Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

A service composition method using improved hybrid teaching learning optimization algorithm in cloud manufacturing

Jun Zeng*, Juan Yao, Min Gao and Junhao Wen

Abstract

In the cloud manufacturing process, service composition can combine a single service into a complex service to meet the task requirements. An efficient service composition strategy is crucial, as it affects the efficiency of resource and capacity sharing in the cloud manufacturing system. However, in the face of a large-scale environment, the existing methods have the problems of slow convergence and instability. To solve the problems above, we propose an improved optimization method, named improved-TC. Specifically, we are inspired by the horizontal crossover of CSO in the hybrid-TC teaching phase, the Hybrid-TC is proposed in our previous work, which is a hybrid of the teaching-learning-based optimization algorithm (TLBO) and the crisscross optimization algorithm (CSO). Improved-TC is an improvement on the learning phase of hybrid-TC algorithm, we change the search method of hybrid-TC in the learning phase to a one-dimensional search, thereby some dimensions in the population that are trapped in the local optimum have the chance to jump out of the iteration. Experiments show that our proposed method has a faster convergence speed and more stability in the face of service composition in large-scale environments.

Keywords: Cloud manufacturing, Service composition, Quality of service, Teaching-learning-based optimization, The crisscross optimization algorithm

Introduction

In order to cope with the increasing demand for products from users, manufacturers need to collaborate more effectively by sharing their manufacturing resources and technical capabilities to meet user needs [1–3]. Cloud manufacturing (CMfg) is a new service-oriented manufacturing paradigm. In the CMfg system, manufacturing resources and manufacturing capabilities are encapsulated as sharable services and distributed to users in an on-demand manner through the internet to make full use of resources and avoid waste of resources [4]. Service composition and optimization selection (SCOS) [5, 6] is a key technology to realize resource and capacity sharing in CMfg system. SCOS process is to search a service from the candidate service pool for each subtask in CMfg system, then generate a composite service and optimize it to meet the needs of users [7, 8]. That is, according to different composition structure, it can combine various single services with different functions into composite services with comprehensive functions, so that the newly acquired services can meet the requirements of the task. In the process of CMfg service composition, according to task requirements and user needs, the selected services will be constrained by various Quality of service (QoS) attributes, and the service composition solution needs to be optimized through efficient methods to meet the requirements. The optimization objective of the SCOS problem is to find an optimal composition service that meets the task requirements and user needs in a feasible time. How to find the optimal or nearly optimal service composition quickly and effectively, in the mass of composite services, is a problem worth our attention.

*Correspondence: zengjun@cqu.edu.cn

School of Big Data & Software Engineering, Chongqing University, Chongqing, China

So far, there are many research methods on the SCOS problem. Li et al. [9] used a service composition method based on service clustering network to solve the problem of static and dynamic demand in cloud manufacturing. Lu et al. [10] proposed a systematic framework for process capability evaluation and service recommendation in a cloud manufacturing environment, in which the integrated service composition module successfully connected and shared the engineering knowledge of engineers or management teams from different companies. The above work is based on the method of service composition framework. Furthermore, in the SCOS process, the QoS is often used as a criterion when evaluating service composition efficiency. Some researchers have considered QoS to optimize services: Lartigau et al. [11] proposed a cloud manufacturing service composition method to solve the problem of transportation route selection between merchants by considering QoS evaluation. A QoS-aware manufacturers to users (M2U) mode is presented in [12], which improves its ability to dynamically optimally allocate resources. Although the existing SCOS methods have some effect in solving the service composition problem, they still face huge challenges due to the complex personalization requirements of users and the rapidly increasing number of services in cloud manufacturing platforms. And there are still some key issues that need to be addressed: (1) Due to the increasing number of services in cloud manufacturing system, existing methods are inefficient in solving large-scale SCOS problems, and more efficient algorithms need to be found; (2) With the personalized needs of users increase, the service composition conditions in the CMfg system become more complex, and it is difficult to find an accurate solution using the existing service composition methods, and it is necessary to improve the global search strategy of the algorithm. A large-scale service composition problem can be described as follow [13]: If M is the number of service sets in a composition path and $N^{'}$ is the number of candidate services of service sets, the number of all composition paths will be $N^{'M}$. As the numbers of candidate services and service sets increase, the solution space of the optimization problem grows exponentially [14–16]. Some intelligent optimization algorithms [5, 6, 17, 18] due to their characteristics, such as universality, applicability, randomness, and fast search speed, are often used to solve large-scale SCOS problem. However, as the personalized needs of user increase, more and more candidate services are included in composite services. This will cause the existing intelligent algorithms to be no longer suitable for solving the current SCOS problem, and there will be disadvantages, such as poor stability, slow convergence speed, and low solution quality.

To solve the problems above, we propose a hybrid optimization algorithm, named improved-TC, which is an improved algorithm based on the hybrid-TC [19] proposed by us previously. Hybrid-TC is an algorithm that combines the teaching–learning-based optimization algorithm (TLBO) [20] and the crisscross optimization algorithm (CSO) [21] in the teaching stage. It has several prominent advantages such as accurate calculation and free from the algorithm parameters, which can improve the solution of service composition. However, it is unstable and has slow convergence. To enhance the stability and avoid the slow convergence of hybrid-TC, we are inspired by the horizontal crossover of CSO in the hybrid-TC teaching phase, and in the learning phase of improved-TC a similar method was used to improve the algorithm, i.e., changing the multi-dimensional search method the algorithm to a one-dimensional search, so that some dimensions in the population that fall into the local optimum have the opportunity to jump out of the iteration. The novelty of our proposed method is that it considers the optimization of the composition path from a small granularity perspective. In this way, the solutions can be found more accurately, which makes up for the shortcomings of hybrid-TC's insufficient global search ability, and enhances the stability and convergence of the algorithm. At the same time, the solutions of service composition have been improved.

In summary, our contributions in this paper are as follows:

- To increase the global search ability of the algorithm, we combine the teaching–learning-based optimization algorithm (TLBO) [20] and the crisscross optimization algorithm (CSO) in the teaching phase, which considers the optimization of composition path from the perspective of small granularity and makes up for the insufficient search ability of the original TLBO.
- To improve the convergence speed and enhance the stability of the service composition optimization algorithm, the learning phase of improved-TC is improved. The improved algorithm can make some dimensions in the population that fall into the local optimum can be jumped out of the iteration, so that the convergence speed and the stability of the algorithm is enhanced.
- To get the best performance of our algorithm, we determine the population proportion of CSO through experiments.

The rest of our paper is organized as follows. Section 2 introduces the related work of QoS-aware service composition. Section 3 presents the problem statements and evaluation models. Section 4 presents our algorithm model and the detailed description of the model. In Section 5, the comparative experiments are performed to verify the proposed approach. Finally, Section 6 makes the conclusions and future works of the paper. In addition, the Table 1 listing the symbols used in the paper to accommodate a larger audience.

Related work

The meta-heuristic optimization algorithms in optimization problems

The essence of the optimization problem is to select a set of variables or parameters, under the condition of satisfying a series of related constraints, to make the design goal reach the optimal value [22–25]. For this reason, a lot of researchers have designed meta-heuristic methods to solve optimization problems, for example, Rao et al. [20] proposed a Teaching–Learning-Based Optimization (TLBO) to solve large-scale non-linear optimization problems. The algorithm is divided into two phases: teaching and learning, then the global solution is found by continuous optimization in these two stages. Meng et al. [21] proposed the crisscross optimization algorithm (CSO) inspired by Confucian doctrine of gold mean and the crossover operation in genetic algorithm to solve complex optimization problems. In the algorithm, horizontal crossover and vertical crossover are used as search operators. Combining the two search operators, the algorithm has a magical effect on improving the convergence speed and solution accuracy.

