Q-DWSO: hybrid approach for QoS-aware dynamic web services orchestration

Reena Gupta*
School of Computer Science and Information Technology, Devi Ahilya Vishwavidyalaya, 452001, India
Email: gupta.reena865@gmail.com
*Corresponding author

Raj Kamal
School of Computer Science and Information Technology, Devi Ahilya Vishwavidyalaya, 452001, India and
Prestige Institute of Engineering, Management and Research, Indore 452010, India
Email: dr_rajkamal@hotmail.com

Ugrasen Suman
School of Computer Science and Information Technology, Devi Ahilya Vishwavidyalaya, 452001, India
Email: ugrasen123@yahoo.com

Abstract: Many complex business-applications require the composition of tasks invoking multiple web services. The complexity of a composition process increases when a composition needs dynamic functionalities. Web services (WSs) of different service providers can be functionally identical, but have varying Quality of Service (QoS) attributes. A QoS optimisation is thus required at every stage in a composite web service (CWS). Maximising the overall utility functions are the need at both, component and composition levels. Local (task level) or global (composition using tasks in CWS) optimisation is currently used approach during the composition from the tasks. The present paper proposes a hybrid approach using the local as well as global optimisations. While local optimisation ensures the composition from the WSs in the trustworthy environment, the global optimisation satisfies the QoS constraints at the global level. The approach uses a hybrid orchestration structure. It maximises overall utility functions and reduces computational time related complexity.

Keywords: service oriented architecture; SOA; composite web service; CWS; web services composition; hybrid orchestration; quality of service; QoS; dynamic orchestration; composite service; computational time complexity; web-services local optimisation; web-services global optimisation; parallel web-services structure; sequential web-services structure.


1 Introduction

Service oriented architecture (SOA) is an architectural style used for designing a composite web service (CWS) using distributed tasks (Josuttis, 2007). Each task has specific functionality. A complex business-process (BP) design needs multiple functionalities (Rathore and Suman, 2011).

Functionality of a task may accomplish using a service among the available multiple web services (WSs). Tasks execute sequentially or in parallel. A complex BP design uses the interoperable WSs in the distributed tasks. WSs are interoperable over the web. The WSs use standards such as XML, WSDL, SOAP and UDDI (Josuttis, 2007).

One of the most used techniques for composing a CWS is orchestration. An orchestration process coordinates and sequences the WSs (Kamal and Agrawal, 2010) in the tasks. A CWS composition process can be static or dynamic in nature.

The static composition invokes a WS among the available ones in the tasks in a predefined order. One cannot change the execution order of the services at run time. The dynamic composition selects and invokes the services at run time according to the user preferences (Rathore and Suman, 2011). A user can also change the preferences dynamically. Therefore, current researchers are focusing on the dynamic composition.
Functional properties of a WS alone cannot effectively optimise a composition. WSs of different service providers (SPs) can be functionally identical, but have varying quality of service (QoS) attributes. A QoS optimisation is thus required at every stage of composing a CWS.

QoS is a non-functional property of a service and plays an important role in a dynamic WS selection during execution of a task and in the overall composition process. A dynamic selection takes into account the different QoS attributes, which bind with each WS. The basic QoS attributes are the response time, reliability, execution cost, availability and reputation (Mohamed et al., 2014; Sun and Zhao, 2012).

Various QoS-aware approaches (Alrifai and Risse, 2009a, 2009b; Amal et al., 2014; Chitra and Guptha, 2014; Dionisis et al., 2013; Fethallah et al., 2012; Hassine et al., 2006; Kang et al., 2012; Lécué and Mehandjiev, 2009; Margaris et al., 2013, 2015; Mayer et al., 2009; Mohamed et al., 2014; Qi et al., 2013; Qin et al., 2012; Sun and Zhao, 2012; Wang et al., 2010; Wu and Wang, 2011; Wu et al., 2011; Yu et al., 2014; Zhang, 2014a, 2014b; Zhao et al., 2012) have been adopted to solve the dynamic composition problem.

Most researchers consider either the local (task level) or global (CWS tasks) optimisation for the computation time complexities. One level optimisation approach has the limitations. When considering the local level optimisation, the approach does not consider the global QoS constraints. The global optimisation results in higher computation time complexities. Therefore using either local or global may not result into reduced computation time complexities. Recent studies therefore, focused on the combination of these two level optimisations to produce reduced time complexities along with the accounting for the QoS constraints.

Further, the researchers consider non-functional properties. However, the user’s functional preferences and requirements may also need additional considerations during the dynamic CWSs.

Further, the researchers mostly consider only sequential structures when orchestrating the tasks for the composite WSs, additional structures for the composition may also be needed. The existing approaches have little consideration of the computation time complexities. Some approaches are calculating their time complexity but they are producing less time efficient composition.

The objective of presented study is to consider the above four needs during QoS-aware dynamic web services orchestration (Q-DWSO). Present paper considers a hybrid approach for Q-DWSO, which is significant due to the following reasons: the approach considers user’s functional preferences and QoS attributes. The proposed approach also considers a combination of local and global optimisation.

The paper considers the use of QoS-aware WS selection scheme for local optimisation (Gupta et al., 2015). This scheme ensures the composition in trustworthy environment on the basis of trust rate of SPs. The five basic QoS attributes used here are response time, throughput, availability, reliability and execution cost. It satisfies the optimality at local level using QoS attributes and reduces the search time complexity exponentially. Locally optimised service compositions simplify the process of global optimisation for CWS. The proposed approach considers all the orchestration structures such as sequential, parallel, switch, loop and hybrid structures to provide a better performance.

The organisation of remainder of this paper is as follows. Section 2 defines the composition problem with the help of various orchestration structures and QoS...
aggregation functions. Section 3 describes the systematic procedure adopted in the presented approach. Section 4 describes an example for service composition scenario. Section 5 discusses method of implementation and evaluates the proposed approach. Section 6 discusses the derived conclusion.

