QoS-Aware Service Selection for Customisable Multi-Tenant Service-Based Systems: Maturity and Approaches

Qiang He, Jun Han, Feifei Chen, Yanchun Wang, Rajesh Vasa, Yun Yang
School of Software and Electrical Engineering
Swinburne University of Technology
Melbourne, Australia
{qhe, jhan, feifeichen, yanchunwang, rvasa, yyang}@swin.edu.au

Abstract—Multi-tenant service-based systems (SBSs) have become a major paradigm in software engineering in the cloud environment. Instead of serving a single end-user, a multi-tenant SBS provides multiple tenants with similar and yet customised functionalities with potentially different quality-of-service (QoS) values. Thus, existing approaches to service selection for single-tenant SBSs are no longer suitable. Furthermore, the target multi-tenancy maturity level also needs to be considered in the service selection approach for an SBS. In this paper, we propose three novel QoS-aware service selection approaches for composing multi-tenant SBSs that achieve three different multi-tenancy maturity levels. Extensive and comprehensive experiments are conducted and the experimental results show that our approaches outperform the existing approach in both effectiveness and efficiency.

Keywords—Cloud computing; SBS; Service Selection; Quality of Service; Multi-Tenancy; Optimisation

I. INTRODUCTION

The service-oriented paradigm is emerging as a new way to engineer software systems that are composed of services locally or remotely accessed by an execution engine (e.g., a BPEL engine [1]). Through service composition, service-based system (SBS) developers can compose existing services in the form of business processes to construct new SBSs [2, 3]. In order to offer cost-effective solutions to multiple clients in the cloud environment, an SBS must achieve multi-tenancy, i.e., the ability to satisfy multiple clients simultaneously based on a single application instance [4]. Engineering a multi-tenant SBS involves work on different levels, including data-centre level, infrastructure level and application level [5]. In this research we focus on quality management for multi-tenant SBS on the application level where the system quality are guaranteed through composition of services with the “right” quality values. The tenants of a multi-tenant SBS usually have different multi-dimensional quality requirements for the SBS, e.g., response time, throughput and availability. In addition, the SBS provider has its own optimisation goal for the SBS, e.g., least system cost or best system quality [6].

In a market where even 100ms extra delay can cost 1% drop in sales [7], fulfilling tenants’ quality requirements is critical to the success of an SBS. In the cloud environment, the candidate services available for composing an SBS often differ in their quality values. To compose a multi-tenant SBS, the developer needs to select appropriate services that collectively fulfil tenants’ quality requirements and achieve SBS provider’s optimisation goal. In the past decade, many optimisation models and service selection approaches have been proposed to compose single-tenant SBSs [2, 3, 8, 9]. The quality of such an SBS is optimised for only one tenant. However, a multi-tenant SBS in the cloud environment needs to fulfill multiple tenants’ quality requirements, rendering existing approaches obsolete and impractical. In recent years, this important and challenging issue has attracted many researchers’ attention [10-13], including ours [14].

There are two critical limitations to existing quality-aware service selection approaches for multi-tenant SBSs. First, existing service selection approaches assume that the functionalities required by all the tenants from an SBS are exactly the same. This assumption is not realistic. A real-world multi-tenant SBS must be able to provide different tenants with similar yet customised functionalities realised by enacting differentiated execution plans within the SBS [15] Suppose one needs to engineer a Travel Booking SBS for two tenants. One tenant wants to use this SBS to book airline tickets, book accommodation and purchase insurance while the other tenant only wants to book airline tickets and purchase insurance. This SBS needs an airline ticket search service and an insurance quote service shared by both tenants, plus an accommodation booking service specifically for the first tenant. We have developed an approach in [16] that customises the functionality of an SBS for different tenants. However, to our best knowledge, existing quality-aware multi-tenant service selection approaches do not consider functionality customisation, and thus are not suitable for composing such an SBS. A possible solution is to adopt the existing single-tenant approaches to compose multiple instances of the SBS, one for each tenant. However, it is very difficult for such an approach to achieve the SBS provider’s optimisation goal. While the quality delivered to tenants are individually optimised, the overall quality of the SBS are usually not optimal.

