A Dynamic Composite Web Services Selection Method With QoS-Aware Based on AND/OR Graph

Hong Yu *, Qingfeng Zhou, Man Liu

Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, No.2, Chongwen Road, Nan’an District, Chongqing, 400065, P.R. China
E-mail: yuhong@cqupt.edu.cn

Received 8 November 2012
Accepted 29 January 2014

Abstract

Service selection problem refers to the selection of appropriate Web services from a large number of candidates in order to create complex composite services that can satisfy users’ quality-of-service (QoS) requirements. However, the existing services selection methods seldom consider the parallel relations between services as well as the dynamic changing of QoS. To combat these defects, a QoS-aware composite services selection model is presented based on an AND/OR graph model in this paper. The weight of edges is defined by considering QoS attributes, and one shorter path in the model is corresponding to the better service composition that satisfies the service request. Besides, the model is not only capable of dealing with sequence relations and fork relations between services, but also capable of dealing with parallel relations between services. And then a heuristic services selection algorithm is developed based on the framework of the ant colony optimization (ACO) to guide finding the path, and which is able to deal with three cases of the dynamic changing of QoS discussed in this paper. The results of comparison experiments show that the proposed method is effective and efficient.

Keywords: Web Services, Quality of Service, AND/OR Graph, Dynamic Selection Method, Ant Colony Optimization

1. Introduction

Web services are defined as self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. In service-oriented computing (SOC), Web services are fundamental elements of distributed and heterogeneous applications. With the development of theory and technology of Web service, users will demand more value and informative services rather than those offered by single, isolated Web services. Therefore, Web services composition is a main method of implementing service reuse. Developers and users can solve complex problems by composing available elementary services. Take an integrated financial management Web service as an example, it can be created by composing more specialized Web services for payroll, tax preparation, and cash management.

It is argued that services will be composed as part of work flows to build complex applications to achieve client problem requirements. With the growing number of alternative web services that provide the same functionality but differ in quality of service (QoS), the service composition becomes a decision problem on which component services should be selected from numerous candidates such that users’ end-to-end QoS requirements (e.g. availability, response time) and preferences (e.g. price) are satisfied. Thus, the specification and description of the QoS are very important, and QoS at-
tributes/dimensions serve as a key index for discriminating candidate Web services and composite Web services with identical functionality. In this paper, we will use the execution price, the execution duration, the reputation, the successful execution rate and the availability proposed by Zeng et al. as the attributes of QoS, since they are widely used.

The composite service is not a simple combination problem of elementary services. Relations between them will affect the quality and implementation of the composite service. That is, not only the relations between services input and output parameters, but also the parallel relations or the fork relations between elementary services should be considered, too. For example, Zeng et al. have represented an execution path as a directed acyclic graph, and the AND/OR states means parallel relations and fork relations; Yu et al. have considered six service composition structures in their study. Inspired by the existing models, and it is intuitively to remind us thinking the parallel and fork relations as AND/OR relations, we represent the Web services composition model as an AND/OR graph directly in this paper, where AND vertices and OR vertices mean the parallel relation and the fork relation between services, respectively. When talking about the AND/OR graph, Lang and Su had presented a formalization of the Web service composition problem as a search problem in an AND/OR graph. In Lang and Su’s AND/OR graph model, the AND/OR vertices are the services and the input/output documentary, respectively; which is different from the model proposed in this paper. Besides, the approach in Lang and Su neglects the QoS.

Though the above references have considered both the parallel relation and fork relation between services in the composite services, they don’t mention how to deal with the dynamic changing of QoS. Due to many factors such as network and scalability, the QoS is not a fixed value as discussed in the above-mentioned references. Therefore, many achievements have been presented for the dynamic services selection. Liu et al. presented an extensible QoS model which is fair and dynamic for both service requesters and service providers, the dynamic and fair computation of QoS values of web services is mainly through a secure active users’ feedback and active monitoring. Taher and Khatib presented a generic framework for QoS-based selection mechanism for Web services by introducing QoS models. Li et al. presented an agent-based adaptive dynamic semantic Web service selection framework, called AADSS, in which Web service consumers can dynamically select the “right” service, and adaptively change the bound services according to the QoS properties values, service reputation ranking scores and so on. Ardagna and Pernici introduced a new modeling approach to the Web service selection problem that is particularly effective for large processes and when QoS constraints are severe, and they assumed that quality values are real numbers that vary in a bounded range with a minimum and a maximum value. Mabrouk et al. presented an efficient service selection algorithm to address services’ selection and composition in the context of a QoS-aware middleware for dynamic service environments. Alrifai and Risse presented an efficient heuristic for the QoS-based service composition by combining global optimization with local selection methods. Alrifai et al. proposed an approach based on the notion of skyline to effectively and efficiently select services for composition.

Web service selection problem is usually formulated as a multi-objective optimization problem, which is known to be an NP hard problem. Thus, some heuristic algorithms are introduced to solve the optimization problem. In references [3, 11, 13], the authors used integer linear programming method to solve the web service selection problem. In references [15–18], the authors chose genetic algorithms for web service selection. Liu et al. presented a model of a web service based on the particle swarm to solve dynamic Web services selection with QoS global optimal, which can dynamically select and bind the best suitable web service to meet the requirement of different users. Wang et al. proposed an efficient QoS-aware web service selection approach based on an improved particle swarm optimization algorithm. Fan and Fang presented a novel service selection algorithm by integrating simulated annealing (SA) and niche technique into particle swarm optimization (PSO). Li et al. presented the multi-objective evolutionary algorithms for QoS-aware service composition optimization.

On the other hand, the ant colony optimization (ACO) is a novel nature-inspired metaheuristic for the solution of hard combinatorial optimization (CO) problems. The main idea of ACO is to model the problem as a search for a minimum cost path in a graph. ACO ap-
A Dynamic Composite Web Services Selection Method

A Dynamic Composite Web Services Selection Method has been demonstrated by successful applications in a variety of hard combination optimization problems such as traveling salesman problems, vehicle routing problem, constraint satisfaction problem, machine learning, etc.\textsuperscript{24}

Inspired by the character of ant colony optimization, more researchers have focused on solving the service selection problem with ACO. For example, Fang et al.\textsuperscript{27} presented a novel global QoS optimizing and multi-objective Web services selection algorithm based on multi-objective ant colony optimization (MOACO) for the dynamic Web service composition; Liu et al.\textsuperscript{25} proposed an algorithm by integrating Max-Min Ant System into Culture algorithm framework to solve the problem of composite Web services selection.

However, the mentioned methods only consider the sequence relation between services. Wang et al. have take into account the fork relation between services in reference [28], and they presented two dynamic methods of trust-aware composite services selection based on the ant colony system. The method 1 is to provide perfect services under the environment that the number of invocations don’t exceed the threshold, and to reject service under the environment that the number of invocations exceeds the threshold; the method 2 is used under the environment that services’ QoS degrades when the number of invocations increases. The authors considered the two possible cases in dynamic service selection under the environment that the number of invocations increases, but there should exist another situation, that is under the environment that service’s QoS keeps no degrading while the number of invocations increases. Because service providers are able to react by scheduling more resource such as enhancing the network’s bandwidth once the QoS degradation.