Table 1 Notations of this paper

Notations	Definition
$T=\{t_1,t_2,\ldots,t_m,\ldots,t_M\}$	A set of M subtasks
$S=\{S_1,S_2,\ldots,S_m,\ldots,S_M\}$	A set of M candidate service sets
$S_m=\left\{s_{m,1},s_{m,1},\ldots,s_{m,n'},\ldots,s_{m,N'}\right\}$	A set of N' candidate services
$S'=\left\{s_{1,2},s_{2,2},\ldots,s_{m,n'},\ldots,s_{M,N'}\right\}$	A composite service
$P=\{P_1,P_2,\ldots,P_n,\ldots,P_N\}$	A set of N individual
$P_n=\left\{s_{1,2},s_{2,2},\ldots,s_{m,n'},\ldots,s_{M,N'}\right\}$	Mapping of services and individual
$\prod_{m=1}^{M}N'$	The composite service path
T	Task
s_m	Candidate service
$s_{m,n'}$	The n'-th service of service set S_m
t_m	Subtask
$q_{Te}^{S_m},q_{Ct}^{S_m},q_{Rl}^{S_m},q_{Rp}^{S_m}$	Time, cost, reliability and reputation of candidate service
$q_1,\ q_2,q_3,q_4$	Time, cost, reliability and reputation of aggregation values
$\mathrm{Q}_\mathrm{k}^{+},\mathrm{Q}_\mathrm{k}^{-}$	Normalized positive and negative attribute indicators
q_k	The QoS attribute
$\min q_\mathrm{k},\max q_\mathrm{k}$	The minimum and maximum aggregated values of the QoS attribute
ω_k	The weight of the attribute
w_1,w_2,w_3,w_4	The weight of time, cost, reliability and reputation
R_{max}	Maximum iterations
P	The population
P_n	The individual
$P_{T',new}^{i},P_{T',old}^{i}$	The new and old position
P_{mean}	The mean position
$P_{teacher}$	The teacher (the best individual at current iteration)
T_F	The teaching factor
r,r_1,r_j,r_2,c_1,c_2	The random numbers
$P_{c,old}^{l_1},P_{c,old}^{l_2}$	The old position of the l_1-th and l_2-th individual in teaching-phase
$P_{c,old}^{l_1,d},P_{c,old}^{l_2,d}$	The old dimension of teaching-phase
$P_{c,new}^{l_1,d},P_{c,new}^{l_2,d}$	The new dimension of teaching-phase
$P_{L,new}^{j},P_{L,old}^{j},P_{L,old}^{k}$	The new and old position of learning-phase
$P_{L,new}^{j,h},P_{L,old}^{j,h},P_{L,old}^{k,h}$	The new and old dimension of learning-phase

Service composition and optimization selection in cloud manufacturing

As one of the key technologies of cloud manufacturing service platform, SCOS is very important to promote the development of manufacturing industry. A growing number of researchers pay attention to the SCOS problem and study it from multiple perspectives.

QoS indicators are often used to evaluate the effectiveness of service composition. The research methods based on QoS indicators [26], such as cost, time reliability, called QoS-aware methods. Many researchers have carried out research on QoS-aware methods to solve the SCOS problem. The work in this area includes: Que. et al. [12] proposed a QoS-aware manufacturer to users (M2U) mode, in the mode, the QoS indexes are considered, such as time, cost, reliability, and ability, to evaluate model efficiency. Zhou et al. [27] proposed a mathematical model including four QoS metrics of time, cost, availability and reputation, then they used a hybrid multi-population co-evolution approach to solve the SCOS problems. In order to maximize production efficiency while weighing different production orders, Laili et al. [28] studied the multi-stage integrated scheduling of mixed tasks in cloud manufacturing environment based on QoS indicators. The experiments proved that this method not only reduces the production cost also save time. Liang et al. [29] established a deep reinforcement learning-based cloud manufacturing service composition model. This model models the service composition process as a Markov decision process and designs the corresponding reward function considering logistics. A series of experiments verify the effectiveness, efficiency, robustness, adaptability and scalability of the method in solving the SCOS problem.

The research methods based on Qos-aware are constrained by QoS indicators when composition services, and the composition efficiency is low. To improve the composition efficiency, some researchers try to solve SCOS problem using frame-based methods [9, 10]. for example, Zhang et al. [9] proposed a network clustering method (SC-SCN) to combine services, which firstly clustered services with the same function into a set of abstract services by similarity calculation, secondly, the abstract service network is obtained by similarity calculation, finally, search for the services that meet the task requirements to combine services by the search algorithm.

The cloud manufacturing SCOS problem is an NP-hard problem, and it is difficult to find an exact solution in a feasible time. Intelligent optimization algorithms are widely used to solve SCOS problem because of its flexibility and scalability, so that combined results with exact solutions or close to exact solutions can be found in feasible time [8]. An improved grey wolf optimizer algorithm (IGWO) for saving energy consumption is proposed by Yang et al. [6], the method improved the control factor and algorithm position update in the GWO, which leads to the improvement of the algorithm's search ability and ensures the accuracy of the solution. On this basis, they also proposed an enhanced gray wolf optimization algorithm (EMOGWO) [5] for multi-objective SCOS problems, which made three improvements to the basic multi-objective GWO and overcame the shortcoming of local optimum and less diversity in multi-objective CMfg SCOS problems. Zhou et al. [30] considered the high flexibility and complexity of CMfg, a hybrid artificial bee colony method (HABC) is designed to solve the large-scale solution space problem of service composition. It uses archimedean copula estimation of distribution algorithm (ACEDA) probability model and the chaos operators of global best-guided artificial bee colony to generate the offspring individuals, which has the advantages of high performance and high stability. An ensemble optimization approach (EOA) is proposed by Fazeli et al. [17], which used genetic algorithm (GA), particle swarm optimization (POS) and social spider algorithm (SSO) as a black box, then used a new operator to summarize the results of the black box, this algorithm can be flexibly expanded and combined with other algorithms to effectively balance the various indicators in the service composition. Zhou et al. [18] proposed a hybrid teaching-learning-based optimization, which combined two algorithms in the learning phase and the teaching phase respectively, and used the onlooker search of artificial bee to change the search method in the studying phase, so that the algorithm could solve the large-scale cloud manufacturing service composition problems.

So far, the existing work on using the meta-heuristic algorithm to solve large-scale SCOS problems has not achieved satisfactory results. We are inspired by the work [18] and try to improve the quality of the solution in large-scale CMfg environment. Therefore, we develop a hybrid teaching–learning-based optimization algorithm to solve this problem.

Service composition problem description and quality of service evaluation

Cloud manufacturing service composition problem description

In the cloud manufacturing environment, service composition is one of a key technology, and it is necessary to analyze the composition process systematically. As shown in Fig. 1, a complex manufacturing task T is

