Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Service composition considering energy consumption of users and transferring files in a multicloud environment

Jianmin Li* and Shunzhi Zhu

Abstract

In the internet and cloud environment, service composition is always used to enhance the function and processing ability of clouds. Those clouds work together for a user and provide different functions. A service request may involve multiple clouds. The past work focuses on the method of service composition and ignores the energy composition when files are transferred between clouds, including the energy consumption for transferring files (sending files from the user to the cloud and receiving files from the cloud to the user) of the user. The paper models the service composition in a multicloud environment. Based on those models, we use the GA (genetic algorithm) algorithm (GA-C) to solve the service composition problem with multiple targets in a multicloud environment. Simulation results show that the GA-C can: (1) reduce the average number of involved clouds and the energy consumption between clouds, and (2) reduce the energy consumption of the user and the failure rate of service composition.

Keywords Multicloud environment, Service composition, Energy consumption, Atomic service, Transferring files

Introduction

With the development of the Internet and cloud computing, increasingly more services are provided on the Internet [1–5]. As a method of utilizing services, service composition has been widely used in different fields [6–8]. It combines atomic services in different areas to meet the needs of users with multiple functions and different QoSs. The service composition approach has been widely used in different industries, such as cloud manufacturing [7, 9, 10] and metro services [11].

The service composition approach is important both to the user and to the benefits of the entire system. For the user, it is necessary to meet the user's functional and performance requirements. For service providers, it is necessary to improve overall efficiency while meeting the needs of users. Most service composition methods take into account the user's QoS. These QoSs include both hard requirements [12–15], such as time limits [3, 16–18] and costs [6, 19–22], and soft requirements [11, 18, 23–32], such as security [4, 5, 9, 10, 33], reliability and flexibility.

Past work has mainly focused on researching service composition methods to meet user requirements and achieve system scheduling goals. These methods include Markov method [6], GA (genetic algorithm) [34], blockchain method [12], and deep learning method [11]. The scheduling targets of those service composition methods include (1) meeting time requirements, (2) reducing cost, (3) ensuring security, (4) satisfying reliability, and (5) a tradeoff between some of them [5, 19, 21, 35–37].

On the internet, different clouds provide different services. A single cloud cannot provide enough services for a user. Most of the time, a service request may need multiple clouds to obtain the service. For example, when we buy clothes on the internet, we need to visit the website to look for the related clothes, then we pay the money

*Correspondence:
Jianmin Li
lijianmin2006@sina.cn
School of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, China

through the service provided by someone bank service, last, a service provided by a courier services company to send the clothes to us.

In the past, service composition methods rarely considered energy consumption [38], especially in a multicloud environment. The energy consumption of transferring files between clouds and the energy consumption of users are due to sending and receiving files. In this paper, we focus on the energy consumption for transferring files between clouds and the energy consumption for transferring files of users. We pay attention to the energy consumption of users, because reducing the energy consumption of mobile devices (including IoT devices) is very important. The mobile device has a limitation of the energy supply; if we can reduce the energy consumption, the work time for the mobile device would be lengthened. When we support every cloud that has the same energy efficiency, then, the energy consumption for transferring between clouds is the most important aspect of energy consumption for the service. So, in the paper, we major pay attention to the two kinds of energy consumption for transferring files.

The paper is organized as follows: "Introduction" section is the introduction. "Related works" section gives the related work of the service composition. It also gives an introduction to the service composition method that considers energy consumption. "System framework for service composition in multiclouds" section introduces the framework of service composition under a multicloud environment. We give the models used for service composition in "System model of service composition in the multicloud environment" section. "Service composition method under multicloud environment" section presents the proposed service composition method based on GA. "Simulation and comparisons" section presents the simulation and comparison. "Conclusions and future work" section presents the conclusion and future work.

Related works

Service composition methods have been widely researched in recent decades. Most of the methods try to meet the QoS requirement [4]. S. Sefati et al. [6] used a hidden Markov model (HMM) and an ant colony optimization (ACO) for the service composition that targets enhancing the QoS. They used the trained HMM (hidden Markov model) to predict QoS. Based on the QoS estimation result, they used the ACO algorithm to find a suitable service composition result. To address the complexity (the number of candidate services is very large) of the service composition, R. Boucetti et al. [34] combined genetic algorithms and neural networks (NN) for QoS-aware services composition for the context of large-scale environments. They decomposed the QoS intervals into M QoS-levels. GA obtains the ideal theoretical composition with global QoS optimization, and NN is used to eliminate the IoT services, and only keeps the services having the same QoS level of the atomic services to the ideal composition given by GA.

With the development of the cloud, the service composition under multicloud environment has also received attention, and some methods have been proposed to solve the service composition problem under multicloud environments [39–43]. K. Kritikos et al. [41] considered the different types of requirements of the service request, and gave a service composition approach according to the requirement. B. Pang et al. [42] gave a formal concept for service composition under the multicloud environment about multiple users. They first adopted collaborative filtering to obtain the services request of the target user, then they constructed the service–provider concept, and finally, a way is given to select the best multicloud composition and give recommended results for the target user. P. Kendrick et al. [39] used agent-matchmakers and agent representatives for a multiagent-based service composition method. With the help of multiple agents, it can obtian the service composition efficiently. To solve the problem of the security of the service composition problem under the multicloud environment, F. Lahmar et al. [40] proposed a security-aware multicloud service composition approach using fuzzy formal concept analysis (fuzzy FCA) and rough theory (RS) set to guarantee security. A. Souri et al. [43] presented a hybrid formal verification approach to assess the service composition in multicloud environments targeted at reducing the number of clouds of a service request and meeting the required level of quality of service (QoS). Other service composition methods we introduce in "Simulation and comparisons" section include All-C [44–46], Bas-C [44–46], Smt-C [44–46], Swm-C [47], and Fast-C [48].

With more attention being given to the energy consumption of computing systems, energy-aware service composition has also been emphasized by some researchers [49, 50]. D. Zeng et al. [51] solved the energy-efficient service composition problem in a green energy-powered cyber–physical fog system, and jointly considered source rate control, load balancing, and service replica deployment. They formulated the problem as a mixed integer linear programming problem and used a heuristic to reduce the complexity of the problem. For the same reason, J. Ibrahim et al. [52] proposed an energy-aware mechanism to optimize the mobile cloud service composition by using a hybrid shuffled frog leaping algorithm and genetic algorithm. To solve the limitations in storage, computation, and energy, E. Tong et al. [53] proposed hierarchical energy-efficient service selection to address the dynamic characteristics of the Internet of Things. The method works well under the dynamic network state. S.

Deng et al. [54] focused on the problem of mobile service composition in terms of energy consumption. It presented energy consumption aggregation rules for composite services with different structures. They used GA to solve the problem. Those studies always try to reduce the energy consumption of all service requests.

Different from past research, our target is to reduce the energy consumption of users and the energy consumption of transferring files between clouds. Because the network is always dynamic, so the energy consumption between users and clouds is dynamic. Especially when every cloud has the same energy efficiency, the energy consumption for transferring files is the most important aspect of energy consumption.

System framework for service composition in multiclouds

The system framework

In the system, there are multiple clouds and every cloud provides some atomic services. A service request needs to visit some clouds to obtain all atomic services in the service request.

