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

An Approach to Resource and QoS-aware Services Optimal Composition in the Big Service and Internet of Things

Xunyou Min[1], Xiaofei Xu[1], (Member, IEEE), Zhizhong Liu[2], Dianhui Chu[1], and Zhongjie Wang[1], (Member, IEEE)

[1]School of Computer Science and Technology Harbin Institute of Technology, Harbin, China (e-mail: min_xun_you@sina.com, {xiaofei, rainy}@hit.edu.cn)
[2]College of Computer Science and Engineering Henan Polytechnic University Jiaozuo, Henan, China (e-mail: lzzmff@126.com)

Corresponding author: Xiaofei Xu (e-mail: xiaofei@hit.edu.cn).

We thank the editors and the anonymous reviewers for their helpful comments and suggestions that have led to this improved version of the paper. This work is supported by: the National Key Research and Development Program of China (No.2017YFB1400600), the National Science Foundation of China (No.61772155 and 61472106), and the National Natural Science Foundation Fund China (No.61772159).

ABSTRACT Recently, various types of services belonging to different domains intertwine together and constitute the Big Service. In the Big Service and Internet of Things, services optimal composition is a key technology to create value-added services to satisfy users' complex requests. However, massive services that possess the same functionalities but different quality of service (**QoS**) are emerging on the internet. Moreover, the online performance of many online services is determined by their distribute resources. Therefore, the expected performance of a composite service depends on the creation of the optimal composite service that can meet end-to-end quality requirements while ensuring that component services have sufficient resources to support their successful execution. To this end, resource and QoS-aware services optimal composition (**RQ-SOC**) becomes an important issue in the Big Service and Internet of Things. Moreover, with the evolution of service industries, service features in various service domains (**SFSD**) (priori features, correlation features and similarity features) are gradually formed. These SFSD have great influences on RQ-SOC. Thus, to effectively solve the RQ-SOC problem, this paper first defines the SFSD and describes the important influences of SFSD on RQ-SOC. Then, the improved artificial bee colony algorithm (**ABC**) for RQ-SOC is proposed, and a resource checking operator based on the analysis of the mutual relations between services and resources is presented. Third, the resources checking operator is integrated into the improved ABC to solve the RQ-SOC problem effectively. Finally, the experimental results show that the proposed method for RQ-SOC is feasible and effective.

INDEX TERMS Artificial Bee Colony Algorithm, Service, Service Optimal Composition, Service Domain Features, Internet of Things

I. INTRODUCTION

With the rapid development of new information technologies and computing paradigms (such as the internet of things (IoT) [1], big data [2], social network services [3], mobile networks, intelligent terminals, service computing [4] and cloud computing [5]), massive services from different domains are emerging on the internet. These services intertwine together and constitute the Big Service [6], which is a complex ecosystem with massive diverse services.

The Big Service brings together both virtual services (such as Web services, cloud services, SaaS, PaaS, IaaS, SNS, Internet of Tings services, etc.) and physical services (such as public transportation services, logistics services, enterprise services, medical services, human services, manufacturing services, financial services, etc.). In the Big Service, the concept of "service" in traditional service-oriented architecture is extended from software services to generalized "service" with the inclusion of network services, virtual services and entity services with good interoperability, self-organization and scalability.

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.2018.2855807, IEEE Access

The Big Service is customer focused so that when it receives a customer's request, it aggregates services across domains to create an optimal composite service on demand. The general process of creating composite services in the Big Service and Internet of Things consists of three layers, as illustrated as FIGURE 1.

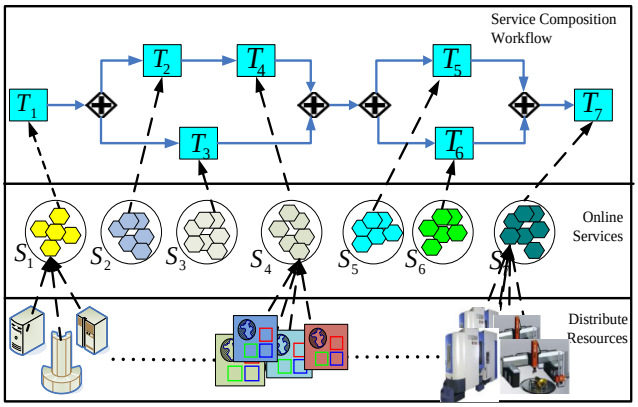

FIGURE 1: Schematic diagram of service composition in the Big Service and Internet of Things.

The first layer is the workflow layer, which concludes various service composition workflows that could satisfy users' different complex application requirements; the second layer is the online candidate services layer, where massive services exist for each task comprising the composition workflow. Candidate services can include Web services, SaaS, IaaS, PaaS, or other virtual services. Candidate services for the same task possess similar functionalities but possess different QoS. Services optimal composition is constructed in the second layer by aggregating several online services belonging to different domains according to the workflow in the first layer. The last layer is the resources layer in the Internet of Things, where distributed resource pools exist, which can provide suitable resources to support the execution of online services.

Currently, massive services with similar functionalities but different QoS are emerging; thus, QoS-aware services optimal composition has become more complex. Moreover, resources are the foundation of most of the online services. When the resources of online services are busy, the online performance of online services will be poor; when there are insufficient resources for the online services, then the online services will not work. Therefore, in the Big Service and Internet of Things, the availability and performance of online services depend on their distributed resources.

Most of the existing work on services optimal composition considers only how to construct the optimal schemes by binding a specific service to each task [7]–[9], with the result that the services optimal composition is not only able to satisfy user QoS global constraints but can also achieve the optimal QoS. Existing research assumes that the selected services always have enough resources in the Internet of Things to support their work. However, in reality, this is not the case because many services are virtual services, and the execution of these services requires the support of a certain amount of resources in the Internet of Things to satisfy the SLA between the service provider and the user. In fact, resources may not often exist or cannot be obtained for component services when being executed. This condition leads to low availability of composite service.

Therefore, the creation of composite services should consider not only how to create optimal service composition that can satisfy user functional and nonfunctional demands perfectly in the service layer but must also how to bind enough resources in the Internet of Things for component services to ensure their smooth implementation. Therefore, RQ-SOC becomes a key problem in the Big Service and Internet of Things. Moreover, with the rapid development and continuous evolution of service industries, SFSD are gradually formed, and these features have great influences on improving the effect and efficiency of RQ-SOC.

To effectively solve the RQ-SOC in the Big Service and Internet of Things, this paper innovatively proposes an effective method for RQ-SOC based on our improved ABC. First, SFSD are defined and then their important influences on RQ-SOC are deeply mined. Second, based on the SFSD, the ABC is improved for solving RQ-SOC, the mutual relations between services and resources are analyzed, and the resources checking operator is proposed. Finally, the resources checking operator is integrated into the improved ABC to find services with appropriate resources to complete a certain task.

The remainder of this paper is organized as follows. Section 2 reviews related works. In Section 3, a motivation example is presented. Section 4 describes the problem of RQ-SOC in the Big Service and Internet of Things. SFSD and then their important influences on RQ-SOC are all described in detail in Section 5. The improved ABC and resources checking operator are all presented in Section 6. An innovative approach for the RQ-SOC is described in Section 7. Section 8 shows the relevant experimental results. Finally, we conclude the research of this paper and then give an overview of future work in Section 9.

