Parameter Tuning for ABC-based Service Composition with End-to-End QoS Constraints

Ruilin Liu, Zhongjie Wang, Xiaofei Xu
School of Computer Science and Technology
Harbin Institute of Technology
Harbin, China
gavinliuruilin@hotmail.com, {rainy, xiaofei}@hit.edu.cn

—QoS-aware service composition problem has been drawn great attentions in recent years. As an NP-hard problem, high time complexity is inevitable if global optimization algorithms (such as integer programming) were adopted. Researchers applied various evolutionary algorithms to decrease the time complexity by looking for near optimum solution. However, each evolutionary algorithm has two or more parameters the value of which is to be assigned by algorithm designers and likely has impacts on the optimization results (primarily time complexity and optimality). Our experiments show that there are some dependencies between the features of service composition problems, the value of the evolutionary algorithm's parameters, and the optimization results. In this paper, we use a popular evolutionary algorithm Artificial Bee Colony (ABC) to solve service composition problem and focus on the ABC's parameter turning issue. The objective is to identify the potential dependency to help service composition algorithm designers easily set up the values of ABC parameters to obtain preferable composition solution without many times of tedious attempts. Five features of service composition problem, three ABC parameters and two metrics of the final solution are identified. Based on a large volume of experiment data, ABC parameter tuning for a given service composition problem is conducted using C4.5 algorithm and the dependency between problem features and ABC parameters are established using multiple linear regression method. An experiment on a validation dataset shows the feasibility of our approach.

Keywords-QoS-aware service composition; Artificial Bee Conoly (ABC) algorithm; parameter tuning; C4.5 algorithm

I. INTRODUCTION

In the service era, people's lives rely more and more on the services delivered via Internet. An evident trend is that user requirements are increasingly complicated and no single service could completely fulfill a coarse-grained requirement [1]. It is necessary to compose the functionalities of multiple web services. This is called "*service composition* [2][3][4]". A typical service composition problem is to consider a set of end-to-end QoS constraints (local and global) raised by customers and look for an optimal composite solution to satisfy these QoS constraints, or called QoS-aware service composition problem [5][6].

A key issue in solving QoS-aware service composition problem is the time complexity. This is because the number of candidate Web services is likely to be large [7], and the selection of the best services from candidates to compose a solution that meets the QoS constraints is time-consuming and has been proven to be NP-hard [8]. Researchers have proposed various algorithms to decrease the time complexity, such as skyline-based approach [9], approximation-based approaches [10], evolutionary algorithm based approaches, etc. Among them, evolutionary algorithm shows better performance and has been widely applied.

There are many frequently used evolutionary algorithms, such as genetic algorithm [11], Particle Swarm Optimization [12]. Among them, Artificial Bee Colony (ABC) algorithm is the simplest but a relatively powerful one [13][16]. It has only three control parameters, and the procedure of ABC is simple to understand and implement [14][15]. Researchers have applied ABC into service composition problems and have confirmed its superiority compared with other evolutionary optimization algorithms [16]. This paper uses the TBA approach presented by [16].

However, existing ABC researches have not fully addressed a key issue, namely, *parameter tuning*. More specifically, how to assign appropriate value to the three control parameters? In the past experiments, we have found there is a certain dependency between the parameters' values and the final optimization results. Still, current ABC algorithm designers have to rely on their own experiences to manually and casually assign values to the parameters.

Akay and Karaboga studied the ABC parameter tuning problem in [15] and have got some positive conclusions, e.g.:

> As ABC uses and exploitative process to efficiently converge minima and an explorative process to ensure sufficient diversity in the population, it does not need a large *colony size* (or called *SN*) to solve optimization problems with high dimensions.
>
> The second parameter *Limit* dictates the occurrence of scout bees that are responsible for ensuring the diversity of the population. For relatively small-sized populations, *Limit* should not be low to avoid the situation that there are many randomly-generated solutions in the present populations. However, since large populations have sufficient diversity, *Limit* does not stand out in these situations.