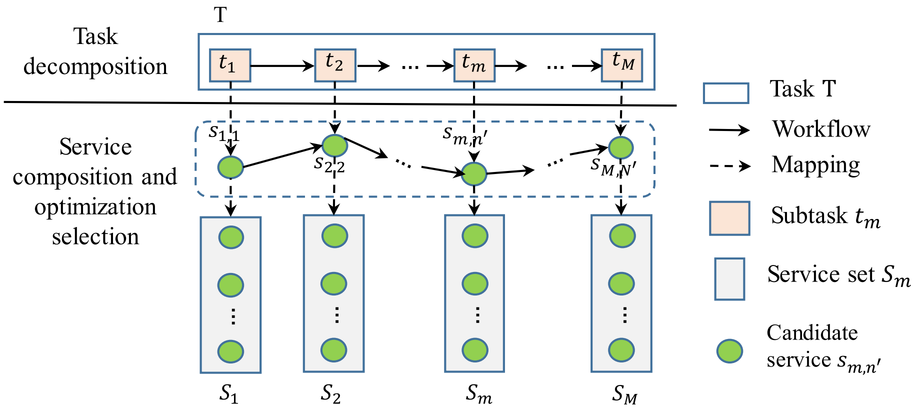

Fig. 1 Service composition flowchart

usually decomposed into several subtasks t, which can be represented by $T=\{t_1,t_2,\dots,t_m,\dots,t_M\}$, where M is the total number of subtasks within the task T and t_m is the m- th subtask of T. Each subtask t_m is executed in some logical order and corresponds to a service set. The SCOS problem is the system searches for services from the candidate service pool for each subtask, then selects a service from the corresponding service set to generate composite services and optimize services according to user requirements. Selecting and deter-mining the optimal service for each subtask from the service set is a key step in the SCOS problems. The process of SCOS can be divided into the following steps [6]: (1) task decomposition, $T=\{t_1,t_2,\dots,t_m,\dots,t_M\}$. (2) Service discovery and matching, the system searches for services from the service pool according to the task requirements, and clusters services with the same function to form a candidate service set, candidate service sets can be represented by $S=\{S_1,S_2,\dots,S_m,\dots,S_M\}$, $S_m=\left\{s_{m,1},s_{m,1},\dots,s_{m,n'},\dots,s_{m,N'}\right\}$, where M is the total number of candidate service sets and S_m is the m- th candidate service set, where $s_{m,n'}$ is the n'-th service of service set S_m. (3) service selection and composition, the system selects a service s_m from the candidate service set S_m that corresponds to the subtask to execute the task, and then generates the composite service, the composite service can be expressed by $S'=\left\{s_{1,2},s_{2,2},\dots,s_{m,n'},\dots,s_{M,N'}\right\}$, where $s_{M,N'}$ is the N'-th service of service set S_M. (4) service optimization, in the service composition process, it is assumed that the subtask t_m has N' candidate services s_m. In theory, it can be combined into $\prod_{m=1}^{M}N'$ composition service, therefore, an optimal composition service from the composition service space needs to be selected as the execution path of the task T. The detailed description of the SCOS process is comprehensively explained in the existing study [2].

Cloud manufacturing service composition quality of service evaluation

QoS aggregation functions The typical cloud manufacturing SCOS problem is that the optimal services are selected by each subtask from the corresponding service sets according to numerous QoS constraints, then the task T is performed together. There are four main types of service composition: sequential, parallel, selective and cyclic, three of which (parallel, selective and cyclic) can be transformed into the sequential structure by simplification [31]. To obtain the optimal service composition path, the QoS indicators of cloud services need to be comprehensively evaluated, that is, the QoS indicators of the individual cloud service in the combined path are aggregated, and then the optimal aggregated QoS value in whole composition paths is selected according to the user needs. There are more than 20 QoS indicators for evaluating service capability [32, 33], where parts of indicators have been applied to manufacturing cloud services [27, 30, 34]. In this paper, we take time, cost, reliability and reputation as QoS criteria of services. The aggregation function [35] of sequential structure is shown in Table 2.

Where s_m is the m-th service of composition services; N', q are the number of services and QoS attributes respectively.

Table 2 QoS aggregation functions

QoS Indicators	Aggregation Functions
Time	$q_1=\sum_{n'=1}^{N'}q_{Te}^{S_m}$
Cost	$q_2=\sum_{n'=1}^{N'}q_{Ct}^{S_m}$
Reliability	$q_3=\prod_{n'=1}^{N'}q_{Rl}^{S_m}$
Reputation	$q_4=\frac{1}{N'}\sum_{n'=1}^{N'}q_{Rp}^{S_m}$

Attribute indexes normalization processing Because of the measurement standards and units of each attribute are different, attributes can be divided into positive attribute indicators(Q_k^+) and negative attribute indicators(Q_k^-). Therefore, before calculating the overall QoS value, it is necessary to normalize the QoS value of each indicator. The normalization calculation formula is as follows

$$Q_k^+ = \begin{cases} \frac{q_k - \min q_k}{\max q_k - \min q_k}, \min q_k \neq \max q_k \\ 1, \min q_k = \max q_k \end{cases} \quad (1)$$

$$Q_k^- = \begin{cases} \frac{\max q_k - q_k}{\max q_k - \min q_k}, \min q_k \neq \max q_k \\ 1, \min q_k = \max q_k \end{cases} \quad (2)$$

In Eqs. (1) and (2) $minq_k$ and $maxq_k$ indicate the minimum and maximum aggregated values of the k-th QoS attribute for all possible composition paths. The greater the QoS value of positive attribute indicators, the better the quality of the service, like reliability and reputation. On the contrary, the smaller the QoS value of negative attribute indicators, the better the quality of the service, such as time and cost.

The most common way to evaluate the overall QoS value is to convert the normalized QoS values of attributes into a single QoS value by a simple weighting method. Multiplying the normalized QoS values by different weights to calculate the QoS utility value. The QoS utility value (fitness value) is calculated by the following formula:

$$\text{Max (QoS)} = \text{Max} \sum \omega_k \times Q_k \quad (3)$$

$$\sum_{k=1}^{K} \omega_k = 1 \quad (4)$$

Where Q_k represents the normalized value of the k-th QoS attribute. ω_k denotes the weight of the $k-th$ attribute, K is the number of indicators, $\omega_k \in [0, 1]$.

Proposed algorithm framework

In this paper, we propose a new cloud manufacturing service composition algorithm (improved-TC) based on the hybrid-TC [19], which combines TLBO [20, 36] and CSO [21] in the teaching phase of the original TLBO and the learning phase of the original TLBO is improved. The whole algorithm framework includes four stages: initialize-phase, teaching-phase, learning-phase and output-phase. Specifically, firstly, to obtain better convergence, the Skyline query is added in the initialize phase to filter high-quality candidate services from the service sets, and generate composition services. Secondly, the composition service generated from the initialize phase is input to the teaching phase, which combines the TLBO algorithm with CSO algorithm. And the teaching phase searches for high-quality services from the candidate service set again to replace the composition service in the initialize phase. Thirdly, the solutions obtained by the teaching-phase will continue to be studied in the learning phase to obtain better solutions that replace the old solutions. To obtain a faster convergence speed and a better solution for the algorithm, we have improved in this phase. Finally, the best result is updated by comparison continuously in the output-phase, and then output. The overall framework of our method is shown in Fig. 2 and the specific operations are indicated in Fig. 4. Next, we will introduce more details of the framework.

Initialize-phase

In the initialization phase, some data required by the algorithm is set, such as the number of subtasks t, maximum iterations R_{max} etc. Besides, the most important is that the initial population P generated, because it is related to the convergence of the algorithm. Generally, P is generated randomly by selecting candidate services s from service sets S, $P=\{P_1, P_2, ..., P_n, ..., P_N\}$, $S_m = \{s_{m,1}, s_{m,1}, \ldots, s_{m,n'}, \ldots, s_{m,N'}\}$, $P_n = \{s_{1,2}, s_{2,2}, \ldots, s_{m,n'}, \ldots, s_{M,N'}\}$, where P_n is the n-th individual, N is the is the total number of population, S_m represents the m-th service set, $s_{m,n'}$ is the n'-th service of service set S_m and M is equal to the number of subtasks in task T. The algorithm has poor convergence and low solution quality due to the fact that the randomly generated initial population is irregular.

To solve the problems above, the Skyline query [37, 38] is added to the initial processing. It theory is that the balance of constraints can be found according to the constraints between QoS, and obtain the best QoS value from the whole service path, the pseudo-code shown in Fig. 3. Skyline query can filter out some high quality services from the service set, the search range is reduced when the algorithm searches the target. However, if the number of candidate services for subtasks is too large, the Skyline query can consume a long time.

