IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Optimization of microservice composition based on artificial immune algorithm considering fuzziness and user preference

Ming Gao[1,2], Mingxia Chen[1], An Liu[1], WAI HUNG IP[3,4] (Senior Member, IEEE), AND KAI LEUNG Yung[3]

[1] School of Management Science and Engineering, Dongbei University of Finance and Economics, China
[2] Center for Post-doctoral Studies of Computer Science, Northeastern University, China
[3]Department of Industrial and System Engineering, The Hong Kong Polytechnic University, Hong Kong SAR
[4]University of Saskatchewan, Canada

Corresponding author: Ming Gao (e-mail: gm@dufe.edu.cn).

This work is supported by the National Natural Science Foundation of China (Grant 71772033, 71831003, 71672023), Scientific and Research Funds of Education Department of Liaoning Province (LN2019Q14), and also supported by a grant from the Department of Industrial and Systems Engineering of the Hong Kong Polytechnic University (H-ZG3K).

ABSTRACT Microservices is a new paradigm in cloud computing that separates traditional monolithic applications into groups of services. These individual services may correlate or cross multi-clouds. Compared to a monolithic architecture, microservices are faster to develop, easier to deploy, and maintain by leveraging modern containers or other lightweight virtualization. To satisfy the requirements of end-users and preferences, appropriate microservices must be selected to compose complicated workflows or processes from within a large space of candidate services. The microservice composition should consider several factors, such as user preference, correlation effects, and fuzziness. Due to this problem being NP-hard, an efficient metaheuristic algorithm to solve large-scale microservice compositions is essential. We describe a microservice composition problem for multi-cloud environments that considers service grouping relations and corresponding correlation effects of the service providers within intra- or inter-clouds. We use the triangular fuzzy number to describe the uncertainty of QoS attributes, the improved fuzzy analytic hierarchy process to calculate multi-attribute QoS, construct fuzzy weights related to user preferences, and transform the multi-optimal problem into a single-optimal problem. We propose a new artificial immune algorithm based on the immune memory clone and clone selection algorithms. We also introduce several optimal strategies and conduct numerical experiments to verify effects and efficiencies. Our proposed method combines the advantages of monoclone, multi-clone, and co-evolution, which are suitable for the large-scale problems addressed in this paper.

INDEX TERMS Fuzzy analytic hierarchy process (FAHP), microservice, microservice group, quality of service (QoS), QoS attribute, the Parallel Cooperative Short-term Memory Injection Multi-Clone Clonal Selection Algorithm (ParaCoSIMCSA).

I. INTRODUCTION

With the development of cloud computing infrastructures along with big data and edge computing, many applications for users must now be quickly deployed in the cloud. However, user requirements are widely varied, and application structures can be complex that may involve the invocation and integration of different services. Traditional monolithic architectures feature simple frameworks and low upfront costs. Nevertheless, reusability is low, granularity is coarse, and the configuration and management of quality of service (QoS) lack flexibility when supporting complex processes comprising massive applications.

In comparison, microservice architectures are more flexible, as they can be independently extended and deployed with easier maintenance by leveraging modern containers or other lightweight virtualizations [1]. Microservice

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

architectures refer to the design of a single, user-facing application through the combination of smaller, individually functional services each running distinct processes [2]. Microservices then become well-suited for large-scale and complex business projects. Therefore, more enterprises concentrate today on building microservice architectures, including Amazon, Netflix, and LinkedIn [3].

With comprehensive applications of the cloud, deploying microservices across multiple clouds becomes a significant challenge. Microservices requires to be hosted on the cloud that can meet the needs of users and microservice to maximize the effect that cover QoS [4]. Consequently, in the process of deploying microservices in multi-clouds, allocation and coordination are significant. From this idea, we propose the concept of microservice composition, a challenge that originates from the web composition problem and intends to satisfy the requirements of end-users and their preferences by selecting appropriate microservices to comprise complicated workflows or processes from a combination of available services. Microservice composition should consider several factors, such as user preference, correlation effects, and fuzziness, and can be measured by QoS.

In a multi-cloud environment, the microservice combination must satisfy a variety of QoS of requirements at the process level and takes the uncertainty of preferences of users into account. QoS is usually described by attributes [5], the evaluation of which depends on the subjective and usually uncertain judgement of users. We introduce the triangular fuzzy number (TFN) to describe this uncertainty, and the improved fuzzy analytic hierarchy process (FAHP) to calculate the values of the attributes such as price, reliability, and response time. We propose a fuzzy weight so that users can obtain the weights of attributes by sorting preferences. This paper focuses on optimising multiple attributes and combines these with a fuzzy characterization of the QoS based on user preferences. We do this by modelling the correlation of microservices within intra- or inter-clouds and consider microservice compositions based on end-to-end user preference perception. In addition, as it is an NP-hard problem that considers multi-clouds, the correlation between microservices and user preference, a meta-heuristic algorithm is needed to provide an efficient and powerful ability to search for results. For the service composition problem, an artificial immune algorithm offers more satisfying results in terms of efficiency and feasibility compared to a genetic algorithm [6].

The remainder of this paper is organized as follows. Section II reviews the literature. Section III defines the microservice composition problem, including a formal description, the calculation of the values for the attributes of QoS, and the method of calculating the fuzzy QoS weight of user preference with the triangular fuzzy number. Section IV introduces the improvements observed with the immune memory clonal algorithm and clonal selection algorithm, and the proposes the Parallel Cooperative Short-term Memory Injection Multi-clone Clonal Selection Algorithm (ParaCoSIMCSA). Section VI describes and analyzes the result of these experiments, and Section VII includes the appendix.

II. Related work

Selecting individual microservices to combine into a certain logical order constitutes a microservice combination that should meet the needs of users. The idea of a microservice combination arises from the combination of web services. While there exists little research on issues related to microservice composition, the research field of web service composition is relatively mature. Therefore, this section introduces web service composition work to support our ideas on microservice combination. The available research in the field of microservices is also discussed.

Sun et al. [7] and Ma et al. [8] proposed a method to assign weights to the QoS attributes for solving deviation in their measurement. Fang et al. [9] established an interval QoS model to describe dynamic changes. Zhuang et al. [10] incorporated environmental factors and proposed a QoS monitoring method for a web service based on a weighted naïve Bayesian algorithm. Xu et al. [11] proposed a cooperative filtering recommendation algorithm for web services based on the preferences of users. Zhang et al. [12] proposed a multi-time sequential QoS prediction method, called MulA-LMRBF, to facilitate users in selecting web services. Tan et al. [13] established a multi-objective model that minimised network delays and total cost and improved a particle swarm optimization.

Web service combinations with cooperation is an area that some scholars have performed relevant researches. Mao et al. [14] and Zou et al. [15] considered a cooperative filtering approach to obtain more accurate predictions of web services. Tan et al. [16] developed an improved self-organizing neural network web service composition method, and Yao et al. [17] combined cooperative filtering with content recommendations.

With the development of web services, accurately predicting QoS values to ensure a higher quality is also a concern of scholars. Ma et al. [18] proposed a prediction algorithm to solve the limitations due to unknown QoS values of existing cooperative filtering algorithms. Xu et al. [19] proposed a reputation-based matrix decomposition to predict the QoS value of an unknown web service. Chen et al. [20] designed the similarity model, JacMinMax, driven by the QoS data range, where two neighbourhood selection strategies were proposed to obtain a predictive QoS system.

Wang et al. [21] proposed a combined model of QoS, the cloud environment, and web service composition based on a genetic algorithm. Liu et al. [22] and Gopal N. Rai et al. [23] studied the composability between service combinations.

In researching the web service composition problem, improvement of the algorithm is crucial. Lu and Kou [24] considered that the QoS attributes were multidimensional and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

contradictory. They proposed a new algorithm by combining multi-attribute decision-making with a genetic algorithm. Shen et al. [25] established a logistics web service composition model based on QoS perception and proposed a new genetic optimization algorithm. Huang and Sun [26] proposed a service selection algorithm based on an improved binary particle swarm optimization (BPSO) to increase efficiency while also optimising the web service composition. Tan et al. [27] proposed a chaotic genetic algorithm to improve QoS quality. Xia et al. [28] proposed an improved ant colony algorithm based on the established service composition model, which could converge to the optimal solution quickly. Liu and Yang [29] proposed a multi-objective genetic algorithm based on the multi-level service deployment problem, which contains a strategy for local search and mutation.

Halfaoui et al. [30] proposed a self-organized migration algorithm for selecting a web service as well as a fuzzy Pareto advantage to improve the algorithm and validate its superiority. Xu et al. [31] proposed a chaotic turbo particle swarm optimization algorithm based on a predatory search to obtain candidate services. Ghobaei-Arani et al. [32] and Dahan et al. [33] proposed the cuckoo algorithm and the artificial bee colony algorithm to search for the best combination of web services.

Compared with a web service, a microservice runs as a separate process that is easier to deploy, develop, and maintain. Recently, scholars have also performed significant research on microservices. Bao et al. [34] put forward a performance model and a heuristic scheduling algorithm based on a microservice to improve the scalability of applications and reduce cost. Lin et al. [35] combined a microservice and container, established a scheduling model and improved resource utilization and other factors into the ant colony algorithm to optimise the microservice scheduling process. Wan et al. [36] and Guerrero et al. [37],[38] built optimization models and optimized application deployments through algorithms for microservice deployment and containers.

Jindal et al. [39] designed a regression model to fit the loads that could be supported by microservices under different configurations after an evaluation to optimise its performance. Pietrantuono et al. [40] considered from the perspective of reliability how to design an adaptive reliability testing algorithm for microservices to evaluate and optimise. Singh and Sateesh [41] and Villamizar et al. [42] compared the performance between monolithic and microservice architectures through experiments with applications and verified the advantages of a microservice architecture.

Much of this research only considered web services or microservices while ignoring the characterization of QoS attributes. We consider dynamic selection and optimal configuration of candidate microservices in a microservice composition problem. We use TFN to calculate the values of QoS attributes and propose the concept of a logical microservice group to represent a microservice pool that processes different microservices from the same microservice packet. This paper is based on the research of Gao [43], which focused on workflow modelling and optimization. Here, the service group, fuzzy calculation, and algorithm are applicable to the current problem, so extend the work based on the previous research.

III. Microservice composition problem description

More scholars and business professionals today focus on applying microservices in cloud computing, but selecting and combining from available microservices is an NP-hard problem. In this paper, we propose the concept of a microservice group. Different types of microservices are organized together, and then these microservices are combined and constructed to meet the quality requirements of users, which describes the microservice composition problem. Figure 1 shows how microservices are grouped and run, and the following provides a formal

A. Definition of the microservice composition problem

Definition 1: A 5-tuple is used to represent the attributes, $QoS = (P, T, Rel, CO, UC)$. For each attribute, TFN corresponds to the minimum value, average value, and maximum value of the attributes, respectively. The definitions for each attribute are listed in the following.

- Price (P). The service cost paid by a service requester.
- Response time (T). The time cost of a unit data or calculation from the initial request of users to the completion of the final service execution.
- Reliability (Rel). The probability that a service request is executed correctly before a timeout.
- Correlation (Co). The possible correlated effects of each inner or outer microservice.
- User customized service attributes (UC). The relevant attributes associated with a microservice, such as service accuracy and security.

Definition 2: $SP = \{sp_1, sp_2 \ldots sp_h\}$ represents the collection of service providers in a correlative workflow across multiple entities of the organization, where h is the number of service providers.

Definition 3: The correlated weight matrix of the service providers is expressed as

$$SP_{cow} = \begin{bmatrix} \mathrm{cow}_{1,1} & \cdots & \mathrm{cow}_{1,\mathrm{SPn}} \\ \vdots & \ddots & \vdots \\ \mathrm{cow}_{\mathrm{SPn},1} & \cdots & \mathrm{cow}_{\mathrm{SPn},\mathrm{SPn}} \end{bmatrix} \quad (1)$$

where $cow_{i,j} = \{-1, 0, +1\}$ represents the weight of the internal service and the external correlation of providers, respectively. $cow_{i,j} = -1$ indicates that there is a negative correlation between sp_i and sp_j, $cow_{i,j} = 0$ indicates that there is no correlation between sp_i and sp_j and $cow_{i,j} = 1$, indicates that there is a positive correlation between sp_i and sp_j.

