, v_a)$ exists
7     for each $v_k$
8     $uti\left(v_k \longrightarrow v_a + p^{backwarduti}_{v_a\longrightarrow v_t}\right) = \text{weighed\_mean}\left(\frac{utilityback^{T}\left(p^{backwarduti}_{v_a\longrightarrow v_t}\right) + SN_{v_s\longrightarrow v_t}(T)[a][k]}{T^{constraint}_{v_s\longrightarrow v_t}}, \frac{utilityback^{C}\left(p^{backwarduti}_{v_a\longrightarrow v_t}\right) + SN_{v_s\longrightarrow v_t}(C)[a][k]}{C^{constraint}_{v_s\longrightarrow v_t}}, \frac{\ln\left(utilityback^{R}\left(p^{backwarduti}_{v_a\longrightarrow v_t}\right) \times SN_{v_s\longrightarrow v_t}(R)[a][k]\right)}{\ln\left(R^{constraint}_{v_s\longrightarrow v_t}\right)}\right)$
9     if $uti(v_k \longrightarrow v_a + p^{backwarduti}_{v_a\longrightarrow v_t}) < uti(p^{backwarduti}_{v_k\longrightarrow v_t})$
10      $uti\left(p^{backwarduti}_{v_k\longrightarrow v_t}\right) = uti\left(v_k \longrightarrow v_a + p^{backwarduti}_{v_a\longrightarrow v_t}\right)$
11      $utilityback^{\mu}\left(p^{backwarduti}_{v_k\longrightarrow v_t}\right) = utilityback^{\mu}\left(p^{backwarduti}_{v_a\longrightarrow v_t}\right) + SN_{v_s\longrightarrow v_t}(\mu)[a][k], \mu \in \{T, C\}$
12      $utilityback^{R}\left(p^{backwarduti}_{v_k\longrightarrow v_t}\right) = utilityback^{R}\left(p^{backwarduti}_{v_a\longrightarrow v_t}\right) \times SN_{v_s\longrightarrow v_t}(R)[a][k]$
13      $p^{backwarduti}_{v_k\longrightarrow v_t} = v_k \longrightarrow v_a + p^{backwarduti}_{v_a\longrightarrow v_t}$
14      end if (line 9)
15      end (line 7)
16      end if (line 6)
17      Remove $v_a$ from VS
18    end (line 4)
19    return $p^{backwarduti}_{v_k\longrightarrow v_t}, uti(p^{backwarduti}_{v_k\longrightarrow v_t}), utilityback^{\mu}(p^{backwarduti}_{v_k\longrightarrow v_t}), \mu \in \{T, C, R\}$
20    end
```

Algorithm 3: Utility _Backward_Search.

C.

Proof. Assume that p∗ is the feasible solution in the network and $p^{backward}_{v_s\longrightarrow v_t}$ is the path with the minimal δ; then $\delta(p^{backward}_{v_s\longrightarrow v_t}) \leq \delta(p^*)$. Assume that $p^{backward}_{v_s\longrightarrow v_t}$ is not feasible; then $\delta(p^{backward}_{v_s\longrightarrow v_t}) > 1$. Since p^* is a feasible solution with $\delta(p^*) \leq 1$, we can get the conclusion that $\delta(p^{backward}_{v_s\longrightarrow v_t}) > \delta(p^*)$. This contradicts $\delta(p^{backward}_{v_s\longrightarrow v_t}) \leq \delta(p^*)$. So $p^{backward}_{v_s\longrightarrow v_t}$ is a feasible solution.

The feasible backward search can identify whether there is a feasible path that satisfies all QoS constraints. If $\delta(p^{backward}_{v_s\longrightarrow v_t}) > 1$, then there is no feasible solution in the network and the algorithm will terminate. So the feasible backward search process can avoid useless search processes and make the algorithm more efficient. If $\delta(p^{backward}_{v_s\longrightarrow v_t}) \leq 1$, there is a feasible solution $p^{backward}_{v_k\longrightarrow v_t}$ for each node v_k. The feasible backward path set $p^{backward}_{v_k\longrightarrow v_t}$ can participate in the forward search process and can be used as one of the heuristic functions of the A∗ forward search. □

D.