Therefore, there should be three cases considering the dynamic variations of QoS under the environment that the number of invocations increases, which will be discussed in this paper. It is reasonable that there exist the lower and upper thresholds of the number of invocations which provide high quality physical services. The Case 1 is under the environment that the number of invocations exceeds the upper threshold then service providers reject service, or less than the lower threshold then service providers provide perfect service; the Case 2 is under the environment that the number of invocations is between the lower and upper bound, the service’s QoS is degraded; the Case 3 is under the environment that the number of invocations is between the lower and upper bound, but service’s QoS remains unchanged when considering the interactive with the service providers who hope to do the business.

In this paper, we will present a Web service composition selection algorithm based on the ACO and the AND/OR graph model to adapt the dynamic variations of QoS. The AND/OR graph model is convenient to represent the sequence relation, the fork relation and the parallel relation between services. Identifying the best candidate Web services from a set of functionally equivalent services is a multi-criteria decision making problem. Then, the shortest path in the AND/OR graph model is expected to be the optimal service composition selection that satisfies users’ QoS requirements. And the QoS attributes values are assigned on the edges to indicate the dynamics of QoS. And, the multi-objective function is defined and used as the heuristic information in the algorithm which based on the ACO framework.

The rest of this paper is structured as follows. First, we introduce some basic concepts about the quality criterion and the logical relation between services. Then, a composite service selection model is proposed based on the AND/OR graph in Section 3, and the services selection problem is formulated as a multi-objective optimization problem. In Section 4, after discussing the QoS changes of three cases under the environment that the number of invocations increases, we give the details of our selection algorithm which based on ACO, and we conduct a set of experiments to evaluate the algorithm in Section 5. Finally, some conclusions will be given in Section 6.

2. Basic Concepts

The quality attributes of elementary services and the logical relations between services will be discussed in this section.

2.1. Meta-control Logical Relations Between Services

Definition 1 (Elementary Web service) If a Web service includes one function (operation), it is called an elementary Web service, and WS for short.

In order to express conveniently, two virtual elementary services are given as the initial service and the termi-
nal service of a composite service, expressed as $WS_0$ and $WS_{n+1}$, respectively.

**Definition 2 (Composite Web Service)** A composite Web service (CWS) is defined as a two-tuple: $CWS = (WS, CR)$. The $WS$ is a set of elementary services, namely, $WS = \{WS_0, WS_1, \ldots, WS_i, \ldots, WS_n, WS_{n+1}\}$, where $WS_i$ is an elementary Web service.

$CR$ is a set of the control logical relation between services. For two components $WS_i$ and $WS_j$, there exists one of the four following relations between them, which is called meta-control logical relation between $WS_i$ and $WS_j$. The relations are also described in Fig. 1.

$$\begin{align*}
Q(CWS, op) &= (q_{pr}(CWS, op), q_{du}(CWS, op), q_{rep}(CWS), q_{rat}(CWS), q_{av}(CWS))
\end{align*}$$

A determined composite service concludes some elementary services and some logical relations. Since the composite service is determined after services selection, the relations between elementary services are also determined. Obviously, there only exists sequence and parallel relations among them, then we will present the methods to compute the QoS under the two different meta-control logical relations.

If a meta composite Web service contains sequence relation as the part (1) in Fig 1. The QoS of the composite Web service is calculated as follows.

$$\begin{align*}
q_{pr}(CWS_{seq}, op) &= q_{pr}(WS_i, op) + q_{pr}(WS_j, op) \\
q_{du}(CWS_{seq}, op) &= q_{du}(WS_i, op) + q_{du}(WS_j, op) \\
q_{rep}(CWS_{seq}) &= (q_{rep}(WS_i) + q_{rep}(WS_j))/2 \\
q_{rat}(CWS_{seq}) &= q_{rat}(WS_i) \cdot q_{rat}(WS_j) \\
q_{av}(CWS_{seq}) &= q_{av}(WS_i) \cdot q_{av}(WS_j)
\end{align*}$$

If a meta composite Web service contains parallel relation as the part (3) in Fig 1. The QoS of the composite Web service is shown as follows.

$$\begin{align*}
q_{pr}(CWS_{par}, op) &= q_{pr}(WS_i, op) + \max(q_{pr}(WS_i, op), q_{pr}(WS_j, op)) \\
q_{du}(CWS_{par}, op) &= q_{du}(WS_i, op) + \max(q_{du}(WS_i, op), q_{du}(WS_j, op)) \\
q_{rep}(CWS_{par}) &= (q_{rep}(WS_i) + \min(q_{rep}(WS_i), q_{rep}(WS_j)))/2 \\
q_{rat}(CWS_{par}) &= q_{rat}(WS_i) \cdot \min(q_{rat}(WS_i), q_{rat}(WS_j)) \\
q_{av}(CWS_{par}) &= q_{av}(WS_i) \cdot \min(q_{av}(WS_i), q_{av}(WS_j))
\end{align*}$$

### 3. Composite Services Selection Model Using AND/OR Graph

In this section, we will describe the composite service model based on the AND/OR graph and formalize the composite service selection problem.

#### 3.1. AND/OR Graph Model

There are some solutions to deal with parallel relations or fork relations, but there are few solutions to dispose both parallel relations and fork relations in the current composite service models.
As we have seen in Fig. 1, there is a logical AND relationship among the parallel relation services, and there is a logical OR relationship among the fork relation services. It inspires us to choose an AND/OR graph to form the composite services model. That is, the parallel relation is expressed by “AND” logic, and the fork relation is expressed by “OR” logic.

Intuitively, we can represent a composite service as an AND/OR graph, where a vertex is a Web service $WS_i$, and the direction of an edge means the data logical relation between the two adjacent vertices. Generally speaking, a vertex is called an AND vertex if whose succeeding vertices are expected to be executed in parallel, and a vertex is called an OR vertex if whose succeeding vertices are fork to each other.

In other hand, there are more than one meta-control logic relation in most cases. Considering a mixed relationship, that is, there are parallel relation and the fork relation among the successors of a service vertex, we can transfer the mixed relationship to meta-control logic relations. Then we just need to discuss the model including meta-control logic relations.

Let us take Fig. 2 as an example to explain the transfer method. In the part (1) of Fig. 2, $WS_j$ and $WS_p$ are AND logical vertices, but $WS_q$ is OR logical with $WS_p$ or $WS_j$. We just need to add a new vertex, $WS_s$, as the previous point of the AND vertex. Then, there just leave the meta-control logical relations in the part (2) of Fig. 2.

Here is an example to illustrate a Web composite service system structured by an AND/OR graph in Fig. 3. In the example, the Web services system is composed of 7 elementary services, two virtual services denoting the initial service and the terminal service as $WS_0$ and $WS_8$, respectively. There are lots of choices from the initial elementary service to the terminal elementary service. In addition, there are different logical relations among the elementary services; some are AND logic, and some are OR logic.