II. RELATED WORKS

In the research field of the service computing, service optimal composition(SOC) is a critical problem. Liu *et al*. [10] proposed an extensible service quality computation model that supports a fair and open management of QoS data by incorporating user feedback. However, the QoS-based composition problem is not addressed in their work. In another paper [11], the multidimensional multiple-choice knapsack problem was used to model this problem. In other works [12], [13], nonlinear and linear programming methods were respectively used to find the solution to this problem. Ardagna *et al*. [14] proposed an approach to the selection problem with QoS constraints in the global view. However, these methods suffer from poor scalability because of the exponential time complexity.

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.2018.2855807, IEEE Access

To improve scalability, evolutionary algorithms are frequently used to effectively solve the SOC. Liang *et al*. [15] provided a novel hybrid method to effectively solve the multi-constraint SOC. Zhou *et al*. [16] demonstrated a class of corresponding problems that are within the scope of the secure SOC changing to an NP-complete problem by converting these problems into the corresponding knapsack problem. The simulated annealing theory was presented and changed as a typical heuristic method to tackle the problem. Faruk *et al*. [17] proposed a novel enhanced GPSO to address the SOC, and the performance of the proposed algorithm was evaluated and exemplified. Huo *et al*. [18] formalized the SOC as a typical nonlinear integer programming and proposed the DGABC. This model simulates the process of searching for each corresponding solution. However, these SOC methods do not consider how to make use of SFSD to greatly improve the effect of SOC. Therefore, several research works on service composition task with the effective guidance of SFSD have arisen. In [19], statistic correlation and its important influence on SOC are analyzed, and an approach is proposed to greatly improve the corresponding performance. In [20], the formal descriptions of quality correlation and selection correlation are provided, and a novel selection approach that can perceive correlation is proposed. In [21], similarity measures are used to effectively measure how close two composite schemes are. In [22], a correlation and dot-pattern-based approach is proposed to select services. In [23], the recurring patterns in a composition process are mined using an approach according to these execution logs.

Research on SOC has obtained great progress and promoted the rapid development of cloud computing and service computing. However, the above review shows that the research works conducted so far employed only single SFSD to raise the optimization effect of SOC [24], [25], and research work employing all three SFSD to design an algorithm for solving SOC is still very rare. Moreover, most SOC methods never consider the important influence of resources on the execution composite services, leading to a lower success rate of SOC. Therefore, this paper aims to propose a novel approach for RQ-SOC.

III. MOTIVATION DISCUSSION

We use an example to show the importance of considering resource and SFSD in SOC. This composition process includes five service nodes: (a) the E-commerce node, (b) the payment node, (c) the manufacturing node, (d) the storage node, and (e) the logistics node.

Each node of the corresponding composition process has many candidate services. For example, the concrete candidate services of the E-commerce node are *JD.com*, *Suning*, *Alibaba*, etc., while the concrete candidate services of the manufacturing node are *FOUNDER*, *IBM*, *HP*, etc. When a request arrives, services coming from these service sets should be selected in order to construct an optimal composite scheme. To clarify the priori, we will discuss two scenarios below for the purchasing computer composite process. In

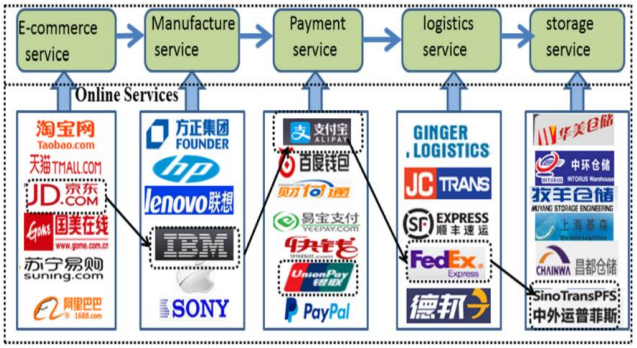

FIGURE 2: Computer purchasing through online services.

our first scenario, the final solutions frequently select *IBM* as the manufacture service (represented as S_{24}) through the E-commerce service *JD.com* (represented as S_{13}), pay money through the payment service PayPal (represented as S_{37}), apply the service SF (represented as S_{43}) for the logistics, and then use storage service *SinotransPFS* (represented as S_{56}). Thus, the composite scheme of this request is $<S_{13}, S_{24}, S_{37}, S_{43}, S_{56}>$. Assume that most users are all satisfied with the above composite scheme; then, $<S_{13}, S_{24}, S_{37}, S_{43}, S_{56}>$ could serve as a priori composite scheme. When a request which is similar to it comes, we could take $<S_{13}, S_{24}, S_{37}, S_{43}, S_{56}>$ as a composite solution directly. This behavior greatly improves the efficiency. In our second scenario, we assume that most composite schemes use the logistics service *SF* or *FedEx*, and the corresponding users are satisfied with them. Therefore, these services could serve as priori services of the delivery node. When a new request arrives, there will be a greater probability for these services to be chosen. This process would help the service system improve the efficiency to a certain extent. Therefore, the proposed priori is very useful for SOC.

Here, similarity refers to that a set of services possess the exact same functionalities as well as similar QoS. For example, considering the manufacturing services *FOUNDER* and *IBM*, both of them have the ability to produce computers and they have similar QoS. Suppose that users are satisfied with *FOUNDER* and *IBM*; if *FOUNDER* is unavailable, *IBM* could serve as a satisfactory alternative. Thus, once an unavailable priori service appears when composing the solution, replacing it with a similar one is a good approach. Correlation includes statistical correlation and business correlation. In statistical correlation, once a service is selected (e.g., *hp*), then a certain specific service would be selected with bigger probability (e.g., *SinoTransPFS*). Business correlation means contractual relationships that exist between services. For example, there is a strategic commercial agreement between *JD.com* and *PayPal*. If a user purchases goods through the JD Web portal, then he can use *PayPal* for free. Clearly, applying correlation could improve the performance of the composition algorithm.

Candidate services should have suitable resources to en-

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.2018.2855807, IEEE Access

sure their successful execution. Take the computer manufacture service S_2 for example; manufacturing resources should be sufficient for the manufacturing company according to the user requirement (such as computer's brand, quantity, supply time and so on). For the warehousing service S_5, for instance, appropriate storage space should be prepared for the selected storage service according to the storage requirement of the goods (such as the amount of computers, the location of the supplier, temperature, humidity and other environmental information). However, if there are not sufficient resources for the component services, even though the composite service <$S_{13}, S_{24}, S_{37}, S_{43}, S_{56}$> is constructed optimally, the process will not work. Therefore, when constructing optimal service composition in the Big Service, it is necessary to consider how to find the suitable service with appropriate resources.

IV. PROBLEM MODEL OF RESOURCES AND QOS-AWARE SERVICE OPTIMAL COMPOSITION

In the Big Service, resources information for each service can be specified in a configuration file. Resources information includes resource identifier, resource type, quantity of available resource, processing ability of one unit resource and resource location. The resource model is defined in Definition 1.

Definition 1. Resource Model:

$$RM =< ID, RT, ARQ, PoU, RL >, \tag{1}$$

where

ID – Resource Identifier, a serial number that denotes the number of the resource;

RT – Resource Type, a unique identifier that specifies the type of the resource;

ARQ – Available Resource Quantity, a positive integer number expressing how many units of the resource are available to be accessed;

PoU – Processing ability of unit resource, the process ability of unit resource;

RL – Resources Location, where the resources are deployed.

Each component service in the composite service should complete a certain task. Here, we define the task model in Definition 2.

Definition 2. Task Model:

$$TM =< TT, TA, PT >, \tag{2}$$

where

TT – Task Type, a unique identifier that specifies the type of the task;

TA – Task Amount, a positive integer number expressing how many units of the task to be processed;

PT – Processing Time, it indicates when the task will be processed.