Figure 1 is the service architecture that we adopt in the multicloud environment:

- Each cloud center ($C_1 \sim C_9$) has multiple services, and each service has multiple instances of atomic services. For example, cloud center C_4 provides two atomic services $Service_4$ and $Service_5$.
- A multicloud environment consists of multiple clouds with different $MCE = \{C_1, C_2, ..., C_{MCEN}\}$, and the services provided by these clouds have different QoS.
- The user (User) puts forward the information of requirements to the cloud management center (Cloud manager), and this information includes the required web services and their QoSs, even including cost and other information;
- The service management center is responsible for service composition work according to the requirements of the cloud center. It not only completes the system targets, but also needs to meet the user's requirements.

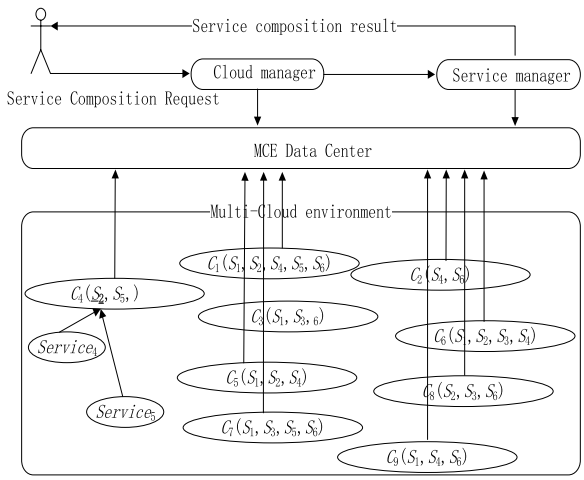

Fig. 1 System model of the service composition under a multicloud

QoSs used in the paper

We consider six QoSs in the paper:

- Cost (Price): Expenses are usually measured in monetary terms, and generally maintain a relatively stable state. Usually, the cost always is related to the processing time and the type of atomic services.
- Availability: Availability is usually expressed as the ratio of the time it can be accessed over a while to the total time, so it can be quantified as an exact value in the interval [0, 1]. Availability is the ratio of the time the system can operate normally to the total time.
- Response time (Time): Time refers to the time from the start of obtaining the right to use the resource to the time when the resource is released. The lower the response time of the cloud service, the higher the satisfaction of the general user, and the more likely the system will get better benefits.
- Reliability: Reliability is a user-perceived indicator that is subjective, so it is difficult to quantify. Generally, reliability is divided into several levels.
- Reputation: Usually obtained from customer feedback, it is subjective. Since this is related to the needs of users, there may be differences in the perception of different users. Generally, users give feedback (score) after they have used some atomic services and the system gives a reputation according to the feedback of various users. Since reputation is also a generally subjective description, it is sometimes not feasible.
- Security (Security): Many current researchers also regard security as a basic QoS attribute. With the wide application of cloud computing, any individual, enterprise, or unit can obtain related services from the cloud through the Internet. The openness of the system makes its security of the system more important than ever.

System model of service composition in the multicloud environment

Structure of services

In this section, the QoS fitting calculation methods of several common combined structures are introduced: sequence structure (Fig. 2a), loop structure (Fig. 2b), and choice structure (Fig. 2c).

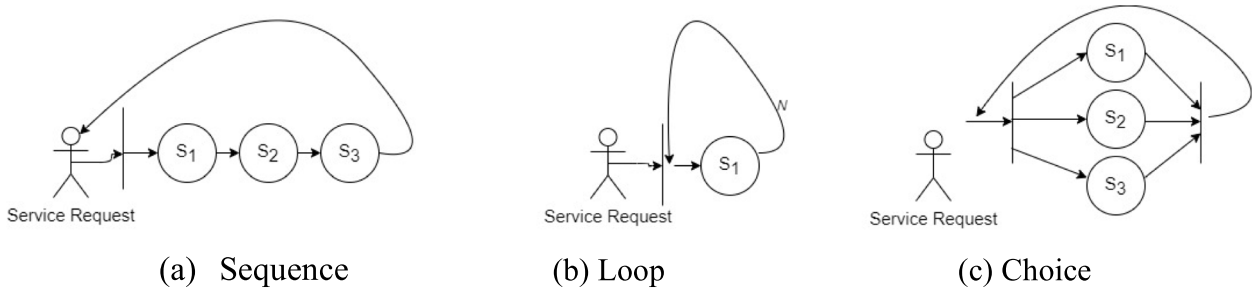

Fig. 2 Three kinds of service composition structures

Sequence structure

The sequence structure is shown in Fig. 2a. Its execution time and price are obtained by accumulating the corresponding values of the three atomic services, its reliability and availability are obtained by multiplying the corresponding values of the three atomic services, and the reputation takes the minimum value of the three atomic services.

Loop structure

The loop structure is shown in Fig. 2b, assuming that the atomic service loops N times. The execution time and price are N times the corresponding values of the three atomic services. The reliability and availability are obtained by multiplying the corresponding values of the three atomic services. Reputation is the reputation of the atomic service.

Choice structure

The loop structure is shown in Fig. 2c, assuming that the probability of execution of the i th atomic service is $p(i)$. The six attributes are obtained by multiplying the corresponding value and the relative probability of all atomic services. In Table 1, QoS_i^{RT} is the value of the response time of the i th atmospheric service (same as other parameters).

In fact, many complex service structures can be regarded as a combination of these three basic structures.

Table 1 QoS composition methods for three structures

Structure	Sequence	Loop	Choice
Response time	$\sum_{i=1}^{3} QoS_i^{RT}$	$N * QoS_i^{RT}$	$\sum_{i=1}^{3} p(i) * QoS_i^{RT}$
Cost	$\sum_{i=1}^{3} QoS_i^{CT}$	$N * QoS_i^{CT}$	$\sum_{i=1}^{3} p(i) * QoS_i^{CT}$
Reliability	$\prod_{i=1}^{3} QoS_i^{REL}$	$\prod_{i=1}^{N} QoS_i^{REL}$	$\sum_{i=1}^{3} p(i) * QoS_i^{REL}$
Availability	$\prod_{i=1}^{3} QoS_i^{AV}$	$\prod_{i=1}^{N} QoS_i^{AV}$	$\sum_{i=1}^{3} p(i) * QoS_i^{AV}$
Reputation	$\text{Min}(QoS_i^{REP})$	QoS_i^{REP}	$\sum_{i=1}^{3} p(i) * QoS_i^{REP}$
Security	$\text{Min}(QoS_i^{SE})$	$\text{Min}(QoS_i^{SE})$	$\text{Min}(QoS_i^{SE})$

In other words, we can obtain services with higher complexity by extending those three structures.

Energy consumption model for service composition

Since services are executed in different clouds, data are transferred between various cloud centers, that is, a large amount of data needs to be transferred (Fig. 2). While transferring data, energy consumption occurs for sending files and receiving files. Because we suppose that the energy efficiency of each virtual center is the same, we do not consider the cost of executing jobs caused by the heterogeneity of virtual centers [53, 54].