2 Problem formulations

This section describes basic orchestration structures, QoS aggregation functions and formulation of composition problem. The section describes utility functions and an outline for proposed algorithm. The section also highlights the major contribution of present work.

2.1 Orchestration structures

Four basic orchestration structures for invoking WSs of the tasks for its functionality are sequential, parallel, conditional and loop (Sun and Zhao, 2012; Wang et al., 2010; Wu and Wang, 2011; Wu et al., 2011; Yu et al., 2014; Zhao et al., 2012).

Various orchestration structures are as follows:

- **Sequential structure** allows the execution of \( n \) tasks in sequence, i.e., orchestrator will invoke the services one after another.
- **Parallel structure** allows the execution of \( n \) tasks in parallel, i.e., at the same time.
- **Conditional structure** allows the execution of only one condition satisfying task.
- **Loop structure** allows the execution of a particular task repeatedly on some condition.

2.2 QoS aggregation functions for various basic orchestration structures

The present paper, considers five QoS attributes (Bin et al., 2010; Mohamed et al., 2014; Qin et al., 2012; Zhao et al., 2012): response time \( (RT) \), throughput \( (TP) \), availability \( (Avl) \), reliability \( (Rel) \) and execution cost \( (C) \).

<table>
<thead>
<tr>
<th>QoS attributes</th>
<th>Definition</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response time</td>
<td>Response time of a WS is the difference between time of receiving response and time of making the request</td>
<td>Millisecond</td>
</tr>
<tr>
<td>Throughput</td>
<td>Throughput of a WS is the number of handled requests in a particular amount of time</td>
<td>Invocations/second</td>
</tr>
<tr>
<td>Availability</td>
<td>Availability of a WS is the number of successful invocation for a particular amount of invocation</td>
<td>Percent</td>
</tr>
<tr>
<td>Reliability</td>
<td>Reliability is the probability that a service is correctly responded with an error message in an expected time period</td>
<td>Percent</td>
</tr>
<tr>
<td>Execution cost</td>
<td>It is the cost of executing a WS. It is advertised by its SP in advance</td>
<td>Pound</td>
</tr>
</tbody>
</table>

Table 1 describes the basic definitions of QoS attributes and unit for a QoS attribute.
The aggregated QoS function of a composition or orchestration structure can be determined by calculation of QoS of the services in the tasks. QoS aggregation functions of the tasks for a CWS using different orchestration structures are as per the following four sets of equations (1) to (4), each set have five expressions (a) to (e) for five QoS attributes.

Suppose that a CWS composes using a set of \( n \) tasks \( T = \{T_1, T_2, \ldots, T_n\} \). An optimisation algorithm will select one optimum service \( S \) for each \( T \). A service \( S \) is atomic that cannot be partitioned and to be implements from start to finish.

A task \( T \) is one of functionality in a process. Each task may have a condition (true or false) for its usage during the composition. Thus a set of maximum \( n \) conditions \( C = \{C_1, C_2, \ldots, C_n\} \) may be invoked during a composition of a process task. Abbreviations used therein are as follows:

\[ CS \] CWS composition of services invoked in the tasks
\[ Q_{RT}(CS) \] Response time of a CS
\[ Q_{TP}(CS) \] Throughput of a CS
\[ Q_{Av}(CS) \] Availability of a CS
\[ Q_{Rel}(CS) \] Reliability of a CS
\[ Q_{C}(CS) \] Execution cost of a CS
\[ S_i \] \( i^{th} \) component service selected in a task among a set of among \( n \)-services
\[ Q_{RT}(S_i) \] Response time of \( S_i \)
\[ Q_{TP}(S_i) \] Throughput of \( S_i \)
\[ Q_{Av}(S_i) \] Availability of \( S_i \)
\[ Q_{Rel}(S_i) \] Reliability of \( S_i \)
\[ Q_{C}(S_i) \] Execution cost of \( S_i \)
\[ C_i \] Probability that given condition is true for execution of \( S_i \)
\[ k \] Number of times iterations takes place for using the \( S_i \).

QoS attributes can be classified into four major categories based on their functionality. \( Q_{RT}(CS) \) defines the aggregated value for \( r^{th} \) QoS attribute of the composite service (CS). \( Q_{RT}(S_i) \) defines the value of \( r^{th} \) QoS attribute for \( i^{th} \) component service (\( S_i \)). Therefore, the QoS aggregation functions in four orchestration structures are as follows:

a  **Sequential structure:** response time in a CS equals the sum of individual response times of services.

\[ Q_{RT}(CS) = \sum_{i=1}^{n} Q_{RT}(S_i) \]  \hspace{1cm} (Additive attribute) \hspace{1cm} (1a)

Throughput of a sequential structure cannot exceed the throughput of any individual service.

\[ Q_{TP}(CS) = \min \{Q_{TP}(S_i)\} \]  \hspace{1cm} (Min-operator attribute) \hspace{1cm} (1b)
Availability in a CS equals the multiplication of individual availability of an $S$.

$$Q_{Avl}(CS) = \prod_{i=1}^{n} Q_{Avl}(S_i) \quad \text{(Multiplicative attributes)} \quad (1c)$$

Reliability in a CS equals the multiplication of individual reliability.

$$Q_{Rel}(CS) = \prod_{i=1}^{n} Q_{Rel}(S_i) \quad \text{(Multiplicative attributes)} \quad (1d)$$

Execution cost in a CS equals the sum of individual costs of component services.

$$Q_C(CS) = \sum_{i=1}^{n} Q_C(S_i) \quad \text{(Additive attribute)} \quad (1e)$$

b **Parallel structure**: response time of a parallel structure is the maximum response time from its individual component service.

$$Q_{RT}(CS) = \max \{Q_{RT}(S_i)\} \quad \text{(Max-operator attribute)} \quad (2a)$$

Throughput of a parallel structure cannot exceed the throughput of any individual component service.

$$Q_{TP}(CS) = \min \{Q_{TP}(S_i)\} \quad \text{(Min-operator attribute)} \quad (2b)$$

Availability in a CS equals the multiplication of individual availability.

$$Q_{Avl}(CS) = \prod_{i=1}^{n} Q_{Avl}(S_i) \quad \text{(Multiplicative attributes)} \quad (2c)$$

Equation (2c) is a sequential structure’s availability [from equation 1(c)].