In the Big Service, to construct optimal service composition and ensure successful execution of the optimal composite service, it is necessary to consider the states of resources of candidate services during the creation of optimal service composition. The problem of RQ-SOC is to select concrete services from each candidate service set so that the constructed composite service can achieve the global optimal service quality and meet the user-specified constraints while ensuring these component services have enough resources to support their successful execution. This problem can be defined as the following eq.(3):

$$RQ - SOC =< T, S, R, M, QC >, \tag{3}$$

where

- T=<$T_1, T_2, \cdots, T_M$> represents tasks comprising the service composition work-flow, where M is the total number of tasks. Note that the task here is different from the task in the "Task Model" section. Specifically, the task in the "Task Model" section refers to the task that should be completed by a certain component service in the composite service. For example, the task that should be completed by T_m (m=1, 2, $\cdots$, 7) in FIGURE 1. When each T_m (i.e., T_1 in FIGURE 1) completes its own task, it means that the RQ-SOC has already been completed successfully. The task here refers to the abstract component service in the composite service.
- S=<$S_1, \cdots, S_i, \cdots, S_m$> represents the service class for tasks, where S_i means candidate service set for task T_i and S_i=<$s_{i1}, s_{i2}, \cdots, s_{ij}, \cdots, s_{ik}$>; services s_{ij} in the same set have the same functionality and different QoS.
- R=<$R_{S_1}, \cdots, R_{S_i}, \cdots, R_{S_m}$> and R_{S_i}=<$r_{S_{i1}}, r_{S_{i2}}, \cdots, r_{S_{ij}}, \cdots, r_{S_{ik}}$> where $r_{S_{ij}}$ indicates the resource for service s_{ij} in service class S_i.
- M=<$M_1, \cdots, M_i, \cdots, M_m$> represents the amount of tasks that need to be completed for each task node.
- QC=<$C_1, \cdots, C_j, \cdots, C_p$> represents the global QoS constraints for the composite service.

The required resources amount (RRA) of S_i can be calculated according to the eq. (4):

$$RRA(S_i) = \frac{M_i}{PoU}, \tag{4}$$

where $RRA(S_i)$ is a positive decimal number that indicates how many units of resource will be required for service class S_i. Through comparing the required resources amount of S_i ($RRA(S_i)$) with available resources amount (ARA) of a service s_{ij} belonging to service class S_i, it can be determined whether service s_{ij} has sufficient resources to complete the task. Here, we can obtain ARA(s_{ij}) by analyzing $r_{s_{ij}}$.

When QS satisfy the following constraints:

$$\begin{cases} q_1 \leq C_1, \\ q_2 \leq C_2, \\ \cdots\cdots, \\ q_i \geq C_i, \\ \cdots\cdots, \\ q_k > C_k, \end{cases} \tag{5}$$

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.2018.2855807, IEEE Access

TABLE 1: Summary of abbreviations used in the paper.

Abbreviations	Implication
QoS	quality of service
RQ-SOC	resource and QoS-aware services optimal composition
SFSD	service features in various service domains
ABC	artificial bee colony algorithm
IoT	internet of things
SOC	service optimal composition
RRA(S_i)	the required resources amount of S_i
ARA(s_{ij})	available resources amount of a service s_{ij}
PrCS	priori composite schemes
PrS	priori service subset
SiS	similarity service subset
GeS	general service subset
sed	the colony's search direction
RCO	resource checking operator
PrSs	all the PrS set of task nodes
SiSs	all the SiS set of task nodes
GeSs	all the GeS set of task nodes
PoU	processing ability of unit resource
ID	resource identifier
RT	resource type
ARQ	available resource quantity
RL	resources location
TT	task type
TA	task amount
PT	processing time

the mathematic model of RQ-SOC is defined as the following eq.(6):

$$F(SOC) = \max(\sum_{i=1}^{k} w_i \times \frac{q_i^{max} - q_i}{q_i^{max} - q_i^{min}}), \quad (6)$$

where *SOC* indicates a service optimal composition, k indicates the number of QoS attributes, w_i indicates the user-specified weight for the *i*-th attribute, q_i indicates the aggregated value of the *i*-th attribute of *SOC*, q_i^{max} indicates the maximum value of the aggregated value of the ith attribute of *SOC*, and q_i^{min} indicates the minimum value of the aggregated value of the *i*-th attribute of *SOC*. The QoS aggregation formulas for a composite service were described previously [26].

V. SFSD AND THEIR INFLUENCES ON SERVICE OPTIMAL COMPOSITION

A. SFSD

Before the details, we show a summary of the abbreviations used in the paper in Table 1.

1) Priori

The priori means the experience and knowledge that is produced during the course of the service application. Specifically, with regard to a request class R, there often exist certain corresponding services (service schemes) that have higher frequency of use and better user satisfaction. The priori between service s and request class R is formally defined as:

$$Pr < s, R > \Leftrightarrow (PMF(s) \geq PMF_0) \wedge (USMF(s) \geq USMF_0)) \quad (7)$$

Here, $PMF()$ refers to a priori measure function, PMF_0 refers to the corresponding threshold, $USMF()$ refers to the measure function of user satisfaction, and $USMF_0$ refers to the corresponding threshold.

2) Correlation

Correlation is the commercially associated relationships during the mutual cooperation between upstream and downstream services. Suppose s_i and s_k are services in different domains; then, the correlation between them is formally defined as:

$$Co < s_i, s_k > \Leftrightarrow cp(s_i, s_k) \geq cp_0 \quad (8)$$

where $cp(s_i, s_k)$ is the collaboration probability between them and cp_0 means the corresponding threshold.

3) Similarity

In a service domain, there are many services that have the exactly same functionality as well as similar QoS; this universal phenomenon is named similarity. Suppose s_i and s_k are services in the same service domain; then, the similarity between them is determined by:

$$Simi < s_i, s_k > \Leftrightarrow (FJF(s_i) = FJF(s_k)) \wedge Diff_{QoS}(s_i, s_k) \leq Diff_0 \quad (9)$$

where $FJF()$ means the functional judgment function, $Diff_{QoS}(s_i, s_k)$ means the QoS diversity judgment function between them, and $Diff_0$ means the corresponding QoS diversity threshold.

B. SFSD'S INFLUENCES ON SERVICE OPTIMAL COMPOSITION

SFSD are important knowledge and objective law in the service domain; they have important influences on the construction of optimal composite schemes. Therefore, it is very necessary to apply them to construct the final solution. In the sections that follow, the influences of SFSD on creating the optimal solutions are analyzed in detail.

1) Influences of priori on SOC

First, the priori influences the initial population generation strategy, and certain high-quality initial individuals could be generated according to the priori. Second, the priori influences the search strategy of service space and the neighborhood search. With the priori, the service space could be artificially divided into many smaller spaces. This step is very useful to narrow the search to improve the efficiency. Moreover, heuristic novel neighborhood search could be designed under the proper guidance of priori, and this is very useful to avoid random or blind neighborhood search and improve the algorithm's search capability.

2) Influences of correlation on SOC

Correlation has important influences on generating initial solutions, computing the fitness of composite services 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.2018.2855807, IEEE Access

neighborhood searching. Correlation could bring the service composition closer to practical application.