Based on the problem above, we choose 1/5 of P (n of P_n is $4/5N$) to be generated by Skyline query in the experiment. Moreover, the population generated by the Skyline query is generated offline, which saved more time during the service composition phase. After generating the population, we calculate and sort the fitness of all the populations, and save the optimal fitness value for comparison in the fourth phase.

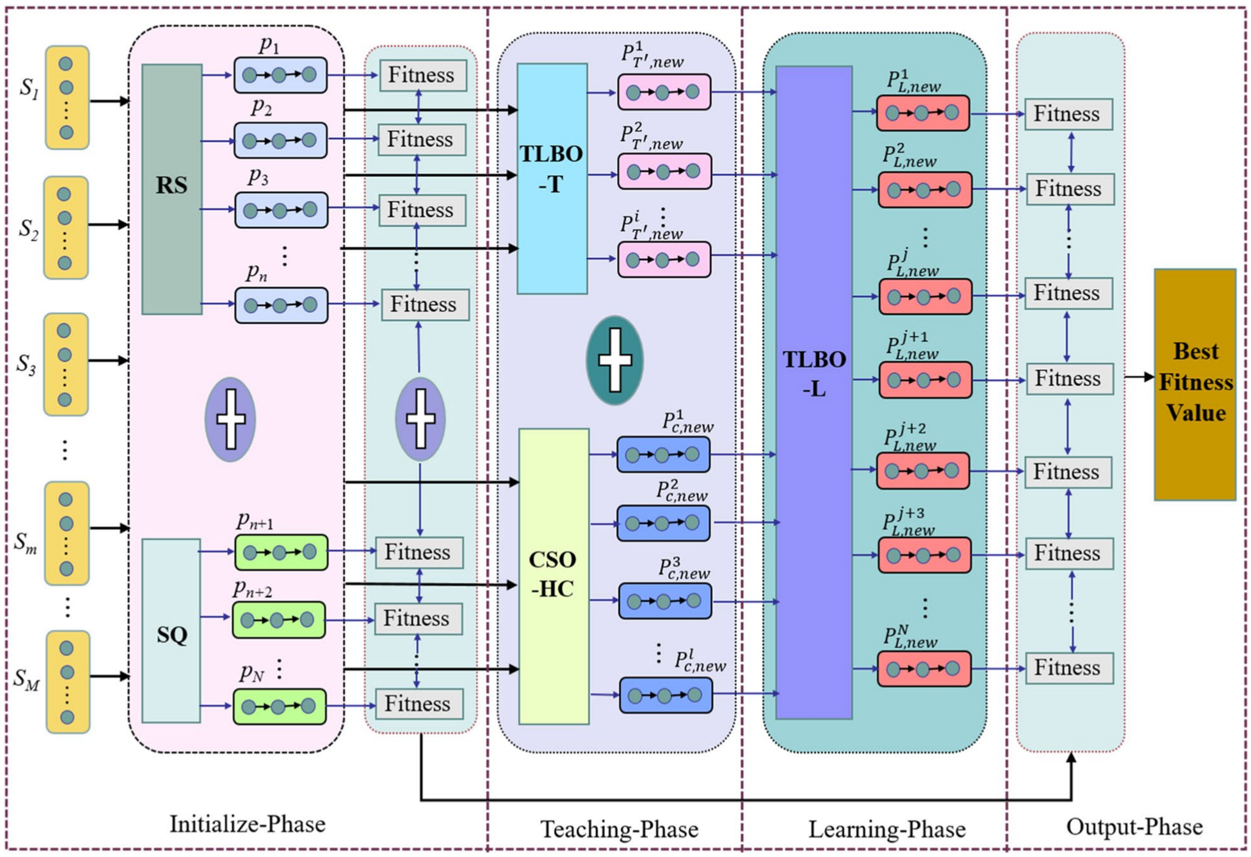

Fig. 2 Framework of the proposed improved-TC, RS is Random Selection, SQ is Skyline Query, TLBO-T is TLBO-Teaching, CSO-HC is CSO-Horizontal Crossover and TLBO-L is TLBO-Learning

```
Algorithm 1. Skyline query
1: Input the service sets S;
2: Find non-dominated solution according to dominate regulation:
3:     i iterate from 1 to size(S);
4:         flag←0;
5:         j iterate from 1 to size(S);
6:         if servie $s_i$ dominate $s_j$
7:             flag←1;
8:         end
9:         if flag= =0
10:            Save the non-dominated solution into Skyline set;
11:        end
12: Output Skyline set.
```

Fig. 3 Skyline query Algorithm

Teaching-phase

In the Teaching-Phase of original TLBO, only the individual with the best fitness value work by Eq. (5), in this way, the new individuals learn nothing or learn all the things taught by the best individual. Actually, the new individual obtained by Teaching-Phase may be random

```
Algorithm 2. Improved-TC operation steps
Step1: Initialization
1.1: Initialize the population $P$ according to the Skyline query and random, and initialize the parameters $t$, maximum
     iterations $R_{max}$;
1.2: Calculate the QoS values of composition services in population $P$ according to Eq. (3);
1.3: Rank the QoS values and save the best QoS value Q ($P_{best}$);
Step2: Teaching-Phase
2.1: Find the best individual;
2.2: Divide the $P$ into two parts according to the proportion of the population (TLBO: CSO);
2.3: Generate the new individuals $P^{i}_{T',new}$ by Eq. (5) and Eq. (6);
2.4: Generate the other new individual dimensions $P^{l_1,d}_{c,new}$, $P^{l_2,d}_{c,new}$ using Eq. (7) and Eq. (8);
2.5: Update QoS values of composition services in $P$ according to Eq. (3) and generate offspring;
Step3: Learning-Phase
3.1: Find a new individual dimensions $P^{k,h}_{L,old}$ randomly in the offspring;
3.2: Generate the new individual dimensions $P^{j,h}_{L,new}$ using Eq. (11) and Eq. (12);
3.2: Compare fitness values of the population and generate new offspring;
Step4: Output-Phase
4.1: Update QoS values of composition services in $P$ according to Eq. (3) and rank the values;
4.2: Output the best QoS value if the iterations of improved-TC is greater than $R_{max}$;
4.3: Otherwise, go to step 2.
```

Fig. 4 Improved-TC operation steps

proportion. Therefore, the mechanism of the Teaching-Phase may lead to the phenomenon that the poor convergence speed and premature may occur for the results to a certain extent. We found that the stability of the TLBO is very poor in the experiment, in short, the results of TLBO obtained by run experiments with the same experimental settings are quite different. Therefore, the horizontal crossover of CSO is added in this stage, which can solve this problem.

Iterative procedure for the CSO algorithm, each iteration has two crossing methods: Horizontal crossover and vertical crossover. Some dimensions in the population that fall into the local optimum will have the opportunity to jump out of the iteration by these two crossover methods. In this paper, we only use the horizontal crossover of CSO, the formulas horizontal crossover of CSO are defined in Eqs. (7) and (8). Base on Eqs. (7) and (8), the horizontal crossover searches for the new individuals (i.e. $P^{l_1}_{c,new}$) in the population P. The new individuals (i.e. $P^{l_1}_{c,new}$) are obtained by using two pairs of old individuals (i.e. $P^{l_1}_{c,old}$ and $P^{l_2}_{c,old}$,) and then execute the competition operator. The competition operator is that if the fitness of the new individual is better than the fitness of the old individual, using the new individual replaces the old individual, else selecting the old individual as the offspring individual. It can be seen that the updating of individuals is one-dimensional by Eqs. (7) and (8), which means that the blind spots that cannot be searched by the old individuals can be reduced, enabling the CSO algorithm to have strong global search ability.