See Algorithm 3.

E.

See Algorithm 4.

Data Availability

The experiment 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 in part by the Zhejiang University/University of Illinois at Urbana-Champaign Institute and was led by Principal Supervisor Prof. Hongwei Wang. This work was supported by National Key R&D Program of China [Grant No. 2018YFB1701602] and State Key Laboratory of

```
Data: $\mathrm{SN}_{v_s \longrightarrow v_t}(T,C,R), T^{constraint}_{v_s \longrightarrow v_t}, C^{constraint}_{v_s \longrightarrow v_t}, R^{constraint}_{v_s \longrightarrow v_t}, p^{backward}_{v_k \longrightarrow v_t}, coe(T,C,R), \delta(p^{backward}_{v_k \longrightarrow v_t}),$
$deltaback^{\mu}(p^{backward}_{v_k \longrightarrow v_t}) p^{backwarduti}_{v_k \longrightarrow v_t}, uti(p^{backwarduti}_{v_k \longrightarrow v_t}), utilityback^{\mu}(p^{backwarduti}_{v_k \longrightarrow v_t}),\ \mu \in \{\mathrm{T,C,R}\}$
Result: $p^{forward}_{v_s \longrightarrow v_t}, U(p^{forward}_{v_s \longrightarrow v_t})$
1   begin
2    Set $\mathrm{VS}_1 = \mathrm{VS}_2 = \emptyset, U1(p^{forward1}_{v_s \longrightarrow v_k}) = \infty\ (v_k \neq v_s), U1(p^{forward1}_{v_s \longrightarrow v_s}) = 0,\ p^{forward1}_{v_s \longrightarrow v_s} = v_s$
      $U2(p^{forward2}_{v_s \longrightarrow v_k}) = \infty\ (v_k \neq v_s), U2(p^{forward2}_{v_s \longrightarrow v_s}) = 0, p^{forward2}_{v_s \longrightarrow v_s} = v_s$
3    move $v_s$ into $\mathrm{VS}_1$ and $\mathrm{VS}_2$
4    while $\mathrm{VS}_1 \neq \emptyset$ and $\mathrm{VS}_2 \neq \emptyset$ then
5      find $v_a(value) = \min(v^*(value))\ v^* \in \mathrm{VS}_1$
6      find $v_b(value) = \min(v^*(value))\ v^* \in \mathrm{VS}_2$
7      if $v_a = v_b$ and $U(v_a) = U(v_b)$
8      if edge$(v_k, v_a)$ exists
9      for each $v_k$
10     start case ($\mu \in \{\mathrm{T,C,R}\}$)
11     Case 1$p^{forward1+backward}_{v_s \longrightarrow v_a \longrightarrow v_t}$ is feasible and $p^{forward2+backwarduti}_{v_s \longrightarrow v_b \longrightarrow v_t}$ is feasible
12         if $U1(p^{forward1+backward}_{v_s \longrightarrow v_a \longrightarrow v_t}) < U1(p^{forward1}_{v_s \longrightarrow v_t})$
13         $p^{forward1}_{v_s \longrightarrow v_t} = p^{forward1}_{v_s \longrightarrow v_a} + p^{backward}_{v_a \longrightarrow v_t}$
14         $Forward^{\mu}\left(p^{forward1}_{v_s \longrightarrow v_t}\right) = combine\left(Forward^{\mu}\left(p^{forward1}_{v_s \longrightarrow v_a}\right), deltaback^{\mu}\left(p^{backward}_{v_a \longrightarrow v_t}\right)\right)$
15         end if (line 12)
16         if $U2(p^{forward2+backwarduti}_{v_s \longrightarrow v_b \longrightarrow v_t}) < U2(p^{forward2}_{v_s \longrightarrow v_t})$
17         $p^{forward2}_{v_s \longrightarrow v_t} = p^{forward2}_{v_s \longrightarrow v_b} + p^{backwarduti}_{v_b \longrightarrow v_t}$
18         $Forward^{\mu}\left(p^{forward2}_{v_s \longrightarrow v_t}\right) = combine\left(Forward^{\mu}\left(p^{forward2}_{v_s \longrightarrow v_b}\right), utilityback^{\mu}\left(p^{backwarduti}_{v_b \longrightarrow v_t}\right)\right)$
19         end if (line 16)
20       Case 2$p^{forward1+backward}_{v_s \longrightarrow v_a \longrightarrow v_t}$ is feasible and $p^{forward2+backwarduti}_{v_s \longrightarrow v_b \longrightarrow v_t}$ is infeasible
21       if $U1(p^{forward1+backward}_{v_s \longrightarrow v_a \longrightarrow v_t}) < U1(p^{forward1}_{v_s \longrightarrow v_t})$