Definition 4: $SO = \{(so_1, so_{qos_1}, so_{sp_1}), (so_2, so_{qos_2}, so_{sp_2}), \ldots, (so_m, so_{qos_m}, so_{sp_m})\}$ represents the binding objects of candidate tasks of a service node in a multi-entity correlated workflow across an organization, where m is the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

number of candidates. so_i represents the identification of the service object, and $so_{qos_i} = (Q_{P_i}, Q_{T_i}, Q_{Rel_i}, Q_{Co_i}, Q_{UC_i})$ represents the attributes of QoS where each component is TFN. so_{sp_i} represents the service providers corresponding to a service object, and $so_{sp_i} \in SP$.

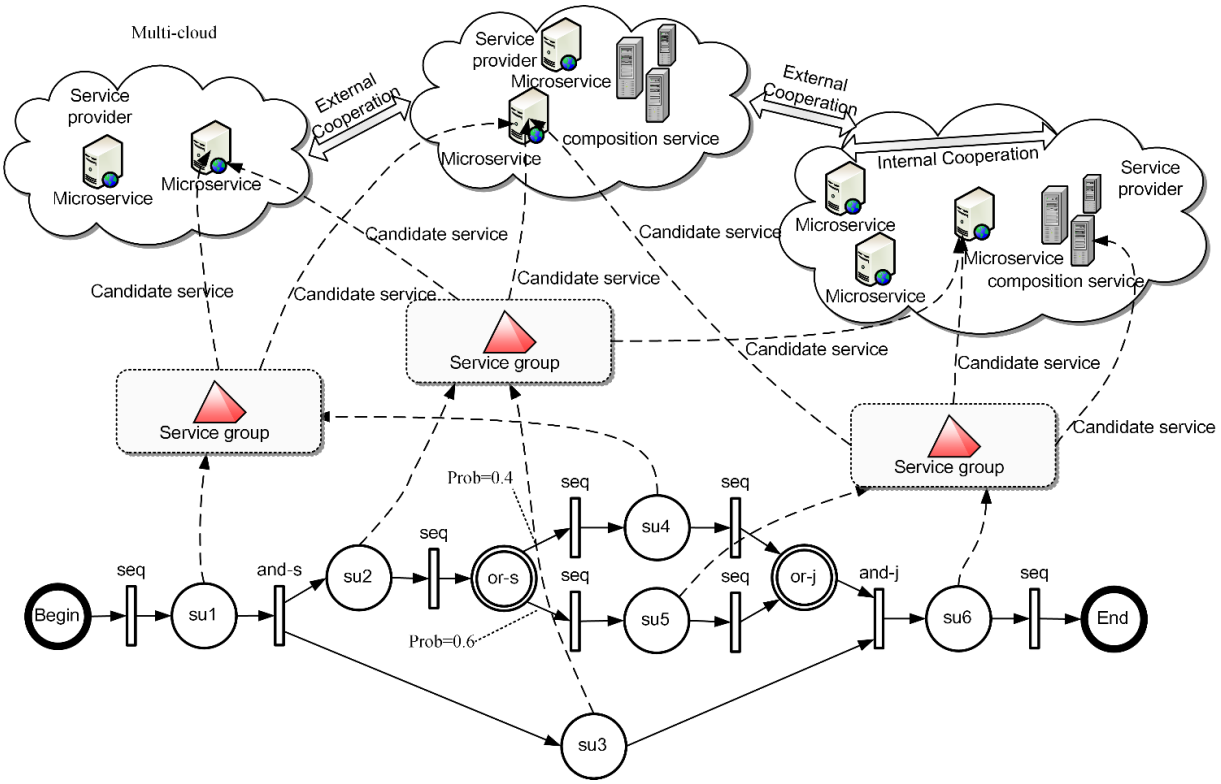


FIGURE 1. **Service grouping and correlation within intra- or inner-clouds of microservice composition problem.**

Definition 5: Define the weight matrix of the QoS attributes involved in the service composition as the following matrix where each row vector corresponds to the weight of the TFN of the QoS attribute.

$$QoS_w = \begin{bmatrix} \mathrm{qw}_{P,l} & \mathrm{qw}_{P,m} & \mathrm{qw}_{P,u} \\ \mathrm{qw}_{T,l} & \mathrm{qw}_{T,m} & \mathrm{qw}_{T,u} \\ \mathrm{qw}_{Rel,l} & \mathrm{qw}_{Rel,m} & \mathrm{qw}_{Rel,u} \\ \mathrm{qw}_{Co,l} & \mathrm{qw}_{Co,m} & \mathrm{qw}_{Co,u} \\ \mathrm{qw}_{UC,l} & \mathrm{qw}_{UC,m} & \mathrm{qw}_{UC,u} \end{bmatrix} \quad (2)$$

Definition 6: $R = \{(r_1, r_{sos_1}), (r_2, r_{sos_2}), \ldots, (r_n, r_{sos_n})\}$ represents the identification of the service group corresponding to the service node in the multi-entity correlated workflow across the organization where n is the number of the service group. $r_{sos_i} = \{so_1, so_2, so_{gi}\}$ is the identification set of the candidate service objects of the service group, and g represents the number of candidate service objects of r_i.

Definition 7: $BP = (p, nf, flo)$ represents an example of a process to be bound in a correlated workflow of multiple entities across an organization, as illustrated in Figure 2.

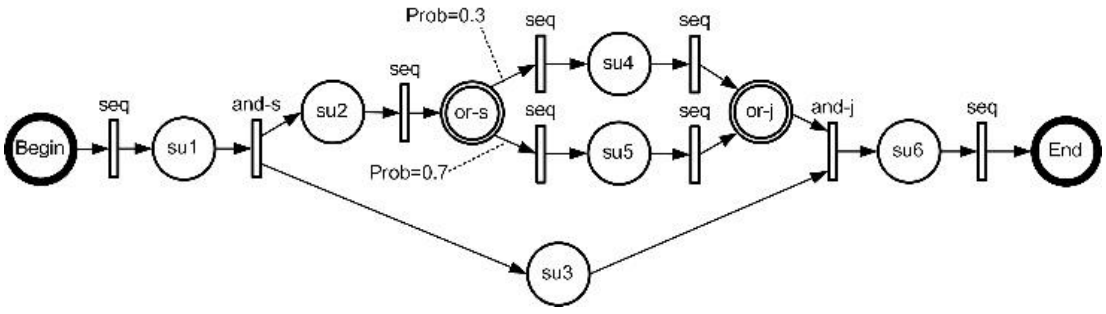


FIGURE 2. **Sample process for service composition optimization.**

$p = \{(n_1, n_{r_1}, n_{p_1}), (n_2, n_{r_2}, n_{p_2}), \ldots, (n_v, n_{r_v}, n_{p_v})\}$ is the triple comprised of each service node identification, the identification of the corresponding service group, and the occurrence probability of the service node. v is the number of service nodes of the process instance to be bound, and n_{v_i} represents the service group corresponding to the service node, where $n_i \in$R. n_{p_i} is the probability of occurrence of the service node.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

The occurrence probability and the parent probability of the nested structure of each service node executing sequentially or serially, in parallel, or cyclically are equal to the parent probability of the structure. If there does not exist nesting in any parent structure, then they are equal to 1.

In the selective or branch execution structure, the probability of the occurrence and the parent probability of the nested structure are equal to the parent probability of the structure multiplied by the selective or branch probability from experience or statistics.

nf is a multi-value function described as $\{n_i\} \rightarrow 2^{\{so_j\}}$ where $2^{\{so_j\}}$ is the power set of candidate service objects from all nodes of the process instance to be bound. This function represents a mapping of a service node of the process instance to the effective identification set of candidate service objects of this node. The implementation steps of this function include the following.

- For n_i, the service node in N is matched to obtain (n_i, n_{r_i}) that corresponds with r_j.
- For r_j, the service node in R is matched to obtain (r_j, r_{sos_j}) that corresponds with $\{so_1, so_2, so_{wj}\}$, which is the identification set of candidate service objects of a given n_i.

B. Calculation of the QoS attributes of the binding process instances

The candidate service objects corresponding to multiple service nodes are bound together to form process instances. For the four local structures, the calculation of the QoS attribute values of price, response time, reliability, and the user-customized service attributes are defined in Tables 1 to 4, respectively.

TABLE 1. Calculation of price.

Process execution structure	Price
Sequential or serial	$(p * \sum Q_{P_{i,l}}, p * \sum Q_{P_{i,m}}, p * \sum Q_{P_{i,u}})$
Parallel	$(p * \sum Q_{P_{i,l}}, p * \sum Q_{P_{i,m}}, p * \sum Q_{P_{i,u}})$
Selective or branch	$\left(p * \sum(p_i * Q_{P_{i,l}}), p * \sum(p_i * Q_{P_{i,m}}), p * \sum(p_i * Q_{P_{i,u}})\right)$
Executed cyclically n times	$(p * n * Q_{P_{i,l}}, p * n * Q_{P_{i,m}}, p * n * (p_i * Q_{P_{i,u}}))$

TABLE 2. Calculation of response time.

Process execution structure	Response time
Sequential or serial	$(p * \sum Q_{T_{i,l}}, p * \sum Q_{T_{i,m}}, p * \sum Q_{T_{i,u}})$
Parallel	$(p * \max(Q_{T_{i,l}}), p * \max(Q_{T_{i,m}}), p * \max(Q_{T_{i,u}}))$
Selective or branch	$(p * \sum(p_i * Q_{T_{i,l}}), p * \sum(p_i * Q_{T_{i,m}}), p * \sum(p_i * Q_{T_{i,u}}))$
Executed cyclically n times	$(p * n * Q_{T_{i,l}}, p * n * Q_{T_{i,m}}, p * n * Q_{T_{i,u}})$

Table 3. Calculation of reliability.

Process execution structure	Reliability
Sequential or serial	$(p * \prod Q_{Rel_{i,l}}, p * \prod Q_{Rel_{i,m}}, p * \prod Q_{Rel_{i,u}})$
Parallel	$(p * \prod Q_{Rel_{i,l}}, p * \prod Q_{Rel_{i,m}}, p * \prod Q_{Rel_{i,u}})$
Selective or branch	$(p * \sum(p_i * Q_{Rel_{i,l}}), p * \sum(p_i * Q_{Rel_{i,m}}), p * \sum(p_i * Q_{Rel_{i,u}}))$
Executed cyclically n times	$(p * (Q_{Rel_{i,l}})^n, p * (Q_{Rel_{i,m}})^n, p * (Q_{Rel_{i,u}})^n)$

Table 4. Calculation of the user-customized service attribute.

Process execution structure	User customized service attribute
Sequential or serial	$(p * \frac{1}{n} * \sum Q_{UC_{i,l}}, p * \frac{1}{n} * \sum Q_{UC_{i,m}}, p * \frac{1}{n} * \sum Q_{UC_{i,u}})$
Parallel	$(p * \frac{1}{n} * \sum Q_{UC_{i,l}}, p * \frac{1}{n} * \sum Q_{UC_{i,m}}, p * \frac{1}{n} * Q_{UC_{i,u}})$
Selective or branch	$(p * \sum(p_i * Q_{UC_{i,l}}), p * \sum(p_i * Q_{UC_{i,m}}), p * \sum(p_i * Q_{UC_{i,u}}))$
Executed cyclically n times	$(p * Q_{UC_{i,l}}, p * Q_{UC_{i,m}}, p * Q_{UC_{i,u}})$

C. Calculation of the fuzzy QoS weight based on a fuzzy analytic hierarchy process

1) TRIANGULAR FUZZY NUMBERS

The values of the QoS attributes typically depend on the preference of users where TFN described the uncertain QoS attributes. Therefore, we propose a comparison method of TFN and a modified formula of TFN defuzzification based on a conservative strategy.

TFN is expressed in as a triple, such as $TFN = (l, m, u)$, where l, m, u represent the minimum, average, and maximum, respectively, of the statistics or experience. TFN must be converted into a single value when performing the pairwise comparison, which is called defuzzification and often uses the centre of gravity method. For $TFN = (l, m, u)$, the center of gravity is $CG = \frac{(l + m + u)}{3}$.

For our proposed comparison approach for TFN, we define $TFN_1 = (l_1, m_1, u_1)$ and $TFN_2 = (l_2, m_2, u_2)$, and the centers of gravity are CG_1 and CG_2, respectively.

a) If $CG_1 < CG_2$, then TFN_1 is inferior to TFN_2.

b) If $CG_1 \geq CG_2$, then the comparison is divided into four cases:

- If $l_1 < l_2$ and $m_1 < m_2$, then TFN_1 is inferior to TFN_2 and the risk is relatively high.
- If $l_1 < l_2$ and $m_1 > m_2$, then TFN_1 is superior to TFN_2 and the average is satisfied.
- If $l_1 > l_2$ and $m_1 < m_2$, then TFN_1 is superior to TFN_2, the risk is relatively low, and the minimum is more optimistic.
- If $l_1 > l_2$ and $m_1 > m_2$, then TFN_1 is superior to TFN_2, which is strictly optimal.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

On this comparison basis, we propose a modified formula of the TFN defuzzification calculation. For $TFN = (l, m, u)$, we introduce the defuzzification weight vector $TFN_W = (w_l, w_m, w_u,)$, where

$$Defuzzy\ (TFN) = \frac{(w_l * l + w_m * m + w_u * u)}{(w_l + w_m + w_u)} \quad (3)$$

When $w_l \geq w_m, w_u$, the defuzzification presents a conservative strategy, reflecting the tendency to avoid risks.