Parameter Tuning for ABC-based Service Composition with End-to-End QoS Constraints

Ruilin Liu, Zhongjie Wang, Xiaofei Xu
School of Computer Science and Technology
Harbin Institute of Technology
Harbin, China
gavinliuruilin@hotmail.com, {rainy, xiaofei}@hit.edu.cn

—QoS-aware service composition problem has been drawn great attentions in recent years. As an NP-hard problem, high time complexity is inevitable if global optimization algorithms (such as integer programming) were adopted. Researchers applied various evolutionary algorithms to decrease the time complexity by looking for near optimum solution. However, each evolutionary algorithm has two or more parameters the value of which is to be assigned by algorithm designers and likely has impacts on the optimization results (primarily time complexity and optimality). Our experiments show that there are some dependencies between the features of service composition problems, the value of the evolutionary algorithm's parameters, and the optimization results. In this paper, we use a popular evolutionary algorithm Artificial Bee Colony (ABC) to solve service composition problem and focus on the ABC's parameter turning issue. The objective is to identify the potential dependency to help service composition algorithm designers easily set up the values of ABC parameters to obtain preferable composition solution without many times of tedious attempts. Five features of service composition problem, three ABC parameters and two metrics of the final solution are identified. Based on a large volume of experiment data, ABC parameter tuning for a given service composition problem is conducted using C4.5 algorithm and the dependency between problem features and ABC parameters are established using multiple linear regression method. An experiment on a validation dataset shows the feasibility of our approach.

Keywords-QoS-aware service composition; Artificial Bee Conoly (ABC) algorithm; parameter tuning; C4.5 algorithm

I. INTRODUCTION

In the service era, people's lives rely more and more on the services delivered via Internet. An evident trend is that user requirements are increasingly complicated and no single service could completely fulfill a coarse-grained requirement [1]. It is necessary to compose the functionalities of multiple web services. This is called "*service composition* [2][3][4]". A typical service composition problem is to consider a set of end-to-end QoS constraints (local and global) raised by customers and look for an optimal composite solution to satisfy these QoS constraints, or called QoS-aware service composition problem [5][6].

A key issue in solving QoS-aware service composition problem is the time complexity. This is because the number of candidate Web services is likely to be large [7], and the selection of the best services from candidates to compose a solution that meets the QoS constraints is time-consuming and has been proven to be NP-hard [8]. Researchers have proposed various algorithms to decrease the time complexity, such as skyline-based approach [9], approximation-based approaches [10], evolutionary algorithm based approaches, etc. Among them, evolutionary algorithm shows better performance and has been widely applied.

There are many frequently used evolutionary algorithms, such as genetic algorithm [11], Particle Swarm Optimization [12]. Among them, Artificial Bee Colony (ABC) algorithm is the simplest but a relatively powerful one [13][16]. It has only three control parameters, and the procedure of ABC is simple to understand and implement [14][15]. Researchers have applied ABC into service composition problems and have confirmed its superiority compared with other evolutionary optimization algorithms [16]. This paper uses the TBA approach presented by [16].

However, existing ABC researches have not fully addressed a key issue, namely, *parameter tuning*. More specifically, how to assign appropriate value to the three control parameters? In the past experiments, we have found there is a certain dependency between the parameters' values and the final optimization results. Still, current ABC algorithm designers have to rely on their own experiences to manually and casually assign values to the parameters.

Akay and Karaboga studied the ABC parameter tuning problem in [15] and have got some positive conclusions, e.g.:

- As ABC uses and exploitative process to efficiently converge minima and an explorative process to ensure sufficient diversity in the population, it does not need a large *colony size* (or called *SN*) to solve optimization problems with high dimensions.
- The second parameter *Limit* dictates the occurrence of scout bees that are responsible for ensuring the diversity of the population. For relatively small-sized populations, *Limit* should not be low to avoid the situation that there are many randomly-generated solutions in the present populations. However, since large populations have sufficient diversity, *Limit* does not stand out in these situations.