Reliability in a CS equals the multiplication of individual reliability.

$$Q_{Rel}(CS) = \prod_{i=1}^{n} Q_{Rel}(S_i) \quad \text{(Multiplicative attributes)} \quad (2d)$$

Equation (2d) is a sequential structure’s reliability [from equation 1(d)].

Execution cost in a CS equals the sum of individual costs of component services.

$$Q_C(CS) = \sum_{i=1}^{n} Q_C(S_i) \quad \text{(Additive attribute)} \quad (2e)$$

Equation (2e) is sequential structure’s execution cost [from equation 1(e)].

c **Conditional structure**: response time of a conditional structure is the sum of individual response times of services multiplied by its execution condition.

$$Q_{RT}(CS) = \sum_{i=1}^{n} Q_{RT}(S_i) \times C_i \quad \text{(Additive attribute)} \quad (3a)$$

Throughput of a conditional structure is the sum of individual throughputs of component services multiplied by its execution condition.

$$Q_{TP}(CS) = \sum_{i=1}^{n} Q_{TP}(S_i) \times C_i \quad \text{(Additive attribute)} \quad (3b)$$
Availability of a conditional structure is the sum of individual availability of component services multiplied by its execution condition.

\[ Q_{avl}(CS) = \sum_{i=1}^{n} Q_{avl}(S_i) \times C_i \quad \text{(Additive attribute)} \]  

(3c)

Reliability of a conditional structure is the sum of individual reliability of component services multiplied by its execution condition.

\[ Q_{rel}(CS) = \sum_{i=1}^{n} Q_{rel}(S_i) \times C_i \quad \text{(Additive attribute)} \]  

(3d)

Execution cost of a conditional structure is the sum of individual execution cost of component services multiplied by its execution condition.

\[ Q_{e}(CS) = \sum_{i=1}^{n} Q_{e}(S_i) \times C_i \quad \text{(Additive attribute)} \]  

(3e)

d Loop structure: Response time of a loop structure is the response times of any of its services \( S \) multiplied by its number of execution iteration.

\[ Q_{rt}(CS) = k \times Q_{rt}(S) \]  

(4a)

Throughput of a loop structure is the throughput of any of its services \( S \) with exponent of its number of execution iteration.

\[ Q_{tp}(CS) = (Q_{tp}(S))^k \]  

(4b)

Availability of a loop structure is the availability of any of its component services with exponent of its number of execution iteration.

\[ Q_{avl}(CS) = (Q_{avl}(S))^k \]  

(4c)

Reliability of a loop structure is the reliability of any of its component services with exponent of its number of execution iteration.

\[ Q_{rel}(CS) = (Q_{rel}(S))^k \]  

(4d)

Execution cost of a loop structure is the execution cost of any of its component services multiplied by its number of execution iteration.

\[ Q_{e}(CS) = k \times Q_{e}(S) \]  

(4e)

The number \( k \) and constant \( C_i \), the probability that a branch executes depends on specific task during CWS and is quite complex.

Healy et al. (2000) suggested an efficient approach of automatic bounding of loop iterations and then statically compute the worst case execution time (WCET) and perform the timing analysis. Healy and Whalley (2002) suggested tighter timing predictions by automatic detection and exploitation of value-dependent constraints. Keim et al. (2009) suggested soft WCET using path analysis technique.

Actually usable values of \( k \) and \( C_i \) should be computed using these approaches during a CWS. For simplicity of computations, \( k \) and \( C_i \) can be taken as 1, though agreeably this is not effective, since it practically drops support for loops and conditionals.
A linear transformation of multiplicative attributes using logarithm is described as in equation (5).

$$\log(Q_r(CS)) = \log\prod_{i=1}^{n}Q_r (S_i) = \sum_{i=1}^{n}\log(Q_r (S_i))$$  \hspace{1cm} (5)

The logarithm conversion of multiplicative attributes is functionally similar to the additive attributes. Now, it can be used as an additive attribute for various orchestration structures as per needed.

2.3 Utility functions: QoS attributes normalisation, weighting and summation steps and defining the local and global utility functions

A utility function is defined to optimise the result. We have introduced two utility functions one for local optimisation and another for global optimisation. The concept behind introducing two utility functions is to improve the results of composition. This section contains three steps are given as follows:

- **Normalisation step:** First, there is a need to standardise the entire QoS attributes to maintain the integrity. Normalisation function is given to achieve the standardisation. There are two categories of QoS attributes: positive and negative. The aim is to minimise the value of negative attributes and maximise the value of positive attributes. Normalisation function $N_r(S)$ for negative and positive attributes is given in the equations (6) and (7), respectively:

$$N_r(S) = \frac{\max(Q_r(S_i)) - Q_r(S)}{\max(Q_r(S_i)) - \min(Q_r(S_i))} \quad \text{(negative attribute)}$$  \hspace{1cm} (6)

$$N_r(S) = \frac{Q_r(S) - \min(Q_r(S_i))}{\max(Q_r(S_i)) - \min(Q_r(S_i))} \quad \text{(positive attribute)}$$  \hspace{1cm} (7)

where $\min(Q_r(S_i))$ and $\max(Q_r(S_i))$ are the minimum and maximum values for the $r^{th}$ attribute of service class $S_i$. The $Q_r(S)$ is the $r^{th}$ attribute of the service $S$ where $S \in S_i$.

The proposed method only consider negative QoS attributes for the simplicity, because a positive $r^{th}$ attribute can be converted in negative $r^{th}$ attribute by the multiplication of $-1$.