2) FUZZY ANALYTIC HIERARCHY PROCESS (FAHP)

FAHP is a fuzzy, multi-attribute decision-making method that combines fuzzy set theory and the Analytic Hierarchy Process (AHP). We use an improved FAHP method to construct fuzzy QoS weights with the following steps.

Step 1. FAHP defines the hierarchical relationship between attributes and sub-attributes.

Step 2. Set the corresponding table between the fuzzy linguistic value and TFN. In this improved FAHP table, the elements of the upper and lower symmetry are logically inverse relations recorded as (l_1, m_1, u_1) and (l_2, m_2, u_2) where

$$l_1 + u_2 = 1 \quad (4)$$
$$m_1 + m_2 = 1 \quad (5)$$
$$u_1 + l_2 = 1 \quad (6)$$

In the fuzzy root of the FAHP table, the two TFNs of the inverse relationship satisfy the following equations.

$$l_1 * u_2 = 1 \quad (7)$$
$$m_1 * m_1 = 1 \quad (8)$$
$$u_1 * l_2 = 1 \quad (9)$$

Step 3. Construct a fuzzy preference relationship decision matrix such that the rows and columns are QoS attributes.

Step 4. Replace the fuzzy linguistic value in the fuzzy preference relation decision matrix to the corresponding TFN. Let $(p_L(i,j), p_M(i,j), p_U(i,j))$ be the elements representing the TFN in the fuzzy preference relational decision matrix where n attributes participate in the pairwise comparison, i and j represent the rows and columns, and $\forall i, j \in \{1, \ldots, n\}$.

$$p_L(i,j) + p_U(j,i) = 1 \quad (10)$$
$$p_M(i,j) + p_M(j,i) = 1 \quad (11)$$
$$p_U(i,j) + p_L(j,i) = 1 \quad (12)$$
$$p_L(i,j) + p_L(j,k) + p_U(k,i) = \frac{3}{2}, i < j < k \quad (13)$$
$$p_M(i,j) + p_M(j,k) + p_M(k,i) = \frac{3}{2}, i < j < k \quad (14)$$
$$pU(i,j) + p_U(j,k) + p_L(k,i) = \frac{3}{2}, i < j < k \quad (15)$$
$$p_L(i,i+1) + p_L(i+1,i+2) + \cdots + p_L(j-1,j) + p_U(j,i) = \frac{j-i+1}{2}, i < j \quad (16)$$
$$p_M(i,i+1) + p_M(i+1,i+2) + \cdots + p_M(j-1,j) + p_M(j,i) = \frac{j-i+1}{2}, i < j \quad (17)$$
$$p_U(i,i+1) + p_U(i+1,i+2) + \cdots + p_U(j-1,j) + p_L(j,i) = \frac{j-i+1}{2}, i < j \quad (18)$$

Step 5. All elements of the fuzzy preference relation decision matrix are given or calculated. If the matrix can be found and the negative values or the values are greater than 1, then it requires further normalization. For all elements $(p_L(i,j), p_M(i,j), p_U(i,j))$, there exists $c(c > 0)$ and $p_{L/M/U}(i,j) \in [-c, 1+c]$.

$$TFNNorm(x_L) = \frac{(x_L + c)}{(1+2*c)} \quad (19)$$
$$TFNNorm(x_M) = \frac{(x_M + c)}{(1+2*c)} \quad (20)$$
$$TFNNorm(x_U) = \frac{(x_U + c)}{(1+2*c)} \quad (21)$$

The TFN components of each element in the matrix are processed according to the normalization formula. After normalization, $p_{L/M/U}(i,j) \in [0,1]$.

Step 6. Calculate the weight of each QoS attribute. First, average each row of the fuzzy preference relational decision matrix.

$$A_{L/M/U}(i) = \frac{1}{n}\sum_{j=1}^{n} p_{L/M/U}(i,j) \quad (22)$$

Then, we obtain the weight of the QoS attributes corresponding to rows.

$$W_{L/M/U}(i) = \frac{A_{L/M/U}(i)}{\sum_{i=1}^{n} A_{U/M/L}(i)} \quad (23)$$

D. Calculation of the correlative QoS attributes of the binding process instances

The calculation of correlation is complicated, as it takes the overall consideration of the binding process instance. We present the following calculation steps for the correlation of the binding process instance in a microservice.

Step 1. According to the structure descriptor, the loop execution is expanded into a sequential or serial structure. The probability of occurrence for each service node is calculated recursively from the external.

Step 2. Initialize the occurrence probability table so_p of the candidates with the occurrence probability of each service node, and calculate the probability from inside out. In this process, the same service objects are merged, and their occurrence probabilities are adjusted. Finally, we obtain the occurrence probability of different candidate service objects. During this step, we apply the following rules.

Rule 1. Sequential or serial execution structure. If there is a nested sequential or serial execution substructure, then expand it to merge with the structure. If the same service object so_i exists, then the corresponding nodes are set as node$_1$ and node$_2$. Let $so_{p_{new}} = max\{so_{p_1}, so_{p_2}\}$ and only keep the so_i. If the remaining node$_1$ corresponds to so_i, then update the so_p table and the probability of node$_1$ becomes $so_{p_{new}}$. The probability of node$_2$ is 0. Then, it is processed by the parent structure's matching rule of this structure. If there is no parent structure, then the algorithm ends.

Rule 2. Parallel execution structure. Connect each branch to a sequential or serial execution structure. If there is a sequential or serial substructure, then expand and merge with the current structure, and apply Rule 1.

Rule 3. Selective or branch execution structure. Connect each branch to a sequential or serial execution structure. If there exists a nested sequential or serial execution substructure, then the expansion is merged with the current structure. If the same service object so_i exists, then set the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

corresponding service node as node$_1$ and node$_2$ to create $so_{p_{new}} = so_{p_1} + so_{p_2}$, so that so_i is reserved. If node$_1$ corresponding to so_i is reserved, then the so_p table is updated. The probability of node$_1$ is $so_{p_{new}}$, and the probability of node$_2$ is 0. This is then processed by the parent structure's matching rule of this structure. If there is no parent structure, then the algorithm ends.

Step 3. For the so_p table obtained in the previous step, remove the column corresponding to 0 to obtain the probability of different candidates.

Step 4. Find pairwise combinations of internal and external service providers in the table and construct the probability matrix of the service providers corresponding to candidate service objects. The matrix is an upper triangular matrix, and the probability of service providers is calculated by the following two principles.

- Internal correlation of service providers.

Initialising the correlated probability matrix of the service providers, each element of the diagonal line is 0. In the appearance probability table of candidate service objects, search for different sets with the same service provider to denote as $\{so_i\}$. The number of the set element is labelled as a satisfying $2 \le a \le b$ where b is the number of candidate service objects. There exists $sp\ (so_i) = sp\ (so_j)$ and the service provider is sp_k.

$$sp_{co_{p(sp_k)}} = 1 - \prod_{i=1}^{m}\left(1 - so_p(so_i)\right) - \sum_{i=1}^{m}(so_p(so_i) * \prod_{j=1,j\neq i}^{m}(1 - so_p(so_j))) \quad (24)$$

- External correlation of service providers.

After initialising the correlated probability matrix of the service providers, the upper right part of the matrix, except for the diagonal, is 0. In the probability table of candidate service objects, search for those corresponding to the set and denote as $\{so_a\}$ and $\{so_b\}$ for the two service combinations of service providers sp_i and sp_j, where the numbers of the two set elements are a and b. If the candidate service object set is empty, then the probability of correlation corresponding to the two service providers is 0. Otherwise, it meets the following condition.

$$sp_{co_{p(sp_i,sp_j)}} = \left[1 - \prod_{i=1}^{m}\left(1 - so_p(so_i)\right)\right] * [1 - \prod_{i=1}^{n}\left(1 - so_p(so_i)\right)] \quad (25)$$

Step 5. Each element in the correlated probability matrix of service providers is multiplied by each element of the correlated weight matrix to obtain a correlation matrix of the providers.

Step 6. Sum over each element of the correlated matrix of service providers denoted as CO_{sum}.

5) Antibody and antibody affinity. The smaller the affinity between antibodies, the poorer the diversity of antibody populations become. The larger the affinity between antibodies, the diversity becomes more abundant.

6) Assignment of the number of antibody clones. The cloning probability q_i of the antibody A_i is normalized as

$$CO_{sum} = \frac{1}{2} * \{\frac{CO_{sum}}{C(s_{p_n},2)} + s_{p_n}] + 1\} \quad (26)$$

where $C(s_{p_n}, 2)$ is a combination representing the value of the correlation degree QoS attribute, or the component Q_{co}.

E. Optimization model

$$\max solution = Defuzzy\left(TFNSumWeighted\left(QoS(solution_i)\right)\right) \quad (27)$$

where $QoS\ (solution_i)$ is recorded as $(Q_P, Q_T, Q_{Rel}, Q_{Co}, Q_{Uc})$. The TFNSumWeighted function utilizes the weight matrix of QoS attributes of microservice group to get the weighted values. Weight the $QoS\ (solution_i)$ to get a new weight matrix.

$$QoS_v = \begin{bmatrix} qw_{P,l} * Q_{P_{i,l}} & qw_{P,m} * Q_{P_{i,m}} & qw_{P,u} * Q_{P_{i,u}} \\ qw_{T,l} * Q_{T_{i,l}} & qw_{T,m} * Q_{T_{i,m}} & qw_{T,u} * Q_{T_{i,u}} \\ qw_{Rel,l} * Q_{Rel_{i,l}} & qw_{Rel,m} * Q_{Rel_{i,m}} & qw_{Rel,u} * Q_{Rel_{i,u}} \\ qw_{Co,l} * Q_{Co_{i,l}} & qw_{Co,m} * Q_{Co_{i,m}} & qw_{Co,u} * Q_{Co_{i,u}} \\ qw_{UC,l} * Q_{Uc_{i,l}} & qw_{UC,m} * Q_{Uc_{i,m}} & qw_{UC,u} * Q_{Uc_{i,u}} \end{bmatrix} \quad (28)$$

The matrix is summed by columns, which is recorded as $\left(sw_{qos_l}, sw_{qos_m}, sw_{qos_u}\right)$. Defuzzy the weighted vector, we get

$$Defuzzy\left(TFNSumWeighted\left(QoS(solutioni)\right)\right) = \left(w_l * sw_{qos_l} + w_m * sw_{qos_m} + w_u * sw_{qos_u}\right) / (w_l + w_m + w_u) \quad (29)$$

IV. Parallel Cooperative Short-term Memory Injection Polyclone Clonal Selection Algorithm

A. Improvement of the clonal selection algorithm

The clonal selection algorithm is based on an artificial immune principle that offers an efficient algorithm for discrete combinatorial optimization problems. In most cases, it can achieve enough optimization results compared to standard genetic algorithms. For the microservice combination optimization problem presented in this paper, we apply the following provisions.

1) Antibody code. The antibody encodes the solution space for the optimization problem.

2) Antibody decode. The specific service object identifier of the sequential number mapping of the candidate service is indicated by $solution = \{SO_{ID_i}\}$.

3) Antigen. Refers to the optimization problem domain.

4) Antibody and antigen affinity. The degree of adaptation of the antibody to the antigen is the objective function.

$q_i = \frac{q_i}{\sum_{i=1}^{n} q_i}$. The cloning scale of antibody A_i is $n_{c_i} = INT[q_i * n_c]$, where n_c is the total cloning number of the antibody population and the term $INT[\dots]$ is rounded up or down. Adjusting n_{c_i} will make it satisfy $n_c = \sum_{i=1}^{n} n_{c_i}$.

7) Adaptive hypermutation strategy. Du et al. [43] proposed this approach that comprehensively evaluated the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

mutation probability based on antibody affinity and population diversity. Guided by this concept, we define the following definition for transformation.