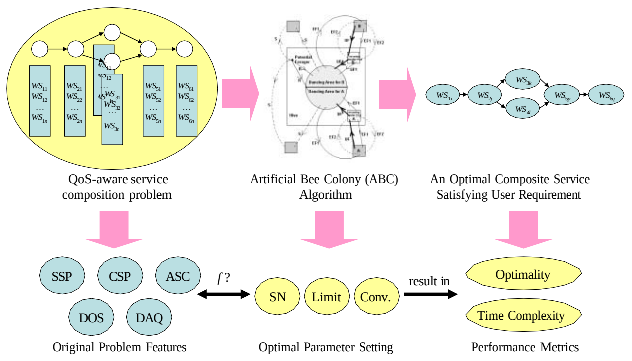
Unfortunately, their work aims at those continuous objective functions and it is doubtful that these conclusions were still true to the discrete combinatorial optimization problems such as QoS-aware service composition. This is just the objective of this paper: we try to find out what kind of settings of the three control parameters (*SN*, *Limit* and *Convergence*) might result in a composition solution with good performance, not only a higher optimality of the solution, but also a lower time complexity required to get the solution.

To note that, the value setting of the three control parameters is not standalone but depends on some features of service composition problems, e.g., the number of the candidate services, the structural complexity of the service process. The possibly optimal parameter setting is not a simple value or a value range, but might be a linear or non- linear function with the independent variables being the features of service composition problem. If we can identify such functions, an optimal parameter setting could be easily suggested to the algorithm designer, thereby avoiding their tedious attempts and at the same time ensuring an optimal composition solution. This issue is schematically shown in Figure 1(a) where our objective is to find the function *f* between the features of service composition problems and the three parameters of ABC algorithms.

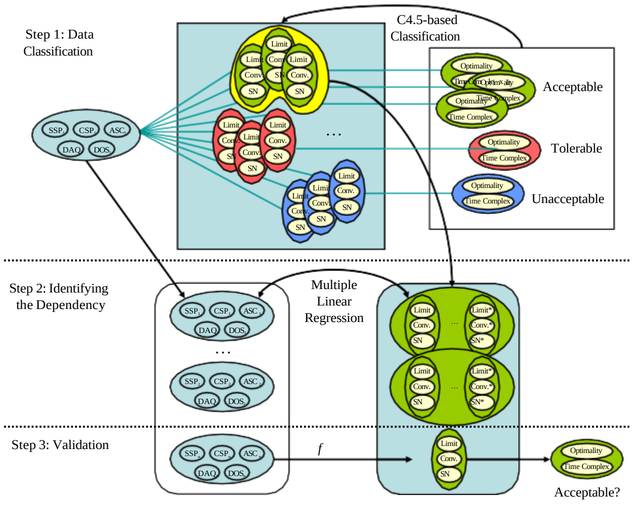
For simplicity, in the subsequent sections we use "*problem features*" as the abbreviation of "the features of a service composition problem", and "*ABC parameters*" as the one of "the three control parameters of the ABC-based service composition algorithm".

To solve this issue, we adopt simulation techniques combined with data mining methods. Based on the massive experiment data collected from above 9,000,000 times of experiments on the ABC-based service composition algorithms, C4.5 algorithm and multiple linear regression method are employed together to make the classification and find the potential dependencies. The work is decomposed into three steps: (1) in terms of one specific composition problem (meaning that the values of the problem features are fixed), we use C4.5 algorithm to extract the acceptable ABC parameters settings that could result in acceptable performance; (2) to identify the dependency between problem features and ABC parameters, we use the multiple linear regression method to process the experiment data that covers multiple service composition problems and the corresponding acceptable ABC parameter settings, and get an approximate *f*; (3) for validation, we use *f* to recommend the ABC parameter settings for a set of service composition problems that were not covered by the data used in step (2) and execute the ABC algorithm, then check the final results to see whether it is acceptable compared with the actual data. The three steps are shown in Figure 1(b) from top to bottom, respectively.

Remainder of the paper is organized as follows. Section II introduces an ABC algorithm to solve the service composition problems with end-to-end QoS constraints, and the definition and metrics of five problem features. Section III introduces the approach of generating and collecting the experiment data. Section IV puts forward a C4.5-based classification algorithm for identifying the optimal parameter settings for a given QoS-aware service composition problem. Section V presents a multiple linear regression method to identify dependencies between problem features and ABC parameter, and the obtained regression equations are used to predicate optimal ABC parameter settings in terms of a new service composition problem. Finally are the conclusions and future work.