22         $p^{forward1}_{v_s \longrightarrow v_t} = p^{forward1}_{v_s \longrightarrow v_a} + p^{backward}_{v_a \longrightarrow v_t}$
23         $Forward^{\mu}\left(p^{forward1}_{v_s \longrightarrow v_t}\right) = combine\left(Forward^{\mu}\left(p^{forward1}_{v_s \longrightarrow v_a}\right), deltaback^{\mu}\left(p^{backward}_{v_a \longrightarrow v_t}\right)\right)$
24         end if (line 21)
25         if $U2(p^{forward2+backward}_{v_s \longrightarrow v_b \longrightarrow v_t}) < U2(p^{forward2}_{v_s \longrightarrow v_t})$
26         $p^{forward2}_{v_s \longrightarrow v_t} = p^{forward2}_{v_s \longrightarrow v_b} + p^{backward}_{v_b \longrightarrow v_t}$
27         $Forward^{\mu}\left(p^{forward2}_{v_s \longrightarrow v_t}\right) = combine\left(Forward^{\mu}\left(p^{forward2}_{v_s \longrightarrow v_b}\right), deltaback^{\mu}\left(p^{backward}_{v_b \longrightarrow v_t}\right)\right)$
28       end if (line 25)
29       Case 3$p^{forward1+backward}_{v_s \longrightarrow v_a \longrightarrow v_t}$ is infeasible and $p^{forward2+backwarduti}_{v_s \longrightarrow v_b \longrightarrow v_t}$ is feasible
30       if $U1(p^{forward1+backwarduti}_{v_s \longrightarrow v_a \longrightarrow v_t}) < U1(p^{forward1}_{v_s \longrightarrow v_t})$
31       $p^{forward1}_{v_s \longrightarrow v_t} = p^{forward1}_{v_s \longrightarrow v_a} + p^{backwarduti}_{v_a \longrightarrow v_t}$
32       $Forward^{\mu}\left(p^{forward1}_{v_s \longrightarrow v_t}\right) = combine\left(Forward^{\mu}\left(p^{forward1}_{v_s \longrightarrow v_a}\right), utilityback^{\mu}\left(p^{backwarduti}_{v_a \longrightarrow v_t}\right)\right)$
33         end if (line 30)
34       if $U2(p^{forward2+backwarduti}_{v_s \longrightarrow v_b \longrightarrow v_t}) < U2(p^{forward2}_{v_s \longrightarrow v_t})$
35       $p^{forward2}_{v_s \longrightarrow v_t} = p^{forward2}_{v_s \longrightarrow v_b} + p^{backwarduti}_{v_b \longrightarrow v_t}$
36       $Forward^{\mu}\left(p^{forward2}_{v_s \longrightarrow v_t}\right) = combine\left(Forward^{\mu}\left(p^{forward2}_{v_s \longrightarrow v_b}\right), utilityback^{\mu}\left(p^{backwarduti}_{v_b \longrightarrow v_t}\right)\right)$
37       end if (line 34)
38       end case (line 10)
39       end (line 9)
40       end if (line 8)
41       else (line 7)
42       if edge$(v_k, v_a)$ exists
43       for each $v_k$
44       if $p^{forward1+backward}_{v_s \longrightarrow v_a \longrightarrow v_t}$ is feasible and $U1(p^{forward1+backward}_{v_s \longrightarrow v_a \longrightarrow v_t}) < U1(p^{forward1}_{v_s \longrightarrow v_t})$
45       $p^{forward1}_{v_s \longrightarrow v_t} = p^{forward1}_{v_s \longrightarrow v_a} + p^{backward}_{v_a \longrightarrow v_t}$
46       $Forward^{\mu}\left(p^{forward1}_{v_s \longrightarrow v_t}\right) = combine\left(Forward^{\mu}\left(p^{forward1}_{v_s \longrightarrow v_a}\right), deltaback^{\mu}\left(p^{backward}_{v_a \longrightarrow v_t}\right)\right)$
47       end if (line 44)
48       if $p^{forward1+backwarduti}_{v_s \longrightarrow v_a \longrightarrow v_t}$ is feasible and $U1(p^{forward1+backwarduti}_{v_s \longrightarrow v_a \longrightarrow v_t}) < U1(p^{forward1}_{v_s \longrightarrow v_t})$
49       $p^{forward1}_{v_s \longrightarrow v_t} = p^{forward1}_{v_s \longrightarrow v_a} + p^{backwarduti}_{v_a \longrightarrow v_t}$
```

Algorithm 4: Continued.