3) Influences of similarity on SOC

Similarity influences the search strategy. With the priori knowledge and similarity knowledge, the solution space could be artificially divided into many small spaces, which is helpful for narrowing the algorithm's search space. Moreover, according to the similarity, heuristic novel neighborhood search could be designed to enhance the algorithm's search capability.

C. PREPROCESSING THE CANDIDATE SERVICE SPACE

When solving an SOC, we can classify the user service request as a request class R by its characteristics. In the service system, there usually exists a service workflow that corresponds to the R. Due to the priori, we can mine priori composite schemes (PrCS) for the service workflow and priori services for each task of the service workflow with respect to R. Due to the similarity, we can identify similar services with respect to priori services. Therefore, it is possible to divide the candidate service set of each task in the service workflow into three subsets: the priori Service SubSet (PrS), the Similarity Service SubSet (SiS) and the General Service SubSet (GeS). This step helps to improve the efficiency of the optimization algorithm and provides knowledge preparation for designing the optimization strategy with the guidance of domain knowledge. The definitions of the three service subspaces are presented as follows.

1) Priori Service SubSet (PrS)

$$PrS(T_m, R) = \{s_{m_k} | s_{m_k} \in S_m \wedge PMF(s_{m_k}) \geq PMF_0 \wedge (USMF(s_{m_k} \geq USMF_0))\}, \quad (10)$$

where, s_{m_k} is a concrete service within S_m for the task T_m. The service subset consisting of services which could satisfy eq.(10) is called the PrS.

2) Similarity Service SubSet (SiS)

$$SiS(T_m, R) = \{s_{m_r} | \forall s_{m_s} \in Prs(T_m, R), s_{m_r} \in S_m \wedge s_{m_r} \notin Prs(T_m, R) \wedge Diff_{QoS}(s_{m_r}, s_{m_s}) \leq Diff_0\}, \quad (11)$$

SiS is the service subset composed of services that have similar service quality to services within the PrS.

3) General Service SubSet (GeS)

$$GeS(T_m, R) = S_m - PrS(T_m, R) - SiS(T_m, R). \quad (12)$$

It is a subset of the whole service set S_m; that is, remove the services in PrS and the services in SiS from S_m.

In the PrS and the SiS, existing potential partial order among services is discovered. Establishing the partial order can provide domain knowledge guidance for the improved neighborhood search strategy designed in this paper and raise the efficiency of problem solving. In the PrS, this order between services on the basis of priori measure value of service and user satisfaction measure value is established as follows:

$$PO(s_{m_k}) = \lambda_1 PMF(s_{m_k}) + \lambda_2 USMF(s_{m_k}), \quad (13)$$

where s_{m_k} is a service that comes from PrS, λ_1 is the weight of priori measure value, and λ_2 is the weight of user satisfaction measure value.

In the SiS, to build the order, first, the service that has the optimum priori measure value within the PrS is found, which we call the optimum priori service. Second, we calculate the QoS similarity degrees between services within the SiS and the optimum priori service. Finally, the order is established based on the calculated QoS similarity degrees.

VI. IMPROVED ABC AND RESOURCES CHECKING OPERATOR

ABC is proposed as a swarm intelligence evolutionary algorithm which is enlightened by the searching process of bees [27]. The solution space is mapped to the food distribution environment, and each solution in the solution space is mapped to a certain food source. The solution's fitness is mapped to a food's nectar amount. Bees have three castes for the division of labor: employed bees, onlooker bees, and scout bees. Employed bees carefully exploit the food sources that they have previously explored, and then share the food's quality information with the corresponding onlooker bees. Once the onlooker bees obtain this information, they choose the suitable food source for exploiting. Once an employed bee abandons its food source, it turns into a scout one. Scout bees will search the environment randomly.

Studies [28]–[30] have proved that the performance of ABC is better than the performances of other algorithms (PSO, ACO, GA, DE) when solving certain complex problems. Therefore, this paper takes ABC as the algorithm basis, improves the key optimization strategies of ABC with the useful knowledge provided by SFSD, and proposes a resources checking operator, ultimately forming an algorithm for RQ-SOC. This section describes the improved ABC algorithm for SOC and the resources checking operator. We will present the algorithm for RQ-SOC in Section7.

A. IMPROVEMENTS ON ABC

1) Search strategy of service space

In ABC, bees search for new and better food sources in the whole solution space randomly, and this original search strategy does not make use of SFSD and leads to low efficiency. For the RQ-SOC, given the existence of the SFSD such as PrS, PrCS and SiS, taking the PrS and SiS as the key area of searching can reveal the user satisfied solution with high probability. As a result, the algorithm conducts the search in the PrS, SiS and GeS successively following the definite rule to increase the solution efficiency of the problem. Specifically, the proposed search strategy and its

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.2018.2855807, IEEE Access

corresponding rule is: bees search in the PrS first; if they can find the user satisfied solution in PrS, they do not need to search in the SiS or the GeS; otherwise, if they cannot find the user-satisfied solution in the PrS and the degree of the exploration and development of PrS is greater than the specified threshold, then they move to the SiS to continue the search. For the following search process, the algorithm conducts the search that implements gradual phased transfer in the order of PrS, SiS and GeS until the user-satisfied solution is found.

2) Improved initial population generation strategy

RQ-SOC's food source within the population is modeled as:

$$X_t =< s_{l_d}, \cdots, s_{f_a}, \cdots, s_{m_o} >, \tag{14}$$

where X_t represents a food that is an n-dimensional vector, and it is also a solution for RQ-SOC. For the RQ-SOC, we can generate initial high-quality foods by making full use of the priori. Certain initial foods could be produced based upon the PrCS, and this part accounts for a proportion of α. Then, other initial foods are randomly generated based upon all of the PrSs, and this part accounts for a proportion of $\beta \cdot \alpha + \beta$=1.

Assume the total number of foods is SN, generating initial foods based upon PrCS from which to select the top $\alpha * SN$ priori schemes as the initial foods according to the priori measure values of priori schemes. Generating initial foods based upon all of the PrSs involves randomly extracting a concrete service from each PrS of the task and constructing initial foods that could satisfy the global constraints.

3) Improved employed bees

Since partial order between services in the PrS and SiS always exists, an improved neighborhood search approach with a heuristic mechanism is proposed for the RQ-SOC. The proposed neighborhood search approach in the PrS and SiS is: first, determine the i-th dimension's search direction through the direction determination rule; then, along the direction and determined search step $\mathcal{X}$, select a concrete new service that is used to substitute for the original service, and finally acquire a food.

- Determine the direction for the search
  Assume that X_o is a food and the *i*-th dimension is s_{i_k}; we take it as the specified reference point and then select a service $s_{i_k+\mathcal{X}}$ on one side of s_{i_k} with step $\mathcal{X}$. We substitute it for s_{i_k} and then generate a food X'_o. Therefore, the search direction in PrS (SiS) is:

$$sed = \begin{cases} s_{i_k} \rightarrow s_{i_k+\mathcal{X}}, Fit(X_o) < Fit(X'_o), \\ s_{i_k+\mathcal{X}} \rightarrow s_{i_k}, Fit(X'_o) < Fit(X_o), \end{cases} \tag{15}$$

  Where, sed is the colony's search direction.
- Determine the step for the search
  The search step $\mathcal{X}$ could be adjusted adaptively along with the search to further improve the search speed. If the colony's search direction is not changed, the colony's search step is increased. The adjustment rule of colony's search step is:

$$\mathcal{X} = \mathcal{X} + \gamma, if\ sed\ is\ unchanged, \tag{16}$$

  where γ means a step-length increment, which is a round number, and $0 < \gamma < \mathcal{X}$. $\mathcal{X}$ and γ could be decided using the trial-and-error method. The above is the proposed search approach in the PrS and SiS, but we cannot find any available domain knowledge in the GeS; therefore, we still implement the traditional search method in GeS.