- **Weighting and summation steps:** Sum up the normalise values of each QoS attributes and assign a weight to each attribute to calculate the QoS performance. The weighted sum is as follows:

$$U(S) = \sum_{r=1}^{m}N_r(S) \times W_r$$  \hspace{1cm} (8)

where $W_r \in [0,1]$ and $\sum_{r=1}^{m}W_r = 1$. The equation defines that $W_r$ is the weight of $r^{th}$ attribute. A service user assigns the weights to the QoS attributes. Here, $m$ is the total number of QoS attributes. The above discussed utility function is similar to than that
of the reviewed literature (Amal et al., 2014; Sun and Zhao, 2012; Wang et al., 2010; Zhang, 2014a, 2014b).

- **Defining the local and global utility function:** The proposed approach considers two utility functions for efficient composition: local and global utility functions. Local utility is the utility function defined for the component service. Global utility is the utility function defined for the CWS using quality attributes.

  The objective of local utility function is to maximise the overall quality of locally selected component services. The constraints are that the response time and execution cost of selected component service should be minimised and throughput, availability and reliability should be maximised. Local utility function ensures the optimisation at the local level.

  The objective of global utility function is to maximise the overall quality of composite service at the global level. The constraints are that the response time and execution cost of the composite service should be minimised and throughput, availability and reliability should be maximised. Global utility function ensures the optimisation at the global level.

  The idea behind defining two utility functions is that once we optimised the component services locally, then it is easy to optimise composite service globally. It improves efficiency and decreases the complexity. The local utility function can be determined using equation (9).

  \[
  U(L) = \sum_{r,k} \frac{\max(Q_r(S_i)) - Q_r(S_i)}{\max(Q_r(S_i)) - \min(Q_r(S_i))} \times W_k
  \]

  where \(Q_r(S_i)\) defines the value of \(r^{th}\) QoS attribute for \(i^{th}\) component service \((S_i)\). \(W_k\) is the weight of \(k^{th}\) attribute.

  The global utility functions can be determined using equation (10)

  \[
  U(G) = \sum_{r,k} \frac{\max(Q_r(CS)) - Q_r(CS)}{\max(Q_r(CS)) - \min(Q_r(CS))} \times W_k
  \]

  where \(X\) is the set of QoS attributes considered in the proposed approach. Further, \(\min(Q_r(CS))\) defines the minimum value of \(r^{th}\) attribute for \(CS\) and measured by the aggregation of minimum value of the \(r^{th}\) attribute of each service class involved in \(CS\). Similarly, \(\max(Q_r(CS))\) defines the maximum value of \(r^{th}\) attribute of \(CS\) and evaluated by the aggregation of maximum value of the \(r^{th}\) attribute of each service class involved in \(CS\).

2.4 **Algorithm outline**

A service \(S\) is one that cannot be partitioned and to be implements from start to finish. A task \(T\) is one of functionality in a process. There may be multiple \(S\) available for each task from different SPs. An optimisation algorithm will select one optimum \(S\) for each \(T\).
Inputs of the proposed algorithm are list of task $T = \{T_1, T_2, \ldots, T_n\}$, user preferences $P_u$, trustworthy SP’s service $S_{sp}$ and an expected output is composite service $CS$.

The proposed method considers that in a $CS$, there are $n$ services $S_1, S_2, \ldots, S_n$ and $n$ tasks $T_1, T_2, \ldots, T_n$. The $m$ QoS attributes correspond to each component service. The $CS$ therefore, needs to satisfy each $m$ QoS constraints, i.e., threshold value for each QoS attribute. We explicitly setup a threshold value for each QoS constraint based on some condition. The condition is that the throughput, availability and reliability of a $CS$ should be maximised and response time and execution cost of a $CS$ should be minimised. The objective of optimisation is the selection of an optimum component service for each task $t$ in such manner that:

1. The overall global utility function $U(G)$ of a $CS$ is maximised.
2. The $CS$ satisfies $m$, QoS constraints.

An algorithm selects optimum component services for each task that maximises the overall utility while satisfying the QoS constraints.

Present work main contributions are as follows: The present work considers a QoS-aware hybrid approach and uses hybrid orchestration structure for the composition of tasks in CWS. The hybrid approach is a combination of local and global optimisations. Two utility functions are defined for optimisations. Hybrid orchestration structure is used for the composition, which is a combination of all defined basic orchestration structures.

3 Procedure for dynamic web services orchestration

Figure 1 shows the overall procedure for dynamic composition. Initially user feeds their functional requirement in the form of request. A pre-filtration technique based on trust rate is applied before going for the search process for the user’s functional requirements. A trust rate evaluation is based on the reliability of services of its SP. The SPs, characterised with the highest trust rate are trustworthy. This kind of filtration ensures the trustworthiness of SP. Next, it will search the user preferences only for the services of a selected trustworthy SP. Binary search is applied for the search process. This selection ensures the optimisation at the local level. Then selected optimum services are composed using orchestration. Global optimisation optimises the $CS$ at the global level.

![Dynamic orchestration process](image)

3.1 Proposed architecture

Figure 2 shows the QoS-aware dynamic web services orchestration architecture (Q-DWSOA) in detail. The architecture is the combination of major three components:
pre-selection process, optimal service selection (local selection and local optimisation) and global optimisation. Figure 2(a) represents the detailed functionality of pre-selection process component. Figure 2(b) shows the process of local selection and local optimisation. Figure 2(c) illustrates the global optimisation component. The description of different components is as follows.

**Figure 2** Q-DWSOA, (a) pre-selection process (b) local selection and local optimisation (c) global optimisation
3.1.1 Pre-selection process for trusted environment

Considering that there are \( n \) SPs provision for the multiple functionally identical services. A filtration technique is required to simplify the search process. The proposed approach uses an additional QoS attribute trust rate to filter the trustworthy SP’s services. The trust rate is calculated based on the reliability of its services. Binary search algorithm applies for finding trustworthy environment. SPs with the highest trust rate are considered trustworthy. This pre-selection process filters trustworthy SPs. The trustworthy SP’s services now can be a part of the local selection and local optimisation process.