From Fig. 3, we can see that there is one meta-control logic relation between arbitrary services $WS_i$ and $WS_j$, and the logic relations are denoted by the directed arrows. For example, after operating the service $WS_1$, the composite service should choose the next one from $WS_2$ and $WS_6$; and they are the “OR” logic to each other. However, Both $WS_3$ and $WS_4$ must be executed after $WS_2$, because they are the “AND” logic to each other.

On the other hand, there are usually more than one physical service for a Web service $WS_i$. Namely, $WS_i = \{PS_{i1}, PS_{i2}, \ldots, PS_{|PS_i|}\}$, where $|PS_i|$ denotes the number of physical services for $WS_i$. Fig. 4 is a schematic diagram of physical services, which is based on the edges from elementary services $WS_i$ to $WS_j$ as an example. Suppose there are $M$ and $N$ physical services for $WS_i$ to $WS_j$, respectively. There exists paths from each physical service of $WS_i$ to all physical services of $WS_j$, and they form a complete directed bipartite graph.

### 3.2. Weight of Edge

We will introduce how to distribute QoS values into the edges in this subsection.
Let WS\textsubscript{j} be a Web service but not the initial service in a CWS system, and WS\textsubscript{i} contains M physical services. Let the physical service of the initial service be PS\textsubscript{0,1}, and it is also a virtual vertex. Usually, for a physical service PS\textsubscript{i,m} of the WS\textsubscript{i}, there might exists many combinations from the initial service PS\textsubscript{0,1} to it. In other words, PS\textsubscript{i,m} is the joint end vertex of those combinations (paths). Suppose CWS\textsubscript{i,m} is the set of all these combinations, CWS\textsubscript{i,m} = \{CWS\textsubscript{1}, \cdots, CWS\textsubscript{1}, \cdots, CWS\textsubscript{1}\}, and there are L combinations. That is, CWS\textsubscript{i,m} is the set of all combinations/paths from the PS\textsubscript{0,1} to the PS\textsubscript{i,m}. Generally speaking, those composite services have different qualities of service.

There are many combinations/paths from the PS\textsubscript{0,1} to the PS\textsubscript{i,m}, every combination must have its QoS values since it is also a composite service according to Definition 3. We choose the best QoS values from all of them, and name it as the best QoS of PS\textsubscript{i,m}, denoted by Q\textsubscript{best} (PS\textsubscript{i,m}). That is, Q\textsubscript{best} (PS\textsubscript{i,m,op}) = (q\textsubscript{best,pr} (PS\textsubscript{i,m,op}), q\textsubscript{best,du} (PS\textsubscript{i,m,op}), q\textsubscript{best,rat} (PS\textsubscript{i,m}), q\textsubscript{best,av} (PS\textsubscript{i,m})). Especially, if WS\textsubscript{j} is the first elementary service WS\textsubscript{1} (not the virtual service WS\textsubscript{0}), the best QoS of every physical service in WS\textsubscript{1} is its QoS, namely Q\textsubscript{best} (PS\textsubscript{1,m,op}) = (q\textsubscript{pr} (PS\textsubscript{1,m,op}), q\textsubscript{du} (PS\textsubscript{1,m,op}), q\textsubscript{rat} (PS\textsubscript{1,m}), q\textsubscript{av} (PS\textsubscript{1,m})). For every PS\textsubscript{i,m}, we also attach the best QoS of PS\textsubscript{i,m} on it except its inherent QoS values.

No matter that WS\textsubscript{j} is an AND vertex or OR vertex, suppose one of its successors to be WS\textsubscript{j} as shown in Fig.4. Simultaneously, suppose the current visited service/vertex is the PS\textsubscript{i,m} and its Q\textsubscript{best} (PS\textsubscript{i,m,op}) has been obtained. Thus, we need to calculate the best QoS of PS\textsubscript{i,n} based on the Q\textsubscript{best} (PS\textsubscript{i,m,op}). Obviously, it is the sequence relation between them, the PS\textsubscript{i,n} can be obtained according to the Equation (3) as follows.

\[
\begin{align*}
q\textsubscript{best,pr} (PS\textsubscript{i,n,op}) &= q\textsubscript{best,pr} (PS\textsubscript{i,m,op}) + q\textsubscript{pr} (PS\textsubscript{i,n}) \\
q\textsubscript{best,du} (PS\textsubscript{i,n,op}) &= q\textsubscript{best,du} (PS\textsubscript{i,m,op}) + q\textsubscript{du} (PS\textsubscript{i,n}) \\
q\text{best,rat} (PS\textsubscript{i,n}) &= (q\text{best,rat} (PS\textsubscript{i,m}) + q\text{rat} (PS\textsubscript{i,n}))/2 \quad (3) \\
q\text{best,av} (PS\textsubscript{i,n}) &= q\text{best,av} (PS\textsubscript{i,m}) \cdot q\text{av} (PS\textsubscript{i,n})
\end{align*}
\]

When more and more other composite services (combinations/paths) arrive at the PS\textsubscript{i,n}, the best QoS of PS\textsubscript{i,n} may be changed. The best QoS of services following the WS\textsubscript{j} will also change.

Therefore, we use Q\textsubscript{best} (PS\textsubscript{i,n}) to depict the weight of edge between PS\textsubscript{i,m} and PS\textsubscript{i,n}. Let d(PS\textsubscript{i,m}, PS\textsubscript{i,n}) denote the weight of edge between PS\textsubscript{i,m} and PS\textsubscript{i,n}, which is defined as follows.

\[
d(PS\textsubscript{i,m}, PS\textsubscript{i,n}) = \frac{q\text{best,pr} (PS\textsubscript{i,m,op}) + q\text{best,du} (PS\textsubscript{i,m,op})}{q\text{best,pr} (PS\textsubscript{i,n}) \cdot q\text{best,du} (PS\textsubscript{i,n})}
\] (4)

From the above equation, we know that the weight of edge is determined by the best QoS of the end vertex of the edge. Since the best QoS of service might change in selection processing, it results that weights of edges are also dynamically changed and tend to the best composite Web service.

As we have discussed, the five attributes are divided into the negative attributes and the positive attributes. In other words, the higher the prices and the execution durations are, the larger the weight is. On the contrary, the larger positive criteria, the smaller the weight is.

3.3. QoS Distance of a Composite Service

We introduce the QoS distance of a composite Web service to score the quality of a composite service in this section.

Definition 4 (QoS Distance of a Composite Web Service) Let CWS\textsubscript{i} be a composite Web service, we define its QoS distance as follows.

\[
L_i = K \cdot \frac{q_{pr}(CWS\textsubscript{i})}{q_{pr}(CWS\textsubscript{i}) + q_{du}(CWS\textsubscript{i})} \\
\] (5)

where K = Rep \cdot Rat \cdot AV / (Pr \cdot Du). Pr, Du, Rep, Rat and AV denote constraint values of the execution price, the execution duration, the reputation, the successful execution rate and the availability, respectively.