According to the relevant literature, the energy consumption of file uploading is different from that of downloading. If only a small file is uploaded, and the large files are all in a cloud center, the energy consumption of file transmission will be reduced. We assume that the user's sending power consumption is U^{sp}, and the receiving power consumption is U^{rp}. The sending power consumption and the receiving power consumption of cloud C_j are C_j^{sw} and C_j^{rw}, respectively. Suppose the service request sequence is $R = \{r_1, r_2, \ldots, r_l\}$. The service is a sequence structure (if it is not a sequence structure, it can be changed to linear in chronological order), and the output and output file sizes of r_l are r_l^{in} and r_l^{out}. $E_U(R)$, $E_R(R)$, $E_S(R)$ represent user energy consumption, (between different clouds) file sending energy consumption, and (between different clouds) file receiving energy consumption, respectively. $cksc(l_{k-1}, l_k)$ returns whether the subservice (l_{k-1}, l_k) is in a cloud; if so, it returns 0; otherwise, it returns 1. Formulas (1, 2, 3 and 4) are used to calculate the energy consumption of different aspects (Fig. 3).

$$E_U(R) = l_1^{in} * U^{sp} + l_k^{in} * U^{rp} \tag{1}$$

$$E_R(R) = \sum_{2}^{lend} l_k^{in} * cksc(l_{k-1}, l_k) * PWR(l_k) \tag{2}$$

$$E_S(R) = \sum_{1}^{lend-1} l_k^{out} * cksc(l_{k-1}, l_k) * PWS(l_k) \tag{3}$$

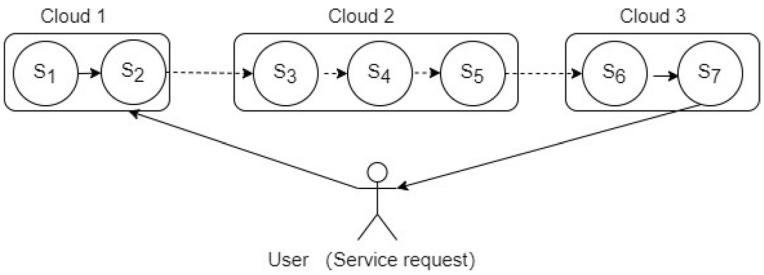

Fig. 3 Energy consumption model for the multicloud environment

$$E(R) = E_U(R) + E_R(R) + E_S(R) \tag{4}$$

Service composition method under multicloud environment

The QoS management for service composition under a multicloud environment

When multiple clouds can provide the service to the user. How to judge which atomic service is the best that is a problem for us. We suppose all the services provided by various clouds are denoted as:

$$C_i = \{< P^{RT}_{i,sid}, P^{CT}_{i,sid}, P^{REL}_{i,sid}, P^{PAV}_{i,sid}, P^{PEP}_{i,sid}, P^{PSE}_{i,sid} >\} \tag{5}$$

i and sid are the identifiers of the cloud and the atomic service, respectively. $P^{RT}_{i,sid}$, $P^{CT}_{i,sid}$, $PS^{REL}_{i,sid}$, $P^{PAV}_{i,sid}$, $P^{PEP}_{i,sid}$, and $P^{PSE}_{i,sid}$ are the response time, cost, reliability, availability, reputation, and security of service sid in cloud i, and we take the service as $P^{sid}_i =< P^{RT}_{i,sid}, P^{CT}_{i,sid}, P^{REL}_{i,sid}, P^{PAV}_{i,sid}, P^{PEP}_{i,sid}, P^{PSE}_{i,sid} >$.

The service request R_j can be denoted as follows:

$$R_j = \{< R^{RT}_{j,sid}, R^{CT}_{j,sid}, R^{REL}_{j,sid}, R^{RAV}_{j,sid}, R^{REP}_{j,sid}, R^{RSE}_{j,sid} >\} \tag{6}$$

j is the identifier of a service request. $R^{RT}_{j,sid}$, $R^{CT}_{j,sid}$, $R^{REL}_{j,sid}$, $R^{RAV}_{j,sid}$, $R^{REP}_{j,sid}$, and $R^{RSE}_{j,sid}$ are the response time, cost, reliability, availability, reputation, and security of the atomic service request R_j.

For the response time, $\overline{RT}$ and σ_{RT} are the average value and variance of the response time of the atomic service in the service requests (same to other parameters). To normalize the QoS value of the service request R_j and the QoS in the cloud C_i, we give the definition:

$$nP^{RT}_{i,sid} = 0.5 + \frac{P^{RT}_{i,sid} - \overline{RT}}{2 \times \beta\sigma_{RT}} \tag{7}$$

$$nR^{RT}_{j,sid} = 0.5 + \frac{R^{T}_{j,sid} - \overline{RT}}{2 \times \beta\sigma_{RT}} \tag{8}$$

Same for other QoSs used in the paper:

$$nP^{CT}_{i,sid} = 0.5 + \frac{P^{CT}_{i,sid} - \overline{CT}}{2 \times \beta\sigma_{CT}} \tag{9}$$

$$nR^{CT}_{j,sid} = 0.5 + \frac{R^{CT}_{j,sid} - \overline{CT}}{2 \times \beta\sigma_{CT}} \tag{10}$$

$$nP^{REL}_{i,sid} = 0.5 + \frac{P^{REL}_{i,sid} - \overline{REL}}{2 \times \beta\sigma_{REL}} \tag{11}$$

$$nR^{REL}_{j,sid} = 0.5 + \frac{R^{REL}_{j,sid} - \overline{REL}}{2 \times \beta\sigma_{REL}} \tag{12}$$

$$nP^{AV}_{i,sid} = 0.5 + \frac{P^{AV}_{i,sid} - \overline{AV}}{2 \times \beta\sigma_{AV}} \tag{13}$$

$$nR^{AV}_{j,sid} = 0.5 + \frac{R^{AV}_{j,sid} - AV}{2 \times \beta\sigma_{AV}} \tag{14}$$

$$nP^{REP}_{i,sid} = 0.5 + \frac{P^{REP}_{i,sid} - \overline{REP}}{2 \times \beta\sigma_{REP}} \tag{15}$$

$$nR^{REP}_{i,sid} = 0.5 + \frac{R^{REP}_{i,sid} - \overline{REP}}{2 \times \beta\sigma_{REP}} \tag{16}$$

$$nP^{SE}_{i,sid} = 0.5 + \frac{P^{SE}_{i,sid} - \overline{SE}}{2 \times \beta\sigma_{SE}} \tag{17}$$

$$nR^{SE}_{i,sid} = 0.5 + \frac{R^{SE}_{i,sid} - \overline{SE}}{2 \times \beta\sigma_{SE}} \tag{18}$$

Some attributes are positive attributes (larger is better, such as stability), and others are negative attributes (smaller is better, such as price and cost). In our paper, the response time and cost are the negative attributes and others are position attributes. We give a weight to the cost:

$$W_{CT} = {num(P_{i,sid}^{CT})}\Big/{\sum_A num(A_i^{sid})} \tag{19}$$

where $num(P_{i,sid}^{CT})$ is the number of atomic services provided by the cloud which has a lower cost than $P_{i,sid}^{CT}$ and $num(P_{i,sid}^{CT})$ is the number of atomic services that meet the cost of service request RCT_i^{sid}. $A = \{PCT, PRT, PREL, PAV, PREP, PSE\}$ W_{RT},W_{REL},W_{AV} ,W_{REP}, and W_{SE} are the weights of the response time, reliability, availability, reputation, and security of the service request R_j. They have the same equations as formula (19), so we do not give the formulas here.