3.1.2 Local selection and local optimisation

An idea of local selection technique is to apply the search process of user preferences only on the services of filtered SPs after the pre-selection process. This step considers five QoS attributes: response time, throughput, availability, reliability and execution cost. The concept behind the local optimisation is to select one optimum service from each group. The different QoS values are mapped to a single utility value with the help of a utility function. A service with the maximum utility value is selected as the optimum one. Although, this approach results in reduced computation time complexity yet, it is not efficient for QoS-based service composition at the global level.

3.1.3 Global optimisation

The concept behind global optimisation is to compute the aggregated QoS values for all possible service combinations. A service combination that gives highest value of
aggregated utility, while satisfying the global constraints, i.e., threshold value for each aggregated QoS is selected. This approach is not efficient in terms of computation time complexity. This is because the computations are done for aggregated QoS for all possible service combinations without taking into account WS filtering mechanism. A WS filtration technique is needed for the reduced search-time complexity.

3.2 Combining local and global utility

Following describes the procedure for how local utility can combined into global utility, i.e., hybridisation of local and global utility.

3.2.1 Sequential structure

This section considers only sequential composition structure for the composition process. The \( r \)th attribute for the CS of sequential structure can be computed as follows in equation (11)

\[
Q_r(CS) = \sum_{i=1}^{n} Q_r(S_i)
\]

Note that as discussed in Section 2.2 after the logarithm transformation the entire multiplicative attributes for the sequential structure can be converted into the additive attributes. The value of minimum and maximum aggregated value for \( r \)th attribute of a composite service is evaluated using equations

\[
\min(Q_r(CS)) = \sum_{i=1}^{n} \min(Q_r(S_i))
\]

\[
\max(Q_r(CS)) = \sum_{i=1}^{n} \max(Q_r(S_i))
\]

The aggregate value of min-operator attribute, i.e., throughput can be calculated differently. Following equations calculates aggregated value for the min-operator attribute, maximum and minimum value for \( r \)th attribute of \( j \)th component service. Where, \( j \)th component service is considered for an instance to show the utility calculation for the set of min-operator attribute.

\[
Q_r(CS) = Q_r(S_j)
\]

\[
\min(Q_r(CS)) = \min(Q_r(S_j))
\]

\[
\max(Q_r(CS)) = \max(Q_r(S_j))
\]

Suppose, \( V \) is the set of additive and multiplicative attributes and \( W \) is the set of \( \min \) operator attribute. The derivation of global utility by combining the local utility calculates as follows:
Thus, from given equations (11)–(16), we have

\[
\sum_{i}^{n} U(S_i) = \sum_{i \neq j}^{n} U(S_i) + U(S_j)
\]

\[
= \sum_{i \neq j}^{n} \sum_{k = 1}^{n} \left( \frac{\max(Q_r(S_i))}{\max(Q_r(S_i))} - \frac{\min(Q_r(S_i))}{\min(Q_r(S_i))} \right) \times W_k
\]

\[
+ \sum_{k \neq l}^{n} \left( \frac{\max(Q_r(S_l))}{\max(Q_r(S_l))} - \frac{\min(Q_r(S_l))}{\min(Q_r(S_l))} \right) \times W_k
\]

\[
= \sum_{k \neq l}^{n} \sum_{i = 1}^{n} \max(Q_r(S_i)) - \sum_{i = 1}^{n} \min(Q_r(S_i)) \times W_k
\]

\[
+ \sum_{k \neq l}^{n} \left( \frac{\max(Q_r(S_l))}{\max(Q_r(S_l))} - \frac{\min(Q_r(S_l))}{\min(Q_r(S_l))} \right) \times W_k
\]

Thus, from given equations (11)–(16), we have

\[
= \sum_{k \neq l}^{n} \left( \frac{\max(Q_r(CS)) - Q_r(CS)}{\max(Q_r(CS)) - \min(Q_r(CS))} \right) \times W_k + \sum_{k \neq l}^{n} \left( \frac{\max(Q_r(CS)) - Q_r(CS)}{\max(Q_r(CS)) - \min(Q_r(CS))} \right) \times W_k
\]

\[
= \sum_{k \neq l}^{n} \left( \frac{\max(Q_r(CS)) - Q_r(CS)}{\max(Q_r(CS)) - \min(Q_r(CS))} \right) \times W_k
\]

\[
= \sum_{k \neq l}^{n} \left( \frac{\max(Q_r(CS)) - Q_r(CS)}{\max(Q_r(CS)) - \min(Q_r(CS))} \right) \times W_k
\]

\[
= U(G)
\]

The equation shows that the global utility of a CS can be derived from the summation of local utility of its component service.

3.2.2 Parallel structure

This section describes only parallel composition structure for the composition process. Note that, as we have discussed in Section 2.2 after the logarithm transformation the entire multiplicative attributes can be transformed into the additive attributes. Then, the value of the attribute of a CS for additive and multiplicative attributes in parallel structure can be calculated in a similar way as for the sequential structure using equations (11)–(13). Following equations evaluate aggregated value of max-operator attribute i.e., response time, minimum and maximum value for the attribute of the component service. Where the component service is considered for an instance to show the utility calculation for the set of max-operator attribute.

\[
Q_r(CS) = Q_r(S_p)
\]

\[
\min(Q_r(CS)) = \min(Q_r(S_p))
\]

\[
\max(Q_r(CS)) = \max(Q_r(S_p))
\]
Let $E$ be the set of additive and multiplicative attributes and $F$ be the set of $\max$ operator attribute. The global utility can derive by combining the local utility as follows:

$$
\sum_{i=1}^{n} U(S_i) = \sum_{i=1, i \neq p}^{n} U(S_i) + U(S_p)
$$

$$
= \sum_{i=1, i \neq p}^{n} \sum_{k \in E} \frac{\max(Q_r(S_i)) - Q_r(S_i)}{\max(Q_r(S_i)) - \min(Q_r(S_i))} \times W_k
$$

$$
+ \sum_{k \in F} \frac{\max(Q_r(S_p)) - Q_r(S_p)}{\max(Q_r(S_p)) - \min(Q_r(S_p))} \times W_k
$$

$$
= \sum_{k \in E} \sum_{i=1}^{n} \frac{\max(Q_r(S_i)) - \sum_{i=1}^{n} Q_r(S_i)}{\max(Q_r(S_i)) - \sum_{i=1}^{n} \min(Q_r(S_i))} \times W_k
$$

$$
+ \sum_{k \in F} \frac{\max(Q_r(S_p)) - Q_r(S_p)}{\max(Q_r(S_p)) - \min(Q_r(S_p))} \times W_k
$$