Step 1. Set the threshold of the variation probability range. The lower limit of mutation probability of an antibody is $p_0^m = \frac{1}{antibody\ coding\ dimension}$, and the upper limit of mutation probability of an antibody is $p_{max}^m = \frac{d}{antibody\ coding\ dimension}$. If the dimension of variation is too high, then it will miss the global optimum value. Therefore, the maximum variation dimension of the hypervariation is limited to 2, such that d = 2, so $p_{max}^m = \frac{2}{antibody\ coding\ dimension}$.

Step 2. Calculate the normalized affinity D_i of the average distance for the antibody A_i in the antibody population $\{A_1, A_2, \dots, A_n\}$.

Step 3. Calculate $QoS\ (solution\ (A_i))$ for the antibody A_i in the antibody population $\{A_1, A_2, \dots, A_n\}$, denoted as qos_i. The sum of QoS of all antibodies is $\sum_{i=1}^{n} \mathrm{qos}_i$.

Step 4. $boost_i = \frac{1}{\exp(\mathrm{D}_i)} * \frac{\mathrm{qos}_i}{\sum_{i=1}^{n} \mathrm{qos}_i} * \mathrm{n}$, where n is the number of antibodies.

Step 5. The $boost_i$ of antibody A_i is normalized to obtain $boost_i = \frac{boost_i}{\sum_{i=1}^{n} \mathrm{boost}_i}$.

Step 6. Let $c_i = boost_i * n$, then assign the hypermutation sensitive threshold parameter to c_0, which is typically set between 1.05 to 1.1. If $c_i > c_0$, then we make $p_i^m = p_{max}^m$. Otherwise, $p_i^m = p_0^m$. p_i^m is the adaptive hypermutation probability of the antibody.

In our experimentation, we observe that the standard clonal selection algorithm can easily miss the global optimum when using the adaptive mutation strategy in a larger variation of the step. The space of the solution of the coverage is not comprehensive enough, so the average solution is not stable. Therefore, we suggest the following improvement strategy based on the standard clonal selection algorithm.

1) Uniform distribution variation strategy. The mutation probability of each dimension of the antibody is set to its reciprocal. The random selection of the antibody with a minimum can achieve better optimization results. We adopt a uniform distribution variation strategy when the dimension is selected.

2) Short term memory injection combined with clonal variation full storage strategy. After the cloning mutation of an antibody A_i, the optimal antibody in the subpopulation of the clonal selection algorithm is recorded as $group_{best}$. Regardless of if $QoS(solution(A_i))$ is greater than $QoS(solution(group_{besti}))$, the value of $group_{besti}$ is unconditionally accepted to join the next generation of the antibody population. To prevent population degradation, the previous generation's optimal antibody is injected into the next generation during the final step of the clonal selection to replace the worst population.

3) Improvement of cross strategy. Multi-point crossover can offer an improved optimization effect. When $p_{crossover} = Int\left[\frac{gene\ coding\ dimension}{2}\right] * p_{mutation} = Int\left[\frac{gene\ coding\ dimension}{2}\right] * \frac{1/}{gene\ coding\ dimension}$, then $Int[\dots]$ is rounded down. The gene pairs exchange the encoded values with the most significant random selection to minimize the impact of crossover on population diversity.

The results of this experiment suggest that the multi-clone strategy with multi-point crossover at the largest dimension improves more significantly than a genetic algorithm. After comprehensively using the three strategies, the improved multi-clone selection algorithm appears superior to the improved genetic algorithm.

B. Improvement of the immune memory clone programming algorithm

Jiao [47] proposed an Immune Memory Clonal Selection Algorithm (IMCSA) based on the principle of clone selection. The IMCSA introduces the immune memory mechanism that divides the antibody into two subpopulations of the common antibody population and the memory unit. The algorithm enables the common antibody population to evolve with a higher mutation probability for maintaining the greatest diversity possible. The memory unit maintains the historical global optimal solution of the current status and sets a small mutation probability or a mutation strategy that is different from the common antibody population. The memory unit continuously absorbs the optimal antibody of each generation of the common group and infuses the optimal antibody into the common antibody group. When the memory unit is no longer improved, the algorithm gradually converges to the global optimum.

Based on the IMCSA, we propose the following strategies.

1) The clonal selection process for the memory cell and common antibody populations adopts this improved clonal selection process.

2) The immune memory injection mechanism occurs before the improvement. At the beginning of each iteration, only the optimal antibody of the memory unit is injected into the common antibody population to replace the worst antibody.

3) Improved immune memory maturation mechanism. At the end of each iteration, the memory unit absorbs the optimal antibody from the common antibody population and replaces its worst antibody.

4) Short-term immune memory injection mechanism and improved immune memory maturation mechanism. The memory unit must have an optimal antibody in both populations after each iteration. Before the next iteration, the optimal antibody is injected into the common antibody population. If the antibody of the common antibody population that is absorbed by the memory unit is not optimum, then the optimal antibody must exist in the memory unit. So, the optimal antibody injected into the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

common population must come from the memory unit before the iteration. Then, these two populations achieve the exchange of the optimal antibody.

5) The correlated short-term immune memory injection strategy. After each iteration, the memory unit and the common antibody population exchange their optimal antibodies to replace the worst antibodies.

C. The Parallel Cooperative Short-term Memory Injection Multi-clone Clonal Selection Algorithm

After combining the improved clonal selection process and the cooperative short-term immune memory injection strategy, the improved clonal immune memory programming algorithm reflects cooperative evolution between two identical populations. The experimental results demonstrate the effectiveness of the algorithm. The results also show that the IMCSA is inferior to the improved clonal selection algorithm under the same computing cost and number of antibodies. The split of the immune memory unit and ordinary antibody group causes the two populations to be smaller. The mutation and crossover are conducted in the smaller populations, which leads to each antibody group being more sensitive to population diversity.

Based on these experimental results, we propose a moderate cooperative mechanism of short-term memory. A configurable inter-group cooperative mechanism is provided according to the size of the antibody populations that can flexibly set which populations exchange either unidirectionally or bidirectionally. After each iteration, the exchange the optimal antibodies from the subpopulations occur to replace the worst from each group. Finally, we propose ParaCoSIMCSA based on the improved IMCSA. The principle of the algorithm is shown in Figure 3, and the steps are as follows.

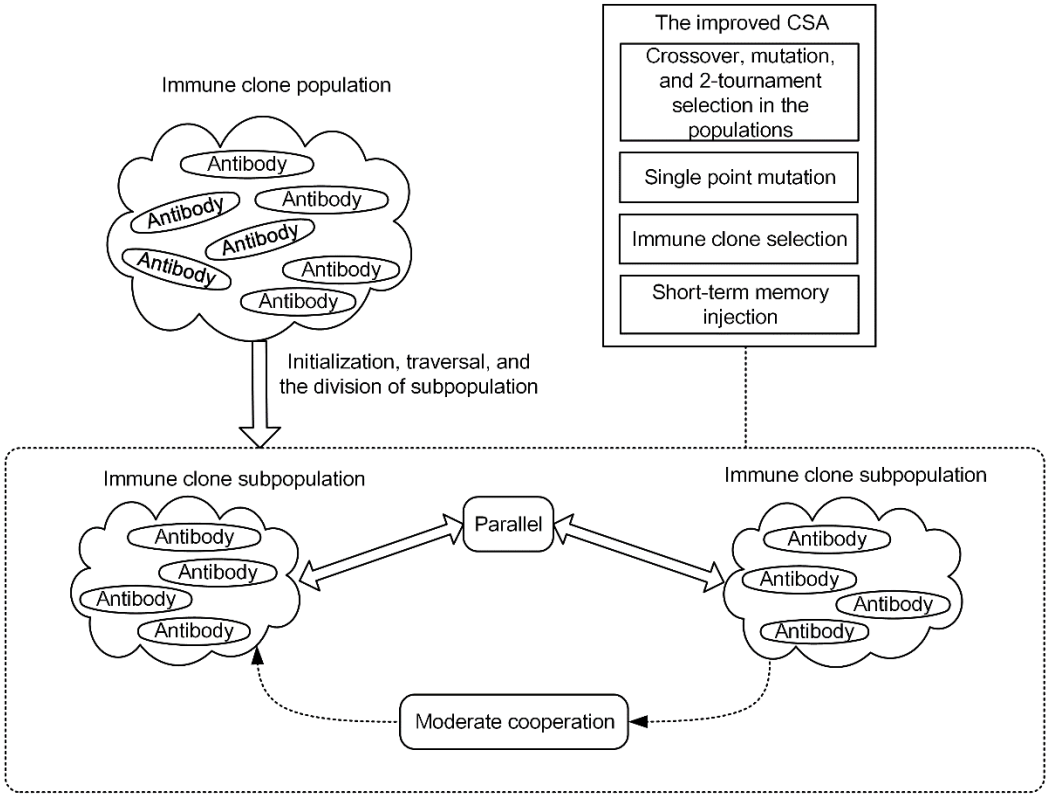


FIGURE 3. **Schematic diagram of ParaCoSIMCSA based on the hybrid clone and selective moderate cooperation.**

Step 1. Initialize the antibody population arranged in reverse order of affinity.

Step 2. Traverse the antibody population and divide the immune clonal population.

Step 3. Each population evolves correlatively in parallel. At each iteration, the population is processed in parallel based on the improved clone selection process. If the maximum number of iterations is reached, then the algorithm ends.

In this third step, the following operations are specifically performed.

1) Each parallel population implements the maximum dimension multi-point crossover. The crossed offerings perform a single-point mutation and select the optimal antibody to place into the new population after the two-selection match.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

2) Each parallel population executes the immunising cloning process. All cloned antibodies perform the single-point mutation.

3) Each parallel population implements the immune clonal selection process. The new antibody is unconditionally accepted into the subpopulations.

4) To avoid the degeneration of the populations, short-term memory injection replaces the worst antibody with the global optimum from the previous population.

5) After each generation of populations creates new antibodies, they update the population optimal value.

Step 4. All populations end in parallel and update the global optimal solution. Each population replaces the worst antibodies with the optimums. Turn Step 3.

V. Experiment

A. Experimental setup

In this experiment, a desktop PC running the Windows7-64bit operating system is used with an Intel core i5 760（OC 4.0 GHz, 4 cores and 4 threads), 8 GB memory and 120 GB SSD hard disk. The emulator is implemented in Java written with the Eclipse IDE running 64-bit JDK-1.8.0. The 64-bit JVM can configure a large heap memory, which will improve the time and performance of the Java garbage collection mechanism over multiple iterations of the algorithm for multiple trials. This will eliminate much of the fluctuations of the JVM during multiple runs of the same or different algorithms. The related modules of vector, matrix, and statistical calculation use the Apache Commons Math open source library and the random number generation uses its RandomDataGenerator class. The partition of roles of the service objects, the TFN of the QoS attributes of the service objects, and the service group candidate pool are generated by pseudo-random numbers. To ensure the repeatability and consistency of each experiment, the simulation using the pseudo-random number sets a fixed random seed to generate the same random value for each run.

B. Preset parameters and comparison principles

1) PRESET PARAMETERS

The experimental parameters are preset, and the values of each are as follows.

i) The processes to be combined. Take the process constituting the six service nodes, as an example, that is shown in Figure 2, which can be expressed as SEQ (1, AND (SEQ (2, OR (4, 5)), 3), 6). The probabilities of the selective structure are set to 0.3 and 0.7. The optimised value QoS of the service composition for the process target belongs in [0, 1], and the optimization direction is maximized.

ii) Service object candidate pool. $h = 1000$.

iii) Service Provider. $y = 4$, which includes {0, 1, 2, 3}.

vi) Service provider cooperative matrix. $SP_{cow} = \{\{1, 0, 1, -1\}, \{0, 0, -1, 0\}, \{0, 0, -1, 1\}, \{0, 0, 0, 1\}\}$.

v) QoS weight matrix of the service combination. $QoS_w = \{\{0.12, 0.21, 0.37\}, \{0.07, 0.12, 0.23\}, \{0.16, 0.24, 0.37\}, \{0.11, 0.19, 0.32\}, \{0.13, 0.24, 0.41\}\}$. The row vector is the TFN weight of each QoS attribute.

vi) The service group of the process service node, represented by x. The number of service groups is set to 5, including {0, 1, 2, 3, 4} to obtain $x = \{2, 1, 3, 4, 3, 0\}$.

vii) The candidate pool of service groups. $g = \{SO(0), SO(1), SO(2), SO(3), SO(4)\}$, where $SO(i)$ represents a list of candidate services corresponding to the service group. Each element in the array randomly selects $SO(i)$ different service objects from the candidate pool.