According to the common sense, the better composite Web service (path) provides the better QoS. In other words, a better composite service relates to a shorter path. Namely, it has less execution price and execution duration, and higher reputation, successful execution rate and availability. Obviously, the above fraction is consistent to the general knowledge. On the other hand, Equation (5) also integrates the five factors together, it is helpful to assign the QoS attributes values to the edges between services.

3.4. Choose the Optimal Composite Service

Because composite Web services with different QoS values may have the same QoS distance according to Equa-
tion (5), we need a strategy to determine which composite Web service is the best one. We will discuss this problem in this subsection.

Suppose there are two composite Web services \(CW_{Sm}\) and \(CW_{Sn}\). The QoSes of \(CW_{Sm}\) and \(CW_{Sn}\) are respectively shown as follows:

\[
Q(CW_{Sm},op) = (q_{pr}(CW_{Sm},op), q_{du}(CW_{Sm},op), q_{rep}(CW_{Sm}), q_{rat}(CW_{Sm}), q_{av}(CW_{Sm}))
\]

\[
Q(CW_{Sn},op) = (q_{pr}(CW_{Sn},op), q_{du}(CW_{Sn},op), q_{rep}(CW_{Sn}), q_{rat}(CW_{Sn}), q_{av}(CW_{Sn}))
\]

Distance of \(CW_{Sm}\) and \(CW_{Sn}\) are also respectively calculated as follows according to Equation (5):

\[
L_m = K \cdot \frac{q_{du}(CW_{Sm})+q_{du}(CW_{Sn})}{q_{pr}(CW_{Sm})+q_{pr}(CW_{Sn})+q_{rep}(CW_{Sm})+q_{rat}(CW_{Sm})}
\]

\[
L_n = K \cdot \frac{q_{du}(CW_{Sn})+q_{du}(CW_{Sm})}{q_{pr}(CW_{Sn})+q_{pr}(CW_{Sm})+q_{rep}(CW_{Sn})+q_{rat}(CW_{Sm})}
\]

If every QoS attribute value of \(CW_{Sm}\) and \(CW_{Sn}\) are equal, their QoSes are equal, namely \(Q(CW_{Sm}) = Q(CW_{Sn})\); otherwise, their QoSes are not equal. There will be three cases, and we describe the corresponding strategies as follows.

- If \(Q(CW_{Sm}) = Q(CW_{Sn})\) and \(L_m = L_n\), namely both the QoSes and the distances of \(CW_{Sm}\) and \(CW_{Sn}\) are equal, we will randomly choose one as the best composite Web service or choose one which contains less Web services as the best composite Web service.
- If \(Q(CW_{Sm}) \neq Q(CW_{Sn})\) but \(L_m = L_n\), namely the QoSes are not equal but the distances of them are equal, we can make sure that there must exist at least two different QoS attribute values. In this case, we leave the choice to users, and users can choose one according to their preferences.
- For example, suppose that \(Q(CW_{Sm}) = (4, 4, 1, 1, 25), Q(CW_{Sn}) = (1, 7, 1, 1, 25)\). We find that the two composite services have the same QoS values, but different QoSes. Compared to \(CW_{Sn}\), \(CW_{Sm}\) needs more execution price, but it uses less execution duration. At this time, users can choose which composite Web service is the better one according to their preferences. If users would like to sacrifice more execution price in order to obtain less execution duration, they will choose \(CW_{Sm}\) as the best composite Web service; if users hope to use less money, they would like to choose \(CW_{Sn}\) as the best composite Web service.
- If \(Q(CW_{Sm}) \neq Q(CW_{Sn})\) and \(L_m \neq L_n\), namely neither the QoSes nor the distances of \(CW_{Sm}\) and \(CW_{Sn}\) are equal, we choose one whose distance is the minimum as the best composite Web service.

Therefore, we can choose the best one from many composite Web services according to the above strategies.

3.5. Composite Services Selection Model

Let \(G(N,E,W)\) be an AND/OR graph, where \(N\) is a set of vertices, \(E\) is a set of edges, \(W\) is a set of edge weights.

The composite services selection problem can be formulated as a multi-objective optimization problem with QoS constraints, which is described as follows.

**Objective functions:** \(\min(q_{pr}(P)), \min(q_{du}(P)), \max(q_{rep}(P)), \max(q_{rat}(P)), \max(q_{av}(P))\).

**Constraint conditions:** \(q_{pr}(P) < Pr, q_{du}(P) < Du, q_{rep}(P) > Rep, q_{rat}(P) > Rat, q_{av}(P) > Av\).

\(P\) is the target composite Web service. \(Pr, Du, Rep, Rat\) and \(Av\) denote constraint values of the execution price, the execution duration, the reputation, the successful execution rate and the availability, respectively.

In order to satisfy the multi-objectives, we distribute the QoS values into the edges of the directed graph by considering constraint conditions at the same time. The weights of edges between services is calculated by Eq. (4). That is, the goal of searching optimal composite selection is to find a best composite Web service in the AND/OR graph, according to strategies introduced in subsection 3.4.

3.6. An Example for Composite Service Selection

Take the composite service system in Fig.3 as an example again, we will explain some definitions proposed in this section.

Suppose every elementary service only contains one physical service to simplify the example. \(Q(P_{S_{0x}},op)\) means the QoS of the \(x\)th physical service in the \(WS_i\). The inherent initial QoS of every physical in the composite service system in Fig.3 are shown as follows:

\(Q(P_{S_{01}},op) = (0, 0, 0, 1, 1), Q(P_{S_{11}},op) = (40, 50, 4, 0.4, 0.3), Q(P_{S_{21}},op) = (20, 30, 6, 0.7, 0.8), Q(P_{S_{12}},op) = (30, 40, 5, 0.6, 0.7), Q(P_{S_{22}},op) = (90, 20, 3, 0.4, 0.5), Q(P_{S_{31}},op) = (30, 20, 4, 0.8, 0.9), Q(P_{S_{61}},op) = (50, 50, 7, 0.7, 0.7), Q(P_{S_{71}},op) = (70, 80, 9, 0.1, 0.1), Q(P_{S_{81}},op) = (0, 0, 0, 1, 1)\).

At the beginning, we only know the best QoS of every physical service in the first service, namely \(WS_1\). And the
best QoS of every physical service in other services is not known, but we can calculate their best QoSes according to Eq. (3) while choosing the next vertex. Because the $WS_1$ is the first service following the initial virtual service $WS_0$, so the best QoS of $PS_{1,1}$ is $Q(PS_{1,1}, op)$, namely $Q_{best}(PS_{1,1}, op) = Q(PS_{1,1}, op)$. Then we can calculate the weight of edge between $PS_{1,1}$ and $PS_{1,1}$ according to Equation (4).