Thus, from given equations (11)–(13) and equations (18)–(20), we have

$$
U(G) = \sum_{k \in E} \frac{\max(Q_r(CS)) - Q_r(CS)}{\max(Q_r(CS)) - \min(Q_r(CS))} \times W_k + \sum_{k \in F} \frac{\max(Q_r(CS)) - Q_r(CS)}{\max(Q_r(CS)) - \min(Q_r(CS))} \times W_k
$$

$$
= \sum_{k \in E} \frac{\max(Q_r(CS)) - Q_r(CS)}{\max(Q_r(CS)) - \min(Q_r(CS))} \times W_k
$$

$$
= \sum_{k \in F} \frac{\max(Q_r(CS)) - Q_r(CS)}{\max(Q_r(CS)) - \min(Q_r(CS))} \times W_k
$$

$$
= U(G)
$$

(21)

The equation indicates that the global utility of a CS can be derived from the summation of local utility of its component service.

### 3.2.3 Conditional structure

The global utility function for the conditional structure is the summation of utility of each execution path with the condition of execution of path as follows in equation (22)

$$
U(G) = \sum_{i=1}^{n} U(EP_i) \times C_i
$$

(22)

where $EP_i$ is the execution path and $C_i$ is the probability that this condition is true for executing the path $EP_i$. If each execution path contains exactly one service $S_i$ then the calculation of global utility function is as follows in equation (23)
The global utility of a CS for the conditional structure can thus be derived by summing up the local utility of its component service and multiplying it by the execution condition.

3.2.4 Loop structure
Note that multiplicative attribute for loop structure as discussed in Section 2.2 can be transformed into additive attribute by applying logarithms as follows:

\[ Q_r(CS) = (Q_r(S_i))^k \]  
\[ \log Q_r(CS) = \log(Q_r(S_i))^k \]  
\[ \log Q_r(CS) = k \log(Q_r(S_i)) \]

Now, we can define global utility of a CS for the loop structure using equation (27)

\[ U(G) = k \times Q_r \]  
\[ U(G) = k \times Q_i(S_i) \]  
\[ U(G) = k \times U(S_i) \]

3.2.5 Hybrid structure
The above subsections describe the procedure of computing the global utility of a CS when only one basic composition structure involved, i.e., sequential, parallel, conditional or loop structure. This section focuses on the determination of the utility of a general structure that is the combination of sequential, parallel, conditional and loop structure.

The utility function of a hybrid orchestration is the summation of utility function of four basic orchestration structures defined earlier. Suppose, there are totally \( n \) tasks, \( m \) parallel tasks, \( p \) sequential tasks, \( l \)\( m - p - l \) conditional tasks and a set of \( n \) component services \( S = \{ S_1, S_2, ..., S_n \} \) involved in a hybrid orchestration. Where \( k \) is the number of iteration for the repeated execution of \( l \) task and \( C_m + p + l + 1, C_{m+p+1}, ..., C_m \) are the execution conditions corresponding to the \( T_m + p + l + 1, T_{m+p+1}, ..., T_m \). Figure 3 shows the basic functionality of hybrid orchestration structure.
We can calculate this summation with the help of equations (17), (21), (23) and (29). The utility summation of various orchestration structures is defined in the following equation.

\[
\sum_{i=1}^{n} Q_{r_i}(S_i) = \sum_{i=m+1}^{m+p} Q_{r_i}(S_i) + k \times U(G)
\]

Here, we are considering that the loop structure will execute only once for the simplicity, i.e., \(k = 1\) because for more iteration it can be multiplied by the value of \(k\). A component service will execute in conditional structures only after satisfying the condition \(C_i\) and \(C_i = 1\) for its true value.

Similar to sequential and parallel structure cases [equations (17) and (21)] is the derived equation (30) gives the result that the global utility of a hybrid orchestration can also be computed by taking the summation of utility of various available basic orchestration structures.

### 3.3 Handling of constraints

The proposed approach considering constraint classifications: first, we need to maximise and second, we need to minimise in order to maximise overall utility. There are five QoS constraints such as: response time, throughput, availability, reliability and execution cost.
Our optimisation objective is to maximise the throughput, availability and reliability of a CS and minimise the response time and execution cost of a CS. To achieve this objective we setup threshold values for each constraint of component service.

1. Maximisation constraints are that the throughput, availability and reliability of component service should not less than the $TP_{\text{max}}(60)$, $Avl_{\text{max}}(60)$ and $Rel_{\text{max}}(60)$, respectively.

2. Minimisation constraints are that the response time and execution cost should not more than $RT_{\text{min}}(35)$ and $C_{\text{min}}(25)$, respectively.

When several trustworthy SPs for a task are present, then number availability at an instant can be lower than 100% as choice of service from other SP exists that may also offer reliable service. Assume that $Avl_{\text{max}}$ 60% of the required one is the availability constraint at any instant. Similarly, at the maximum instances, constraints that TP and Rel must be 60% of the possible one are assumed.

### 3.4 Proposed algorithm

The overall procedure for the dynamic orchestration is described in Algorithm 1. Inputs to the algorithm are as follows:

1. The algorithm considers user preferences, $U_p$, as inputs.

2. Tao et al. (2012) suggested an approach, which can be used for prediction of trustworthy QoS of the WSs and for incorporating that in the existing UDDI. Later, Gupta et al. (2015) gave details of a method using a trust model, which selects the highest trust rate SP service, $S_{stsp}$. The algorithm considers $S_{stsp}$ values for the services used for composition in the UDDI registry as the inputs.