The important factor to be considered is that the strong global searching ability of horizontal crossover improves the solution quality but increases the time consumption of algorithm. A good population ratio of TLBO and CSO is very important to the algorithm, so we will determine the population ratio of TLBO and CSO according to the experiment, in form:

Teaching-phase:

$$P^{i}_{T',new} = P^{i}_{T',old} + r * (P_{teacher} - T_F * P_{mean}) \tag{5}$$

$$T_F = round(1 + \mathrm{rand}(0,1)) \tag{6}$$

Where $P^{i}_{T',new}$ and $P^{i}_{T',old}$ represent the new and old position of the i-th individual respectively; the mean position P_{mean} can be calculated by the population mean; $P_{teacher}$ is the teacher, r is a random number in the interval [0,1]; T_F is the teaching factor that determines the change in the mean value, T_F can be 1 or 2.

Horizontal crossover of CSO:

$$P^{l_1,d}_{c,new} = r_1 * P^{l_1,d}_{c,old} + (1 - r_1) * P^{l_2,d}_{c,old} + c_1 * \left(P^{l_1,d}_{c,old} - P^{l_2,d}_{c,old}\right) \tag{7}$$

$$P^{l_2,d}_{c,new} = r_2 * P^{l_2,d}_{c,old} + (1 - r_2) * P^{l_1,d}_{c,old} + c_2 * \left(P^{l_2,d}_{c,old} - P^{l_1,d}_{c,old}\right) \tag{8}$$

Where r_1, r_2 are random numbers between [0,1]; c_1, c_2 are random numbers between [−1,1]; $P^{l_1,d}_{c,old}$ and $P^{l_2,d}_{c,old}$ are the d-th dimension of $P^{l_1}_{c,old}$ and $P^{l_2}_{c,old}$ in the parent population respectively; $P^{l_1}_{c,old}$ and $P^{l_2}_{c,old}$ represent the old position of the l_1-th and l_2-th individual respectively; $P^{l_1,d}_{c,new}$ and $P^{l_2,d}_{c,new}$ are the d-th dimension offspring of $P^{l_1,d}_{c,old}$ and $P^{l_2,d}_{c,old}$ generated by horizontal crossover searches. Obviously, the perturbation to the solution by horizontal crossover is one-dimensional, so the offspring can be generated from a more fine-grained space.

Learning-phase

In this stage, all offspring individuals increase their QoS fitness values by interacting with a randomly selected individual from the whole old individuals, such doing has the problems of poor stability and slow convergence. However, in the literature [19], we found that the one-dimensional interaction of individuals is very helpful to improve the convergence speed and enhance the stability of the algorithm. Therefore, to improve the convergence speed and enhance the stability of the algorithm, we are inspired by Eqs. (7) and (8), so the Eqs. (9) and (10) are improved to Eqs. (11) and (12). Similar to the horizontal crossover, the perturbation of the solution in Eqs. (11) and (12) are one-dimensional, which can let some dimensions in the population that fall into the local optimum can jump out of the iteration, thus the convergence speed and the stability of the algorithm is enhanced. Specifically, when an one-dimensional individual $P^{j,h}_{L,old}$ selected randomly another one-dimensional individual $P^{k,h}_{L,old}$ at any iteration (where $j \neq k$), two states can occur according to fitness values relative to each other: if the fitness of $P^{j}_{L,old}$ is better than $P^{k}_{L,old}$ then $P^{j,h}_{L,new}$ will be updated by Eq. (11), else by Eq. (12). Next, it is need to further judge whether the fitness value of $P^{j}_{L,new}$ is better than $P^{j}_{L,old}$, if better, replace $P^{j}_{L,old}$, in form:

Learning of TLBO:

$$P^{j}_{L,new} = P^{j}_{L,old} + r_j \times \left(P^{j}_{L,old} - P^{k}_{L,old}\right) \tag{9}$$

$$P^{j}_{L,new} = P^{j}_{L,old} + r_j \times \left(P^{k}_{L,old} - P^{j}_{L,old}\right) \tag{10}$$

Where r_j is a random number in the range [0,1]; $P^{j}_{L,new}$, and $P^{j}_{L,old}$ represent the new and old position of the j-th individual respectively; $P^{k}_{L,old}$ is the old position of the k-th individual.

Improved learning of TLBO:

$$P^{j,h}_{L,new} = P^{j,h}_{L,old} + r_j \times \left(P^{j,h}_{L,old} - P^{k,h}_{L,old}\right) \tag{11}$$

$$P^{j,h}_{L,new} = P^{j,h}_{L,old} + r_j \times \left(P^{k,h}_{L,old} - P^{j,h}_{L,old}\right) \tag{12}$$

Where r_j is a random number in the range [0,1]; $P^{j,h}_{L,new}$ is the h-th dimension offspring of $P^{j,h}_{L,old}$; $P^{j,h}_{L,old}$ and $P^{k,h}_{L,old}$ represent the h-th dimension of the j-th individual and the k-th individual respectively.

Output-phase

This Output-Phase will output the best result. Before that, we sort the individual fitness values learned in the learning stage, select the maximum fitness value, and compare it with the maximum fitness saved in the initialize-phase. If it better than the maximum fitness with the initialize-phase, save it as the optimal fitness value, else save the maximum fitness with the initialize-phase as the optimal fitness value, the algorithm will continue to iterate. Then, in each subsequent iteration, the fitness value of the current iteration is compared with the last optimal fitness value, the best fitness value is saved. In this way, the old fitness value will be updated with new fitness value until the maximum iteration is reached. Finally, output the optimal value.

Experiments

In this section, three experiments are designed to verify the effectiveness of improved-TC for cloud manufacturing SCOS problems in the large-scale environment. The first experiment is to observe the influence of the selection of the population ratio (TLBO: CSO) on the results. The second experiment is to verify the effectiveness of the improved Learning-Phase and Skyline query. The third experiment evaluates the performance of improved-TC in large-scale SCOS problem.

Experimental settings

Since there is no public dataset for cloud manufacturing service composition at present, our experimental data are randomly generated according to factual laws. Without loss of generality, all QoS attribute values in our dataset are randomly generated between [0.7, 0.95] [17]. In this dataset, including 20,000 services, each service has 6 attributes: time, price, reliability, success rate, cost, and reputation. In the experiments of this paper, we only use the following four properties. The weights of time, cost, reliability and reputation in the user's QoS preference for the task are chosen as $w_1 = 0.35$, $w_2 = 0.3$, $w_3 = 0.2$, and $w_4 = 0.15$ [30]. Besides, $R_{max} = 1000$, $P = 40$. Due to the random-ness of generated data, the experiments for each test data are performed to run 20 times to make a fair comparison.

The influence of improved-TC population proportion on quality of service values

Because of the horizontal crossover searches of CSO algorithm from small granularity searches, the best solution can be found in this way. However, if the proportion of CSO population is too large, many candidate services will be included in the CSO search, making the search time longer. In this case, it is difficult to guarantee the quality of the model. To obtain the best ratio of the population in the model, the proportion of CSO changes from 0 to 100 with an increment of 10, as shown in Fig. 5, where the meaning of the green rectangle rectangle is the average QoS value corresponding to the proportion of the CSO population. Each request is decomposed into 20 subtasks, each subtask corresponds to 50 candidate services.

As one can see from Fig. 5, the optimal QoS values are obtained as the proportion of CSO population grew from 0 to 100. According to Fig. 5, the average QoS rises rapidly as the proportion of CSO population increases from 0 to 10. This can be due to the fact that the CSO improves the search ability of the algorithm. When the proportion of CSO population changes from 10 to 20, the average QoS value increases rapidly, which is because the search ability of the algorithm becomes stronger as the proportion of CSO population increases. When the proportion of CSO population increases from 20 to 70, the average QoS value increases slowly. However, the QoS results show almost parallel over the span of 70–100, and even the average QoS value has a downward trend at 80. This indicates that the performance of the algorithm will reach the bottleneck when the proportion of CSO population reaches a certain number. It can be seen that when the proportion of CSO population is equal to 70, the average QoS value obtained by the algorithm is the highest. Therefore, the population ratio of 30:70 (TLBO: CSO) is selected in this paper.