When the selection arrives at one service except the first service, we should firstly calculate the best QoS of every physical in its successor services and then calculate weights of edges between them. Then we move to the next vertex after choosing one physical service. For an OR vertex, we should choose one physical service from all of its successor services; but for an AND vertex, we should choose one physical service from its every successor. For example, $WS_2$ is an AND vertex, we should choose one physical service in $WS_2$ and $WS_3$, respectively. Besides, for an AND vertex, its parallel branches must be executed simultaneously.

Suppose the selected service is the $PS_{1,1}$ in the current. Because the $WS_1$ is an OR vertex, so we should choose one physical service from $WS_2$ and $WS_6$.

Firstly, we calculate the QoS of every physical service in $WS_2$ and $WS_6$. For example, we calculate the best QoS of $PS_{6,1}$ according to Eq. (3), namely $Q_{best}(PS_{6,1}, op) = (40+50, 50+50, (4+7)/2, 0.4 \cdot 0.7, 0.3 \cdot 0.7) = (90, 100, 5.5, 0.28, 0.21)$.

Secondly, we calculate weight of edge between $PS_{1,1}$ and every physical service in $WS_2$ and $WS_6$. For example, we calculate the weight of edge between $PS_{1,1}$ and $PS_{6,1}$. It is calculated as follows.

$$d(PS_{1,1}, PS_{6,1}) = \frac{q_{best}(PS_{6,1}, op) + q_{best}(PS_{6,1}, op)}{q_{best}(PS_{1,1}, op) + q_{best}(PS_{1,1}, op)} = (90 + 100)/(5.5 \cdot 0.28 - 0.21) = 587.51$$

In the same way, we have $d(PS_{1,1}, PS_{2,1}) = 416.67$. Because $d(PS_{1,1}, PS_{2,1})$ is less than $d(PS_{1,1}, PS_{6,1})$, the probability of choosing $WS_2$ as the next service is higher.

When the selection arrives at the terminal vertex, we obtain a composite Web service. For this system, we can obtain two composite Web services $CW_S$ and $CW_S$. They are shown as follows.

$CW_S = (WS_0, WS_1, WS_2, WS_3, WS_4, WS_5, WS_6, WS_7)$;

$CW_S = (WS_0, WS_1, WS_2, WS_3, WS_4, WS_5, WS_6, WS_7)$.

Obviously, $CW_S$ contains AND service and $CW_S$ doesn’t contain AND service.

Which one is better? Firstly, We calculate their QoS distances respectively according to Eq. (5). Because $WS_0$ and $WS_8$ are virtual vertices, so when calculating their QoS distances, we don’t include them. The QoS of $CW_S$ is calculated as follows.

$$q_{pr}(CW_S, op) = q_{pr}(WS_1, op) + q_{pr}(WS_2, op) + max(q_{pr}(WS_3, op), q_{pr}(WS_4, op)) + q_{pr}(WS_5, op) + q_{pr}(WS_7, op)$$

$$= 40 + 20 + max(30, 90) + 30 + 70 = 250$$

$$q_{da}(CW_S, op) = q_{da}(WS_1, op) + q_{da}(WS_2, op) + max(q_{da}(WS_3, op), q_{da}(WS_4, op)) + q_{da}(WS_5, op) + q_{da}(WS_7, op)$$

$$= 50 + 30 + max(40, 20) + 20 + 80 = 220$$

$$q_{rep}(CW_S) = (q_{rep}(WS_1) + q_{rep}(WS_2) + min(q_{rep}(WS_3), q_{rep}(WS_4)) + q_{rep}(WS_5) + q_{rep}(WS_7))/6$$

$$= (4 + 6 + min(5, 5) + 4 + 9)/6 = 4.33$$

$$q_{rat}(CW_S) = q_{rat}(WS_1) \cdot q_{rat}(WS_2) \cdot min(q_{rat}(WS_3), q_{rat}(WS_4)) \cdot q_{rat}(WS_5) \cdot q_{rat}(WS_7) \cdot q_{rat}(WS_8)$$

$$= 0.4 \cdot 0.7 \cdot min(0.6, 0.4) \cdot 0.8 \cdot 0.1 = 0.00896$$

$$q_{av}(CW_S) = q_{av}(WS_1) \cdot q_{av}(WS_2) \cdot min(q_{av}(WS_3), q_{av}(WS_4)) \cdot q_{av}(WS_5) \cdot q_{av}(WS_7) \cdot q_{av}(WS_8)$$

$$= 0.3 \cdot 0.8 \cdot min(0.7, 0.5) \cdot 0.9 \cdot 0.1 = 0.0108$$

We obtain the QoS of $CW_S$, namely $Q(CW_S, op) = (250, 220, 4.33, 0.00896, 0.0108)$.

In the same way, we can obtain the QoS of $CW_S$, namely $Q(CW_S, op) = (160, 180, 6.67, 0.028, 0.021)$.

Then we calculate their QoS distances according to Equation (5). The QoS distance of $CW_S$ is $L_1 = 1121703.814$, and the QoS distance of $CW_S$ is $L_2 = 68691.348$.

Secondly, We find that nor their QoSes or their distances are equal, so we choose $CW_S$ as the optimal composite Web service according to strategies in subsection 3.4. It is obviously to see that every QoS attribute of $CW_S$ is better than $CW_S$’s, which is also support the selection made by the strategies proposed in this section.

4. Service Selection Algorithm Based on ACO

In this section, we will present a dynamic composition service selection algorithm using AND/OR graph based on ant colony optimization, the DA Algorithm for short in the following descriptions. The algorithm considers not only sequence relations, but also fork relations as
well as parallel relations between services, and the services selection is formulated a multi-objective optimization problem as discussed in the last section. In addition, the approaches to deal with the dynamic variations of QoS under the environment that the number of invocations increases are also presented in this section.

4.1. The Framework of the Algorithm

Ants are set on the initial service vertex of the composite service system, which is represented as an AND/OR graph. The task of each ant is to find a path from the initial vertex to the terminal vertex, which satisfies the multi-objective composition optimization functions and the QoS constraint conditions. If ants arrive at an OR vertex, they visit one of the successors according to the transition rules; and if ants arrive at an AND vertex, they visit all the successors. Every ant updates the local pheromone when it passes an edge. After the colony complete an iteration, the global pheromone is updated. The framework of the DA algorithm is described in Algorithm 1.

As we have discussed in the preceded section, the weights of edges are dynamically changed in the proposed model, and the changing QoS while the number of invocations increases is also considered in this paper.

Suppose there exists the lower and upper thresholds of the number of invocations from every physical service provider. Obviously, this assumption is consistent with the actual. Set the lower threshold is $Th_{low}$, and the upper bound threshold is $Th_{app}$. There are three cases will happen.

Case 1 is under the environment that the number of invocations is less than $Th_{low}$ or more than $Th_{app}$. If the number of invocations is less than $Th_{low}$, the service can provide high quality of service obviously; if the number of invocations is more than $Th_{app}$, then the service system will refuse to provide service since request services exceed the capabilities of supplier.