4) Improved scout bees

For the RQ-SOC with SFSD, the new scout bees' strategy is:

- If the scout bees of the algorithm are searching within their own PrSs, then a food based on these sets is generated.
- If they are searching within their own SiSs, then a food based on these sets is generated.
- If they are searching within their own GeSs, then, a food based on these sets is generated.

B. RESOURCE CHECKING OPERATOR

During the searching process, it is very necessary to check whether candidate services have enough available resources to support their successful execution. If the available resource of a concrete service cannot satisfy the requirement of the task, then this service will be filtered out. To find the composite scheme with the appropriate resources to ensure the successful execution, this paper proposes a resource checking operator to realize the resources-aware function in the searching process. The resource checking operator is the following Algorithm.1: The RCO will be integrated into the improved ABC, which is carried out after the operation of initial population generation, neighborhood search and scout bees of the improved ABC, to ensure the selected candidate services have enough resources to support their execution.

Algorithm 1 Resource Checking Operator (RCO).

Require: information of tasks to be processed; resources information of services;
Ensure: availabilities of services;
1: Compute required resource amount *RRA* of S_i;
2: Get the available resource amount ARA of the current candidate service;
3: **if** (**then** $RRA(S_i) \leq ARA(s_{ij})$) The current candidate service could be selected;
4: **end if**
5: **if** (**then** $RRA(S_i) > ARA(s_{ij})$) The current candidate service could not be selected;
6: **end if**
7: **Return** E_n;

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.2018.2855807, IEEE Access

VII. AN APPROACH TO THE RQ-SOC: S-ABCR$_{RSC}$

Based on the resource checking operator and the improved ABC, this paper proposes an approach to the RQ-SOC (named S-ABCR$_{RSC}$). S-ABCR$_{RSC}$ is described in Algorithm.2 and Algorithm.3.

Algorithm 2 RQ-SOC based on improved ABC (Part-1).

Require: PrCS, all of the PrSs, all of the SiSs, all of the GeSs, QoS values, QoS constraints, available resource amount of concrete services; required resource for each node; user QoS preferences, values of $SN, \alpha, \beta, \gamma, \mathcal{X}, \mathcal{E}$, Limit;
Ensure: The optimal composite scheme;
1: **Step1: Initialization**
2: Generate $\alpha * SN$ initial foods based upon the PrCS; carry out the resource checking operator; replace the component service which doesn't have enough resource with a concrete service which has enough resource;
3: Generate $\beta * SN$ initial foods based upon all of the PrSs with the resource checking operator;
4: **for** (**do** $k = 1$ to SN) initialize a tree T_i with only a leaf (the root);
5: **end for**
6: **Repeat**
7: **Step2: improved employed bees**
8: **while** the mining extent of the PrS is less than threshold $\mathcal{E}$ **do**
9: **for** $k = 1$ to SN **do**
10: Determine the searching direction with the eq.(15);
11: Determine the step with the eq.(16);
12: Generate a food with the proposed neighborhood search approach; carry out resource checking operator to ensure each component service has suitable resources to support its execution; evaluate ;
13: Compare the new one with the old one and keep the better one;
14: **if** (k-th food is not improved) **then** $Count_k$= $Count_k$+1;
15: **end for**
16: **end while**
17: **while** the mining extent of the SiS is less than threshold $\mathcal{E}$ **do**
18: **for** $k = 1$ to SN **do**
19: Determine the searching direction with the eq.(15);
20: Determine the step with the eq.(16);
21: Generate a food with the proposed neighborhood search approach; Carry out resource checking operator to ensure each component service has suitable resources to support its execution; evaluate the new one;
22: Compare the new one with the old one and keep the better one;
23: **if** (k-th food is not improved) **then** $Count_k$= $Count_k$+1;
24: **end for**
25: **end while**
26: **while** the searching space == GeS **do**
27: **for** (**do** $k = 1$ to SN)
28: Generate a food with resource checking operator; evaluate the new one;
29: Compare the new one with the old one and keep the better one;
30: **if** (k-th food is not improved) **then** $Count_k$= $Count_k$+1;
31: **end for**
32: **end while**

Algorithm 3 RQ-SOC based on improved ABC (Part-2)

1: **Step3: Onlooker bees**
2: **for** $k = 1$ to SN **do**
3: Compute the choice probability and select a food randomly with the roulette method; execute the same operation as employed bee;
4: **end for**
5: **Step4: Improved scout bees**
6: **for** $k = 1$ to SN **do**
7: **if** (**then** $Count_k$>Limit && the searching space ==PrSs)
8: Generate a food randomly based upon all of the PrSs with the resource checking operator;
9: **end if**
10: **if** (**then** $Count_k$>Limit && the searching space ==SiSs)
11: Generate a food randomly based upon all of the SiSs with the resource checking operator;
12: **end if**
13: **if** (**then** $Count_k$>Limit && the searching space ==GeSs)
14: Generate a food randomly based upon all of the GeSs with the resource checking operator;
15: **end if**
16: **end for**
17: **Step:5 Record the best scheme**
18: **if** (**then** X_o is superior to the memorized best solution)
19: X_B=X_o // $X_o$is the current best scheme
20: **end if**
21: **Until**
22: **Step6: termination condition judgment**
23: **if** (**then** the pre-defined termination condition is satisfied)
24: stop computing and return the global optimal scheme;
25: **end if**

VIII. EXPERIMENT

A. EXPERIMENTAL DESIGN

In the experiment, a sequential SOC process extending from the process shown in FIGURE 2 and consisting of nine tasks is used as the experimental object, which is illustrated as FIGURE 3. Each task of the workflow has 1000 candidate services. In the experiment, three service quality attributes

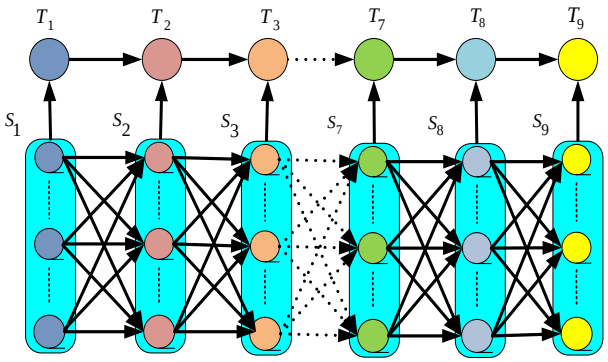

FIGURE 3: SOC process.

(Cost, Response Time, and Reliability) are considered, and the value ranges of these attributes are generated in [0,20], [0,100] and [0.75,1] randomly. All the quality values are all

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.2018.2855807, IEEE Access

normalized to [0.0, 1.0], and the cost type attributes are transferred to the benefit type attributes. The required resource of each task is randomly generated in [10, 80]. The available resource of each service is generated in [0, 85]. We assume there are 20 services in each service set that have correlations with their downstream service class, and each service has correlations with 40 services. The aggregated QoS values of these attributes of two correlated services will be reduced to 5, 10 and 0.001. We assume that the user preferences for these three attributes are 0.4, 0.3 and 0.3 respectively.