```
50          $Forward^{\mu}\left(p^{forward1}_{v_s\longrightarrow v_t}\right) = combine\left(Forward^{\mu}\left(p^{forward1}_{v_s\longrightarrow v_a}\right), utilityback^{\mu}\left(p^{backwarduti}_{v_a\longrightarrow v_t}\right)\right)$
51         **end if (line 48)**
52         **end (line 43)**
53         **end if (line 42)**
54         **if** edge$(v_k, v_b)$ **exists**
55         **for** each $v_k$
56         **if** $p^{forward2+backwarduti}_{v_s\longrightarrow v_b\longrightarrow v_t}$ is feasible **and** $U2(p^{forward2+backwarduti}_{v_s\longrightarrow v_b\longrightarrow v_t}) < U2(p^{forward2}_{v_s\longrightarrow v_t})$
57         $p^{forward2}_{v_s\longrightarrow v_t} = p^{forward2}_{v_s\longrightarrow v_b} + p^{backwarduti}_{v_b\longrightarrow v_t}$
58         $Forward^{\mu}\left(p^{forward2}_{v_s\longrightarrow v_t}\right) = combine\left(Forward^{\mu}\left(p^{forward2}_{v_s\longrightarrow v_b}\right), utilityback^{\mu}\left(p^{backwarduti}_{v_b\longrightarrow v_t}\right)\right)$
59         **end if (line 56)**
60         **if** $p^{forward2+backward}_{v_s\longrightarrow v_b\longrightarrow v_t}$ is feasible **and** $U2(p^{forward2+backward}_{v_s\longrightarrow v_b\longrightarrow v_t}) < U2(p^{forward2}_{v_s\longrightarrow v_t})$
61         $p^{forward2}_{v_s\longrightarrow v_t} = p^{forward2}_{v_s\longrightarrow v_b} + p^{backward}_{v_b\longrightarrow v_t}$
62         $Forward^{\mu}\left(p^{forward2}_{v_s\longrightarrow v_t}\right) = combine\left(Forward^{\mu}\left(p^{forward2}_{v_s\longrightarrow v_b}\right), deltaback^{\mu}\left(p^{backward}_{v_b\longrightarrow v_t}\right)\right)$
63         **end if (line 60)**
64         **end (line 55)**
65         **end if (line 54)**
66         **end if (line 7)**
67         **Remove** $v_a$ from $VS_1$ and $v_b$ from $VS_2$
68       **End (line 4)**
69       **return** $p^{forward}_{v_s\longrightarrow v_t}$ = better_utility$(p^{forward1}_{v_s\longrightarrow v_t}, p^{forward2}_{v_s\longrightarrow v_t})$ and $U(p^{forward}_{v_s\longrightarrow v_t})$
70       **end**
```

Algorithm 4: Forward_Search.

Intelligent Manufacturing System Technology [Grant No. QYYE1601].

References

[1] L. Zhang et al., "Cloud manufacturing: a new manufacturing paradigm," *Enterprise Information Systems*, vol. 8, no. 2, pp. 167–187, 2014.

[2] F. Tao, L. Zhang, V. C. Venkatesh, Y. Luo, and Y. Cheng, "Cloud manufacturing: a computing and service-oriented manufacturing model," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 225, no. 10, pp. 1969–1976, 2011.

[3] X. Xu, "From cloud computing to cloud manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 1, pp. 75–86, 2012.