Here we give a score when P_i^{sid} is allocated to R_j:

$$score(P_i^{sid}, R_j) = W_{RT} * sc(nP_{i,sid}^{RT}, nR_{j,sid}^{RT}) + W_{CT} * sc\left(nP_{i,sid}^{CT}, nR_{j,sid}^{CT}\right) + W_{REL} * sc\left(nP_{i,sid}^{REL}, nR_{j,sid}^{REL}\right) + W_{AV} * sc\left(nP_{i,sid}^{AV}, nR_{j,sid}^{AV}\right) + W_{REP} * sc\left(nP_{i,sid}^{REP}, nR_{j,sid}^{REP}\right) + W_{SE} * sc(nP_{i,sid}^{SE}, nR_{j,sid}^{SE}) \tag{20}$$

Suppose A is the attribute of a service provider and B is the attribute of a service request. The function $sc(A, B)$ gives scores from different aspects (suppose the attribute is the benefit attribute: larger is better):

$$sc(A,B) = \begin{cases} \frac{1}{2} + \frac{1}{2} * 1/e^{A-B} & ifA \geq B \\ \frac{1}{2} * A/B & ifA < B \end{cases} \tag{21}$$

Our scheduling target is to maximize:

$$tar_1 = \sum score(P_i^{sid}, R_j) \tag{22}$$

Subject to:

$$nP_{i,sid}^{RT} \leq nR_{j,sid}^{RT} \tag{23}$$

$$nP_{i,sid}^{CT} \leq nR_{j,sid}^{CT} \tag{24}$$

$$nP_{i,sid}^{REL} \geq nR_{j,sid}^{REL} \tag{25}$$

$$nP_{i,sid}^{AV} \geq nR_{j,sid}^{AV} \tag{26}$$

$$nP_{i,sid}^{REP} \geq nR_{j,sid}^{REP} \tag{27}$$

$$nP_{i,sid}^{SE} \geq nR_{j,sid}^{Se} \tag{28}$$

As the analysis in "Energy consumption model for service composition" section, we also have two other targets:

$$\text{Minimization}: tar_2 = \sum E_R(R) + E_S(R) \tag{29}$$

$$tar_3 = \sum E_U(R) \tag{30}$$

tar_2 and tar_3 minimize the energy consumption of users and the energy consumption for transferring files.

In this paper, we only consider six QoSs. However, we can add any QoS to our model, regardless of whether the QoS belongs to the benefit type (such as security) or the cost type (such as execution time).

The service composition method based on GA

Our problem is a multiple-objective problem with multiple conditions. In addition, we try to use NSGA III to solve the problem.

1: Input the initial datasets: tasks, Clouds
2: Normalized data about the response time, cost, reliability, availability, reputation, and security
3: Generate NN initial service composition solutions as the first population and set to S
4: **While** $i \leq M$ do
// when the iteration time is less than 200
5: Calculate the fitness of service composition solutions in S and select TOP NN service composition solutions according to the fitness
6: **While** $j \leq \alpha * NS$
7: Randomly select two service composition solutions as ss_1 and ss_2
8: $ss = ss_1$;
//Selection
9: **If** $rand(0,1) \leq R_{cro}$
//Crossover
10: Update ss_1 through "multipoint crossover " for ss_1 and ss_2 and add new individuals to S;
11: **EndIf**
//Mutation
12: **If** $rand(0,1) \leq R_{mut}$
//Crossover
13: Update ss_1 and ss_2 through the "multipoint " mutation strategy and add new individuals to S;
14: **EndIf**
15: **EndWhile**
16: **EndWhile**
17: Output the solution with the largest fitness in NN service composition solutions

Algorithm1. Service composition GA

In Algorithm 1, NN is the total number of individuals, M is the maximized number of iteractions,R_{cro} and R_{mut} are the rates of crossover and mutation, respectively. NS is the number of atmonic services of all the service requests ($\alpha > 1$). A is a constant Lines $1 \sim 3$ initialize the population with NN individuals. One individual is a service composition solution. Lines $4 \sim 16$ are M times for our GA. The rates of crossover and mutation are R_{cro} and R_{mut}, respectivey. Function $rand(0, 1)$ returns a random number in the range of (0,1). Figure 4 is the flowchart of service composition methods based on GA.

Individuals for scheduling solution

An individual gives the location of every service in a service request. Suppose there are N clouds. We use $\sqrt{N+1}$ genes to denote the location of every service. 0 denotes when the service is executed on the local mobile devices, and others denote that the service is allocated to the relative Clouds.

Fitness function

The fitness function is based on the scheduling targets: tar_1, tar_2 and tar_3. We give weight to every target, and then the scheduling target is maximized:

$$tar = w_1 * tar_1 - w_2 * tar_2 - w_3 * tar_3 \tag{31}$$

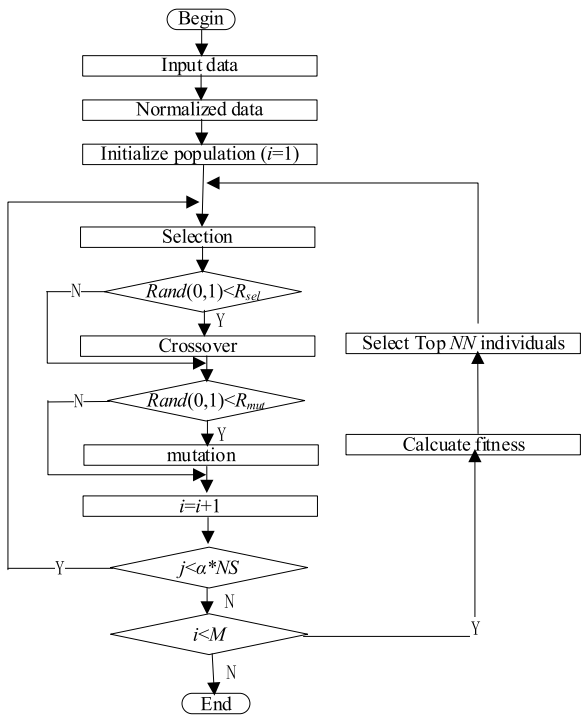

Fig. 4 Flowchart of service composition based on GA

Genetic operators

The genetic operators include selection, crossover, mutations, and updating.

Selection: In the paper, we randomly select two individuals for crossover and mutation.

Crossover: Recombination (plus variation) of biological genes plays a central role in biological evolution in nature. Additionally, the crossover operator of the genetic operations palys a central role in of the genetic algorithm. The so-called crossover refers to the operation of replacing and recombining part of the structure of the two parent individuals to generate a new individual. Through crossover, the searchability of GA can be improved by leaps and bounds. We use the multipoint crossover method for crossover operation in our paper.

Mutation: The basic content of the mutation operator is to change the gene values of individuals in the population. In the simulation, we change from 1 to 0, or change from 0 to 1 for one bit.

Updating: The operator selects superior individuals from a group and eliminates inferior individuals. The selection operator is sometimes referred to as the reproduction operator. The purpose of selection is to directly inherit the optimized individual (or solution) to the next generation or to generate new individuals through pairing and crossover to the next generation. The selection operation is based on the fitness evaluation of individuals in the group. Our selection operator is based on the value of the scheduling target function (Formula (30)). We always select the individuals with bigger values in the fitness function.

GA for service composition

The basic operation process of GA for service composition is as follows: [2]

(1) Initialization: Set the evolutionary time $t = 0$, set the maximum evolutionary time T, and randomly generate M individuals as the initial population $P(0)$.
(2) Individual evaluation: Calculate the fitness of each individual in group $P(t)$ according to the scheduling target (Formula (19)).
(3) Selection operation: The purpose of selection is to directly inherit the optimized individuals to the next generation or to generate new individuals through pairing and crossover and then inherit them to the next generation. The selection operation is based on the fitness evaluation of the individuals in the population. We select the top M individuals in descending order of the target function of individuals.