D-ABC [31] is iterated a certain number of times to generate 100 feasible composite schemes, and we randomly select 20 schemes and take them as the PrCS. Then, based upon the above definition of PrS, SiS and GeS and these 100 feasible schemes, we obtain all of the PrSs, SiSs and GeSs. The cardinality of these sets is 20, 40 and 1000, respectively. In the experiment, all of the PrSs, SiSs and GeSs are all stored in text files, and these useful data could be read directly by S-ABCR$_{RSC}$ whenever they are needed. In the experiment, the corresponding parameters are configured: $SN = 100, \alpha = 0.2, beta = 0.8$, Limit=3, and $\mathcal{E} = 0.7$. S-ABCR$_{RSC}$ is programmed in Java, and the PC's configuration is: OS: Microsoft Windows 7; CPU: Intel(R) Core(TM) i5-4570 @ 3.20 GHZ; Memory:4.00 GB.

B. VALIDATION OF THE RESOURCE PERCEPTION

To validate the effect of sensing resources in increasing the success rate of SOC, ten SOC requests are considered, in this condition, each task has the different resource requirements and each candidate service has the different available resources. The above ten different resource requirements and ten different available resources are produced randomly, as illustrated in Table 2.

TABLE 2: The ranges of the ten RRA and ARA

R&A1	R&A2	R&A3	R&A4
RRA:[10,15] ARA:[0,20]	RRA:[20,35] ARA:[10,40]	RRA:[15,28] ARA:[5,30]	RRA:[28,35] ARA:[20,40]
R&A5	R&A6	R&A7	R&A8
RRA:[30,45] ARA:[33,50]	RRA:[32,48] ARA:[27,55]	RRA:[28,48] ARA:[40,60]	RRA:[33,42] ARA:[30,55]
R&A9	R&A10		
RRA:[20,60] ARA:[25,50]	RRA:[45,80] ARA:[50,85]		

SOC with the ten resource requirements and available resource ranges are solved by S-ABC$_{SC}$ and S-ABCR$_{RSC}$. S-ABC$_{SC}$ means the removal of RCO from S-ABCR$_{RSC}$, that is, without considering resources in solving the ten SOC. In the experiment, we find 100 solutions for each SOC with S-ABC$_{SC}$ and S-ABCR$_{RSC}$ when they are iterated 100 times. Then, we compute the composition failure rate, which means the percentage of solutions which do not satisfy the resources requirement. FIGURE 4 provides the results, where the vertical axis represents the composition failure rate and the horizontal axis denotes the serial number of resource requirements.

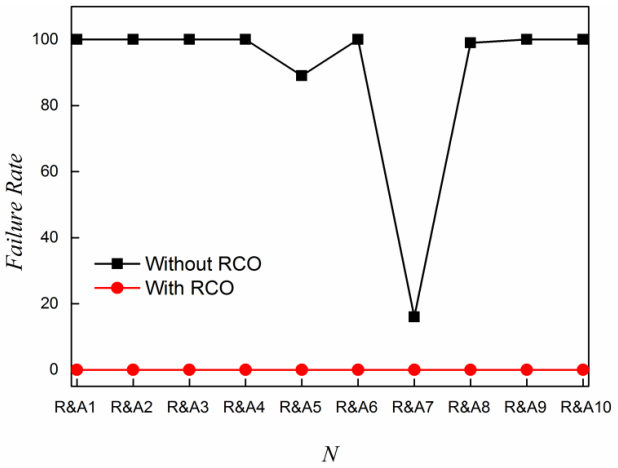

FIGURE 4: Validation of resource perception.

From FIGURE 4, it could find that most of the composition failure rates of the ten SOC solved by S-ABC$_{SC}$ are 100%, and the composition failure rates of the ten SOC solved by S-ABCR$_{RSC}$ are 0%. From these results we could find that considering resources in SOC is very useful for increasing the success rate of SOC.

C. VERIFICATION OF PRIORI

In the experiment, the important influence of the priori on the SOC solving is verified. Here, D-ABC serves to create the PrCS and all of the PrSs for the SOC. The iterative number of D-ABC indicates the intensity of priori knowledge, and the iterative number of S-ABCR$_{RSC}$ is 50. Then, D-ABC is iterated for different numbers of times for generating PrCS and all of the PrSs.Then, we apply S-ABCR$_{RSC}$ to conduct the SOC with different PrCS and PrSs. FIGURE 5 is the results, where the vertical axis is the fitness function value of solutions of S-ABCR$_{RSC}$, and the horizontal axis denotes different iterative numbers of D-ABC.

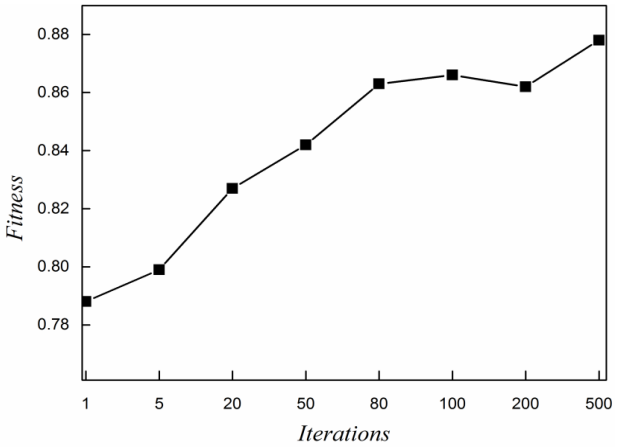

FIGURE 5: Verification of priori.

From FIGURE 5, we could find that along with the gradual increase in the priori intensity, the solutions of S-ABCR$_{RSC}$

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.2018.2855807, IEEE Access

become increasingly better. From these experiment results we could find that the intensity of priori has a considerable impact on the algorithm's solving process; abundant priori knowledge is very helpful for increasing the effect of SOC algorithm.

D. VERIFICATION OF CORRELATION

In the experiment, the important influence of correlation on the SOC solving is verified. Here, S-ABCR$_{RSC}$ with considering correlation and S-ABCR$_{RSC}$ without considering correlation are all applied to conduct the same SOC. FIGURE 6 is the results, where the vertical axis is the fitness function value of solutions of S-ABCR$_{RSC}$ with or without considering correlation and the horizontal axis denotes different iterative number of S-ABCR$_{RSC}$.

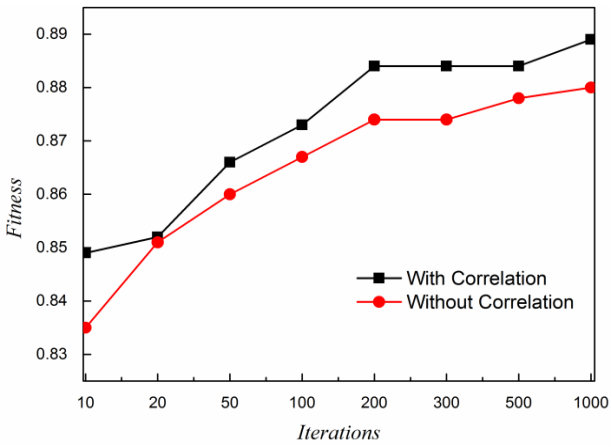

FIGURE 6: Verification of correlation.

From FIGURE 6, we could find that the fitness function values of solutions of S-ABCR$_{RSC}$ with considering correlation are superior to those found by S-ABCR$_{RSC}$ without considering correlation when the iterative number is the same. These experimental results reveal that considering correlation is very helpful for increasing the effect of SOC.