**Algorithm 1  Dynamic web services orchestration (DWSO) procedure algorithm**

Input: List of task $T = \{T_1, T_2, \ldots, T_n\}$, user preferences $P_u$, trustworthy service provider’s service $S_{stsp}$
Output: Composite service $CS$

1. Request for $P_u$/* initially a user feeds its functional preferences in the form of GUI*/
2. REPEAT UNTILL $T \neq \text{null}$ do /*A search process will execute for all task associated with each user preferences */
3. Pre-selection process to get $S_{stsp}$ /*A pre-selection technique will filter services of trustworthy service providers before going for the search process */
4. FOREACH $T$ do
5. FOREACH $S_{stsp}$ do
6. Search for $P_u$ in all $S_{stsp}$/* Now user preferences will be searched for each task for only filtered services */
7. Result[] ← invokeservice ($P_u$, $S_{stsp}$); /* An invokeservice function will invoke the desired services for user preferences from trustworthy service providers */
8. Calculate QoS attribute for each Result[] /* Calculate QoS attribute for each resultant services */
9. Update trust rate using reliability $Rel$/* It will also update the database with new calculated trust rate */
Check for QoS constraints at local level /* Now QoS constraints of the resultant services will be check for the optimality at the local level with their threshold values. If they satisfied threshold then a database the value will be stored in a database */

IF(rt ≤ RT_{\text{max}} and cost ≤ C_{\text{max}} and \text{tp} ≥ TP_{\text{max}} and rel ≥ Rel_{\text{max}} and \text{avl} ≥ Avl_{\text{max}})

THEN

Store the resultant QoS attributes

END IF

Normalise the resultant QoS attributes /* A normalisation process will apply on stored resultant QoS and a service with optimum QoS will be selected*/

Select the service, which is optimum one for each task T

Fetch the total_qos of selected component service for the task T /* Total QoS of selected component service is calculated for each task */

Calculate aggregate QoS agrqos for the CS using orchestration /* Now aggregated QoS of a CS is calculated using orchestration and user will get the resultant aggregated QoS for the CS */

agrqos ← agrqos + total_qos;

END FOR

END LOOP

Registry for the available services provides the QoS parameters for the services of each SP. The proposed algorithm takes into account the advantages of both the approaches local optimisation and global optimisation. Firstly, local optimisation approach selects the optimal service for each task T locally. It leads to the reduction in computation time complexity as optimum services are selected locally. Next, global optimisation calculates the aggregated QoS for only locally selected optimal services to maximise the utility function. So, overall approach will reduce the computational time complexity.

4 An example: e-SCM

Following describes an example of e-supply chain management (e-SCM) system. Figure 4 illustrates a composition process for e-SCM system. Consider that there are 11 tasks T = \{T_1, T_2, \ldots, T_{11}\} and a set of n component services S = \{S_1, S_2, \ldots, S_n\} involved in a composition process. Tasks T_1, T_3, T_5, T_{11} are sequential, tasks T_2, T_4, T_7 are parallel, tasks T_8, T_9, T_{10} are conditional and task T_6 executed repeatedly. Each task has its own candidate services along with their five QoS attributes response time (Q_1), throughput (Q_2), availability (Q_3), reliability (Q_4) and execution cost (Q_5). The proposed approach optimises and selects the component services locally on the basis of these QoS, trust rate of a SP and local utility function for different orchestration structures. Locally optimise services easily aggregates the QoS at the global level.
**Figure 4** A composition process of e-SCM

**Figure 5** Details of procedure for e-SCM
Figure 5 presents a detailed functional procedure of e-SCM system. The user searches for a required product. Every product is made up of different raw materials. Parallel orchestration is required at this stage. There may be multiple services of different SPs are available for the required raw materials. The proposed approach filters the SPs on the basis of the trust rate. Next, an optimal service is selected for different raw materials with respect to the required product. Now, the sequential orchestration leads to the calculation of the total QoS for the services of different raw materials. Further, user places its order and a total QoS is generated. On the basis of order placed user will pay the amount. A user can place multiple orders. There are different modes of payment based on some condition. There may be multiple services for different payment modes. Conditional orchestration is applied at this stage. An optimum service is selected for the specified condition. A total QoS is generated for the composite service at this stage.

5 Implementation and experimental evaluations

5.1 Implementation setup

We developed a business process for e-SCM example to assess the performance of proposed Q-DWSO approach. We created 50 WSs and ten tasks for the experimentation. The services created for every task of e-SCM process are login, register, trust, raw materials for different products, searching, customer order and various payment modes. Q-DWSO is implemented on advanced JAVA development platform with eclipse editor. An Axis2 server is used for the WSs creation. Windows 8.1 is the OS used for simulation. An experimentation is carried out on the 64-bit Intel (R) Core (TM) i3-3217U Processor 1.80 GHz and 4.00 GB RAM along 500 GB hard disk. The approach is tested in a LAN environment. MYSQL is the database server used for the WSs.

Initially, user feeds its functional requirements using GUI. The proposed approach divided into three steps. First step is to find the trustworthy SP’s services. Second step, is only for the trustworthy SP’s services. This step optimises the selected services at the local level with the help of QoS attributes and local utility function. Once optimisation is done at the local level, next orchestration process optimises the result at global level using global utility function.

5.2 Performance evaluation

Let $m$ is the number of QoS constraints, $t$ is the number of tasks, $s$ is the number of services for each task $t$ and $k$ is the number of occurrences of trust rate.

Gupta et al. (2015) gave the calculations of the complexity of selecting trustworthy services. The calculation showed that the complexity equals $O(t \log s k)$. The complexity of composing optimum services for each task along their QoS constraints will equal $O(m t \log s)$ using the same procedure. Therefore, for an hybrid approach, since complexities are additive, the complexity will equal $O(m t \log s + t \log s k)$.