(4) Crossover operation: The crossover operator is applied to the group. The core function of the GA is the crossover operator. We select two individuals and cross with a point, and we obtain two new individuals. The crossover happens at a rate of R_{cro}.
(5) Mutation operation: The mutation operator is applied to the population. That is, we randomly select some genes in the individual and change their values. After the population $P(t)$ is selected, crossed, and mutated, the next generation population $P(t+1)$ is obtained. The mutation happens at a rate of R_{mut}.
(6) Judgment of termination condition: if $t = T$, the individual with the maximum fitness obtained in the evolution process is used as the output of the optimal solution, and the calculation is terminated.

When the GA is terminated, we select the individual that has the largest value in the target function.

Simulation and comparisons

Simulation environment

In this section, we will compare our method GA-C with All-C [44–46], Bas-C [44–46], Smt-C [44–46], Swm-C [47], and Fast-C [48].

All-C: All clouds containing composite objects are used as input to service composition until the service composition request is completed.

Bas-C: First, we check whether any cloud meets the requirements of the service composition object, if it is, we select the cloud; otherwise, we select any two clouds, check whether it meets the requirements of the service composition object, and if so, select these two clouds until the service composition request is met.

Smt-C: Smt-C models the multicloud environment into a tree structure by cloud, service cluster, and service levels, and realizes service composition by searching for combinations with the minimum number of clouds.

Swm-C [47]: Apply the swarm algorithm to the discrete optimization problem of service composition. In a multicloud environment, consider the QoS of service requests to select clouds and related services.

Fast-C [48]: Fast-C is a heuristic method that considers multiple criteria services selection, and tries to satisfy as many QoS requirements as possible. It takes a global-aware utility cost based on expected compositional QoSs and improves the solution by iterations.

The simulation environment is as follows: we assume that the number of copies of atomic services provided by each web service type is a random integer between [100 ~ 200], and 20 to 80 service composition requests are randomly generated each time. The number of atomic services in all service requests is a random integer, and the maximum number of atomic services is 50. The test environment contains 100 clouds (an atomic service may have multiple atomic service copies in a cloud), and the atomic services contained in these clouds are also generated by random numbers (0 or 1:0 means the cloud does not provide the service, and 1 denotes the cloud provides the service).

We simulated the service composition process 5000 times, and the results in this section are the average of the 5000 execution results. The simulation environment is as follows: operating system WIN10, CPU 8 core 2.6 GHz, 8 GB of memory.

Comparisons and discussion

In this section, we will give comparisons of the six methods from different metrics: ACC (average number of combined Clouds), AVS (average number of visited services), the failure rate of service composition, the energy consumption of users and transferring files, and the average score of atomic services.

Average number of combined clouds and average number of visited services

Two parameters will be compared in this section: ACC (Fig. 5), and AVS (Fig. 6). The larger in the ACC is, the more data need to be transferred between clouds, thus increasing the time and energy consumption for transferring files. The larger the AVS is, the longer the search service time and the longer the algorithm execution time.

GA-C has the smallest ACC value among the five methods (Fig. 5), making it the smallest number of clouds involved. The descending order of the five methods is ALL-C, Bas-C, Smt-C, Swm-C, and GA-C. Compared

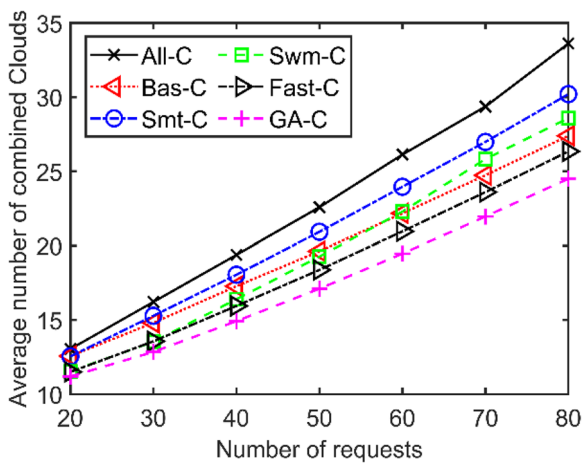

Fig. 5 ACC

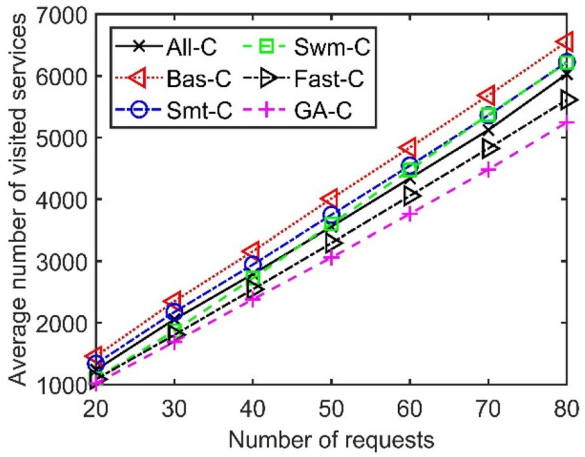

Fig. 6 AVS

with the ACC of ALL-C, Bas-C, Smt-C, Swm-C, and Fast-C, the GA-C average decreases by 31.65%, 13.85%, 21.52%, 12.47%, and 6.39%, respectively. GA-C has the smallest ACC value because: (1) the score makes every service request obtain only the atomic service that meets its QoS requirement and (2) GA ensures obtaining the ideal scheduling solution. Although Swm-C performs better when the number of service requests is small, it does not perform as well as Bas-C and GA-C as service requests increase. Therefore, the Swm-C algorithm is suitable in the case of low complexity. Although the purpose of ALL-C is to ensure that all services are in one cloud, it leads to many fragmented atomic services. It may be that each atomic service of a service request needs to access a cloud, so the ACC value is expected to be the largest. Smt-C may also encounter the same problem, that is, it wants to serve as much as possible in one cloud, but the subsequent subservices need to be provided by multiple clouds, which also leads to a relatively large ACC. The Swm-C algorithm is suitable for the case where the search range is small, but as the number of services increasing, its performance is not good.

Figure 6 shows the AVS for the five methods with different numbers of web service requests. From Fig. 6, we find that GA-C has the smallest AVS value among the five methods, making it the least number of services accessed. The descending order of AVS values is Bas-C, Smt-C, ALL-C, Swm-C, Fast-C and GA-C. Compared with ALL-C, Bas-C, Smt-C, Swm-C, and Fast-C, GA-C reduces the AVS value by 16.25%, 30.19%, 21.98%, 16.86%, and 7.02%, respectively.

In general, GA-C performs well in the two parameters, because the GA-C ensures the three targets, which not only ensures that the number of clouds involved in the service is small, but also ensures that the number of services accessed is small. Although ALL-C has a relatively high ACC value, it also has a relatively low AVS value. The Bas-C and Smt-C methods perform almost the same on the two comparison parameters. Although Swm-C performs well when the number of server requests is small, when the number of service requests increases, the performance is not very good. Fast-C also has good performance in ACC and AVS. It has an advantage in the speed of obtaining the service composition result, at the same time, it loses some efficiency, especially when the number of service requests is large.

Comparisons considering six QoSs

"Average number of combined Clouds and average number of visited services" section does not consider QoSs. Here, we will compare those methods when the relative QoS is a random integer in Table 2. When a QoS is only half (when the QoS belongs to the benefit type) of the requirement, we consider the service composition to fail (for the QoS belonging to the cost type, it is more than 1.5 times the QoS, it is a failure).

We use the method of Sect. 4.3 to normalize these values. Other simulation parameters are the same as in "Average number of combined Clouds and average number of visited services" section.

From Fig. 7, when considering QoSs, our proposed method reduces the probability of combination failure. Compared with the FRs of ALL-C, Bas-C, Smt-C, Swm-C, and Fast-C, the average FR of GA-C decreases by 17.13%, 36.09%, 30.57%, 19.70%, and 7.42%, respectively.