The number of candidate services in the candidate pool from the configured service groups determines the complexity of the solution space of the service composition optimization problem. In this experiment, we design four combinatorial space configurations.

Scale 1. $x = \{25, 35, 30, 20, 15\}$ with approximately 1.58 million combinations.

Scale 2. $x = \{50, 70, 60, 40, 45\}$ with approximately 151 million combinations.

Scale 3. $x = \{100, 140, 120, 80, 90\}$with approximately 967.7 million combinations.

Scale 4. $x = \{200, 280, 240, 160, 180\}$ with approximately 62 trillion combinations, which refers to the data set size of the Ali dispatch competition.

2) COMPARISON PRINCIPLE

a) The measurement of the average execution time and average solution. After multiple experiments, we find that the results of 200 repeated operations tend to be stable and achieve a good compromise in terms of time consumption and accuracy. Therefore, the total execution time and the average solution from 200 runs are taken for horizontal and vertical comparisons of the algorithm.

b) The cost of the calculation. The calculation cost examines the number of iterations, the size of the population, the number of variation and crossover, the calculation of QoS, the complexity, and the characteristics of the algorithm.

C. Comparison and analysis of the ParaCoSIMCSA results

The ParaCoSIMCSA is compared to the standard genetic algorithm, the monoclone selection algorithm, and the multi-clonal selection algorithm. Also, comparisons are provided to the improvements of these three algorithms as well as the monoclone immune memory cloning programming algorithm and multi-clone immune memory cloning programming algorithm at the same scale, which represent situation 1 with results shown in Tables 5 and 6.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

TABLE 5. Comparison of algorithms for situation 1.

Algorithm	Scale 3 (Calculation cost limited to 37800)	Notes
Standard genetic algorithm (POP_SIZE=100, Tmax=180, p_sele_m=0.1, single point crossing)	Average solution：0.7642817093806431 Execution time：80620ms	
Monoclone selection algorithm (ANTI_COUNT=30, ANTI_ CLONE_COUNT =300, Tmax=126)	Average solution：0.7666678743490252 Execution time：66370ms	Monoclone selection algorithm is superior to the standard genetic algorithm.
Multi-clone selection algorithm (ANTI_COUNT=30, ANTI_ CLONE_COUNT =150, Tmax=20, single point crossing)	Average solution：0.7677261155894745 Execution time：62681ms	Multi-clone creates a bottleneck of random antibody variations in population diversity.
Multi-clone selection algorithm (ANTI_COUNT=30, ANTI_ CLONE_COUNT =150, Tmax=20, maximum dimension multi-point crossing.)	Average solution：0.7677440102752684 Execution time：63264ms	Maximum dimensional multi-point crossing slightly improves the effect of single-point crossing on population diversity.
Monoclone immune memory clone programming algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =5, ANTI_CLONE_COUNT=300, ANTI_ MEMORY_CLONE_COUNT =50, Tmax=126, memory_recall=1.0)	Average solution：0.7655001089042788 Execution time：67718ms	Immune memory mechanism suppresses the diversity of the population.
Multi-clone immune memory cloning programming algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =5, ANTI_CLONE_COUNT=160, ANTI_ MEMORY_CLONE_COUNT =26, Tmax=27, memory _recall=1.0, single point crossing)	Average solution：0.767174665620872 Execution time：66394ms	Optimization performance of the multi-clone is superior to that of the monoclone.
Multi-clone immune memory cloning programming algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =5, ANTI_CLONE_COUNT=160, ANTI_ MEMORY_CLONE_COUNT =26, Tmax=27, memory_recall=1.0, maximum dimension multi-point crossing)	Average solution：0.7673251033882784 Execution time：66465ms	Maximum dimension multipoint crossing strategy improves the optimization performance.
Multi-clone immune memory cloning programming algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =5, ANTI_CLONE_COUNT=160, ANTI_ MEMORY_CLONE_COUNT =26, Tmax=27, memory_recall=0.0, maximum dimension multi-point crossing.)	Average solution：0.7677055472946981 Execution time：65768ms	*memory_recall* = 0.0 represents blocking the immune memory injection mechanism. However, performance optimization significantly increases, which validates the immune memory injection mechanism for the inhibition of population diversity.
Multi-clone immune memory cloning planning algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =25, ANTI_CLONE_COUNT=160, ANTI_ MEMORY_CLONE_COUNT =134, Tmax=27, memory_recall=0.0, maximum dimension multi-point crossing.)	Average solution：0.7676739743278953 Execution time：67232ms	Results suggest that the introduction of this mechanism degrades the optimization performance.
ParaCoSIMCSA (WORKERS_COUNT = 2, WORKERS_ANTI_COUNT = {20,10}, WORKERS_ANTI_CLONE_COUNT = {330,620}, WORKERS_COWORKERS = {{1},{}}, Tmax=20)	**Average solution：0.7678094096752802 Execution time：38265ms**	**Moderate cooperation reduces the cost of communication and enhances scalability.**
ParaCoSIMCSA (WORKERS_COUNT = 2, WORKERS_ANTI_COUNT = {20,10},	Average solution：0.7677709556923623 Execution time：38787ms	Too much cooperation reduces diversity and increases the possibility of local convergence.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

Algorithm	Scale 3 (Calculation cost limited to 37800)	Notes
WORKERS_ANTI_CLONE_COUNT = {330,620}, WORKERS_COWORKERS = {{1},{0}}, Tmax=20)		
ParaCoSIMCSA (WORKERS_COUNT = 2, WORKERS_ANTI_COUNT = {20,10}, WORKERS_ANTI_CLONE_COUNT = {330,620}, WORKERS_COWORKERS = {{},{}}, Tmax=20)	Average solution：0.7677959253372842 Execution time：38133ms	With independent evolution, the results are superior to excessive cooperation, but inferior to moderate cooperation.

Table 6. Comparison of the recall rate of the empirical global optimal solution of the algorithms for situation 2.

Algorithm (optimal parameter configuration)	Scale 4 (Calculation cost limited to 37800*3)	Recall rate of empirical global optimal solution
Standard genetic algorithm	Average solution：0.7815668820299694 Execution time：235570ms	33%
Monoclonal selection algorithm	Average solution：0.7847607018209204 Execution time：192097ms	90%
Multi-clone selection algorithm	Average solution：0.7848161560857695 Execution time：187921ms	98.5%
Monoclonal immune memory Clone Planning algorithm	Average solution：0.7823816022110257 Execution time：193697ms	44.5%
Multi-clone immune memory cloning planning algorithm	Average solution：0.7839956027857973 Execution time：186113ms	75.5%
ParaCoSIMCSA (WORKERS_COUNT = 3，incomplete interaction, WORKERS_COWORKERS = {{1},{},{1}})	**Average solution：0.7848657722552929 Execution time：66989ms**	**100%**
ParaCoSIMCSA (WORKERS_COUNT = 3，circular interaction, WORKERS_COWORKERS = {{1},{2},{0}})	Average solution：0.7848080892602625 Execution time：68165ms	98%
ParaCoSIMCSA (WORKERS_COUNT = 3，full interaction, WORKERS_COWORKERS = {{1,2},{0,2},{0,1}})	Average solution：0.7848124957805951 Execution time：69765ms	97.5%
ParaCoSIMCSA (WORKERS_COUNT = 3，no interaction, WORKERS_COWORKERS = {{},{},{}})	Average solution：0.7848651468165974 Execution time：66237ms	99.5%

In these experiments, the execution time is the total of 200 repetitions. The global optimal solution cannot be obtained from the global search, so the experiential maximum value of multiple runs of 0.7848657722552929 is taken as the benchmark. The recall rate of the global optimal is calculated as $\frac{the\ number\ of\ times\ obtained\ by\ empirical\ global\ optimal\ solution}{200}$ * 100%.

From these results, we conclude the following.

a) The mechanism of the ParaCoSIMCSA is effective. If there is no population cooperation, then each population evolves independently. The optimization performance is better than with too much cooperation but is inferior to moderate cooperation. Excessive cooperation reduces diversity and increases the possibility of local convergence. Moderate cooperation improves the optimization performance, reduces the cost of communication, and enhances the scalability.

b) The ParaCoSIMCSA offers the advantages of strong configuration, high parallel efficiency, and ease to use. With proper configuration, compared to the improved polyclone selection algorithm for a single population, our algorithm can obtain the same optimization effect in less computing time or a better optimization for a larger population.

Figure 4 compares the global optimal solution of the selective moderate cooperation and no cooperation of the hybrid clone of the ParaCoSIMCSA with iterations. The former is beneficial for breaking through the limitation of local optimal values.

Figure 5 shows the global optimal solution of selective moderate cooperation of the hybrid clone ParaCoSIMCSA with iterations. The cooperative behaviour is tracked in the experiment with the observation that cooperation between the two populations is relatively frequent, reflecting the complementarity of the population during the later stage of optimization.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

Figure 6 shows the global optimal solution of the hybrid clone ParaCoSIMCSA without cooperation through iterations. In the late stage of optimization, no elite antibody injection exists in the monoclonal population, and the multi-clone population fails to remove the attraction of a local optimal value, so it converges in advance.

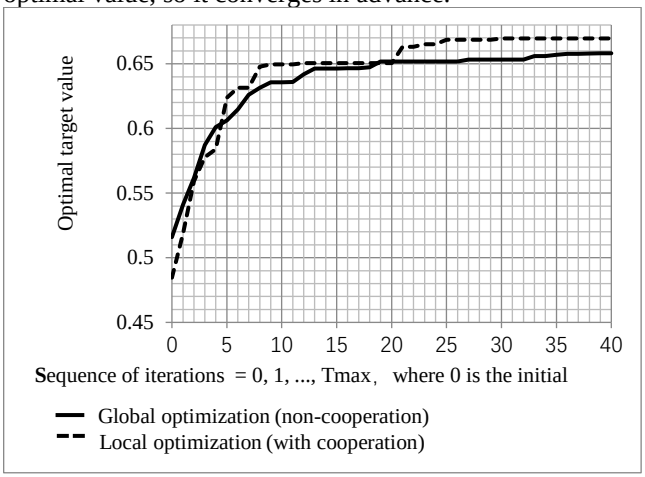


FIGURE 4. **Comparison of the selective moderate cooperation and non-cooperation in the hybrid clone ParaCoSIMCSA.**

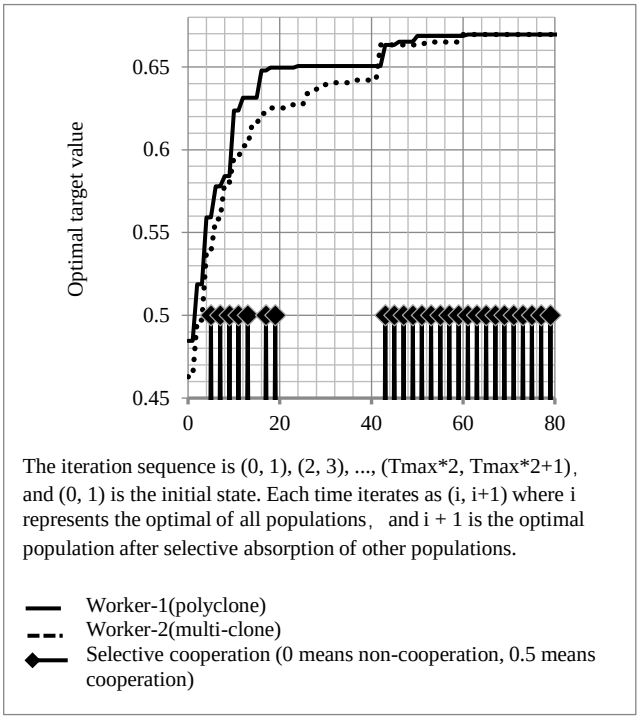


FIGURE 5. **Hybrid clone ParaCoSIMCSA with selective moderate cooperation.**

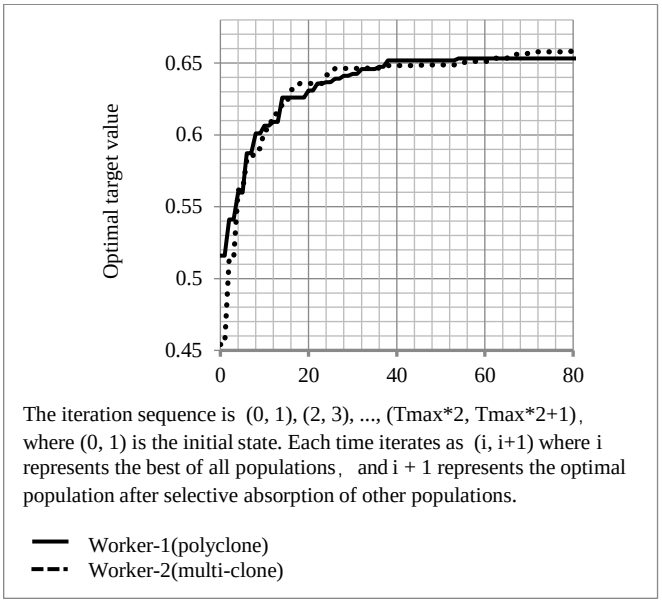


FIGURE 6. **Hybrid clone ParaCoSIMCSA without selective cooperation.**

VI. Conclusion

This research focused on the optimization of microservice combinations and proposed a concept of the microservice group to take the relationship and correlation of service groups of providers by considering intra- or inter-clouds. We utilised the TFN to describe the uncertainty of QoS attributes and determined the multi-attribute QoS with an improved FAHP. The users only need to provide the order of importance of attributes to obtain their weights. Based on the IMCSA, we also proposed the ParaCoSIMCSA that demonstrated superior performance compared to traditional clonal selection algorithms and the immune memory cloning algorithm.