QoS-aware service composition problem
Artificial Bee Colony (ABC) Algorithm
An Optimal Composite Service Satisfying User Requirement
SSP
CSP
ASC
DOS
DAQ
f ?
SN
Limit
Conv.
result in
Optimality
Time Complexity
Original Problem Features
Optimal Parameter Setting
Performance Metrics

(a) Issues to be Solved in this Paper


Step 1: Data Classification
C4.5-based Classification
Acceptable
Tolerable
Unacceptable
Optimality
Time Complex
Step 2: Identifying the Dependency
Multiple Linear Regression
Step 3: Validation
f
Acceptable?

(b) Approaches Adopted in this Paper

Figure 1. Issues and Approaches of this Paper

II. ABC-Based Algorithm for QoS-Aware Service Composition Problem

A. *Mathematical Model*

A QoS-aware service composition problem is briefly defined as: given a fixed service process consisting of a set of activities and the structural relationships between the activities, the algorithm selects one concrete service component for each activity from its candidate service set, and the final result satisfy the QoS constraints raised by a customer. The QoS attributes we consider here include price, response time, availability, reputation and throughput.

1) *Input*

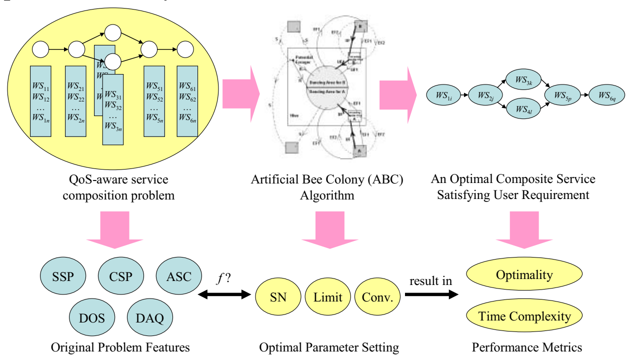
Unfortunately, their work aims at those continuous objective functions and it is doubtful that these conclusions were still true to the discrete combinatorial optimization problems such as QoS-aware service composition. This is just the objective of this paper: we try to find out what kind of settings of the three control parameters (*SN*, *Limit* and *Convergence*) might result in a composition solution with good performance, not only a higher optimality of the solution, but also a lower time complexity required to get the solution.

To note that, the value setting of the three control parameters is not standalone but depends on some features of service composition problems, e.g., the number of the candidate services, the structural complexity of the service process. The possibly optimal parameter setting is not a simple value or a value range, but might be a linear or non-linear function with the independent variables being the features of service composition problem. If we can identify such functions, an optimal parameter setting could be easily suggested to the algorithm designer, thereby avoiding their tedious attempts and at the same time ensuring an optimal composition solution. This issue is schematically shown in Figure 1(a) where our objective is to find the function *f* between the features of service composition problems and the three parameters of ABC algorithms.

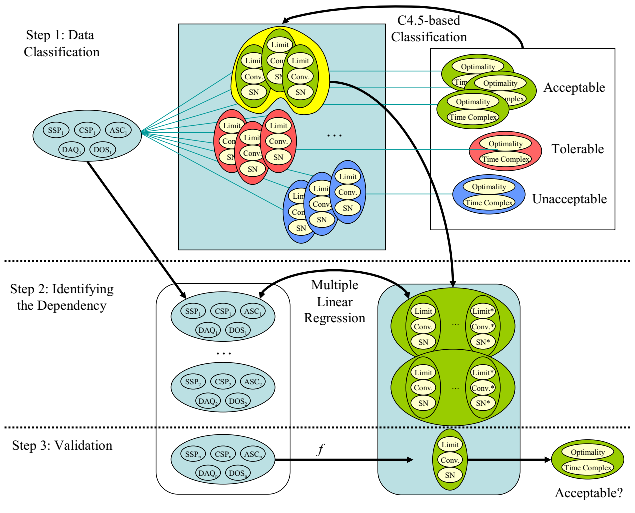
For simplicity, in the subsequent sections we use "*problem features*" as the abbreviation of "the features of a service composition problem", and "*ABC parameters*" as the one of "the three control parameters of the ABC-based service composition algorithm".