Case 2 is under the environment that the number of invocations is between the $Th_{low}$ and the $Th_{app}$, and providers have no react. For this case, the service’s QoS will degrade with the service request increasing.

Case 3 is under the environment that the number of invocations between the $Th_{low}$ and the $Th_{app}$, but providers have some positive actions to keep the QoS. That is, the quality of service is not degraded when considering the interactive with the service providers, which also means the providers hope to do the business. For example, the service providers increase hardware resources to keep the competitions of services.

Since the different ant colony system differs from each other by the three main aspects as the transition rule, the global pheromone updating rule and the local pheromone updating rule. The solutions to the three aspects will be described in the following subsections. The transition rule is the same one for the three cases, but the pheromone updating rules are different.

4.2. Case 1

Let $P_{ij}^k$ denote the state transition probability, that is, the ant $k$ moves from service $i$ to service $j$. $q$ is a random variable, $q_0$ is a constant, $q_\in [0, 1]$, and $q_0 (0 \leq q_0 \leq 1)$. $allowed_k$ is the set of the nodes (physical services) allowed to visit by the ant $k$. When an ant $k$ arrives at an OR vertex, $allowed_k$ is composed of the physical services in the all succeed vertices (services), and the ant chooses one node to visit according to the following transition rule. When an ant arrives at an AND vertex, the $allowed_k$ is the union of the composed of the physical services in every succeed vertices (services), the ant must choose one node to visit from every succeed vertices.

Transition Rule is given as follows.

$$p_{ij}^k = \begin{cases} \arg \max_{\in allowed_k}[\frac{\tau_{ij}}{\sum_{l \in allowed_k} \tau_{il}}]^{\alpha}[\eta_{ij}]^{\beta} & \text{if } q < q_0 \\ J & \text{otherwise} \end{cases}$$

(6)

where $J$, a random variable, is computed by the following probability distribution equation.

$$J = \begin{cases} \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in allowed_k} [\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} & l \in allowed_k, \\ 0 & \text{otherwise} \end{cases}$$

(7)

$\eta_{ij} = \frac{1}{d_{ij}}$ is the heuristic information, $d_{ij}$ is computed according to the Equation (4). $\tau_{ij}$ is the pheromone intensity of the path between service $i$ and $j$, $\alpha$ is a parameter controlling the importance of the pheromone, and $\beta$ is a parameter controlling the importance of heuristic information.
Local Pheromone Updating Rule The ant updates the pheromone on the passing edges according to the following equation.

\[ \tau_{ij} = (1 - \rho) \tau_{ij} + \rho \tau_0 \]  

(8)

where \( \tau_0 \) is the initial pheromone level, \( 0 < \rho < 1 \) is the pheromone decay parameter.

Global Pheromone Updating Rule After an iteration, the pheromone is updated by the following equation.

\[ \Delta \tau_{ij} = \sum_{k=1}^{m} \Delta \tau_{ij}^k \]

\[ \Delta \tau_{ij}^k = \begin{cases} \frac{1}{m} & (i, j) \in L_k \\ 0 & \text{otherwise} \end{cases} \]  

(9)

where \( L_k \) denotes the length of optimal execution path of the ant \( k \) in this iteration.

4.4. Case 3

Under this case, though the number of invocations \( Th \) is also between the lower threshold \( Th_{low} \) and upper threshold \( Th_{app} \), but the service provider makes positive countermeasures to keep the competition of the service. For example, the provider may increase the hardware resources. In other words, the pheromone updating rule should be designed to lead ants still choose the visited path.

Local Pheromone Updating Rule The ant updates the pheromone on the passing edges according to the following equation.

\[ \tau_{ij} = (1 - \rho) \tau_{ij} + \rho \tau_0 \times \log(U^{Th}) \]  

(12)

where \( \tau_0 \) is the initial pheromone level, \( 0 < \rho < 1 \) is the pheromone decay parameter, \( U (U > 1) \) is a parameter controlling the no descending of pheromone, \( Th \) is the number of invocations from the service \( j \).

Global Pheromone Updating Rule After an iteration, the pheromone is updated by the following equation.

\[ \Delta \tau_{ij} = \sum_{k=1}^{m} \Delta \tau_{ij}^k \times \log(U^{Th}) \]

\[ \Delta \tau_{ij}^k = \begin{cases} \frac{1}{m} & (i, j) \in L_k \\ 0 & \text{otherwise} \end{cases} \]  

(13)

where \( L_k \) denotes the length of optimal execution path of the ant \( k \) in this iteration.

5. Experiments and Analysis

In order to evaluate the approach, the DA algorithm is programmed in Visual C++ and running on a PC, which is 2.60GHz and 1.96G Memory. To guarantee an equal distribution and statistical correctness, each test is run 20 times, and the average results are recorded as the result of this test in every experiment.

There are 4 services instance systems used in experiments, denoted by S11, S12, S13, S14. The number of elementary services in these composite services systems are 10, 20, 30 and 40, respectively. The number of candidate physical services for each elementary service can be different in practice. We set the number is 20 simply in experiments. The service execution price and execution duration are integers which are generated randomly.
Algorithm 1: Dynamic Web Service Composition Selection Algorithm Using AND/OR Graph Based on ACO

Input: $G = (N, E, W)$.
Output: a service path $GS$.

begin
  // Step 1: Initially
  $\tau_{ij}(0) =$ const; $t = 0$;
  $LS_0 = \emptyset$; $TS_0 = \emptyset$; //local solution, temp solution
  $GS, GS' = \emptyset$; //global solution
  $Th_{low} =$ const; $Th_{upp} =$ const; //the lower threshold and upper threshold
  currentnode = $PS_{0,1}$; //save current vertex
  $Th =$ const; //the threshold of 3 cases
  Set $M, T$; //the number of ants, and the number of maximum iteration