VII. APPENDIX

A. Fuzzy QoS weight calculation of user preferences based on the improved FAHP

Users have different preferences for each attribute of the QoS for the binding process instances. The degree of preference is usually fuzzy, so it cannot be described accurately. Therefore, we use the fuzzy language value of the TFN to express the preference of users. FAHP is a fuzzy multi-attribute decision-making method combining fuzzy set theory and AHP. In this paper, we use an improved FAHP method to construct fuzzy QoS weights that integrate user preferences.

Feng and Kong [46] used the fuzzy root method to construct the corresponding relationship between the fuzzy language value and TFN and then calculated the fuzzy QoS weights of user preferences with FAHP. In this paper, to obtain the QoS weights, we construct the fuzzy judgment matrix for pairwise comparison of the QoS natures and then calculate the weight vectors on this basis.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

The fuzzy judgment matrix is a symmetric matrix, and all diagonal elements are 1. Each element in the matrix is the TFN corresponding to the fuzzy language value of the evaluation of the QoS attributes of the users in pairs. The users must fill the upper right triangle portion of the matrix, except the diagonal elements. If the number of elements in the QoS natures set is $\boldsymbol{n}$, then the users must make pair-to-pair comparison evaluations, which have $\frac{\boldsymbol{n}*(\boldsymbol{n}-\boldsymbol{1})}{\boldsymbol{2}}$ QoS attributes. Therefore, with the increase in the number of QoS attributes the accuracy of computation decreases.

Wang and Chen [47] improved the shortcomings of the FAHP by replacing the fuzzy judgment matrix with a fuzzy preference relation decision matrix. They redefined the corresponding relation of the fuzzy language values and the TFN. The users only need to do (n – 1) chains in the fuzzy preference relation decision matrix and compare two chains, such as $(\boldsymbol{A1}, \boldsymbol{A2}) -> (\boldsymbol{A2}, \boldsymbol{A3}) -> (\boldsymbol{A3}, \boldsymbol{A4})$. This method reduces the number of comparisons and improves the accuracy of the evaluation. Therefore, we propose the calculation steps of the fuzzy QoS weight based on the improved FAHP.

Step 1. FAHP defines the hierarchical relationship between the attributes and sub-attributes.

Step 2. Set the corresponding table between the fuzzy linguistic value and TFN.

In the improved FAHP table, the elements of the upper and lower symmetry are logically inverse relations and are recorded as (l_1, m_1, u_1) and (l_2, m_2, u_2), which refer to Equation (4) - (6) in the paper.

TABLE 7. Comparison of the language value and the TFN between two attributes in the improved FAHP.

The language value of the comparison between two attributes	Triangular fuzzy number
Very bad	(0.0, 0.0, 0.1)
Bad	(0.0, 0.1, 0.3)
A little bad	(0.1, 0.3, 0.5)
Equal	(0.5, 0.5, 0.5)
A little good	(0.5, 0.7, 0.9)
good	(0.7, 0.9, 1.0)
Very good	(0.9, 1.0, 1.0)

In the fuzzy root of the FAHP table, the two TFNs of the inverse relationship satisfies Equation (7) – (9).

Table 8. Comparison of the language value and the TFN between two attributes in the FAHP of the fuzzy root.

The language value of the comparison between two attributes	Triangular fuzzy number
Very bad	$(\frac{1}{9}, \frac{1}{9}, \frac{1}{7})$
Bad	$(\frac{1}{9}, \frac{1}{7}, \frac{1}{5})$
A little bad	$(\frac{1}{7}, \frac{1}{5}, \frac{1}{3})$
Equal	(1, 3, 5)
A little good	(3, 5, 7)
good	(5, 7, 9)
Very good	(7, 9, 9)

Step 3. Construct a fuzzy preference relationship decision matrix, such that the rows and columns are QoS attributes.

Table 9. Fuzzy preference relation decision matrix.

	Q_P	Q_T	Q_{Rel}	Q_{Co}	Q_{UC}
Q_P		Good			
Q_T			Very bad		
Q_{Rel}				A little good	
Q_{Co}					A little bad
Q_{UC}					

Step 4. Replace the fuzzy linguistic value in the fuzzy preference relation decision matrix to the corresponding TFN. Let $(p_L(i,j), p_M(i,j), p_U(i,j))$ be the elements representing the TFN in the fuzzy preference relational decision matrix, n attributes participate in the pairwise comparison, i and j represent rows and columns, and $\forall i,j \in \{1,\ldots,n\}$. The detailed calculation formulas refer to Equation (10) – (18). Next, fill the values of (0.5, 0.5, 0.5) into the diagonal elements, calculate the symmetric elements of the known elements by the formulas, and complete the other elements using the remaining formulas. The result of this calculation is shown in Table 10.

Step 5. All the elements of the fuzzy preference relation decision matrix are given or calculated. If the matrix can be found such that the negative values or the values are greater than 1, then it requires further normalization.

For all elements $(p_L(i,j), p_M(i,j), p_U(i,j))$, there exists $c(c>0)$ and $p_{L/M/U}(i,j) \in [-c, 1+c]$. The detailed calculation formulas refer to (19) – (21).

The TFN components of all elements in the matrix are processed according to this normalization formula. After normalization, $p_{L/M/U}(i,j) \in [0,1]$. According to this method, we obtain $c = 0.4$, and the normalization results are shown in Table 11.

Step 6. Calculate the weight of each QoS attribute. The averages are of each row in the fuzzy preference relational decision matrix, and the detailed calculation refers to Equation (22). Then, we obtain the weight of the QoS attributes corresponding to the rows. The detailed calculation formula refers to Equation (23), and the QoS attribute weights are calculated in Table 11, with results presented in Table 12. Finally, the fuzzy QoS weight vector corresponding to this user preference is obtained as ((0.12,0.21,0.37), (0.07,0.12,0.23), (0.16,0.24,0.37), (0.11,0.19,0.32), and (0.13,0.24,0.41).

B. Experiments

1) THE STANDARD GENETIC ALGORITHM

The standard genetic algorithm is the benchmark algorithm for this research and has developed a variety of deformations. Based on the algorithm framework proposed by Russell et al. [48], we perform experiments on scale 1 and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

TABLE 10. Fuzzy preference relation decision matrix containing the TFN.

	Q_P	Q_T	Q_Rel	Q_CO	Q_UC
Q_P	0.5, 0.5, 0.5	0.7, 0.9, 1	0.2, 0.4, 0.6	0.2, 0.6, 1	-0.2, 0.4, 1
Q_T	0, 0.1, 0.3	0.5, 0.5, 0.5	0, 0, 0.1	0, 0.2, 0.5	-0.4, 0, 0.5
Q_Rel	0.4, 0.6, 0.8	0.9, 1, 1	0.5, 0.5, 0.5	0.5, 0.7, 0.9	0.1, 0.5, 0.9
Q_CO	0, 0.4, 0.8	0.5, 0.8, 1	0.1, 0.3, 0.5	0.5, 0.5, 0.5	0.1, 0.3, 0.5
Q_UC	0, 0.6, 1.2	0.5, 1, 1.4	0.1, 0.5, 0.9	0.5, 0.7, 0.9	0.5, 0.5, 0.5

TABLE 11. Fuzzy preference relation decision matrix after normalization.

	Q_P	Q_T	Q_Rel	Q_CO	Q_UC
Q_P	0.50,0.50,0.50	0.61,0.72,0.78	0.33,0.44,0.56	0.33,0.56,0.78	0.11,0.44,0.78
Q_T	0.22,0.28,0.39	0.50,0.50,0.50	0.22,0.22,0.28	0.22,0.33,0.50	0.00,0.22,0.50
Q_Rel	0.44,0.56,0.67	0.72,0.78,0.78	0.50,0.50,0.50	0.50,0.61,0.72	0.28,0.50,0.72
Q_CO	0.22,0.44,0.67	0.50,0.67,0.78	0.28,0.39,0.50	0.50,0.50,0.50	0.28,0.39,0.50
Q_UC	0.22,0.56,0.89	0.50,0.78,1.00	0.28,0.50,0.72	0.50,0.61,0.72	0.50,0.50,0.50

TABLE 12. QoS attribute weights in the fuzzy preference relation decision matrix.

	Q_P	Q_T	Q_Rel	Q_CO	Q_UC	Average	Weight
Q_P	0.50,0.50,0.50	0.61,0.72,0.78	0.33,0.44,0.56	0.33,0.56,0.78	0.11,0.44,0.78	0.38,0.53,0.68	0.12,0.21,0.37
Q_T	0.22,0.28,0.39	0.50,0.50,0.50	0.22,0.22,0.28	0.22,0.33,0.50	0.00,0.22,0.50	0.23,0.31,0.43	0.07,0.12,0.23
Q_Rel	0.44,0.56,0.67	0.72,0.78,0.78	0.50,0.50,0.50	0.50,0.61,0.72	0.28,0.50,0.72	0.49,0.59,0.68	0.16,0.24,0.37
Q_CO	0.22,0.44,0.67	0.50,0.67,0.78	0.28,0.39,0.50	0.50,0.50,0.50	0.28,0.39,0.50	0.36,0.48,0.59	0.11,0.19,0.32
Q_UC	0.22,0.56,0.89	0.50,0.78,1.00	0.28,0.50,0.72	0.50,0.61,0.72	0.50,0.50,0.50	0.40,0.59,0.77	0.13,0.24,0.41

scale 2 with a limited computing cost of 37,800. These experimental results are shown in Table 13.

The standard genetic algorithm takes crossover as the core and variation as the supplement. Small populations decrease diversity, which easily results in a local convergence. At the same time, it selects a small mutation probability to produce new offspring after the crossover, and the population degradation may occur when the number of iterations is too small, or the proportion of the retained operating genes is too large.

2) IMPROVEMENT OF STANDARD GENETIC ALGORITHM

Given the above problems, we propose improvement strategies for the standard genetic algorithm.

a) Change of variation strategy. We set a large value for the probability of the cross-generated genes being selected for a variation to maintain population diversity. After several experiments, we observe that too large a variation step can readily miss the global optimum of the nearest neighbor, and the coverage of the solution space is not as good as the single-point variation. Uniform distribution variation can cover the solution space through iterations as much as possible. Gaussian mutation variation values are too close to the original point, which leads to the limited cover of the solution space.

Table 13. Comparison of the global search, random search, and standard genetic algorithm.

Algorithm	Scale 1(The cost of calculation is 37800)	Scale 2(The cost of calculation is 37800)	remarks
Global search	Globally optimal solution：0.7296354509474829 Single execution time：436791ms	Time cost is large.	Reduce runtime.
Random search algorithm (Randomly generated 37800 schemes is selected as the best)	Average solution：0.7027633169269976 Execution time：69367 ms	Average solution：0.6981540804396668 Execution time：68494 ms	As the scale increases, the instability and unreliability become prominent.
Standard genetic algorithm (POP_SIZE=100, Tmax=180)	Average solution：0.727650971330434 Execution time：79895 ms	Average solution：0.7344841385106744 Execution time：80278ms	Time consumption increases.
Standard genetic algorithm (POP_SIZE=30, Tmax=630)	Average solution：0. 7253680492292347 Execution time：80076ms	Average solution：0. 7302817093806431 Execution time：80620 ms	Genetic algorithm with a small population is inferior to that of a large population with the same computing cost.

b) Improvement of crossover strategy.