The computational time complexity of the existing decomposition-based approach (Sun and Zhao, 2012) is $O(n m t l + n t l \log l)$. A comparative study shows that the proposed approach reduces computational time complexity exponentially.
5.3 Impact of number of WSs

Table 2 gives the complexity comparison of proposed approach with the existing approach for variable number of WSs $s$ and fixed number of tasks $t$. Figure 6 illustrates the comparison of computation time complexity of existing and proposed approach. The parameters for comparison are specified number of WSs ($s$), fixed number of tasks ($t$), occurrences of trust rate ($k$) and number of QoS constraints ($m$).

**Table 2** Comparative study for variable number of WSs

<table>
<thead>
<tr>
<th>QoS-based service composition approaches</th>
<th>Complexity of various approaches</th>
<th>Web services ($s$), tasks ($t$), occurrences ($k$), QoS ($m$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decomposition-based approach (Sun and Zhao, 2012)</td>
<td>$O(n<em>m</em>l + n<em>l</em>log_{2}l)$-service classes, $l$-candidates, $m$-QoS constraints</td>
<td>600 1,260 1,943 2,640 3,349</td>
</tr>
<tr>
<td>Hybrid approach</td>
<td>$O(m<em>t</em>log_{s} + t*log_{s}k)$-QoS constraints, $t$-tasks, $s$-services, $k$-number of occurrences of highest trust rate</td>
<td>150 195 221 240 254</td>
</tr>
</tbody>
</table>

**Figure 6** Comparison of computation time for variable number of WSs (see online version for colours)

5.4 Impact of number of tasks

Table 3 shows the comparison for the variable number of tasks $t$ and fixed number of WSs $s$. Figure 7 represents a graph for variable number of tasks ($t$), fixed number of services ($s$), occurrences of trust rate ($k$) and number of QoS constraints ($m$).
Table 3 Comparative study for variable number of tasks

<table>
<thead>
<tr>
<th>QoS-based service composition approaches</th>
<th>Complexity of various approaches</th>
<th>Web services (50), tasks (t), occurrences (10), QoS (5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decomposition-based approach (Sun and Zhao, 2012)</td>
<td>$O(n^m<em>l + n^m</em>\log 2/l)$ service classes, $l$-candidates, $m$-QoS constraints</td>
<td>669, 1,339, 2,010, 2,680, 3,349</td>
</tr>
<tr>
<td>Hybrid approach</td>
<td>$O(m^r<em>logs + t^r</em>logs*k)$ $m$-QoS constraints, $t$-tasks, $s$-services, $k$-number of occurrences of highest trust rate</td>
<td>50, 102, 153, 204, 255</td>
</tr>
</tbody>
</table>

Figure 7 Comparison of computation time for variable number of tasks (see online version for colours)

5.5 Discussions of results

The evaluation carried out shows that the suggested approach is efficient due to its time complexity calculation and reduced complexity, which lacks in most of the available approaches. Some approaches are calculating time complexity but they are not time efficient (Bin et al., 2010; Kang et al., 2012; Qi et al., 2013; Sun and Zhao, 2012). Most of the existing approaches (Alrifai and Risse, 2009a, 2009b; Bin et al., 2010; Chitra and Guptha, 2014; Dionisis et al., 2013; Fethallah et al., 2012; Kang et al., 2012; Qin et al., 2012; Wang et al., 2010; Wu et al., 2011; Zhang, 2014a, 2014b; Zhao, 2012) are considering only non-functional properties while user’s functional requirements are also important.

The proposed approach considers functional as well as non-functional parameters. A hybrid structure in proposed approach consists all basic composition structure such as
Q-DWSO: hybrid approach for QoS-aware dynamic web services orchestration

sequential, parallel, condition and loop while most of the available approaches (Alrifai and Risse, 2009a, 2009b; Bin et al., 2010; Dionisis et al., 2013; Fethallah et al., 2012; Hassine et al., 2006; Lécué and Mehandjiev, 2009; Margaris et al., 2013, 2015; Mayer et al., 2009; Mohamed et al., 2014; Qi et al., 2013; Wang et al., 2010; Wu and Wang, 2011; Wu et al., 2011; Yu et al., 2014; Zhang, 2014a, 2014b) are not working for all them. Current approaches are considering either local optimisation or global optimisation. Only some of them are hybrid (Alrifai and Risse, 2009a, 2009b; Amal et al., 2014; Zhao et al., 2012). The proposed approach is hybrid in nature as it is supporting local as well as global optimisations.

6 Conclusions

Dynamic service composition is the challenging issue when developing complex business applications. A solution is QoS-aware optimisation technique. The present paper considers following features.

The paper suggests consideration of hybrid approach for Q-DWSO. The approach thus deploys the orchestration-based sequential, parallel, conditional and loop composition structures for the CWS consisting of multiple services. The approach also includes considerations of the functional preferences of the users dynamically. The selection of trustworthy service from number of services reduces time complexity. The approach also takes into account the advantages of both the approaches local optimisation and global level optimisations taking QoS constraints.

Local optimisation approach selects the optimal service for each task based on user preferences. It reduces the computation time complexity as with the local selection. Then, the global optimisation evaluates the aggregated QoS for only locally selected optimal services to maximise the utility function. Combined effect of these is that computational time complexity reduces effectively. Further, experimental-comparison carried out shows that presented approach is more time efficient than decomposition-based service composition approach due to reduced complexity. The proposed approach thus maximises the overall utility in presence of QoS constraints.

Limitations of the presented approach are as follows:

- **Scaling up**: when the number of services is large the hybrid approach (sequential, parallel, conditional and loop composition structures) results in number of complex computations and scaling up may not be feasible.

- **QoS constraints thresholds**: assignment of threshold values for QoS parameters is subject to specific cases. Required Initial assignments of thresholds may be difficult during CWS orchestration of services.

- **UDDI registry updating**: difficulties exists in maintaining correct trust rate and QoS parameters in dynamic CWS.

- **Loops and conditional structures**: timing computations can be difficult, though it is possible to compute WCET using suggested methods of Healy et al. (2000) and Keim et al. (2009). The presented approach takes for simplicity parameter $k = 1$. 

References


Q-DWSO: hybrid approach for QoS-aware dynamic web services orchestration