[4] 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.

[5] C. Jatoth, G. Gangadharan, and R. Buyya, "Computational Intelligence Based QoS-Aware Web Service Composition: A Systematic Literature Review," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 475–492, 2017.

[6] G. Peng, H. Wang, J. Dong, and H. Zhang, "Knowledge-based resource allocation for collaborative simulation development in a multi-tenant cloud computing environment," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 306–317, 2018.

[7] Y. Yu, J. Chen, S. Lin, and Y. Wang, "A dynamic QoS-aware logistics service composition algorithm based on social network," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 4, pp. 399–410, 2014.

[8] T. Korkmaz and M. Krunz, "Multi-constrained optimal path selection," in *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 834–843, IEEE, 2001.

[9] H. Zheng, Y. Feng, and J. Tan, "A hybrid energy-aware resource allocation approach in cloud manufacturing environment," *IEEE Access*, vol. 5, pp. 12648–12656, 2017.

[10] Y. Wu, G. Peng, L. Chen, and H. Zhang, "Service architecture and evaluation model of distributed 3D printing based on cloud manufacturing," in *Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 002762–002767, IEEE, 2016.

[11] W. Ho, X. Xu, and P. K. Dey, "Multi-criteria decision making approaches for supplier evaluation and selection: a literature review," *European Journal of Operational Research*, vol. 202, no. 1, pp. 16–24, 2010.

[12] T. Yu and K.-J. Lin, "Service selection algorithms for Web services with end-to-end QoS constraints," *Journal of Information Systems and e-Business Management*, vol. 3, no. 2, pp. 103–126, 2005.

[13] V. Gabrel, M. Manouvrier, and C. Murat, "Optimal and automatic transactional web service composition with dependency graph and 0-1 linear programming," in *Proceedings of the International Conference on Service-Oriented Computing*, pp. 108–122, Springer, 2014.

[14] Y. Yang, S. Tang, Y. Xu, W. Zhang, and L. Fang, "An approach to QoS-aware service selection in dynamic web service composition," in *Proceedings of the 3rd International Conference on Networking and Services*, pp. 18-18, IEEE, 2007.

[15] A. Klein, F. Ishikawa, and S. Honiden, "Efficient heuristic approach with improved time complexity for QoS-aware service composition," in *Proceedings of the 2011 IEEE 9th International Conference on Web Services, ICWS 2011*, pp. 436–443, IEEE, USA, July 2011.

[16] Y. Luo, Y. Qi, D. Hou, L. Shen, Y. Chen, and X. Zhong, "A novel heuristic algorithm for QoS-aware end-to-end service composition," *Computer Communications*, vol. 34, no. 9, pp. 1137–1144, 2011.

[17] P. Rodriguez-Mier, M. Mucientes, and M. Lama, "Automatic web service composition with a heuristic-based search algorithm," in *Proceedings of the 2011 IEEE 9th International Conference on Web Services, ICWS 2011*, pp. 81–88, IEEE, USA, July 2011.

[18] P. Rodriguez-Mier, M. Mucientes, and M. Lama, "Hybrid optimization algorithm for large-scale QoS-aware service composition," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 547–559, 2017.

[19] Y.-Q. Li and T. Wen, "An approach of QoS-guaranteed web service composition based on a win-win strategy," in *Proceedings of the 2012 IEEE 19th International Conference on Web Services, ICWS 2012*, pp. 628–630, USA, June 2012.

[20] M. Li, D. Zhu, T. Deng, H. Sun, H. Guo, and X. Liu, "GOS: a global optimal selection strategies for QoS-aware web services composition," *Service Oriented Computing and Applications*, vol. 7, no. 3, pp. 181–197, 2013.

[21] W. Wang, Q. Sun, X. Zhao, and F. Yang, "An improved particle swarm optimization algorithm for QoS-aware web service selection in service oriented communication," *International Journal of Computational Intelligence Systems*, vol. 3, 1, pp. 18–30, 2010.

[22] H. Rezaie, N. NematBaksh, and F. Mardukhi, "A multi-objective particle swarm optimization for web service composition," in *Proceedings of the International Conference on Networked Digital Technologies*, pp. 112–122, Springer, 2010.