  // Step 2: Construct solution
  while $GS = GS'$ or $t = T$ do
    for $k = 1, k \leq M, k + +$ do
      Set ants on the starting vertex $WS_0$; $currentnode = WS_0$;
      while $currentnode \neq \emptyset$ do
        if $Th < Th_{low}$ or $Th > Th_{upp}$ then
          if $currentnode$ is an OR vertex then
            According to Equation (6) choose $PS_{ij}$ from all candidate services:
            $LS_k = LS_k \cup PS_{ij}$;
            According to Equation (8) update the pheromone on the passing edges; $currentnode = PS_{ij}$;
          end
        else
          Obtain every parallel branches of $currentnode$, and then add every parallel branches into $LS_k$.
          Set $currentnode$ as the converge vertex of parallel branches;
        end
        if $Th > Th_{low} and Th \leq Th_{upp} and providers haven't some positive actions$ then
          if $currentnode$ is an OR vertex then
            According to Equation (6) choose $PS_{ij}$ from all candidate services:
            $LS_k = LS_k \cup PS_{ij}$;
            According to Equation (10) update the pheromone on the passing edges; $currentnode = PS_{ij}$;
          end
        else
          Obtain every parallel branches of $currentnode$, and then add every parallel branches into $LS_k$.
          Set $currentnode$ as the converge vertex of parallel branches;
        end
        if $Th > Th_{low} and Th \leq Th_{upp} and providers have some positive actions$ then
          if $currentnode$ is an OR vertex then
            According to Equation (6) choose $PS_{ij}$ from all candidate services:
            $LS_k = LS_k \cup PS_{ij}$;
            According to Equation (12) update the pheromone on the passing edges; $currentnode = PS_{ij}$;
          end
        else
          Obtain every parallel branches of $currentnode$, and then add every parallel branches into $LS_k$.
          Set $currentnode$ as the converge vertex of parallel branches;
        end
      end
      if $LS_k$ is superior to $TS_k$ then
        $TS_k = LS_k$;
      end
      $LS_k = \emptyset$;
    end
  Choose the best one from all of $TS_k$ to $GS'$ according to strategies introduced in subsection 3.4;
  $TS_k = \emptyset$;
  According to 3 different cases, use the Global Pheromone Updating Rule Equation (9) or (11) or (13) to update the pheromone;
  $t + +$;
  if $GS'$ is superior to $GS$ then
    $GS = GS'$;
end
  //Step 3: Output the path $GS$.
end
between 1 and 100, the reputation is an integer which is generated randomly between 1 to 10, the successful execution rate and the availability of a Web service is a float which is generated randomly between 0 and 1. Here, $\alpha = 2.0$, $\beta = 5.0$, $\rho = 0.1$, $q_0 = 0.9$; $Rep = 2$, $Rat = 0.6$, $Av = 0.5$, $Pr = 300$, $Du = 100$.

5.1. Experiment 1

In order to show that the approach based on the ant colony optimization in this paper is also valuable, we do comparison experiments firstly.

The compared algorithm is from the reference [18], MA algorithm shorted, which is also a heuristic algorithm. Considering the capability of the contrasted algorithm, only fork relations and sequence relations between services are tested on data sets.

The results are recorded in Table 1, where we record the iterations times, the distance, the running time for both the algorithms. The “Min” column record the results of the optimal composite Web services obtained in experiments, which also means the distance of the path is the minimal; The “Max” column record the results of the worst composite Web services obtained in experiments, which also means the distance of the path is the maximal; The “Avg” column record the results of the average values for the 20 times tests for the corresponding data set.

From Table 1, we know that the distance of the optimal composite Web services obtained by DA algorithm is shorter than the MA algorithm [18]. Besides, the DA algorithm also obtains much better results compared in the average distance and the maximal distance of composite Web services. The times of iterations in our algorithm are less than the times of iterations in the compared algorithm. To sum up, the proposed method in this paper can obtain better results than the compared algorithm.

5.2. Experiment 2

When more and more users apply for services, the QoS of the Web service hardly keep up the original QoS. Therefore, the DA algorithm considers three cases when the number of requests increasing. Wang et al. [28] considered two cases to deal with the problem only in fork relations and sequence relations between services. Thus, we equitably test the two algorithms in Case 1 and Case 2 in this experiment; and the data set contains only fork relations and sequence relations between services.

The compared algorithm [28], the TA algorithm shorted, uses the sum of weights on edges to guide finding the optimal composite Web services. Thus, we also calculate the QoS distance of the optimal composite Web service obtained in TA algorithm. The parameter $U$ used in our algorithm is set as 0.8.

Form Table 2 and Table 3, we know that the distance of the optimal composite Web services obtained by DA algorithm is much shorter than the compared algorithm both in Case 1 and Case 2; in other words, the result of DA algorithm has a much higher quality of composite service. Though the time cost of DA algorithm is a little bit bigger than the TA's, but it is still acceptable to solve the problem especially considering the high quality service requirements.

### Table 1. The experimental results of the MA and DA algorithms

<table>
<thead>
<tr>
<th>Instances</th>
<th>Algorithms</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Iteration</td>
<td>Distance</td>
<td>Time(s)</td>
</tr>
<tr>
<td>SI1</td>
<td>MA</td>
<td>4</td>
<td>0.013696</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td>4</td>
<td>0.008158</td>
<td>0.250</td>
</tr>
<tr>
<td>SI2</td>
<td>MA</td>
<td>10</td>
<td>0.088874</td>
<td>0.031</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td>3</td>
<td>0.028002</td>
<td>0.281</td>
</tr>
<tr>
<td>SI3</td>
<td>MA</td>
<td>7</td>
<td>0.002616</td>
<td>0.015</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td>2</td>
<td>0.002353</td>
<td>0.156</td>
</tr>
<tr>
<td>SI4</td>
<td>MA</td>
<td>4</td>
<td>0.228849</td>
<td>0.016</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td>6</td>
<td>0.044507</td>
<td>0.657</td>
</tr>
</tbody>
</table>
Table 2. The experimental results of the TA and DA algorithms under Case 1

<table>
<thead>
<tr>
<th>Instances</th>
<th>Algorithms</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI1</td>
<td>TA</td>
<td>16</td>
<td>0.012158</td>
<td>0.046</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td>2</td>
<td>0.007779</td>
<td>0.124</td>
</tr>
<tr>
<td>SI2</td>
<td>TA</td>
<td>63</td>
<td>0.120672</td>
<td>0.297</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td>3</td>
<td>0.028563</td>
<td>0.266</td>
</tr>
<tr>
<td>SI3</td>
<td>TA</td>
<td>2</td>
<td>0.103258</td>
<td>0.015</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td>2</td>
<td>0.005353</td>
<td>0.156</td>
</tr>
<tr>
<td>SI4</td>
<td>TA</td>
<td>6</td>
<td>0.644710</td>
<td>0.047</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td>2</td>
<td>0.041155</td>
<td>0.219</td>
</tr>
</tbody>
</table>

Table 3. The experimental results of the TA and DA algorithms under Case 2

<table>
<thead>
<tr>
<th>Instances</th>
<th>Algorithms</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI1</td>
<td>TA</td>
<td>3</td>
<td>0.033091</td>
<td>0.078</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td>14</td>
<td>0.007780</td>
<td>0.014</td>
</tr>
<tr>
<td>SI2</td>
<td>TA</td>
<td>39</td>
<td>2.79262</td>
<td>0.297</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td>2</td>
<td>0.02996</td>
<td>0.189</td>
</tr>
<tr>
<td>SI3</td>
<td>TA</td>
<td>56</td>
<td>3.37017</td>
<td>0.485</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td>2</td>
<td>0.002353</td>
<td>0.157</td>
</tr>
<tr>
<td>SI4</td>
<td>TA</td>
<td>9</td>
<td>10.5987</td>
<td>0.094</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td>21</td>
<td>0.043315</td>
<td>2.28</td>
</tr>
</tbody>
</table>

5.3. Experiment 3

In order to show that the new algorithm is not only capable of dealing with sequence relations and fork relations, but also capable of dealing with parallel relations between services, we test our algorithm under Case 3.