Convergence verification

In order to verify the Convergence for applying improved learning phase in algorithms, Improved-TC, Hybrid-TC [19] (Hybrid-TC with Skyline query), and NSK-TC (Hybrid-TC without Skyline query) are compared in this section. Each task T is decomposed into 10 subtasks, the number of candidate services for each subtask is fixed as 50.

The convergence curves of the three different algorithms are shown in Fig. 6. It can be observed that when the three algorithms obtain almost equal solutions, Improved-TC has better convergence than Hybrid-TC and NSK-TC. This is because in the small-scale environment, Improved-TC and Hybrid-TC are easier to find the optimal solution of the algorithm. When the solutions are close to the optimal solutions, the improved learning phase enables the algorithm to converge faster. Specifically, Improved-TC algorithm converged at about 220th, but at about 580th and 700th, respectively, for Hybrid-TC, and NSK-TC. Furthermore, it can be observed from Fig. 6 that in the initial stage, the QoS value obtained with Improved-TC and Hybrid-TC are higher than the NSK-TC, Moreover, the algorithms with Skyline query

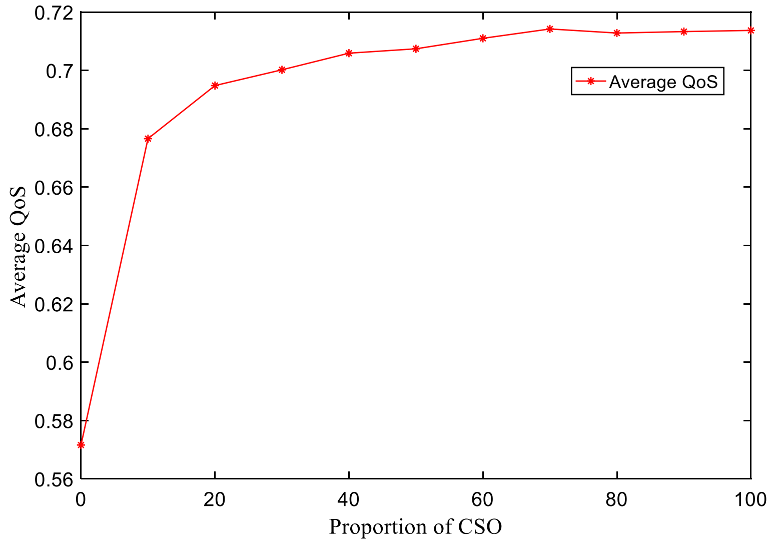

Fig. 5 The influence of CSO population proportion on QoS value

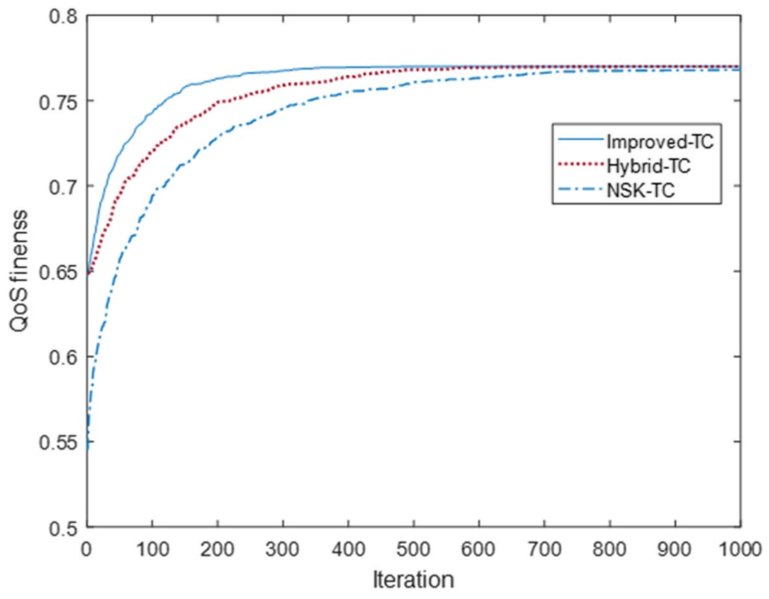

Fig. 6 Algorithms convergence curves

can achieve a slightly high QoS value than the algorithm without Skyline query in the final result. This is because the optimal services are selected from the candidate services by the Skyline query, which improves the solution of the algorithm. Moreover, the algorithm with Skyline query (Improved-TC and Hybrid-TC) converges faster than the algorithm without Skyline query (NSK-TC), which proves that Skyline query can reduce the search space of the algorithm, thus making the convergence of the algorithm better.

The performance of improved-TC in large-scale service composition and optimization selection problems

In order to verify the effectiveness of our proposed method in different size problem spaces. The improved-TC is compared with teaching–learning-based optimization (TLBO), hybrid-TC [19], particle swarm optimization (PSO) [31], and IGWO [6]. These algorithms were chosen for comparison because they perform well in SCOS problems. The parameter settings of all algorithms are shown in Table 3. In particular, because improved-TC is improved on the basis of hybrid-TC, all improved-TC parameter settings are the same as hybrid-TC.

The size of the SCOS problem space has two important dimensions: (1) The number of subtasks for each manufacturing task; (2) The number of candidate services in the service set corresponding to each subtask. All the algorithms will be tested in 9 sizes of the problem space, as shown in Table 4. Where M represents the number of subtasks, and N' represents the number of candidate services corresponding to each subtask.

Figure 7 shows the box plot (median, dispersion, and outliers) about the QoS values of the best solutions obtained for the five algorithms after 20 independent runs. The higher the median value indicates that the algorithm finds a better solution set in terms of QoS fitness value than other algorithms. The lower dispersion value and a small number of outliers mean that the solution set obtained by the algorithm is robust compared with other algorithms.

According to the results, the improved-TC obtained the best results in nine situations. Similarly, Hybrid-TC, TLBO, PSO, and GWO provide the second, third, fourth best and the worst results, respectively. Careful observation reveals that as the problem space increases, the QoS fitness values of the optimal solution are reduced for all algorithms. This is because as the problem space becomes larger, the algorithm has more targets to search, and the algorithm's search strength is insufficient. When the number of subtasks is fixed and only the number of the candidate services in the service set change from 150 to 450, the QoS fitness value of the optimal solution of each algorithm does not decrease significantly, which shows that the number of candidate services corresponding to the subtasks does not interfere much with the algorithms. When the number of subtasks change from 10 to 30, and the number of the candidate service corresponding to each subtask is fixed, the service QoS fitness values of the optimal solutions of each algorithm decreases greatly, which shows that the number of subtasks has a