Energy consumption of different methods

Suppose the file size of each subservice is a random number between [1,100]. According to the relevant literature, the sending rate is much higher than the receiving rate, so it is assumed that the receiving efficiency (energy consumption per unit size file) is a random number between [1, 50], and the sending efficiency (energy consumption per unit size file) is a random number in [50,150]. We mainly examine the total energy consumption of transfer files and their size (Total file size). Since one sends and receives in one direction, so we only consider the sending file size [56] (we do not consider the size of receiving files).

Figure 8 are the file sizes transmitted by the five methods under different service request numbers, and Fig. 9 shows the energy consumption of the files transmitted by the five methods under different service request numbers. Overall, energy consumption is proportional to the size of transferring files. On average, compared with the size of transferring files of ALL-C, Bas-C, Smt-C, Swm-C, and Fast-C, GA-C average reduces by 34.50%, 13.66%, 24.90%, 13.93%, and 7.28%, respectively; compared with

Table 2 Values of six QoSs

Attributes	Scope	Attributes	Scope
Cost	[1, 10]	Availability	[1, 10]
Response time	[1, 10]	Reputation	[1, 5]
Reliability	[1, 5]	Security	[1, 10]

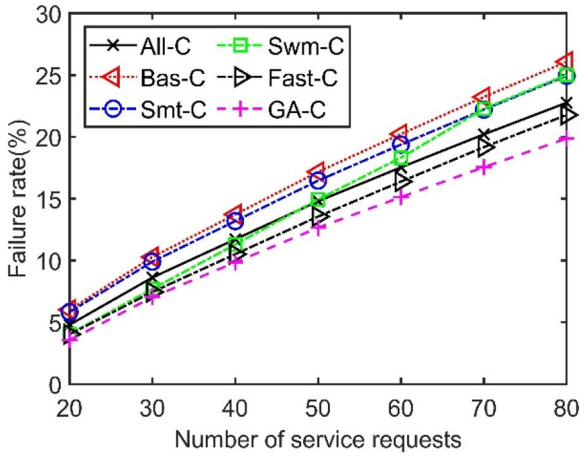

Fig. 7 Failure rate of different methods

energy consumption, GA-C average reduces by 35.43%, 13.75%, 25.88%, 14.72%, and 7.52%, respectively.

Energy consumption of different methods

Figure 10 is the energy consumption of users. As the number of service requests increases, all methods slightly improve in the energy consumption of users. This is because, as the number increases, every atom has fewer choices to obtain atomic services with lower energy

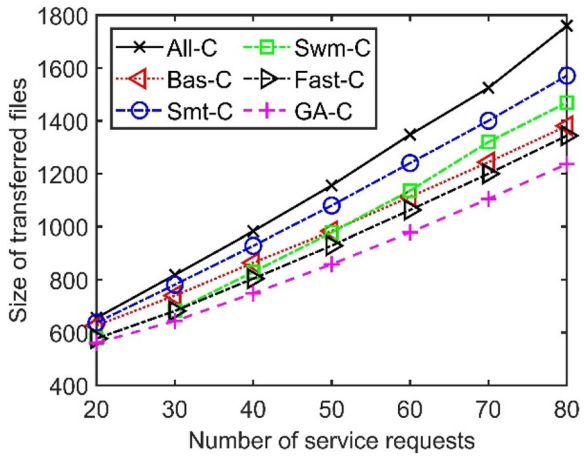

Fig. 8 File size of different methods

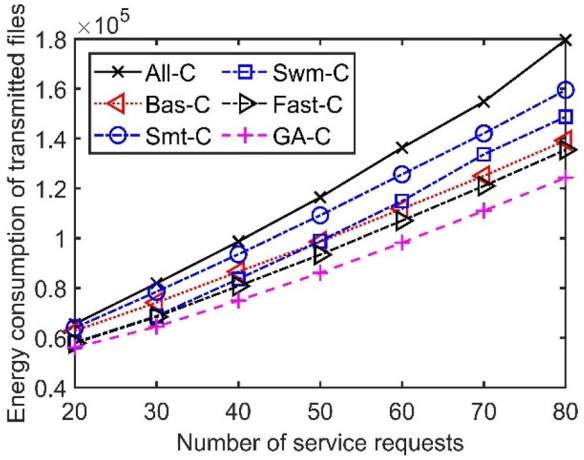

Fig. 9 Energy consumption

consumption. For the energy consumption of users of ALL-C, Bas-C, Smt-C, Swm-C, and Fast-C, the GA-C average decreases by 120.17, 46.11, 168.48, 88.26, and 60.15 respectively.

Average scores of different methods

Figure 11 gives the average score of different methods under different numbers of service requests. In general, all methods have a decreasing trend in the value of the average score with the enhancement of the number of service requests. This is because, with the increase in service requests, fewer atomic services are available for the other service requests, thus decreasing the scores. Our proposed method GA-C always has the highest value in the average score. Compared to the average score of ALL-C, Bas-C, Smt-C, Swm-C, and Fast-C, the GA-C average improves by 31.63%,15.55%, 8.27%, 16.94%, and 8.69%.

All-C and Bas-C both have good performance when the system has a lower load. With the enhancement of

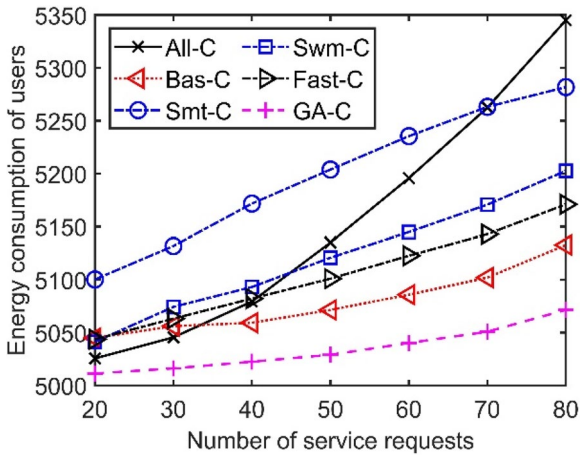

Fig. 10 Energy consumption of users

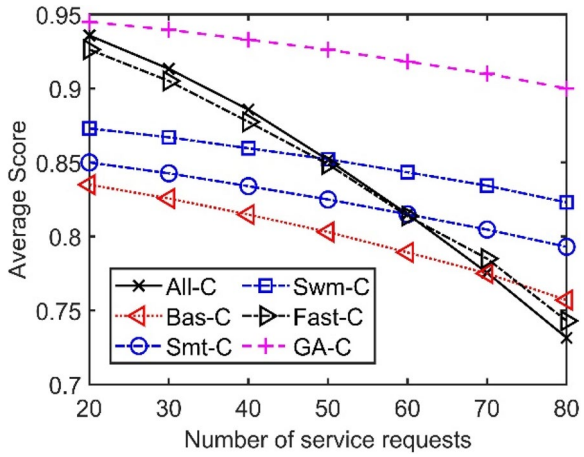

Fig. 11 Average score of different methods

the system load, they have poor performance in all metrics. They both try to ensure good performance of current tasks, and ingnore the influence on future tasks. Smt-C losts its efficiency to search atomic services in the tree structure when the number of services and the number of QoSs is larger. The swarm algorithm in our scheduling cannot find a better solution. Fast-C may have a good speed and efficiency when the system has a lower load, and with the increase of the system load, it lost its efficiency because of its speed. Overall, our proposed algorithm considers reducing the number of access clouds and the number of search services, and has obvious advantages in reducing the probability of task failure and reducing energy consumption.