The other critical limitation of existing service selection approaches for multi-tenant SBSs is the lack of consideration for the multi-tenancy maturity level. In [17], four multi-tenancy maturity levels are introduced, with
regard to how an SBS serves multiple tenants. At the first level, an independent instance of the SBS is specifically composed and optimised for each tenant. This level corresponds to the traditional service selection approaches for SBS [2, 3, 18-20]. At the second level, tenants share the execution engine of the SBS that enacts execution plans customised for the tenants. At the third level, all tenants share the execution engine of the SBS and one set of services that compose the SBS. At the fourth level, all tenants’ workloads are balanced over multiple instances of the SBS. To compose an SBS that fulfils all tenants’ quality requirements at different maturity levels of multi-tenancy, different quality-aware service selection approaches are needed. In this paper, we propose three innovative service selection approaches for composing SBSs that achieve the second, the third and the fourth multi-tenancy maturity levels respectively. Our approaches can be employed at the design phase of system development to help select the appropriate multi-tenancy level and produce optimal system quality at the selected multi-tenancy maturity level.

Focusing on two of the most critical objectives for multi-tenant SBSs, i.e., functionality customisation and multi-tenancy maturity, the major contributions of this paper are threefold:

- Firstly, our approaches allow SBSs to offer multiple tenants customised functionalities at differentiated quality levels.
- Secondly, complementary to existing service selection approaches, which focuses on single-tenant SBSs, our approaches focus on multi-tenant SBSs at the second, third and fourth maturity levels of multi-tenancy.
- Thirdly, extensive and comprehensive experiments are conducted using a published Web service dataset, which contains over 2500 real-world Web services.

The rest of this paper is organised as follows: Section II analyses the problem with a motivating example. Section III introduces our compositional quality model, optimisation model and service selection approaches. Section IV presents the experimental results. Section V introduces related work. Section VI concludes the paper.

II. MOTIVATING EXAMPLE AND PROBLEM ANALYSIS

Figure 1 shows an example of multi-tenant SBS that provides travel booking service. It has three tenants: 1) Webjet, a travel booking company offering flights, car rental and hotel bookings based in Australia; 2) Rail Plus, the leading dedicated rail trip specialist general sales agent throughout Australia & New Zealand; and 3) P&O Cruise, Australia and New Zealand’s leading cruise line.

This SBS serves the three travel agents by processing their customers’ requests. A customer enters their travel requirements, e.g., city of departure, destination, departure date, return date, preferred type of rental car, etc. In response to the request, the SBS returns a list of candidate travel plans for the customer to book.

The functionality of this SBS is represented as a business process that includes six tasks (t1, ..., t6). For customers of different tenants (travel agents), the SBS performs different tasks to generate travel plans. The Airline Ticket Search, Car Rental, Hotel Search and Insurance Quote tasks are performed for Webjet’s customers, the Train Ticket Search and Hotel Search tasks for Rail Plus’s customers, and the Cruise Ticket Search and Insurance Quote tasks for P&O Cruise’s customers. All customers from different travel agents share not all but only some of the tasks. In this SBS, the Insurance Quote task is performed for customers from Webjet and P&O Cruise while the Hotel Search task is performed for customers from Webjet and Rail Plus.

On one hand, these travel agents usually have different multi-dimensional requirements for the quality of the SBS. For instance, P&O Cruise requires a very fast response time despite a high price, while Webjet is more concerned about minimising the cost of using the SBS. The SBS provider, on the other hand, also has its own optimisation goal for the SBS, e.g., to minimise the system cost of the SBS, i.e., the total cost of the services selected to compose the SBS. A set of services must be selected from the candidate services to perform the tasks of the SBS that serves the travel agents with satisfactory quality and achieves the SBS provider’s optimisation goal. Similarly to other research efforts [2, 3, 8, 9, 14], we assume that alternative functionally-equivalent services are available and can be categorised based on their functionalities.

Depending on the target multi-tenancy maturity level, this SBS can be composed in different ways:

Level 1: Three independent systems are composed, one specifically customised for each travel agent. The travel agents do not share an execution engine or any component services.

Level 2: One system instance is composed that enacts three independent execution plans, each specifically customised for one travel agent. The travel agents share an execution but not the component services, as illustrated in Figure 2 later.

Level 3: One system instance is composed that enacts three execution plans that share certain tasks, one specifically customised for each travel agent. The travel agents share the execution engine and certain component services, as presented in Figure 3 later.

Level 4: A set of system instances are composed and shared by all three travel agents. The travel agents share all
the execution engines and certain component services of each system instance, as depicted in Figure 4 later.