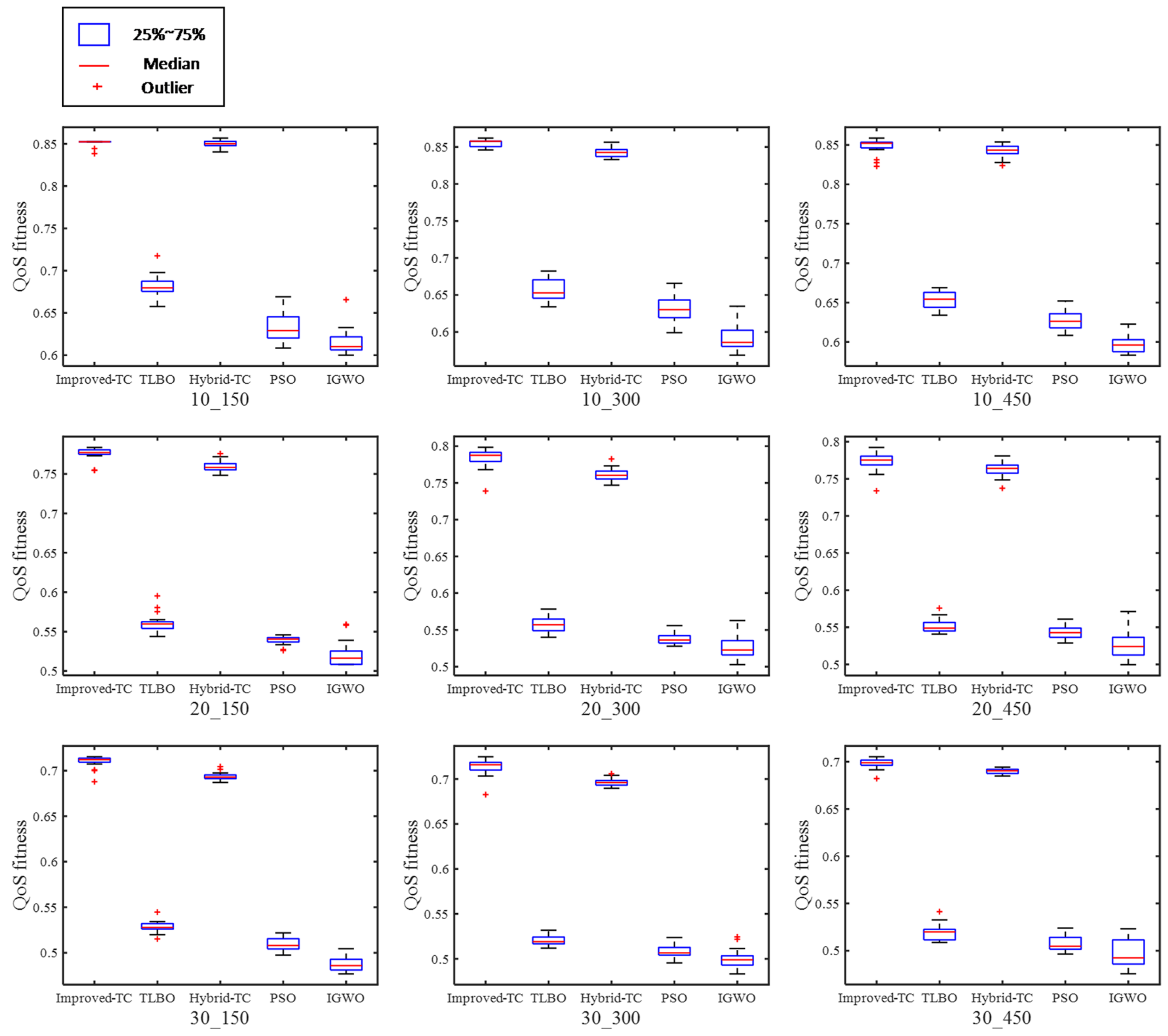

Fig. 7 Performance comparison of different algorithms

Table 3 The settings of algorithm parameters QoS aggregation functions

Algorithm	Parameters
PSO	The inertia weight w varies from 0.9 to 0.7 linearly, the learning factors c_1 varies from 2.5 to 0.5 linearly, while c_2 varies from 0.5 to 2.5.
IGWO	The random vectors r_1,r_2 = rand[0, 1], the control coefficients c_1 = 0.3 and θ = 2, while a varies from 2 to 0 linearly.
TLBO	The random vectors r, r_j = rand[0, 1], the teaching factor T_F is equal to 1 or 2.
Hybrid-TC	The random vectors r, r_j, r_1, r_2 = rand[0, 1], and the random numbers c_1, c_2 = rand[− 1,1], the teaching factor T_F is equal to 1 or 2.
Improved-TC	The random vectors r,r_j, r_1,r_2 = rand[0, 1], and the random numbers c_1, c_2 = rand[−1,1], the teaching factor T_F is equal to 1 or 2.

Table 4 The different sizes of the problem space

Sizes	1	2	3	4	5	6	7	8	9
M	10	10	10	20	20	20	30	30	30
N′	150	300	450	150	300	450	150	300	450

great influence on the composition service. According to Fig. 7, improved-TC, hybrid-TC and TLBO obtain better solutions than the other two algorithms in all situations, which shows that TLBO itself has certain advantages in solving the SCOS problems. In terms of the dispersion of QoS fitness values, there are some differences. The dispersion of PSO takes the worst in the situations 1 to3 while GWO falls to the worst in the situations 2 to 9. This is because the offspring generated by these two algorithms are crossed with individuals as units, the evolutions of PSO and GWO is unstable in some iterations. The improved-TC algorithm always produces the lowest dispersion, so that it is the most stable.

Overall, compared with other algorithms, improved-TC can always search for the best solutions under the same conditions. This is because the target is searched from one dimension, which guarantees the good performance of the improved-TC algorithm. It not only enhances the search range of the algorithm to avoid local optima, but also improves the stability of the algorithm. In addition, it can also be noticed that as the size of the service composition problem space increases, the gap in the solution set between improved-TC and other algorithms becomes more obvious.

Conclusions and future work

Cloud manufacturing is changing the development prospect of manufacturing industry, so it is of great significance to study cloud manufacturing. Under the background of large-scale cloud manufacturing, we have developed a new optimization algorithm (improved-TC) based on the hybrid-TC algorithm to solve the large-scale service composition and optimal selection (SCOS) problems. The algorithm is inspired by the horizontal crossover of CSO in the hybrid-TC teaching phase. In the learning phase of improved-TC, the multi-dimensional search method of the algorithm is changed to one-dimensional search, so that some dimensions in the population that fall into the local optimum have the opportunity to jump out of the iteration. Besides, we adopted the Skyline query in the initialize-phase, which can select the optimal services from the candidate services to improve the convergences of the algorithm. Experiments show the QoS values of our algorithm in different cases, and by changing the number of different subtasks and candidate services, we verify that our algorithm is effective and stable in improving the quality of solutions. During the experiment, we found that our proposed method is effective in improving the QoS value, but it is not suitable for dynamic service composition and optimal selection (SCOS) problems. In the future research, we will try to adopt dynamic adaptive models such as reinforcement learning to solve dynamic service composition and optimal selection (SCOS) problem.

Acknowledgments
The authors would like to thank the staff and postgraduate students at the School of Big Data & Software Engineering, Chongqing University, Chongqing, China for their assistance and valuable advice.
An earlier version of this paper was presented at the International Conference on International Joint Conference on Neural Networks (IJCNN).

Authors' contributions
Jun Zeng is the corresponding author and supervised the work and analyzed the experimental results. Juan Yao designed and performed the experiments and drafted the manuscript. Min Gao and Junhao Wen provided helpful suggestions and revised the manuscript. All authors have read and approved the manuscript.

Funding
This research is supported by the National Key Research and Development Program of China (Grant No.2019YFB1706104), the National Science Foundation of Chongqing, China (No. cstc2020jcyj-msxmX0900) and the Fundamental Research Funds for the Central Universities (Project No. 2020CDJ-LHZZ-040).