E. PERFORMANCE COMPARISON

To validate the optimization performance of S-ABCR$_{RSC}$, this experiment takes S-ABCR$_{RSC}$, discrete ABC (D-ABC) and genetic algorithm with elite reservation (GA) to conduct the same SOC. These algorithms are all realized with JAVA, and the implementation platform of these algorithms is identical. The configuration of D-ABC is: set $SN = 100$, the initial individuals are all randomly generated based upon the concrete service sets, set Limit=3. For GA, the population number is 100 and the mutation operation probability and crossover operation probability are 0.5 and 0.8. PrS is produced by iterating the D-ABC 1000 times.

In the experiment, these algorithms are all executed independently. FIGURE 7 is the results, where the vertical axis is the fitness function value of solutions of these algorithms and the horizontal axis denotes different repetition times.

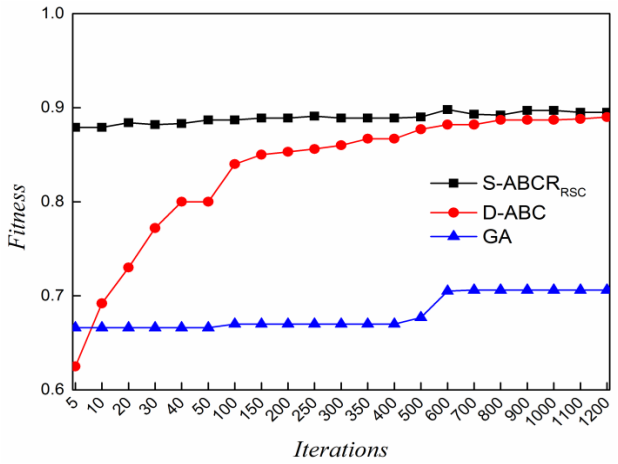

FIGURE 7: Performance comparison.

FIGURE 7, we could find that when the number of iterations is the same, the solutions provided by S-ABCR$_{RSC}$ are superior to the solutions provided by GA and D-ABC. The solutions provided by these algorithms and the corresponding running times are all shown in Table 3.

TABLE 3: Solutions provided by these algorithms and their corresponding running times

S-ABCR$_{RSC}$.		D-ABC		GA	
Fitness	T(ms)	Fitness	T(ms)	Fitness	T(ms)
0.879	13	0.73	13	0.66	13
0.882	19	0.8	18	0.67	24
0.889	52	0.853	54	0.67	51
0.898	148	0.882	147	0.706	142
0.895	285	0.89	270	0.706	268

From Table 3, we could find that S-ABCR$_{RSC}$ could find better schemes with shorter running times. The fitness function value of the best solution of S-ABCR$_{RSC}$ is 0.898 when it runs for 148 microseconds. The fitness function value of the best solution of D-ABC is 0.89 after it runs for 270 microseconds. The fitness function value of the best solution of GA is 0.706 after it runs for 142 microseconds. From these experiment results, we could find that both GA and D-ABC perform worse than S-ABCR$_{RSC}$ in solving RQ-SOC. In fact, there are many fruits about this topic [32]–[42].

F. CONCLUSIONS AND FUTURE WORK

In the Big Service, SOC is a key technology to create value-added services to satisfy users' complex requests. As massive services that possess the same functionalities and different service qualities are emerging, QoS-aware SOC becomes increasing complex. Resources are the foundation of most of the online services and are an important factor in deciding whether a service would execute a given task. Therefore, RQ-SOC becomes a key problem in the Big Service. Moreover, with rapid development and continuous evolution of service industries, SFSD are gradually formed, and these features have great influences on improving the effect and efficiency

of SOC. To effectively solve RQ-SOC in the Big Service, this paper first improves the ABC based upon the SFSD; then, we propose a resources checking operator and finally integrate this operator into the improved ABC to form an algorithm named S-ABCR$_{RSC}$ for RQ-SOC. Experimental results show that S-ABCR$_{RSC}$ is feasible and effective. It's also give a reference to many Internet of Things research field [43]–[53].

Future work will include the design of a more flexible resource checking operator to address highly variable user requirements and highly dynamic resources. Continue to discover more domain features and then deeply analyze the distinctive influences of each feature on service domain optimization problems to guide appropriate design and improvement of the corresponding solving algorithms. Moreover, our future work will further refine the elements of ABC to check which elements(not including the elements in the paper such as search strategy of service space, initial population generation strategy, employed bees and scout bees) can be improved. We will apply S-ABCR$_{RSC}$ to solve RQ-SOC in several practical application domains(e.g., smart home services, smart logistics services, smart city services, and smart healthcare services) to further verify the performance and practical value of S-ABCR$_{RSC}$.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," Computer networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] A. McAfee, E. Brynjolfsson, T. H. Davenport, D. Patil, and D. Barton, "Big data: the management revolution," Harvard business review, vol. 90, no. 10, pp. 60–68, 2012.

[3] S. J. Yu, "The dynamic competitive recommendation algorithm in social network services," Information Sciences, vol. 187, no. 1, pp. 1 – 14, 2012.

[4] M. P. Papazoglou, "Service -oriented computing: Concepts, characteristics and directions," Roma, Italy, 2003, pp. 3 – 12.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, pp. 50 – 58, 2010.

[6] X. Xu, Q. Z. Sheng, L.-J. Zhang, Y. Fan, and S. Dustdar, "From big data to big service," Computer, vol. 48, no. 7, pp. 80 – 83, 2015.

[7] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decade's overview," Information Sciences, vol. 280, pp. 218 – 238, 2014.

[8] V. Gabrel, M. Manouvrier, and C. Murat, "Web services composition: Complexity and models," vol. 196, 2015, pp. 100 – 114.

[9] M. Garriga, A. Flores, A. Cechich, and A. Zunino, "Web services composition mechanisms: A review," IETE Technical Review (Institution of Electronics and Telecommunication Engineers, India), vol. 32, no. 5, pp. 376 – 383, 2015.

[10] Y. Liu, A. H. Ngu, and L. Zeng, "Qos computation and policing in dynamic web service selection," New York, NY, United states, 2004, pp. 66 – 73.

[11] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," ACM Transactions on the Web, vol. 1, no. 1, 2007.

[12] V. Cardellini, E. Casalicchio, V. Grassi, and F. L. Presti, "Flow-based service selection forweb service composition supporting multiple qos classes," in IEEE International Conference on Web Services (ICWS 2007), July 2007, pp. 743–750.

[13] D. A. Menasce, E. Casalicchio, and V. Dubey, "On optimal service selection in service oriented architectures," Performance Evaluation, vol. 67, no. 8, pp. 659 – 675, 2010.

[14] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," IEEE Transactions on Software Engineering, vol. 33, no. 6, pp. 369 – 384, 2007.

[15] Z. Liang, H. Zou, F. Yang, and R. Lin, "A hybrid approach for the multi-constraint web service selection problem in web service composition," Journal of Information and Computational Science, vol. 9, no. 13, pp. 3771 – 3781, 2012.

[16] B. Zhou, D. Llewellyn-Jones, Q. Shi, M. Asim, M. Merabti, and D. Lamb, "Secure service composition adaptation based on simulated annealing," in 6th Layered Assurance Workshop. Citeseer, 2012, p. 49.

[17] M. N. Faruk, G. L. V. Prasad, and G. Divya, "A genetic pso algorithm with qos-aware cluster cloud service composition," vol. 425, Trivandrum, India, 2016, pp. 395 – 405.

[18] Y. Huo, Y. Zhuang, J. Gu, S. Ni, and Y. Xue, "Discrete gbest-guided artificial bee colony algorithm for cloud service composition," Applied Intelligence, vol. 42, no. 4, pp. 661 – 678, 2015.