Crossover strategy generates new gene pairs by randomly selecting pairs from the elite population. In the standard

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

genetic algorithm, this strategy leads to population diversity decline after multiple iterations causing population degradation. Therefore, we apply a single-point evenly distributed variation to the cross-generated genes in the improved algorithm, which alleviates the impact of cross-generated genes on population diversity.

c) Short-term memory effect.

While there exists a population degradation in the standard genetic algorithm, in the improved algorithm, we inject short-term memory at the end of each iteration so that the worst antibody from the new generation is replaced by the global optimum. The optimal historical memory of the previous population is injected into the current generation, ensuring that the new group will not degenerate relative to the previous population. When the new population features a better result, the historical memory of the previous generation is eliminated after the next iteration, and the new historical memory is injected into the next generation, which has a less negative impact on the diversity of the group. When the new population degenerates, the historical memory of the previous population drives the new population to return to the nearest optimal value and then perform a new round of crossover and variation. This approach offers an increased probability and computing cost to bring it out of the local optimal value. Thus, the introduction of the short-term memory effect adjusts the concentration of historical memory adaptively to realize a stable evolution of each generation.

The experimental results of the improved genetic algorithm are compared with those of the standard genetic algorithm, as shown in Table 14. The improved genetic algorithm shows a satisfying optimization result, maintains population diversity, stabilizes high-frequency variation of small sizes, and avoids degradation. However, it must set a larger population size to achieve better optimization results. If the size of the optimization is too small, then an increase in the number of iterations cannot bring about significant improvement to the optimal solution. If the scale of the optimization problem is too large, then the scalability may significantly decrease.

TABLE 14. Comparison of the genetic algorithm and its improvement.

Algorithm	Scale 2 (calculation cost limited to 37,800)	Scale 3(calculation cost limited to 37,800)	Notes
Standard genetic algorithm (POP_SIZE=100, Tmax=180, p_sele_m=0.1, single point crossing)	Average solution：0.7344841385106744 Execution time：80278ms	Average solution：0.7642817093806431 Execution time：80620ms	
Standard genetic algorithm (POP_SIZE=100, Tmax=180, p_sele_m=0.8, single point crossing)	Average solution：0.7383149840558187 Execution time：83331ms	Average solution：0.7665501863565093 Execution time：83576ms	High frequency variation improves the performance.
The improved genetic algorithm (POP_SIZE=100, Tmax=189, p_sele_m=0.8, without short-term memory injection, cross mutation after selection, single-point cross)	Average solution：0.7384065413917745 Execution time：87749ms	Average solution：0.7668057984907042 Execution time：87246ms	High-frequency variation reselection strategy after crossover improves the optimization performance.
The improved genetic algorithm (POP_SIZE=100, Tmax=189, p_sele_m=0.8, short-term memory injection, cross mutation after selection, single-point cross)	**Average solution：0.7386936255393984 Execution time：86983ms**	**Average solution：0.7674369853104762 Execution time：86723ms**	**Injection of short-term memory improves optimization performance.**
The improved genetic algorithm (POP_SIZE=100, Tmax=189, p_sele_m=0.8, short-term memory injection, cross mutation after selection, single point cross, maximum dimension multi-point crossing.	**Average solution：0.7386936255393984 Execution time：87712ms**	**Average solution：0.7675805155726327 Execution time：87586ms**	**Maximum dimension multi-point crossing maintains and improves the diversity of the population and algorithm.**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

3) IMPROVEMENT OF THE IMMUNE MEMORY CLONAL SELECTION ALGORITHM

Jiao [45] proposed the immune memory clonal programming algorithm IMCSA by introducing the immune memory mechanism and dividing the antibody populations into two subpopulations.

a) Normal antibody population, denoted as $A = \{A_1, A_2, \dots, A_{nn}\}$, where nn is the number of antibodies in the common antibody population. The number its clone is denoted as ncn.

b) Memory unit, denoted as $M = \{M_1, M_2, \dots, M_{nm}\}$, where nm is the number of antibodies in the antibody group of the memory units. The number of its clone is denoted as ncm.

The operation of the immune maturation strategy of the algorithm is as follows.

a) The optimal antibody in the common antibody group is denoted as $norm_{gbest}$. Compare with the distance of each antibody M_i in M with the preset distance threshold, denoted as δ_0, which means compare $distance(norm_{gbest}, M_i)$ with δ_0.

When $distance(norm_{gbest}, M_i) < \delta_0$ for all M_i, if $QoS\left(solution\left(norm_{gbest}\right)\right) > QoS(solution(M_i))$, then replace M_i with $norm_{gbest}$.

When $distance(norm_{gbest}, Mi) > \delta_0$ for all M_i, $norm_{gbest}$ replaces the worst antibody in M.

b) A new generation of memory unit antibodies is generated with the above method.

We adjust the algorithm with the following strategies.

i) Memory recall rate.

The memory recall rate is denoted as $memory_{recall}$, which represents the ratio of the memory unit to the normal antibody population injected into the normal antibody group. In the experiment, the number of antibodies exceeding the memory unit means that the excellent individuals of the common antibody group will be injected into the common antibody group again, resulting in the decrease of the diversity of the common antibody group and a decline in the performance of the algorithm. The number of injections is denoted as $nr = memory_{recall} * ncm$, and $memory_{recall}$ is set between 0 and 1.

ii) The variation strategies of the memory units and common antibody populations.

After the experiments, we discover that compared with the gauss variation, the single-point variation has a better performance in the discrete combinatorial optimization problem.

Based on the algorithm framework proposed by Jiao [45], our algorithm is implemented with the above adjustments for considering monoclone and multi-clone immune memory cloning programming. Monoclone and multi-clone variation strategies adopt single-point uniform distribution variations, and multi-clone crossover strategies adopt single-point crossovers. A comparison of algorithms comprised of the standard genetic algorithm, monoclone selection algorithm, and multi-clone selection algorithm along with the improvements of these three algorithms, monoclone immune memory clone programming algorithm, and multi-clone immune memory clone programming algorithm belonging to situation 2 is shown in Table 15.

TABLE 15. Comparison of the algorithms in situation 2.

Algorithm	Scale3 (Calculation cost limited to 37800)	Notes
Standard genetic algorithm (POP_SIZE=100, Tmax=180, p_sele_m=0.1, single point crossing)	Average solution：0.7642817093806431 Execution time：80620ms	
Improved genetic algorithm (POP_SIZE=100, Tmax=189, p_sele_m=0.8, short-term memory injection, selection after cross variation, maximum dimension multi-point crossing)	Average solution：0.7675805155726327 Execution time：87586ms	
Monoclone clone algorithm (ANTI_COUNT=30, ANTI_ CLONE_COUNT =300, Tmax=126)	Average solution：0.7666678743490252 Execution time：66370ms	
Multi-clone selection algorithm (ANTI_COUNT=30, ANTI_ CLONE_COUNT =150, Tmax=20, maximum dimension multi-point crossing.)	Average solution：0.7677440102752684 Execution time：63264ms	
Improved monoclone selection algorithm (ANTI_COUNT=30, ANTI_ CLONE_COUNT =300, Tmax=126，short-term memory injection.)	Average solution：0.7674410412840559 Execution time：68447ms	
Improved multi-clone selection algorithm (ANTI_COUNT=30, ANTI_ CLONE_COUNT =150, Tmax=20，short-term memory injection maximum dimension, multi-point crossing)	**Average solution：0.7677733996258047 Execution time：65191ms**	
Monoclone immune memory clone programming algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =5,	Average solution：0.7655001089042788 Execution time：67718ms	Immune memory mechanism suppresses the diversity of the population.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

Algorithm	Scale3 (Calculation cost limited to 37800)	Notes
ANTI_CLONE_COUNT=300, ANTI_ MEMORY_CLONE_COUNT =50, Tmax=126, memory_recall=1.0)		
Multi-clone immune memory clone programming algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =5, ANTI_CLONE_COUNT=160, ANTI_ MEMORY_CLONE_COUNT =26, Tmax=27, memory_recall=1.0, single point crossing)	Average solution：0.767174665620872 Execution time：66394ms	Optimization performance of the multi-clone is superior to that of monoclone.
Multi-clone immune memory clone programming algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =5, ANTI_CLONE_COUNT=160, ANTI_ MEMORY_CLONE_COUNT =26, Tmax=27, memory_recall=1.0, maximum dimension multi-point crossing)	Average solution：0.7673251033882784 Execution time：66465ms	Maximum dimension multipoint crossing strategy improves the optimization performance.
Multi-clone immune memory clone programming algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =5, ANTI_CLONE_COUNT=160, ANTI_ MEMORY_CLONE_COUNT =26, Tmax=27, memory_recall=0.0, maximum dimension multi-point crossing)	Average solution：0.7677055472946981 Execution time：65768ms	memory_recall = 0.0 representing the block of the immune memory injection mechanism. However, the significant increase in the optimised performance confirms the immune memory injection mechanism inhibits population diversity.
Multi-clone immune memory clone programming algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =25, ANTI_CLONE_COUNT=160, ANTI_ MEMORY_CLONE_COUNT =134, Tmax=27, memory_recall=0.0, maximum dimension multi-point crossing)	Average solution：0.7676739743278953 Execution time：67232ms	Results show that the introduction of this mechanism degrades the optimization performance.

The results of these experiments show that the immune memory mechanism does not improve optimization as expected, and the high recall rate of the immune memory greatly affects the population diversity for the discrete combinatorial optimization problem. The injection of the immune memory reduces new mutated antibodies, inhibits population diversity, and leads to an increase in the possibility of local convergence.

Based on these considerations, we improve the IMCSA and propose specific improvements. A comparison of algorithms comprising the standard genetic algorithm, monoclone selection algorithm, multi-clone selection algorithm, monoclone immune memory clone programming algorithm, multi-clone immune memory clone programming algorithm, and their improvements belonging to situation 3 is shown in Table 16.

TABLE 16. Comparison of algorithms in situation 3.

Algorithm	Scale3 (Calculation cost limited to 37800)	Notes
Improved genetic algorithm (POP_SIZE=100, Tmax=189, p_sele_m=0.8, short-term memory injection, selection after cross variation, maximum dimension multi-point crossing.)	Average solution：0.7675805155726327 Execution time：87586ms	
Monoclone cloning selection algorithm (ANTI_COUNT=30, ANTI_ CLONE_COUNT =300, Tmax=126)	Average solution：0.7666678743490252 Execution time：66370ms	
Multi-clone selection algorithm (ANTI_COUNT=30, ANTI_ CLONE_COUNT =150, Tmax=20, maximum dimension multi-point crossing.)	Average solution：0.7677440102752684 Execution time：63264ms	
Improved monoclone selection algorithm (ANTI_COUNT=30, ANTI_ CLONE_COUNT	Average solution：0.7674410412840559 Execution time：68447ms	

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

Algorithm	Scale3 (Calculation cost limited to 37800)	Notes
=300, Tmax=126，short-term memory injection.)		
Improved multi-clone selection algorithm (ANTI_COUNT=30, ANTI_ CLONE_COUNT =150, Tmax=20，short-term memory injection, maximum dimension multi-point crossing.)	Average solution：0.7677733996258047 Execution time：65191ms	
Monoclone immune memory clone programming algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =5, ANTI_CLONE_COUNT=300, ANTI_ MEMORY_CLONE_COUNT =50, Tmax=126, memory_recall=1.0)	Average solution：0.7655001089042788 Execution time：67718ms	
Multi-clone immune memory clone programming algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =5, ANTI_CLONE_COUNT=160, ANTI_ MEMORY_CLONE_COUNT =26, Tmax=27, memory_recall=1.0, maximum dimension multi-point crossing.)	Average solution：0.7673251033882784 Execution time：66465ms	
Improved monoclone immune memory clone programming algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =5, ANTI_CLONE_COUNT=300, ANTI_ MEMORY_CLONE_COUNT =50, Tmax=126)	Average solution：0.7674111729564063 Execution time：68394ms	Strategy of short-term memory injection is used to improve the sub-process of clonal mutation selection. With the introduction of the mechanism of cooperative short-term immune memory injection, the performance improves.
Improved multi-clone immune memory clone programming algorithm (ANTI_COUNT=30, ANTI_ MEMORY_COUNT =5, ANTI_CLONE_COUNT=160, ANTI_ MEMORY_CLONE_COUNT =26, Tmax=27, maximum dimension multi-point crossing.)	**Average solution：0.7677581483381418 Execution time：67654ms**	**Multi-clone strategy of the multi-point crossing of the maximum dimension further improves the performance.**

REFERENCE

[1] M. El Kholy, and A. El Fatatry, "Framework for interaction between databases and microservice architecture. " *It Prof.*, vol. 24, no. 4, pp. 57-63, 2019.