III. SYSTEM OPTIMISATION

In this section, we first describe the compositional quality model and the optimisation model adopted in this research. Then, we present the service selection approaches for composing multi-tenant SBS at different multi-tenancy maturity levels.

A. Compositional Quality Model

In this research, we adopt the same compositional structures for representing the business processes of SBSs. The compositional structures include sequence, branch, loop and parallel [2], which are included in BPMN [21] and addressed by BPEL [1] - the de facto standards for specifying service-oriented business processes.

A multi-tenant SBS must fulfill multiple tenants’ multi-dimensional quality constraints. Thus, we need to evaluate the quality of the SBS delivered to individual tenants, considering all the execution plans customised for the tenants. Take the SBS presented in Figure 1 as an example, suppose there are three execution plans, eplWebjet, eplP&O for P&O Cruise and eplRailPlus for Rail Plus. The system quality delivered to a tenant can be calculated by aggregating the quality of the services selected for the corresponding execution plans based on the compositional quality model presented in [14]. In this paper, examples are based on cost (or price) and response time, which also have been the basis for quality evaluation in other approaches [2, 3, 8]. Other QoS parameters can be generalised as added dimensions in the quality evaluation.

More details about the compositional structures and the quality evaluation methods can be found in [3, 14].

B. Optimisation Model

Suppose an SBS S consists of n (n≥1) components. Accordingly, there are n service classes S_i, i=1, ..., n, each containing r (r≥1) available candidate services S_j, i=1, ..., r, that provide the same functionality but potentially differ in v quality dimensions q_{j,p}, p=1, ..., v. The service selection problem for S that serves m (m≥1) tenants is a constraint optimisation problem (COP) that aims at finding a set of services, which, when executed according to tenants’ customised execution plans, can fulfill corresponding tenants’ v-dimensional quality constraints c_{k,i}, k=1, ..., m, p=1, ..., v, while achieving the SBS provider’s optimisation goal objective(S).

We first model this problem as a constraint satisfaction problem (CSP), which consists of a finite set of variables X={x_1, ..., x_n}, with respective domains D={D_1, ..., D_n} listing the possible values for each variable, and a set of constraints C={c_1, ..., c_l} over X. A solution to a CSP is an assignment of a value to each variable from its domain such that every constraint is satisfied.

Solving the above CSP may generate many solutions that fulfill all tenants’ quality constraints for the SBS. These solutions usually yield different overall system quality at different system costs. Now we seek to achieve the SBS provider’s optimisation goal for the SBS, which in the model is represented by an objective function objective(S ). The CSP now turns into a COP. In a COP, each solution generated by solving the CSP is associated with a ranking value for the objective function. The solution with the optimal ranking value is the solution to the COP, i.e., the optimal solution to the service selection problem for the SBS. In general, system providers’ optimisation goals can be various, which can be represented using different objective functions. In this paper, we use a typical optimisation objective as an example: to minimise the system cost of the SBS, i.e., the total price of all the selected services.

This COP can be solved by applying Integer Programming techniques [3] (or the Mixed Integer Programming technique [2, 8] if decimal variables are involved). Based on the results, services can be selected to build a multi-tenant SBS that fulfills all tenants’ quality constraints and achieves the SBS provider’s optimisation objective at the target multi-tenancy maturity level.

C. Optimisation at Maturity Level 2

At the second multi-tenancy maturity level, the execution engine of the SBS enacts a separate customised execution plan for each tenant. Figure 2 presents the travel booking SBS that achieves the second multi-tenancy maturity level based on the business process presented in Figure 2. In this system, t_1 is shared by Webjet and Rail Plus. Thus, two services are selected for t_1, one for execution plan #1 to serve Webjet and the other for execution plan #3 to serve Rail Plus. It is the same for t_6, which is shared by Webjet and P&O Cruise. Assume that s_{1,3} is selected for Webjet and s_{5,6} is selected for Rail Plus to execute t_1, s_{6,2} for Webjet and s_{6,5} for P&O Cruise to execute t_6. The execution plans for the three tenants will be epl_{web}(s_{1,3}, s_{2,5}, s_{3,3}, s_{6,2}), epl_{p&O}(s_{1,3}, s_{6,5}) and epl_{rail}(s_{4,2}, s_{5,6}), as presented in Figure 2. Upon the receipt of a request, the execution engine of the SBS enacts the corresponding execution plan based on the sender of the request. Each execution plan is specifically customised for a tenant based on its quality requirement.