In Case 3, the number of invocations $Th$ is between the lower $Th_{low}$ and upper bound $Th_{upp}$, but the providers react to keep the quality; the pheromone updating rules take the Equation (12) and (13). Instances SI1, SI2, SI3 and SI4 contain sequence relations and fork relations as used in the foregoing experiments. In addition, we obtain instances SI1', SI2', SI3' and SI4' respectively by transforming some OR vertexes into And vertexes in instances SI1, SI2, SI3 and SI4.

The results are shown in Table 4, which shows that the DA algorithm proposed in this paper can deal with sequence relations and fork relations as well as parallel relations. We will explain the main ideas of the proposed algorithm on one instance in the next subsection.

Table 4. The results of DA Algorithm Under Case 3

<table>
<thead>
<tr>
<th>Instances</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI1</td>
<td>Iteration</td>
<td>Distance</td>
<td>Time(s)</td>
</tr>
<tr>
<td>SI1'</td>
<td>3</td>
<td>0.007780</td>
<td>0.156</td>
</tr>
<tr>
<td>SI2</td>
<td>3</td>
<td>0.028316</td>
<td>0.579</td>
</tr>
<tr>
<td>SI2'</td>
<td>3</td>
<td>0.028376</td>
<td>0.266</td>
</tr>
<tr>
<td>SI3</td>
<td>4</td>
<td>0.028376</td>
<td>0.673</td>
</tr>
<tr>
<td>SI3'</td>
<td>2</td>
<td>0.002353</td>
<td>0.125</td>
</tr>
<tr>
<td>SI4</td>
<td>3</td>
<td>0.689360</td>
<td>1.438</td>
</tr>
<tr>
<td>SI4'</td>
<td>3</td>
<td>0.115931</td>
<td>56.126</td>
</tr>
</tbody>
</table>
5.4. An Example of the Instance System

We tested our algorithm in the Experiment 3 to show that it can be capable of dealing with sequence relations, fork relations and as parallel relations between services. The DA algorithm proposed in this paper is based on the ACO algorithm to find the optimal composite Web service in the AND/OR graph. We will take the instance systems SI3 and SI3′ as the example to show how to search.

The SI3′ is depicted in Fig.5, which contains 32 services and WS0 and WS31 are virtual vertexes. Besides, it contains one AND service, namely WS1. The structure of SI3′ is same as the SI3′ s except the WS1 in the SI3 is not an AND service but an OR service.

Suppose an ant starts from PS0,1. The ant must choose one physical service from WS1. Because WS1 is the successor of the initial service WS0, the best QoS of every physical service in WS1 is the Qos of itself. Then we calculate weights of edges between PS0,1 and every physical service in WS1 according to Equation (4) after obtaining the best QoS of every physical service in WS1. The ant chooses one physical service according to Equation (6) to arrive at WS1. Besides, the algorithm uses the local pheromone updating rule equation (12) to update the pheromone. The WS1 is an AND service vertex, so the ant must visit all of its branches.

When an ant arrives at one certain physical service as long as it is not the first Web service, the algorithm should calculate the best QoS of its successive physical service according to Equation (3) and then calculate weight of edge between them according to Equation (4).

In Case 3, though service requests exceeds the capacity of service providers, the providers keep up the quality by reacting such as scheduling more resource once they find the Qos degrades. For example, the system can enhance serviceability of the converge service WS30.

Obviously, the ant will find three branches whose initial vertex is the WS1 and the terminal vertex is WS30 in the SI3′ system.

When the ant arrives at the WS30, there are three different physical services of WS30 are chosen by the three branches. The ant chooses the best one from three different physical services of WS30 according to the strategies in subsection 3.4. The ant has find a composite Web service when it arrives at WS31.

When all ants arrive at the terminal service, the algorithm chooses the optimal one as the current optimal solution from all composite Web services obtained by these ants according to the strategies in subsection 3.4.

Then, the algorithm chooses the global pheromone updating rule equation (13) to update the pheromone.

In the Experiment 3, the DA algorithm finds the optimal composite Web service selection of SI3′ is that CWS1 = (PS1,12, PS2,1, PS5,18, PS11,7, PS17,5, PS19,7, PS22,20, PS26,7, PS28,7, PS30,2, PS1,12, PS3,2, PS30,2, PS1,12, PS4,11, PS8,2, PS14,1, PS18,7, PS20,7, PS23,12, PS27,3, PS29,18, PS30,2, PS31,1). Its QoS distance is 0.68936 and the run time is 1.438S. The times of iteration is 3.

Obviously, there existed three parallel branches in the CWs1, they are: \( \{PS1,12, PS2,1, PS5,18, PS11,7, PS17,5, PS19,7, PS22,20, PS26,7, PS28,7, PS30,2\} \), \( \{PS1,12, PS3,2, PS30,2\} \), \( \{PS1,12, PS4,11, PS8,2, PS14,1, PS18,7, PS20,7, PS23,12, PS27,3, PS29,18, PS30,2\} \). That is, the ant selects the physical service PS1,12 in WS1, then selects the parallel subsequent physical services PS2,1 in WS2, the PS3,2 in WS3, the PS4,11 in WS4, respectively.

Besides, our algorithm also found the optimal composite Web service of SI3 is that CWs2 = (PS0,1, PS1,12, PS3,2, PS30,2, PS31,1). Its QoS distance is
0.002353 and the run time is 0.125 second. The times of iteration is 2. Observing the corresponding S13 through the Fig.5, CWS₂ contains evidently less services compared with other composite Web services in S13.

6. Conclusions

QoS-aware service selection is a complex combinatorial optimization problem, which solves the problem how to find a best composition plan that satisfies user’s QoS requirements as much as possible. The existing services selection methods seldom consider the parallel relations between services as well as the dynamic changing of QoS. In order to improve these defects, this paper presents a QoS-aware composite services selection algorithm, which is based on the ant colony optimization and uses an AND/OR graph to model the composite service system.

We uses the AND/OR graph to represent the composite service system, where the AND vertex means the parallel relation and the OR vertex means the fork relation between services. The QoS values are distributed into edges of the graph; the QoS distance of a composite Web service is defined to score the quality of a composite service; and the strategy to choose the optimal composite service is also devised. Based on these work, the composite services selection problem is formulated as a multi-objective optimization problem with QoS constrains. That is, the service selection is to search the shortest solution in the AND/OR graph, so a heuristic algorithm based on ACO is proposed. Furthermore, this paper discusses the three cases of dynamic QoS under the environment that the number of invocations changed.

The results of comparison experiments show that the proposed method can deal with parallel relations between services effectively. Experimental results also show that the novel algorithm based on ant colony optimization algorithm is effective to the compared method based on genetic algorithm. In addition, the experimental results show that the dynamic policy considering the changeable of QoS is reasonable. How to set the appropriate lower threshold and upper threshold adaptively according to the preference of users is our further work.

Acknowledgments

We gratefully thank anonymous reviewers for their constructive comments and suggestions. This work was supported in part by the China NNSFC grant (No.61379114 & 61272060).

References