To solve this issue, we adopt simulation techniques combined with data mining methods. Based on the massive experiment data collected from above 9,000,000 times of experiments on the ABC-based service composition algorithms, C4.5 algorithm and multiple linear regression method are employed together to make the classification and find the potential dependencies. The work is decomposed into three steps: (1) in terms of one specific composition problem (meaning that the values of the problem features are fixed), we use C4.5 algorithm to extract the acceptable ABC parameters settings that could result in acceptable performance; (2) to identify the dependency between problem features and ABC parameters, we use the multiple linear regression method to process the experiment data that covers multiple service composition problems and the corresponding acceptable ABC parameter settings, and get an approximate *f*; (3) for validation, we use *f* to recommend the ABC parameter settings for a set of service composition problems that were not covered by the data used in step (2) and execute the ABC algorithm, then check the final results to see whether it is acceptable compared with the actual data. The three steps are shown in Figure 1(b) from top to bottom, respectively.

Remainder of the paper is organized as follows. Section II introduces an ABC algorithm to solve the service composition problems with end-to-end QoS constraints, and the definition and metrics of five problem features. Section III introduces the approach of generating and collecting the experiment data. Section IV puts forward a C4.5-based classification algorithm for identifying the optimal parameter settings for a given QoS-aware service composition problem. Section V presents a multiple linear regression method to identify dependencies between problem features and ABC parameter, and the obtained regression equations are used to predicate optimal ABC parameter settings in terms of a new service composition problem. Finally are the conclusions and future work.


QoS-aware service composition problem
Artificial Bee Colony (ABC) Algorithm
An Optimal Composite Service Satisfying User Requirement
SSP
CSP
ASC
DOS
DAQ
f ?
SN
Limit
Conv.
result in
Optimality
Time Complexity
Original Problem Features
Optimal Parameter Setting
Performance Metrics

(a) Issues to be Solved in this Paper


Step 1: Data Classification
C4.5-based Classification
Acceptable
Tolerable
Unacceptable
Step 2: Identifying the Dependency
Multiple Linear Regression
Step 3: Validation
f
Acceptable?

(b) Approaches Adopted in this Paper

Figure 1. Issues and Approaches of this Paper

II. ABC-BASED ALGORITHM FOR QOS-AWARE SERVICE COMPOSITION PROBLEM

A. *Mathematical Model*

A QoS-aware service composition problem is briefly defined as: given a fixed service process consisting of a set of activities and the structural relationships between the activities, the algorithm selects one concrete service component for each activity from its candidate service set, and the final result satisfy the QoS constraints raised by a customer. The QoS attributes we consider here include price, response time, availability, reputation and throughput.

1) *Input*

(a) A service process $SP = (A; E)$ where $A = \{S_1; S_2; ...; S_N\}$ is a set of activities and $E = \{(S_i; S_j)\}$ is a set of directed flows between activities.

(b) For each $S_i \in A$, $CDT(S_i) = \{s_{i1}; s_{i2}; ...; s_{iM}\}$ is a set of S_i's candidate services in which all the s_{ij} have the same functionality but have different values of *D*-dimensional QoS parameters $\{Q_1; Q_2; ...; Q_D\}$, such as response time, price, availability, reputation, throughput.

(c) For the QoS parameters, a customer will select a subset $QO \subseteq \{Q_1; Q_2; ...; Q_D\}$ as his optimization objectives, i.e., the customer expects to get a composite solution attaining optimal value on these QoS parameters, andsatisfying the constrains on *D*-dimensional QoS parameters, i.e., for each $Q_i \in Q = \{Q_1; Q_2; ...; Q_D\}$, there is a minimum threshold C_i and a weight ω_i both raised by the customer. Let $T = |QO|$.

2) *Output*

The output is a composite service containing N concrete services $CS = \{s_{1k_1}; s_{2k_2}; ...; s_{Nk_N}\}$ where $s_{ik_i} \in CDT(S_i), 1 \le i \le N$.

3) *Optimization Objective and Constraints*

The objective is to minimize the value of QoS parameters contained in *QO*, i.e.,

$$\min F(CS)$$

$$F(CS) = \sum_{k=1}^{T} \omega_k \times Norm_k^{QO}\left(A_k^{QO}(SP; CS)\right)$$

$$s.t.\ Norm_u^{Q}\left(A_u^{Q}(SP; CS)\right) \ge C_u$$

$$0 \le \omega_i \le 1, \sum_{k=1}^{T} \omega_k = 1, 1 \le u \le D$$

Note that, A is the QoS aggregation functions to calculate the aggregated QoS value based on the service process structure, and $Norm$ is the normalization function to transform the aggregated QoS value into the range $[0; 1]$.