For m tenants, there are m × n × r 0-1 variables X_{k,i,j} (k=1, ..., m, i=1, ..., n, j=1, ..., r and D_{k,i,j}={0, 1}), X_{k,i,j} being 1 if the jth candidate service in the ith service class is selected to create the execution plan for the kth tenant, and 0 otherwise. To compose a multi-tenant SBS at the second multi-tenancy level, the CSP model for service selection is formally expressed as follows:

![Figure 2. Multi-tenant SBS at maturity level 2.](image-url)
The first two constraint families in the CSP model are:

\[ \sum_{k=1}^{m} X_{k,i,j} \leq 1 \quad \forall k \in [1, m], \quad j \in [1, r] \]  

(1)

\[ \sum_{i=1}^{n} X_{k,i,j} \leq 1 \quad \forall i \in [1, n], \quad j \in [1, r] \]  

(2)

\[ \sum_{k=1}^{m} \sum_{i=1}^{n} X_{k,i,j} = m \cdot g_{k,j} \quad \forall j \in [1, r] \]  

(3)

where \( g_{p}(\text{epl}_k) \) is the \( p \)-th quality dimension of the execution plan for the \( k \)-th tenant.

Constraint family (1) guarantees that a maximum of one service is selected in each service class for each tenant. Constraint family (2) guarantees that each service can only be selected for one tenant maximum. Constraint family (3) ensures that for each task \( t \) shared by \( \sum_{k=1}^{m} s_{k,j} \) tenants, the same number of services are selected in the corresponding service class. Constraint family (4) ensures that each tenant’s \( v \)-dimensional quality constraints are fulfilled by the corresponding execution plan.

Suppose the optimisation objective is to minimise the system cost. The corresponding objective function is as follows:

\[ \text{objective(S)}: \minimise \sum_{i=1}^{n} \sum_{j=1}^{r} q_{\text{price}}(s_{i,j}) \times X_{i,j} \]  

(5)

D. Optimisation at Maturity Level 3

At maturity level 3, the execution engine enacts customised execution plans for different tenants that share certain services. Figure 3 presents the travel booking SBS that achieves the third multi-tenancy level.

To optimise an SBS at the third multi-tenancy level, the first two constraint families in the CSP model are:

\[ \sum_{i=1}^{n} X_{i,j} = 1 \quad \forall j \in [1, r] \]  

(6)

\[ q_{k,p}(S) < c_{k,p} \quad \forall k \in [1, m], \quad p \in [1, v] \]  

(7)

where \( q_{k,p}(S) \) is the \( p \)-th quality parameter of \( S \) delivered to the \( k \)-th tenant.

Constraint family (6) guarantees that one and only one candidate service is selected in each service class. Constraint family (7) ensures that each of the \( m \) tenant’s \( v \)-dimensional quality constraints are fulfilled.

At the third multi-tenancy level, some tasks of the SBS are shared by multiple tenants. In the example presented in Figure 1, \( t_3 \) is shared by Webjet and Rail Plus and \( t_5 \) is shared by Webjet and P&O Cruise. Thus, the maximum throughput of the selected services that perform these shared tasks must be large enough to fulfil all tenants’ combined requirements for throughput. In Figure 3, the maximum throughput of \( s_{3,3} \) must be large enough to handle the requests sent by Webjet and Rail Plus. To include this constraint in the CSP model, we build an \( m \times r \) service sharing matrix \( G = [G_1, G_2, \ldots, G_r] \) to record which tasks are shared by which tenants, where \( g_{ij} \) indicates whether the \( j \)-th task is shared by the \( i \)-th tenant. Take the SBS presented in Figure 1 for example:

\[
G = \begin{bmatrix}
111001 \\
000001 \\
001100
\end{bmatrix}
\]

We also build a \( 1 \times m \) throughput constraint matrix \( C_{tp} = [c_{1,tp}, \ldots, c_{n,tp}] \) that records \( m \) tenants’ throughput constraints. Take the tenants in Figure 1 for example: \( C_{tp} = \begin{bmatrix}
e 1, 2, 3 \end{bmatrix} \), where \( c_{kp} \) represents the \( k \)-th tenant’s throughput requirement.