[19] H. f. Li, R. Jiang, and S. y. Ge, "Researches on manufacturing cloud service composition & optimization approach supporting for service statistic correlation," in The 26th Chinese Control and Decision Conference (2014 CCDC), May 2014, pp. 4149–4154.

[20] Q. Wu, Q. Zhu, and M. Zhou, "A correlation-driven optimal service selection approach for virtual enterprise establishment," Journal of Intelligent Manufacturing, vol. 25, no. 6, pp. 1441 – 1453, 2014.

[21] M. Bravo, "Similarity measures for web service composition models," International Journal on Web Service Computing, vol. 5, no. 1, p. 1, 2014.

[22] M. Zhang, B. Zhang, J. Na, X. Zhang, and Z. Zhu, "Composite service selection based on dot pattern mining," in 2009 Congress on Services - I, July 2009, pp. 740–746.

[23] R. Tang and Y. Zou, "An approach for mining web service composition patterns from execution logs," in 2010 12th IEEE International Symposium on Web Systems Evolution (WSE), Sept 2010, pp. 53–62.

[24] Z. Huang, J. Huai, X. Liu, and J. Zhu, "Business process decomposition based on service relevance mining," in 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, vol. 1, Aug 2010, pp. 573–580.

[25] O. A. H. Hassan, L. Ramaswamy, and J. A. Miller, "Mace: A dynamic caching framework for mashups," in 2009 IEEE International Conference on Web Services, July 2009, pp. 75–82.

[26] Z. Z. Liu, Z. P. Jia, X. Xue, and J. Y. An, "Reliable web service composition based on qos dynamic prediction," Soft Computing, vol. 19, no. 5, pp. 1409 – 1425, 2015.

[27] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, Tech. Rep., 2005.

[28] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: Artificial bee colony (abc) algorithm and applications," Artificial Intelligence Review, vol. 42, no. 1, pp. 21 – 57, 2014.

[29] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," Applied Mathematics and Computation, vol. 214, no. 1, pp. 108 – 132, 2009.

[30] B. Akay and D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization," Information Sciences, vol. 192, pp. 120 – 142, 2012.

[31] Q.-K. Pan, M. Fatih Tasgetiren, P. Suganthan, and T. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," Information Sciences, vol. 181, no. 12, pp. 2455 – 2468, 2011.

[32] H. Lian, W. Qiu, D. Yan, Z. Huang and J. Guo, "Efficient Privacy-Preserving Protocol for k-NN Search over Encrypted Data in Location-Based Service," Complexity, vol. 2017, 2017, Art. no. 1490283.

[33] Y. Xu, L. Qi, W. Dou and Jiguo Yu, "Privacy-Preserving and Scalable Service Recommendation Based on SimHash in a Distributed Cloud Environment," Complexity, vol. 2017, 2017, Art. no. 3437854.

[34] Y. H. Chen and C. Y. Chen, "Service oriented cloud vm placement strategy for internet of things," IEEE Access, vol. 5, pp. 25 396–25 407, 2017.

[35] M. Sun, Z. Shi, S. Chen, Z. Zhou, and Y. Duan, "Energy-efficient composition of configurable internet of things services," IEEE Access, vol. 5, pp. 25 609–25 622, 2017.

[36] G. Tian, H. Zhang, Y. Feng, H. Jia, C. Zhang, Z. Jiang, Z. Li, and P. Li, "Operation patterns analysis of automotive components remanufacturing industry development in china," Journal of Cleaner Production, vol. 164, pp. 1363 – 1375, 2017.

[37] J. Ma, W. Di and H. Ren, "Complexity Dynamic Character Analysis of Retailers Based on the Share of Stochastic Demand and Service," Complexity, vol. 2017, 2017 Art. no. 1382689.

[38] R. H. Zhang, Z. C. He, H. W. Wang, F. You, and K. N. Li, "Study on self-tuning tyre friction control for developing main-servo loop integrated chassis control system," IEEE Access, vol. 5, pp. 6649–6660, 2017.

[39] L. Varga, M. Robinson, P. Allen, "Multiutility service companies: A complex systems model of increasing resource efficiency," Complexity, vol. 21, no. S1, pp. 23-33, 2016.

[40] B. Yu, Z.-Z. Yang, and B. Yao, "An improved ant colony optimization for vehicle routing problem," European Journal of Operational Research, vol. 196, no. 1, pp. 171 – 176, 2009.

[41] P. Y. Chen, "Optimal Retail Price Model for Partial Consignment to Multiple Retailers," Complexity, vol. 2017, 2017, Art. no. 1972532.

[42] M. Pouryazdan, C. Fiandrino, B. Kantarci, T. Soyata, D. Kliazovich, and P. Bouvry, "Intelligent gaming for mobile crowd-sensing participants to acquire trustworthy big data in the internet of things," IEEE Access, vol. 5, pp. 22 209–22 223, 2017.

[43] A. Li, X. Ye, and H. Ning, "Thing relation modeling in the internet of things," IEEE Access, vol. 5, pp. 17 117–17 125, 2017.

[44] D. Zhang, J. Jiang, S. Li, X. Li and Qingwen Zhan, "Optimal Investment Timing and Size of a Logistics Park: A Real Options Perspective," Complexity, vol. 2017, 2017, Art. no. 2813816.

[45] A. Taieb, M. Soltani and A. Chaari, "Parameter Optimization of MIMO Fuzzy Optimal Model Predictive Control By APSO," Complexity, vol. 2017, 2017, Art. no. 5813192.

[46] A. Lombardi, S. Tangaro, R. Bellotti, A. Bertolino, G. Blasi, G. Pergola, P. Taurisano and C. Guaragnella, "A Novel Synchronization-Based Approach for Functional Connectivity Analysis," Complexity, vol. 2017, 2017, Art. no. 7190758.

[47] B. Yu, W. H. Lam, and M. L. Tam, "Bus arrival time prediction at bus stop with multiple routes," Transportation Research Part C: Emerging Technologies, vol. 19, no. 6, pp. 1157 – 1170, 2011.

[48] K. Liu, J. Wang, T. Yamamoto, and T. Morikawa, "Exploring the interactive effects of ambient temperature and vehicle auxiliary loads on electric vehicle energy consumption," Applied Energy, 2017.

[49] R. Zhang, J. Wu, L. Huang, and F. You, "Study of bicycle movements in conflicts at mixed traffic unsignalized intersections," IEEE Access, vol. 5, pp. 10 108–10 117, 2017.

[50] Y. X. Zhao, Y. S. Su, and Y. C. Chang, "A real-time bicycle record system of ground conditions based on internet of things," IEEE Access, vol. 5, pp. 17 525–17 533, 2017.

[51] S. K. Bolisetti, M. Patwary, A. H. Soliman, and M. Abdel-Maguid, "Rf sensing based target detector for smart sensing within internet of things in harsh sensing environments," IEEE Access, vol. 5, pp. 13 346–13 363, 2017.

[52] G. Tian, H. Zhang, Y. Feng, D. Wang, Y. Peng, and H. Jia, "Green decoration materials selection under interior environment characteristics: A grey-correlation based hybrid mcdm method," Renewable and Sustainable Energy Reviews, vol. 81, pp. 682 – 692, 2018.

[53] G. Tian, M. Zhou, and P. Li, "Disassembly sequence planning considering fuzzy component quality and varying operational cost," IEEE Transactions on Automation Science and Engineering, vol. 15, no. 2, pp. 748–760, April 2018.