Conclusions and future work

In this paper, based on the analysis of the multicloud environment, we first standardize the QoS of the service; then we analyze the goal of service composition in the multicloud environment; and last, we use the genetic algorithm to complete the service composition problem in the multicloud environment. Simulation experiments show that our proposed service composition method can reduce the number of service composition accesses to the cloud, and reduce the energy consumption of data transmission between different clouds, including reducing user energy consumption. Energy consumption would be reduced if we would obtain a some green energy supply, such as solar, wind, etc. In future work, we will try to find a service composition method when some service providers or users have a green energy supply.

Acknowledgements
This work was supported in part by the Science and Technology Planning Project of Fujian Province (No. 2019J05123, No. 2020H0023, 2020Y9064), and the Joint Funds of 5th Round of Health and Education Research Program of Fujian Province (No. 2019-WJ-41).

Authors' contributions
Jianmin Li wrote the paper and gave the simulation and the main framework of the paper. Shunzhi Zhu gave support for the paper and revised the paper. The author(s) read and approved the final manuscript.

Funding
None.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
None.

Received: 4 September 2022 Accepted: 12 March 2023
Published online: 22 March 2023

References

1. Yang B, Wang S, Li S, Bi F (2022) Digital thread-driven proactive and reactive service composition for Cloud Manufacturing. IEEE Trans Ind Informatics 3203:1–10. https://doi.org/10.1109/TII.2022.3171338
2. Huang J, Duan Q, Guo S, Yan Y, Yu S (2018) Converged Network-Cloud Service Composition with End-to-End Performance Guarantee. IEEE Trans Cloud Comput 6(2):545–557. https://doi.org/10.1109/TCC.2015.2491939
3. Chung JY, Bichler M (2005) Service-oriented enterprise applications and Web service composition. Inf Syst E-bus Manag 3(2):101–102. https://doi.org/10.1007/s10257-005-0051-0
4. Ekie J, Gueye B, Niang I, Ekie T (2021) A survey on QoS-based service composition in Cloud system environment. Proc IEEE Int Conf Softw Eng Serv Sci. 2021:203–210. https://doi.org/10.1109/ICSESS52187.2021.9522246
5. Razian M, Fathian M, Bahsoon R, Toosi AN, Buyya R (2022) Service composition in dynamic environments: A systematic review and future directions. J Syst Softw. 188:111290. https://doi.org/10.1016/j.jss.2022.111290
6. Sefati S, Navimipour NJ (2021) A QoS-Aware Service Composition Mechanism in the Internet of Things Using a Hidden-Markov-Model-Based Optimization Algorithm. IEEE Internet Things J 8(20):15620–15627. https://doi.org/10.1109/JIOT.2021.3074499
7. Wang F, Zhang L, Laili Y (2022) Robotics and Computer-Integrated Manufacturing Multi-granularity service composition in industrial cloud robotics. Robot Comput Integr Manuf. 78:102414. https://doi.org/10.1016/j.rcim.2022.102414
8. Liu L, Zhu H, Chen S, Huang Z (2022) Privacy regulation aware service selection for multi-provision cloud service composition. Futur Gener Comput Syst 126:263–278. https://doi.org/10.1016/j.future.2021.08.010
9. Li J, Zhong Y, Zhu S, Hao Y (2022) Energy-aware service composition in multi-Cloud. J. King Saud Univ Comput Inf Sci. 34:3959–3967. https://doi.org/10.1016/j.jksuci.2022.04.014
10. Lim MK, Xiong W, Wang Y (2022) A three-tier programming model for service composition and optimal selection in cloud manufacturing. Comput Ind Eng. 167:108006. https://doi.org/10.1016/j.cie.2022.108006
11 Ying C, Chow AHF, Nguyen HTM, Chin K-S (2022) Multi-agent deep reinforcement learning for adaptive coordinated metro service operations with flexible train composition. Transp Res Part B Methodol. 161:36–59. https://doi.org/10.1016/j.trb.2022.05.001
12. Wang P, Liu X, Chen J, Zhan Y, Jin Z (2018) "Poster: QoS-Aware service composition using blockchain-based smart contracts." Proc Int Conf Softw Eng. 296–297. https://doi.org/10.1145/3183440.3194978.
13. Xu X, Wang X, Xu H, Wang Z (2021) "Distributed Service Composition in Internet of Services." Proc 2021 IEEE Int Conf Serv Comput SCC. 274–284. https://doi.org/10.1109/SCC53864.2021.00040.