Based on the service sharing matrix and the throughput constraint matrix, tenants’ throughput requirements can be included in the CSP model as constraints:

\[ \sum_{i=1}^{n} X_{i,j} \times q_{\text{tp}}(s_{i,j}) \geq C_{tp} \times G \quad \forall j \in [1, r] \]  

(8)

where \( q_{\text{tp}}(s_{i,j}) \) is the throughput of service \( s_{i,j} \). This constraints family guarantees that the maximum throughput of each shared service is higher or equals to the throughput required by all tenants that share the service.

The corresponding objective function is as follows:

\[ \text{objective(S)}: \minimise \sum_{i=1}^{n} \sum_{j=1}^{r} q_{\text{price}}(s_{i,j}) \times X_{i,j} \]  

(9)

E. Optimisation at Maturity Level 4

At the fourth maturity level multi-tenancy, tenants’ requests are balanced over a group of system instances \( \{S_4^1, S_4^2, \ldots, S_4^4\} \). Each system instance has its own execution engine that enacts the same execution plans using its own set of services. Figure 4 presents an example where Webjet, P&O Cruise and Rail Plus’s requests are balanced over three system instances, i.e., \( S_4^1, S_4^2, S_4^3, S_4^4 \). The COP model for such an SBS can be expressed as below.

For \( m \) tenants, there are \( n \times r \times n-0 \) variables \( X_{h,i,j} \), \( h=1, \ldots, n, \ i=1, \ldots, n, \ j=1, \ldots, r \), and \( D_{ijh} = [0, 1] \), \( X_{h,i,j} \) being 1 if the \( h \)-th candidate service in the \( j \)-th service class is selected to create the execution plan for the \( h \)-th system instance, i.e., \( S_4^h \), and 0 otherwise. The constraints for the
CSP model are:

\[
\sum_{i=1}^{n} X_{h,i,j} = 1 \quad \forall i \in [1, n] \quad j \in [1, r]
\]

(10)

\[
\sum_{i=1}^{n} \sum_{j=1}^{r} \sum_{k=1}^{m} X_{h,i,j} = \sum_{i=1}^{n} \sum_{j=1}^{r} X_{h,i,j} \quad \forall x, y \in [1, r], x \neq y
\]

(11)

\[
\sum_{j=1}^{r} X_{h,i,j} = 1 \quad \forall h \in [1, n] \quad j \in [1, r]
\]

(12)

\[
q_p(S^h_k) < c_{k,p} \quad \forall h \in [1, n], k \in [1, m], p \in [1, v]
\]

(13)

\[
\sum_{h=1}^{n} q_p(S^h_k) \geq \sum_{k=1}^{m} c_{k,p}
\]

(14)

Constraint family (10) guarantees that a maximum of one service is selected in the target service class for each system instance. Constraints family (11) ensures that the same number of services are selected from each service class for the composition despite the number of system instances. Constraints family (12) guarantees only one service from each service class is selected for each system instance. Constraints family (13) makes sure that all tenants’ quality constraints are fulfilled. Constraint (14) is specific to the fourth multi-tenancy level, which expresses the constraint that the total throughput of all the system instances must fulfil all tenants’ throughput requirements combined.

The objective function for minimising the system cost is:

\[
\text{objective(S)} : \text{minimise} \sum_{h=1}^{n} \sum_{j=1}^{r} \sum_{i=1}^{m} d_{price}(s_{i,j}) \times X_{h,i,j}
\]

(15)

F. Discussion

Before selecting services to compose an SBS, the SBS designer needs to decide the target multi-tenancy level. One might expect the fourth multi-tenancy level to be the ultimate goal for all SBSs, but it is not always the case [4]. It is more realistic to think of multi-tenancy maturity level as a tradeoff between system sharing, customisation and scalability. In general, the first level provides the best customisation for tenants, while the third level allows tenants to share an SBS to the highest degree. In terms of system scalability, the second and fourth multi-tenancy levels are the winners. There are advantages and disadvantages to each level. Choosing the target multi-tenancy level can be a complicated decision making process, in which the business, architectural and operational needs must be considered.