[23] F. Rosenberg, M. B. Müller, P. Leitner, A. Michlmayr, A. Bouguettaya, and S. Dustdar, "Metaheuristic optimization of large-scale QoS-aware service compositions," in *Proceedings of the 2010 IEEE 7th International Conference on Services Computing, SCC 2010*, pp. 97–104, IEEE, USA, July 2010.

[24] S. B. Bhushan and P. C. Reddy, "A hybrid meta-heuristic approach for QoS-aware cloud service composition," *International Journal of Web Services Research*, vol. 15, no. 2, pp. 1–20, 2018.

[25] Z. Ye, X. Zhou, and A. Bouguettaya, "Genetic algorithm based QoS-aware service compositions in cloud computing," in *Proceedings of the in International Conference on Database Systems for Advanced Applications*, pp. 321–334, Springer, 2011.

[26] Y. Que, W. Zhong, H. Chen, X. Chen, and X. Ji, "Improved adaptive immune genetic algorithm for optimal QoS-aware service composition selection in cloud manufacturing," *The International Journal of Advanced Manufacturing Technology*, pp. 1–11, 2018.

[27] 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.

[28] C. B. Pop, V. R. Chifu, I. Salomie, M. Dinsoreanu, T. David, and V. Acretoaie, "Ant-inspired technique for automatic Web service composition and selection," in *Proceedings of the 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2010*, pp. 449–455, IEEE, Romania, September 2010.

[29] J. Zhou and X. Yao, "A hybrid artificial bee colony algorithm for optimal selection of QoS-based cloud manufacturing service composition," *The International Journal of Advanced Manufacturing Technology*, vol. 88, no. 9-12, pp. 3371–3387, 2017.

[30] G. Kousalya, D. Palanikkumar, and P. R. Piriyankaa, "Optimal web service selection and composition using multi-objective bees algorithm," in *Proceedings of the 9th IEEE International Symposium on Parallel and Distributed Processing with Applications Workshops, ISPAW 2011*, pp. 193–196, IEEE, Republic of Korea, May 2011.

[31] A. Bekkouche, S. M. Benslimane, M. Huchard, C. Tibermacine, F. Hadjila, and M. Merzoug, "QoS-aware optimal and automated semantic web service composition with user's constraints," *Service Oriented Computing and Applications*, vol. 11, no. 2, pp. 183–201, 2017.

[32] C. Jatoth, G. Gangadharan, U. Fiore, and R. Buyya, "QoS-aware Big service composition using MapReduce based evolutionary algorithm with guided mutation," *Future Generation Computer Systems*, vol. 86, pp. 1008–1018, 2018.

[33] H. Labbaci, B. Medjahed, and Y. Aklouf, "A deep learning approach for long term QoS-compliant service composition," in *Proceedings of the International Conference on Service-Oriented Computing*, pp. 287–294, Springer, 2017.

[34] H. Jin, X. Yao, and Y. Chen, "Correlation-aware QoS modeling and manufacturing cloud service composition," *Journal of Intelligent Manufacturing*, 2015.

[35] G. Liu, Y. Wang, M. A. Orgun, and E.-P. Lim, "A heuristic algorithm for trust-oriented service provider selection in complex social networks," in *Proceedings of the 2010 IEEE 7th International Conference on Services Computing, SCC 2010*, pp. 130–137, IEEE, USA, July 2010.

[36] G. Liu, Y. Wang, M. A. Orgun, and E.-P. Lim, "Finding the optimal social trust path for the selection of trustworthy service providers in complex social networks," *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 152–167, 2013.

[37] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[38] D. Wang, Y. Yang, and Z. Mi, "A genetic-based approach to web service composition in geo-distributed cloud environment," *Computers and Electrical Engineering*, vol. 43, pp. 129–141, 2015.

[39] S. Dustdar and W. Schreiner, "A survey on Web services composition," *International Journal of Web and Grid Services*, vol. 1, no. 1, pp. 1–30, 2005.

[40] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for Web services selection with end-to-end QoS constraints," *ACM Transactions on the Web (TWEB)*, vol. 1, no. 1, p. 6, 2007.