14. Zhang Y, Cui G, Deng S, Chen F, Wang Y, He Q (2021) Efficient Query of Quality Correlation for Service Composition. IEEE Trans Serv Comput 14(3):695–709. https://doi.org/10.1109/TSC.2018.2830773
15. Tondeur J, Van Braak J, Siddiq F, Scherer R (2016) Time for a new approach to prepare future teachers for educational technology use: Its meaning and measurement. Comput Educ 94:134–150. https://doi.org/10.1016/j.compedu.2015.11.009
16. Zeng K, Paik I (2021) Semantic Service Clustering with Lightweight BERT-Based Service Embedding Using Invocation Sequences. IEEE Access 9:54298–54309. https://doi.org/10.1109/ACCESS.2021.3069509
17 Xie N, Tan W, Zheng X, Zhao L, Huang L, Sun Y (2021) An efficient two-phase approach for reliable collaboration-aware service composition in cloud manufacturing. J. Ind. Inf. Integr. 23:100211. https://doi.org/10.1016/j.jii.2021.100211
18. Palade A, Clarke S (2018) "Stigmergy-based qos optimisation for flexible service composition in mobile communities." Proc 2018 IEEE World Congr Serv Serv. 29–30. https://doi.org/10.1109/SERVICES.2018.00027.
19. Hollauf FS, Franceschetti M, Eder J (2021) "Towards Representing Time-Cost Tradeoffs for Service Compositions." Proc - 2021 IEEE Int Conf Serv Comput SCC. 79–88. https://doi.org/10.1109/SCC53864.2021.00020.
20. Shi H, Xu H, Xu X, Wang Z (2021) "Service Composition Considering QoS Fluctuations and Anchoring Cost." Proc 2021 IEEE Int Conf Web Serv ICWS 2021. 370–380. https://doi.org/10.1109/ICWS53863.2021.00056.
21. Wang Z, Cheng B, Zhang W, Chen J (2020) "QoS-aware automatic service composition based on service execution timeline with multi-objective optimization." Proc 2020 IEEE 13th Int Conf Serv Comput SCC 2020. 296–303. https://doi.org/10.1109/SCC49832.2020.00046.
22. Vallejos S, da Rocha Araujo L, Rodríguez G, Berdun L, Toscani R (2020) Preference-based AI planning for web service composition. IEEE Lat Am Trans. 18:1987–1995. https://doi.org/10.1109/TLA.2020.9398640
23. Wang Y, Dong Y, Guo S, Yang Y, Liao X (2020) Latency-Aware Adaptive Video Summarization for Mobile Edge Clouds. IEEE Trans Multimed 22(5):1193–1207. https://doi.org/10.1109/TMM.2019.2939753
24. Xie N (2018) Interval grey number based project scheduling model and algorithm. Grey Syst Theory Appl 8(1):100–109. https://doi.org/10.1108/gs-11-2017-0035
25. Wu S, Mei C, Jin H, Wang D (2018) Android Unikernel: Gearing mobile code offloading towards edge computing. Futur Gener Comput Syst 86:694–703. https://doi.org/10.1016/j.future.2018.04.069
26. Liao LX, Chao HC, Chen MY (2020) Intelligently modeling, detecting, and scheduling elephant flows in software defined energy cloud: A survey. J Parallel Distrib Comput 146:64–78. https://doi.org/10.1016/j.jpdc.2020.07.008
27. Bashari M, Bagheri E, Du W (2018) Self-adaptation of service compositions through product line reconfiguration. J Syst Softw 144(May):84–105. https://doi.org/10.1016/j.jss.2018.05.069
28. Thai L, Varghese B, Barker A (2018) A survey and taxonomy of resource optimisation for executing bag-of-task applications on public clouds. Futur Gener Comput Syst 82:1–11. https://doi.org/10.1016/j.future.2017.11.038
29. Schmid M, Kroeger R (2008) "Decentralised QoS-management in service oriented architectures," Lect. Notes Comput. Sci. (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics). 5053:44–57. https://doi.org/10.1007/978-3-540-68642-2_4.
30. Li X et al (2019) A Novel Workflow-Level Data Placement Strategy for Data-Sharing Scientific Cloud Workflows. IEEE Trans Serv Comput 12(3):370–383. https://doi.org/10.1109/TSC.2016.2625247
31. Casati F, Ilnicki S, Jin LJ, Krishnamoorthy V, Shan MC (2000) "eFlow: A platform for developing and managing composite e-services," Proc. - Acad. Work. Conf. Res. Challenges 2000 Next Gener. Enterp. Virtual Organ. Mobile/Pervasive Technol AIWORC 2000. 341–348. https://doi.org/10.1109/AIWORC.2000.843314.
32. Lin YD, Chu ETH, Lai YC, Huang TJ (2015) Time-and-energy-aware computation offloading in handheld devices to coprocessors and clouds. IEEE Syst J 9(2):393–405. https://doi.org/10.1109/JSYST.2013.2289556
33. Masdari M, Nozad Bonab M, Ozdemir S (2021) QoS-driven metaheuristic service composition schemes: a comprehensive overview, vol. 54, no. 5. Springer, Netherlands
34. Boucetti R, Hemam SM, Hioual O (2022) "An approach based on genetic algorithms and neural networks for QoS-aware IoT services composition." J King Saud Univ Comput Inf Sci. xxxx. https://doi.org/10.1016/j.jksuci.2022.02.012.
35. Chattopadhyay S, Banerjee A (2020) QoS Constrained Large Scale Web Service Composition Using Abstraction Refinement. IEEE Trans Serv Comput 13(3):529–544. https://doi.org/10.1109/TSC.2017.2707548
36. Hao Y, Cao J, Wang Q, Du J (2021) Energy-aware scheduling in edge computing with a clustering method. Futur Gener Comput Syst 117:259–272. https://doi.org/10.1016/j.future.2020.11.029
37. Hao Y, Wang Q, Cao J, Ma T, Du J, Zhang X (2022) "Interval grey number of energy consumption helps task offloading in the mobile environment." ICT Express. xxxx. https://doi.org/10.1016/j.icte.2022.03.005.
38. Hao Y, Cao J, Wang Q, Ma T (2021) Energy-aware offloading based on priority in mobile cloud computing. Sustain Comput Informatics Syst. 31:100563. https://doi.org/10.1016/j.suscom.2021.100563
39. Kendrick P, Baker T, Maamar Z, Hussain A, Buyya R, Al-Jumeily D (2018) An Efficient Multi-Cloud Service Composition Using a Distributed Multiagent-Based, Memory-Driven Approach. IEEE Trans Sustain Comput 6(3):358–369. https://doi.org/10.1109/tsusc.2018.2881416
40. Lahmar F, Mezni H (2021) Security-aware multi-cloud service composition by exploiting rough sets and fuzzy FCA. Soft Comput 25(7):5173–5197. https://doi.org/10.1007/s00500-020-05519-x
41. Kritikos K, Plexousakis D (2015) "Multi-cloud Application Design through Cloud Service Composition." Proc 2015 IEEE 8th Int Conf Cloud Comput CLOUD 2015. 686–693. https://doi.org/10.1109/CLOUD.2015.96.
42. Pang B, Hao F, Yang Y, Park DS (2020) An efficient approach for multi-user multi-cloud service composition in human–land sustainable computational systems. J Supercomput 76(7):5442–5459. https://doi.org/10.1007/s11227-019-03140-w
43. Souri A, Rahmani AM, Navimipour NJ, Rezaei R (2020) A hybrid formal verification approach for QoS-aware multi-cloud service composition. Cluster Comput 23(4):2453–2470. https://doi.org/10.1007/s10586-019-03018-9
44. Zou G, Chen Y, Xiang Y, Huang R, Xu Y (2010) "AI Planning and Combinatorial Optimization for Web Service Composition in Cloud Computing." 28–35. https://doi.org/10.5176/978-981-08-5837-7_166.
45. Yu Q, Chen L, Li B, Li J (2015) Ant colony optimization applied to web service compositions in cloud computing. Comput Electr Eng. 41:18–27. https://doi.org/10.1016/j.compeleceng.2014.12.004
46 Kurdi H, Al-Anazi A, Campbell C, Al Faries A (2015) A combinatorial optimization algorithm for multiple cloud service composition. Comput Electr Eng. 42:107–113. https://doi.org/10.1016/j.compeleceng.2014.11.002
47. Safaei A, Nassiri R, Rahmani AM (2022) Enterprise service composition models in IoT context: solutions comparison. J Supercomput 78(2):2015–2042. https://doi.org/10.1007/s11227-021-03873-7
48. Kurdija AS, Silic M, Delac G, Vladimir K (2022) Fast Multi-Criteria Service Selection for Multi-User Composite Applications. IEEE Trans Serv Comput 15(1):174–187. https://doi.org/10.1109/TSC.2019.2925614
49. Wang S, Zhou A, Bao R, Wu C (2021) "Towards Green Service Composition Approach in the Cloud." Proc 2021 IEEE World Congr Serv Serv 2021. 14. https://doi.org/10.1109/SERVICES51467.2021.00025.
50. Wang S, Zhou A, Bao R, Wu C (2021) Towards Green Service Composition Approach in the Cloud. Proc 2021 IEEE World Congr Serv Serv 14:14. https://doi.org/10.1109/SERVICES51467.2021.00025
51. Zeng D, Gu L, Yao H (2020) Towards energy efficient service composition in green energy powered Cyber-Physical Fog Systems. Futur Gener Comput Syst 105:757–765. https://doi.org/10.1016/j.future.2018.01.060
52. Ibrahim GJ, Rashid TA, Akinsolu MO (2020) An energy efficient service composition mechanism using a hybrid meta-heuristic algorithm in a mobile cloud environment. J Parallel Distrib Comput 143:77–87. https://doi.org/10.1016/j.jpdc.2020.05.002
53. Tong E et al (2021) A Hierarchical Energy-Efficient Service Selection Approach with QoS Constraints for Internet of Things. IEEE Trans Green Commun Netw 5(2):645–657. https://doi.org/10.1109/TGCN.2021.3069121
54. Deng S, Wu H, Tan W, Xiang Z, Wu Z (2017) Mobile Service Selection for Composition: An Energy Consumption Perspective. IEEE Trans Autom Sci Eng 14(3):1478–1490. https://doi.org/10.1109/TASE.2015.2438020

Publisher's Note