Our approaches employ different COP models to optimise SBSs at different multi-tenancy levels. Optimising a single-tenant SBS has been known as a NP-complete problem [22]. Optimising a multi-tenant SBS needs to fulfill multiple tenants’ quality constraints and might need to compose multiple execution plans and/or system instances. Thus, the service selection problem is more complicated for multi-tenant SBSs. Solving the problem might take a significant amount of time depending on its complexity and scale. Unfortunately, there are no common complexity measurements for COP models and formulations. However, the size of the model, measured by the number of its variables and constraints is a direct and practical indicator of its complexity. Table I presents the sizes of the COP models at different multi-tenancy levels. In general, the comparison in size between the models is: level 1 < level 2 < level 3 < level 4. In most cases, the level-4 model is significantly larger than the others. Detailed experimental studies on the efficiency of the approaches at all four multi-tenancy levels are presented in Section IV.B.

IV. EXPERIMENTAL EVALUATION

This section presents the experimental evaluation of our approach, focusing on the comparison between our approach and the existing single-tenant approach in terms of effectiveness (measured by success rate and objective value) and efficiency (measured by computational time).

A. Experimental Setup

We develop a prototype that implements the proposed approaches in Java using JDK 1.6.0 and Eclipse Java EE IDE. For solving the COPs, we use CPLEX v12.6.

We conduct a series of in-lab experiments. The evaluation process mimicked a large number of SBSs on different scales. Candidate services are synthetically generated based on a publicly available Web service dataset QWS [23], which comprises measurements of nine

<table>
<thead>
<tr>
<th>Maturity Level</th>
<th>Number of Tenants</th>
<th>Number of Service Classes</th>
<th>Number of Candidate Services per Service Class</th>
<th>Number of quality constraint dimensions</th>
<th>Number of Instances of S</th>
<th>Number of Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1</td>
<td>1</td>
<td>n</td>
<td>r</td>
<td>v</td>
<td>1</td>
<td>m × n × r</td>
</tr>
<tr>
<td>Level 2</td>
<td>m</td>
<td>n</td>
<td>r</td>
<td>v</td>
<td>m × n × v × r</td>
<td>m × m × n × r</td>
</tr>
<tr>
<td>Level 3</td>
<td>m</td>
<td>n</td>
<td>r</td>
<td>v</td>
<td>m × n × r</td>
<td>m × m × n × r</td>
</tr>
<tr>
<td>Level 4</td>
<td>m</td>
<td>n</td>
<td>r</td>
<td>v</td>
<td>m × n × r × v × r</td>
<td>m × m × n × r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maturity Level</th>
<th>Number of Constraints on Task Coverage</th>
<th>Number of Constraints on Quality</th>
<th>Number of Constraints on Service Exclusiveness</th>
<th>Number of Constraints on Services for Shared Tasks</th>
<th>Total Number of Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1</td>
<td>n</td>
<td>v</td>
<td>0</td>
<td>0</td>
<td>n × v</td>
</tr>
<tr>
<td>Level 2</td>
<td>m × n</td>
<td>m × v</td>
<td>n × r</td>
<td>0</td>
<td>m × n × m × v × r</td>
</tr>
<tr>
<td>Level 3</td>
<td>n</td>
<td>m × v + r</td>
<td>constraint family (7)</td>
<td>0</td>
<td>n × v + n × m × v + n × r</td>
</tr>
<tr>
<td>Level 4</td>
<td>( \frac{n!}{2 \times (n-2)!} )</td>
<td>m × r × v + 1</td>
<td>constraint family (13)</td>
<td>0</td>
<td>n × r × v + 1 × m × r × v</td>
</tr>
</tbody>
</table>

Experimental Setup: We use CPLEX v12.6 for solving the COPs. Our prototype is implemented in Java using JDK 1.6.0 and Eclipse Java EE IDE.
quality parameters of over 2500 real-world Web services. The data was collected from public UDDI registries, search engines and service portals. Their quality values were measured using commercial benchmark tools.

We compare our approaches with the level-1 approach, i.e., single-tenant service selection approach based on Integer Programming, which is as far as we know the most popular and representative technique adopted in research on quality-aware service compositions [2, 3, 8, 9, 18]. To accommodate m tenants, the single-tenant approach generates m execution plans, one at a time. Each of the execution plans must fulﬁl the corresponding tenant’s quality constraints while minimising the system cost. To make the level-4 approach more realistic, we limit the maximum number of instances to the number of tenants.

For effectiveness evaluation, we compare the success rates achieved by the approaches, i.e., the percentage of runs where an optimal solution can be found. We also compare the obtained objective values, i.e., the system cost, as presented in formula (5), (9) and (15). For each service, a cost is randomly generated from a range of 0 to 100. For efficiency evaluation, we compare their computational time, i.e., the time taken to find an optimal solution.