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 31 July 2022 Accepted: 29 September 2022
Published online: 15 October 2022

References
1. Mourad MH, Nassehi A, Schaefer D, Newman ST (2020) Assessment of interoperability in cloud manufacturing. Robotics Comput Integr Manuf. 61:101832. https://doi.org/10.1016/j.rcim.2019.101832
2. Yuan M, Zhou Z, Cai X, Sun C, Gu W (2020) Service composition model and method in cloud manufacturing. Robotics Comput Integr Manuf 61:101840. https://doi.org/10.1016/j.rcim.2019.101840
3. Bouzary H, Chen FF (2020) A classification-based approach for integrated service matching and composition in cloud manufacturing. Robotics

Comput Integr Manuf 66:101989. https://doi.org/10.1016/j.rcim.2020.101989

4. Bouzary H, Frank Chen F (2018) Service optimal selection and composition in cloud manufacturing: a comprehensive survey. Int J Adv Manuf Technol 97(1):795–808
5. Yang Y, Yang B, Wang S, Jin T, Li S (2020) An enhanced multi-objective grey wolf optimizer for service composition in cloud manufacturing. Appl Soft Comput 87:106003. https://doi.org/10.1016/j.asoc.2019.106003
6. Yang Y, Yang B, Wang S, Liu W, Jin T (2019) An improved grey wolf optimizer algorithm for energy-aware service composition in cloud manufacturing. Int J Adv Manuf Technol 105(7):3079–3091
7. Akbaripour H, Houshmand M, Van Woensel T, Mutlu N (2018) Cloud manufacturing service selection optimization and scheduling with transportation considerations: mixed-integer programming models. Int J Adv Manuf Technol 95(1):43–70
8. Liu Y, Wang L, Wang XV, Xu X, Zhang L (2019) Scheduling in cloud manufacturing: state-of-the-art and research challenges. Int J Prod Res 57(15–16):4854–4879. https://doi.org/10.1080/00207543.2018.1449978
9. Li F, Zhang L, Liu Y, Laili Y, Tao F (2017) A clustering network-based approach to service composition in cloud manufacturing. Int J Comput Integr Manuf 30(12):1331–1342. https://doi.org/10.1080/0951192X.2017.1314015
10. Lu Y, Xu X (2017) A semantic web-based framework for service composition in a cloud manufacturing environment. J Manuf Syst 42:69–81
11. Lartigau J, Xu X, Nie L, Zhan D (2015) Cloud manufacturing service composition based on qos with geo-perspective transportation using an improved artificial bee colony optimisation algorithm. Int J Prod Res 53(14):4380–4404
12. Que Y, Zhong W, Chen H, Chen X, Ji X (2018) Improved adaptive immune genetic algorithm for optimal qos-aware service composition selection in cloud manufacturing. Int J Adv Manuf Technol 96(9):4455–4465
13. Wang H, Gu M, Yu Q, Tao Y, Li J, Fei H, Yan J, Zhao W, Hong T (2019) Adaptive and large-scale service composition based on deep reinforcement learning. Knowl Based Syst 180:75–90. https://doi.org/10.1016/j.knosys.2019.05.020
14. Li K, Zhao J, Hu J et al (2022) Dynamic energy efficient task offloading and resource allocation for Noma-enabled iot in smart buildings and environment. Build Environ. https://doi.org/10.1016/j.buildenv.2022.109513
15. Xu J, Li D, Gu W et al (2022) Uav-assisted task offloading for iot in smart buildings and environment via deep reinforcement learning. Build Environ. https://doi.org/10.1016/j.buildenv.2022.109218
16. Chen Y, Gu W, Li K (2022) Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning. Int J Commun Syst e5154. https://doi.org/10.1002/dac.5154
17. Fazeli MM, Farjami Y, Nickray M (2019) An ensemble optimisation approach to service composition in cloud manufacturing. Int J Comput Integr Manuf 32(1):83–91. https://doi.org/10.1080/0951192X.2018.1550679
18. Zhou J, Yao X (2017) Hybrid teaching–learning-based optimization of correlation-aware service composition in cloud manufacturing. Int J Adv Manuf Techno 91(9):3515–3533
19. Yao J, Zeng J, Wen J, Zhou W, Gao M (2021) Hybrid-tc: a hybrid teaching-learning-based optimization algorithm for service composition in cloud manufacturing. In: International Joint Conference on Neural Networks, 2021, Shenzhen, China, July 18–22, 2021, pp 1–8. https://doi.org/10.1109/IJCNN52387.2021.9533977
20. Rao RV, Savsani VJ, Vakharia DP (2012) Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems. Inf Sci 183(1):1–15. https://doi.org/10.1016/j.ins.2011.08.006
21. Meng A, Chen Y, Yin H, Chen S (2014) Crisscross optimization algorithm and its application. Knowl Based Syst 67:218–229. https://doi.org/10.1016/j.knosys.2014.05.004
22. Chen Y, Zhao F, Chen X, Wu Y (2022) Efficient multi-vehicle task offloading for mobile edge computing in 6g networks. IEEE Trans Veh Technol 71(5):4584–4595. https://doi.org/10.1109/TVT.2021.3133586
23. Huang J, Lv B, Wu Y et al (2022) Dynamic admission control and resource allocation for mobile edge computing enabled small cell network. IEEE Trans Veh Technol 71(2):1964–1973.https://doi.org/10.1109/TVT.2021.3133696
24. Huang J, Tong Z, Feng Z (2022) Geographical poi recommendation for internet of things: a federated learning approach using matrix factorization. Int J Commun Syst. https://doi.org/10.1002/dac.5161
25. Chen Y, Zhao F, Lu Y, Chen X (2021) Dynamic task offloading for mobile edge computing with hybrid energy supply. Tsinghua Sci Technol. https://doi.org/10.26599/TST.2021.9010050
26. Chen Y, Xing H, Ma Z et al (2022) Cost-efficient edge caching for Noma-enabled iot services. China. Communications
27. Zhou J, Yao X (2017) Multi-population parallel self-adaptive differential artificial bee colony algorithm with application in large-scale service composition for cloud manufacturing. Appl Soft Comput 56:379–397
28. Laili Y, Lin S, Tang D (2020) Multi-phase integrated scheduling of hybrid tasks in cloud manufacturing environment. Robotics Comput Integr Manuf 61:101850. https://doi.org/10.1016/j.rcim.2019.101850
29. Liang H, Wen X, Liu Y, Zhang H, Zhang L, Wang L (2021) Logistics-involved qos-aware service composition in cloud manufacturing with deep reinforcement learning. Robot Comput Integr Manuf 67:101991
30. Zhou J, Yao X (2017) A hybrid artificial bee colony algorithm for optimal selection of qos-based cloud manufacturing service composition. Int J Adv Manuf Technol 88(9):3371–3387
31. Tao F, Zhao D, Hu Y, Zhou Z (2010) Correlation-aware resource service composition and optimal-selection in manufacturing grid. Eur J Oper Res 201(1):129–143. https://doi.org/10.1016/j.ejor.2009.02.025
32. Garg S, Modi K, Chaudhary S (2016) A qos-aware approach for runtime discovery, selection and composition of semantic web services. Int J Web Inf Syst 12(2):177–200. https://doi.org/10.1108/IJWIS-12-2015-0040
33. Chen Y, Liu Z, Zhang Y, el al. (2021) Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial internet of things. IEEE Trans Industr Inform 17(7):4925–4934
34. Jin H, Yao X, Chen Y (2017) Correlation-aware qos modeling and manufacturing cloud service composition. J Intell Manuf 28(8):1947–1960
35. Hayyolalam V, Kazem AAP (2018) A systematic literature review on qos-aware service composition and selection in cloud environment. J Netw Comput Appl 110:52–74. https://doi.org/10.1016/j.jnca.2018.03.003
36. Rao RV, Savsani VJ, Vakharia DP (2011) Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315. https://doi.org/10.1016/j.cad.2010.12.015
37. Zhang F, Hwang K, Khan SU, Malluhi QM (2016) Skyline discovery and composition of multi-cloud mashup services. IEEE Trans Serv Comput 9(1):72–83. https://doi.org/10.1109/TSC.2015.2449302
38. Wu J, Chen L, Liang T (2014) Selecting dynamic skyline services for qos-based service composition. Appl Math Inform Sci 8(5):2579–2588

Publisher's Note