[2] A. Koschel, I. Astrova, and J. Dotter, "Making the move to microservice architecture." *Int. Conf. Info. Society.*, 74-79, Jul. 2017.

[3] M. Kalske, N. Mäkitalo, and T. Mikkonen, "Challenges When Moving from Monolith to Microservice Architecture". *Current Trends in Web Engineering.*, pp. 32-47, 2017.

[4] C. Esposito, A. Castiglione, and K. K. R. Choo, "Challenges in delivering software in the cloud as microservices." I*EEE Cloud Comput.*, vol. 3, no. 5, pp. 10-14, Oct 2017.

[5] S. Li, "Understanding quality attributes in microservice architecture. " *24th APSEC.* pp:9-10, Doc 2017.

[6] Y. Que et al. "Improved adaptive immune genetic algorithm for optimal QoS-aware service composition selection in cloud manufacturing." The International Journal of Advanced Manufacturing Technology, vol. 96, pp. 4455-4465, Mar 2018.

[7] X. Sun, J. Niu, Q. Gong, and Z. Li, "A web service selection strategy based on composite weighting method." *Appl. Research Comput.*, no. 8, pp. 174-177+194. 2017.

[8] Y. Ma, S. Wang, Q. Sun, and F. Yang, "A web services QoS measurement algorithm that comprehensively considers subjective and objective weights." *J. Software.* vol. 25, no. 11, pp. 2473-2485. 2014.

[9] C. Fang, J. Wang, and Z. Yu, "Web service selection based on dynamic QoS." *Comput. Science.* vol. 44, no. 5, pp. 245-250. 2017.

[10] Y. Zhuang, P. Zhang, W. Li, J. Feng, and Y. Zhu, "An environmentally sensitive approach to web service QoS

monitoring." *J. Software*. vol. 27. no. 8. pp. 1978-1992. May 2017.

[11] K. Xu, X. Zhu, and X. Jin, "Research on personalized web service recommendation method based on improved cooperative e Filtering." *Comput. Tech. Development*. vol. 28, no. 1, Jan 2018.

[12] P. Zhang, L. Wang, S. Ji, and W. Li, "Prediction method of web service QoS based on multivariate time series." *J. Software*. vol. 30, no. 6, pp. 1742-1758. 2019.

[13] B. Tan, H. Ma, M. Yi, and M. Zhang, "Evolutionary multi-objective optimization for web service location allocation problem." *IEEE Trans. Service Computing.* Jan. 2018.

[14] Y. Mao, J. Liu, R. Hu, M. Dong, and M. Shi, "The Sigmoid function is adopted for web service cooperative filtering recommendation algorithm." *J. Frontiers Comput. Science & Tech.* vol. 11, no. 2, pp. 314-322. 2017.

[15] G. Zou, M. Jiang, S. Niu, H. Wu, S. Pang, and Y. Gan. "QoS-aware web service recommendation with reinforced cooperative filtering." *Int. Conf. Service-oriented Computing.* pp. 430-445. 2017.

[16] W. Tan Y. Zhao, X. Hu, L. Xu, A Tang, and T Wang, "A method towards web service combination for cross-organizational business process using QoS and cluster." *Enterprise Int. Syst.* vol. 13, no. 5, pp. 631-649. 2019.

[17] L. Yao, Q. Z. Shen, A. H. H. Ngu, J. Yu, and A. Segev. "Unified cooperative and content-based web service recommendation." *IEEE Trans. Services Computing.* vol. 8, no. 3, pp. 453-466, 2019.

[18] Y. Ma, S. Wang, P. C. K. Huang, C. H. Hsu, and F. Yang, "A highly accurate prediction algorithm for unknown web service QoS values." *IEEE Trans. Services Computing.* vol.99, no.4, pp.511-523. Aug. 2016.

[19] J. Xu, Z. Zheng, and M. R. Lyu, "Web service personalized quality of service prediction via reputation-based matrix factorization." *IEEE Trans. Rel.* vol.65, no.1, pp. 28-37, Aug. 2015.

[20] Z. Chen, L. Shen, and F. Li, "Your neighbors are misunderstood: on modeling accurate similarity driven by data range to cooperative web service QoS prediction." *Future Generation Comput. Syst.* vol.95, no. 6, pp. 404-419, 2019.

[21] D. Wang, Y. Yang, and Z. Mi, "A genetic-based approach to web service composition in geo-distributed cloud environment." *Comput. & Electrical Engineering.* vol. 43, pp. 129-141, Apr. 2014.

[22] Z. Liu, Z., D. H. Chu, Z. P. Jia, J. Q. Shen, and L. Wang, "Two-stage approach for reliable dynamic web service composition." *Knowledge-Based Syst.* vol. 97, pp. 123-143, Apr. 2016

[23] G. N. Rai, G. R. Gangadharan, V. Padmanabhan, and R. Buyya. "Web service interaction modeling and verification using recursive composition algebra." *IEEE Trans. Services Comput.* PP. 1-1. 1939.

[24] C. Lu, and J. Kou, "Multi-attribute decision making and adaptive genetic algorithm for web service composition QoS optimization." *Comput. Sci*. vol. 46, no. 02, pp. 196-204. 2019.

[25] J. Shen, C. Luo, Z. Hou, and Z. Liu, "Improved genetic algorithm for QoS-aware logistics web service combination." *J. Chin. Compu. Sys.* vol. 40, no. 01, pp. 38-41, Jan. 2019.

[26] H. Huang and J. Sun. "Web service composition selection algorithm based on improved BPSO." *Comput. Engineering.* vol. 37, no. 24, pp. 266-268. Dec. 2011.

[27] W. Tan, Z. Yao, and J. Ting, "Constraint-based QoS-aware web service composition in cross-organizational cooperation." *Comput. Engineering.* vol. 44, no. 11, pp. 73-81. Nov. 2018.

[28] Y. Xia, P. Cheng, J. Chen, X. Kong, and D. L, "Service composition optimization based on improved ant colony algorithm." *Chin. J. Comput*. vol. 35, no. 2., pp. 270-281, Jun. 2011.

[29] L. Liu, D. Yang, "Multi-objective genetic algorithm for service level-aware service combination problem." *J. of Jilin University (Engineering and Tech. Edition).* vol. 45, no. 1, pp. 267-273. Jun. 2015.

[30] A. Halfaoui, F. Hadjila, and F. Didi, "QoS-aware web services selection based on fuzzy dominance." *Comput. Sci. and Its App. Springer Int. Publishing.* 2015.

[31] X. Xu, H. Rong, E. Pereira, and M. Trovati, "Predatory search-based chaos turbo particle swarm optimization (PS-CTPSO): A new particle swarm optimization algorithm for Web service combination problems." *Future Generation Comput. Syst.* vol. 89. pp. 375-386. July. 2018.

[32] M. Ghobaei-Arani, A. A. Rahmanian, M. S. Aslanpour, and S. E. Dashti. "Csa-wsc: Cuckoo search algorithm for web service composition in cloud environments." *Soft Comput.* vol. 22, no. 24, pp. 8353–8378. Dec.2018.

[33] F. Dahan, H. Mathkour, and M. Arafah. "Two-step artificial bee colony algorithm enhancement for QoS-aware web service selection problem." *IEEE Access.* 2019.

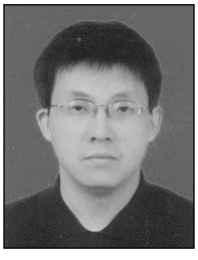
[34] L. Bao, C. Wu, X. Bu, N. Ren, and M. Shen, "Performance modeling and workflow scheduling of microservice-based applications in clouds." *IEEE Trans. Parallel and Distributed Syst*. Feb. 2019.

[35] M. Lin, J. Xi, W. Bai, and J. Wu, "Ant Colony Algorithm for Multi-Objective Optimization of Container-Based Microservice Scheduling in Cloud." *IEEE Access.* 2019.

[36] X. Wang, X. Guan, T. Wang, G. Bai, and B. Choi. "Application deployment using Microservice and Docker containers: Framework and Optimization." *J. Netw. Comput.* Appl. vol. 119, pp. 97-109. Oct. 2018.

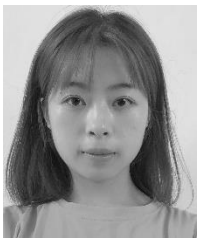
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2971379, IEEE Access

[37] C. Guerrero, I. Lera, and C. Juiz, "Resource optimization of container orchestration: A case study in multi-clouds microservices-based applications." *J. Supercomputing.* vol. 74, no. 7, pp. 2956-2983. July. 2018.
[38] C. Guerrero, I. Lera, and C. Juiz, "Genetic algorithm for multi-objective optimization of container allocation in cloud architecture." *J. Grid Comput.* vol.16, pp. 113-135. Mar. 2018.
[39] A. Jindal, V. Podolskiy, and M. Gerndt, "Performance modeling for cloud microservice applications." *10th ACM/SPEC ICPE.* pp. 25-32. Apr. 2019.
[40] R. Pietrantuono, R. Stefano, and G. Antonio, "Run-time reliability estimation of microservice architectures." *29th ISSRE.* pp. 25-35. Oct. 2018.
[41] V. Singh and K. P. Sateesh. "Container-based microservice architecture for cloud applications." *2017 IICCCA.* May. 2017.
[42] M. Villamizar et. al, "Infrastructure cost comparison of running web applications in the cloud using AWS Lambda and Monolithic and Microservice architectures." *2016 16th IEEE/ACM Int. Symposium CCGrid.* pp. 179-182. July. 2016.
[43] M. Gao, "Knowledge cooperative workflow modeling, service planning and service composition research." *Dongbei University of Finance and Economics*. 2013.
[44] H. Du, L. Jiao, and R. Liu, "Adaptive multi-clone programming algorithm with applications." *ICCIMA 2003. Proc. Fifth Int. Conf. IEEE.* pp. 350-355. 2013.
[45] L. Jiao, "The calculation, learning and recognition of immune optimization." Science Press. 2006.
[46] J. Feng, and L. Kong, "Research on Web service composition method based on fuzzy QoS and preference weight." *J. Chin. Mini-Micro. Comput.* Syst. 2012.
[47] T. Wang, and Y. Chen, "Applying fuzzy linguistic preference relations to the improvement of consistency of fuzzy AHP" *Infor. Sciences.* dol. 128, no. 19, pp. 3755-3765, 2008.
[48] S. Russell et al. "Artificial intelligence: a modern approach". *Englewood Cliffs, NJ: Prentice hall.* 1995.



Ming Gao received the B.E. degree in Management Information System from Department of Economic Information in Dongbei University of Finance and Economics (DUFE), Dalian, China, in 2002, M.S. degree in Information Management from DUFE, Dalian, China, in 2004, and Ph.D. in Information Technology and Management from School of Management Science and Engineering (SMSE) in DUFE in 2013. Now he is an associate Professor in SMSE at DUFE. His research interests include Business Process Management, Web Service Composition, Cloud Computing and Big-data Applications.

Mingxia Chen received the B.E. degree in Engineering management from Liaoning Technical University, Huludao, China, in 2014, is currently studying Management science and engineering from Dongbei University of Finance and Economics. Her research interests include Web Service Composition, Cloud Computing and Big-data Applications.



An Liu received the B.E. degree in Logistics management from Dalian University of Finance and Economics, Dalian, China, in 2014, is currently studying Management science and engineering from Dongbei University of Finance and Economics. Her research interests include Web Service Composition, Cloud Computing and Big-data Applications.

WAI HUNG IP received the LL.B. degree (Hons.) from the University of Wolverhampton, the M.Sc. degree in industrial engineering from Cranfield University, the MBA degree from Brunel University London, and the Ph.D. degree from Loughborough University, U.K.

He is currently a Principal Research Fellow with the Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, and a Professor Emeritus and an Adjunct Professor of Mechanical Engineering with the University of Saskatchewan, Canada.

Prof. Ip is a member of the Hong Kong Institution of Engineering and a senior member of IEEE.

KAI LEUNG YUNG received the B.Sc. degree in electronic engineering from Brighton University, in 1975, the M.Sc. and DIC degrees in automatic control systems from the Imperial College of Science, Technology and Medicine, University of London, in 1976, and the Ph.D. degree in microprocessor applications in process control from Plymouth University, U.K., in 1985. He became a Chartered Engineer (C.Eng., MIEE), in 1982.

He is currently an Associate Head and the Chair Professor of the Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University. His research

interests include space systems, mechatronics, robotics and automatons, AI, and big data.