In the experiments, we utilise the example SBS presented in Figure 1. We randomly partition the candidate services into five categories, each as one of the five service classes of candidate services corresponding to the five tasks. Tenants’ quality requirements are randomly generated from a range of values based the difficulty degree of tenants’ quality requirements. To study our approaches on different scales, we simulate experiment scenarios where the scale varies in four factors: 1) the number of quality dimensions, which determines the number of each tenant’s quality requirements; 2) the number of tenants, which determines the number of quality constraint sets; 3) the number of candidate services per service class, which determines the size of the search space of the COP; and 4) the difficulty degree of tenants’ quality requirements, which determine the difficulty degree of the COP. Table II presents the configuration details of the experiments. For each set of experiment, we average the results obtained from 100 runs.

All experiments are conducted on a machine with Intel i5-4570 CPU 3.20GHz and 8 GB RAM, running Windows 7 x64 Enterprise.

### Table II. Experiment Configuration

<table>
<thead>
<tr>
<th>Factor</th>
<th>Experiment Serial #1</th>
<th>Experiment Serial #2</th>
<th>Experiment Serial #3</th>
<th>Experiment Serial #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Quality Dimensions</td>
<td>1 to 10 in steps of 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Number of Tenants</td>
<td>5</td>
<td>1 to 10 in steps of 1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Number of Candidate Services per Task</td>
<td>50</td>
<td>50</td>
<td>10 to 100 in steps of 10</td>
<td>50</td>
</tr>
<tr>
<td>Difficulty Degree of Quality Constraints</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>10 to 100 in steps of 10</td>
</tr>
</tbody>
</table>

**B. Experiment Results**

**Success Rate**

Figure 5 compares the success rates of four approaches from four sets of experiments where in each set we vary one of the four factors while fixing the other three. It shows that the level-3 approach achieves the worst success rate while the other three approaches achieve 100% success rates most of the time except when the difficulty degrees of tenants’ quality constraints are higher than 60. Figures 5(a) and 5(b) show that the increases in the number of tenants and the difficulty degree of tenants’ quality requirements directly reduce the success rate of level-3 approach. This observation indicates that when the number of tenants is large or/and their quality constraints are severe, it is difficult to fulﬁl their constraints for throughput by selecting just one service to perform each task of the system. Under such circumstances, it is easier to achieve multi-tenancy levels 1, 2 and 4 which allow tenants’ workloads to be shared among multiple services or multiple system instances.

**Objective Value**

In the experiments, the optimisation goal is to achieve a minimum system cost. Figure 6 compares the average objective values obtained by the four approaches. It shows that our approaches signiﬁcantly outperform the level-1 approach in system optimisation. In all the four sets of experiments in this series, the performances of the four approaches in minimising the system are consistent: level-3 approach > level-2 approach > level-4 approach > level-1. The level-3 approach excels in this set of experiments because it requires only one service for each task to compose the system while other approaches employ multiple services for at least some of the tasks. Figure 6 also shows that the number of tenants, the number of candidate services per task and the difficulty degree of quality constraints have huge impact on the objective values obtained by the optimisation approaches, but not the number of quality dimensions.

Please note that some data points are missing from Figures 6(b) and 6(d), as well as Figure 7(b) and 7(d) because no solution can be found in certain cases.

**Computational Time**

Figure 7 compares the computational times taken the four approaches to find an optimal solution. It shows that the level-1, level-2 and level-3 approaches manage to find
an optimal solution within a reasonable amount of time (less than 100 milliseconds). For these approaches, the increases in computational time due to the increases in the changing factors are relatively gentle. However, the time taken by the level-4 approach to find an optimal solution increases drastically as the scenario scales up, especially in the four set of experiments where we increase the difficulty degree of tenants’ quality requirements for the system. This observation is consistent with our analysis in Section III.F.

Overall Comparison

Level-1 approach. This approach does not demonstrate any advantages against our approaches in any situations. It does not have a particularly high success rate or optimise SBSs to a higher degree. Being the simplest and most naïve approach, it is not even the most efficient one under any circumstances.

Level-2 approach. This approach has the best performance in finding a solution in tough situations where the tenants’ quality constraints for the system are extremely severe. It ranks second only to the level-3 approach in system optimisation and efficiency, but the difference is only marginal.