B. *ABC Algorithm for QoS-Aware Service Composition*

The basic ABC algorithm is given below:

```
1: $Count = 1$
2: Initialize the food source populations $f_i$, $i = 1; 2; ...; SN$
3: Evaluate the nectar amount (fitness) of food sources
4: repeat
5:     Employed Bees' phase
6:     Onlooker Bees' phase
7:     if the quality of $f_i$ is not changed in Limit times
           Scout Bees' phase
8:     Memorize the best solution achieved so far
9:     if the best solution is changed
           $Count = 1$;
       Else $Count{+}{+}$;
10: until $Count = Convergence$
```

In this algorithm, the position of a food source represents a possible solution to the optimization problem and a fitness function is used to evaluate the quality of a food source (i.e., the solution). There are three control parameters in the algorithm:

- *SN* is the number of the food sources and represents the size of the populations during the iterations;

Limit is the maximum times of exploration around a food source. During iterations, if a food source has been explored above *Limit* times but its fitness was not improved further, it will be abandoned and replaced with a new food source.

Convergence is a threshold that determines whether the algorithm is terminated or not, i.e., if the optimal solution has not been improved in the consecutive *Convergence* iterations, the algorithm terminates and outputs the current optimal solution.

To note that, in the classical ABC algorithm, the third control parameter is not Convergence but *Maximum Cycle Number* (*MCN*); the reason why we use Convergence to substitute *MCN* is that *Convergence* is more suitable for balancing the tradeoff between the optimality and the execution time.

Aiming at QoS-aware service composition problem, we improve the basic ABC algorithm in the following aspects.

1) *Food source initialization (Step 2)*

Each solution is an *N*-dimensional vector where the value of the *i*-th dimension is the label of the selected service for the *i*-th activity in the process. A new solution $CS = (s_{1k_1}; s_{2k_2}; ...; s_{Nk_N})$ is randomly generated from the solution space $CDT(S_1) \times CDT(S_2) \times ... \times CDT(S_N)$. *SN* new solutions are totally generated.

2) *Fitness of a food source* (*Step 3*)

The fitness of a food source represents its quality, and we use a function based on the optimization objective:

$$Fitness(CS) = \frac{1}{1 + F(CS)}$$

3) *Food source updating* (*Step 6*)

A primitive method of updating a given food source $CS = \{s_{1k_1}; s_{2k_2}; ...; s_{ik_i}; ...; s_{Nk_N}\}$ is to randomly select an activity S_i and select another service s_{ik_j} from its candidate service set $CDT(S_i)$. The new food source $CS' = \left(s_{1k_1}; s_{2k_2}; ...; s_{ik_j}; ...; s_{Nk_N}\right)$ is a neighbor of CS.

However, it has been proven this approach does not keep the feature *Optimality Continuity* that is mandatorily required in ABC. Here we adopt a food source updating method called *Threshold-Based Algorithm* (TBA) [16]. The rationale of TBA is that, any gap in composition's aggregated QoS can be achieved by sufficiently small deviations in its component activity' QoS. It uses predefine threshold to identify neighbors for component activities. Suppose s_{ij} is an arbitrary candidate service for the i-th activity S_i, and s'_{ij} is identified as a neighbor of s_{ij} if and only if $\left|Q_k(s_{ij}) - Q_k\left(s'_{ij}\right)\right| < TD_{ik}$ where $Q_k(s_{ij})$ and $Q_k\left(s'_{ij}\right)$ are the value of the k-th QoS attribute of s_{ij} and s'_{ij}, respectively, and TD_{ik} is a pre-defined threshold for the k-th QoS attribute.

C. *Performance Metrics of the Algorithm*

We focus on two performance metrics:

1) *Optimality:* referring to the relative quality of the composite solution under the condition that all the QoS constraints are satisfied. We use $Optimality(CS)$ being the ratio of $F(CS)$ relative to the quality of global optimal