Level-3 approach. This approach has the worst performance in finding a solution successfully. Unlike other approaches, it guarantees a 100% success rate only in scenarios where the numbers of tenants are small and the difficulty degrees of their quality constraints for the system are low. The advantages of the level-3 approach lie in its ability to optimise the system and do so efficiently.

Level-4 approach. This approach performs consistently and exceptionally well in guaranteeing the success rate of finding an optimal solution and it loses only to the level-2 approach when the tenants’ quality constraints are extremely severe. The SBS composed using this approach balances tenants’ workload over multiple system instances. Thus, it requires multiple services for each of its tasks, resulting in higher system cost than the level-2 and level-3 approaches. Low efficiency is its major disadvantage, making it a less practical choice when a solution needs to be found fast.

V. RELATED WORK

A lot of efforts have been devoted to selecting services to compose a new software system (often referred to as a composite service) that fulfils a single user’s quality requirements. We refer to this type of service selection approaches as single-tenant service selection approaches. Representative works include [2, 3, 8, 9, 18-20]. The common and critical limitation of these existing approaches in the cloud environment is that they only support single-tenant SBSs. A workaround is to employ these approaches to create multiple execution plans or systems for multiple tenants one after another. The result is that, although the created systems can fulfil individual users’ quality requirements, the overall system quality is usually sub-optimal [14]. The SBS provider’s optimisation goal for the SBS cannot be achieved. Our experimental also indicate that this workaround is not efficient either.

The transition from single-tenancy to multi-tenancy in service composition has raised many issues that attracted researchers’ attention in the past few years. To name a few, Pathirage et al. [12] presents the architecture of their multi-tenant business process engine based on BPEL. SBSs built upon this engine can achieve the second multi-tenancy level by sharing the process engine across multiple tenants. This piece of work builds the foundation for the implementation of multi-tenant SBSs based on BPEL. However, this engine does not provide system quality assurance, which relies completely on the quality of the services selected to be executed. In [14], we present an approach to quality-aware service selection for multi-tenant SBSs. We model the service selection problem as a COP where tenants’ differentiated quality requirements are modelled as multiple sets of constraints. Following our work, Lei et al. [10] adopt queuing theory and reliability

![Figure 6. System cost vs. factors.](image)

![Figure 7. Computational time vs. factors.](image)
theory to enable multi-tenant SBS to adapt to changes in individual tenants’ networking environments. Wada et al. [11] design a multi-objective genetic algorithm named $E^3$ to produce a set of Pareto execution plans for multiple tenants of an SBS. Each execution plan fulfills an individual tenant’s quality requirements. Cai et al. [13] propose a new concept named service granularity space (SGS) based on granularity computing. SGS aims to address many issues in multi-tenant service compositions, including multi-granularity, massive customisation, service correlation and system evolution.

The major limitations of the above works are twofold: 1) lack of business process customisation for SBSs; and 2) lack of consideration of the third and fourth maturity levels of multi-tenancy for SBSs. Our approaches allow customisable SBSs to be built to achieve the second, third or the four multi-tenancy maturity level.

VI. CONCLUSIONS

In this paper, we propose three service selection approaches that optimise multi-tenant SBSs at different maturity levels of multi-tenancy. To our best knowledge, our approaches are the first to: 1) allow service-based systems (SBSSs) to offer tenants customised functionalities at different quality levels; and 2) to select services that optimise multi-tenant SBSs at three different multi-tenancy levels. Comprehensive experimental analysis shows the effectiveness and efficiency of our approaches.

In certain scenarios where tenants’ quality constraints are extremely severe or the scales are too large, our approaches cannot find or take too long to find an optimal solution. We will in our future work attempt to 1) employ heuristics to reduce the search space for the optimisation problem; and 2) help SBS designers achieve specific multi-tenancy maturity levels by relaxing tenants’ quality constraints; 3) quantitatively analyse the tradeoff between system scalability, sharing and customisation at different multi-tenancy maturity levels.

ACKNOWLEDGMENT

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

REFERENCES


[16] I. Kumara, J. Han, A. Colman, and M. Kapuruge, "Runtime Evolution of Service-Based Multi-tenant SaaS Applications," Proc. 11th International Conference on Service-Oriented Computing (ICSOC 2013), Berlin, Germany, 2013, pp. 192-206.


