2017 IEEE 14th International Conference on Services Computing

Service Selection based on Correlated QoS Requirements

Yanchun Wang*, Qiang He*, Dayong Ye*, Yun Yang†*
*School of Software and Electrical Engineering, Swinburne University of Technology, Australia
†School of Computer Science and Technology, Anhui University, China
Email: {yanchunwang, qhe, dye, yyang}@swin.edu.au

***Abstract*—With the proliferation of service-based systems (SBSs), more and more functionally equivalent services with different Quality-of-Service (QoS) are emerging. System vendors can select from these services to create service compositions to fulfill users' multi-dimensional QoS requirements. In order to address the QoS-aware service selection problem, various approaches have been proposed in recent years. However, most of them assume that the users' QoS requirements are deterministic and different QoS dimensions of the requirements are independent of each other. In fact, correlations may exist between different QoS dimensions of users' QoS requirements, which make existing service selection approaches impractical. In this paper, we propose SCORE-QoS (Service Selection based on Correlated QoS Requirements), in which the correlated QoS requirements are represented with QoS correlation functions and integrated into the Constraint Optimization Problem that models the service selection process. Two types of correlated QoS requirements, with and without optimization goals, are considered in SCORE-QoS. The experimental results show that our approach can effectively and efficiently select services for composition based on correlated QoS requirements.**

***Keywords*-correlation; QoS; service selection; service-based system; service composition; QoS correlation;**

I. Introduction

Service-based systems (SBSs) have become a predominant way to engineer software systems by composing network-accessible and often distributed Web services, which collectively fulfil users' functional and QoS (Quality of Service) requirements [1]. Due to the proliferation of cloud computing, more and more functional equivalent services are emerging, which are characterized by differentiated multi-dimensional QoS values such as cost and throughput [2]. For each task in an SBS, the SBS vendor can select a proper service from a set of functional equivalent candidate services to create a service composition to fulfil users' multi-dimensional QoS requirements [3]. QoS-aware service selection is considered as a complex multi-criteria decision problem [4], which has been attracting a lot of attention in recent years [5][6][7].

QoS correlations between different services have been well investigated in service selection by many works [2][8][9]. This type of QoS correlation means that some QoS dimensions of a service are not only dependent on the service itself but also correlated to other services [8]. For example, an SBS vendor can offer multiple services with different functionalities as a package with a 20% discount in the total price. However, the correlations between different QoS dimensions of users' QoS requirements are still insufficiently studied. It means that some QoS dimensions of users' QoS requirements are correlated to other dimensions, which is a real-world reflection of the trade-offs between different dimensions of QoS requirements. For example, a user of an SBS may accept a lower throughput at a lower cost, but is willing to pay more if a higher throughput is provided. In this way, a user's multi-dimensional QoS requirements become dynamically varied and correlated. This is significantly different from the scenarios handled by most existing works in service selection [5][6][10], where all the users' QoS requirements are deterministic and different QoS dimensions in the requirements are independent of each other. How to select proper services for compositions based on correlated QoS requirements has become a new challenge, which makes the existing approaches impractical.

In order to address the above issue, this paper proposes a novel approach for service selection based on correlated QoS requirements, named SCORE-QoS. The major contributions of this paper are as follows:

1) SCORE-QoS innovatively formalizes a user's correlated QoS requirement with a QoS correlation function, which from the perspective of spatial geometry can be represented with a graph, e.g., a (straight or curved) line or a surface, in a Euclidean space.
2) The service selection based on a correlated QoS requirement is modeled as a Constraint Optimization Problem (COP), in which the QoS correlations are used as QoS constraints. QoS requirements with and without optimization goals are handled by different COP models. Integer Programming (IP) techniques are employed to find the solutions of the COPs.
3) We conduct comprehensive evaluation of SCORE-QoS with extensive experiments. The results show that SCORE-QoS can effectively and efficiently select services based on correlated QoS requirements.

The rest of this paper is organised as follows. The next section motivates this research with an example. Section III presents the proposed approach for service selection. Section IV presents the experimental evaluation. Section V reviews related work. Finally, Section VI concludes the paper and points out our future work.

2474-2473/17 $31.00 © 2017 IEEE
DOI 10.1109/SCC.2017.38

II. Motivating Example

This section presents an example SBS S to motivate this research. As shown in Fig. 1, this SBS consists of five tasks and provides online video streaming service to users, including Video on Demand (VoD) and video editing services. For each task in this SBS, there is a service class, which includes a set of functional equivalent candidate services with different QoS values. The SBS vendor selects a service from the corresponding service class for each task to formulate a service composition to fulfill a user's multi-dimensional QoS requirement.

There are usually trade-offs between different QoS dimensions of a user' QoS requirement. In this way, the user's QoS requirement is no longer deterministic but becomes dynamically varied and correlated. For example, the cost that the user can accept may increase along with the increase of system throughput. In such context, the SBS vendor need an approach to handle such QoS requirements in service selection in an effective and efficient manner.

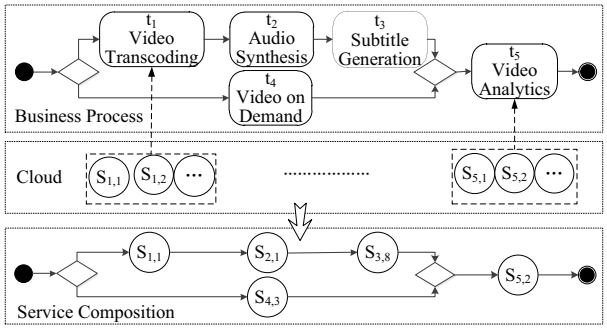

Figure 1. An example SBS for online video streaming services.

III. SCORE-QoS Approach

In this paper, we propose SCORE-QoS for service selection based on correlated QoS requirements. First, the correlated QoS requirements are formalized with QoS correlation functions. Then the service selection based on such QoS requirements is discussed in two scenarios: QoS requirements with and without optimization goals.

A. Formalization of Correlated QoS Requirements

The correlated QoS requirement for an SBS can be initially expressed with qualitative and linguistic terms, which then can be formalized empirically by the system engineers during the Service Level Agreement (SLA) negotiation process. From the perspective of the system engineer, an example of correlated QoS requirement on cost and throughput of the online video streaming system in Fig. 1 is as follows:

"It is acceptable to pay 1 dollar for each increase of 1 Mbps in throughput, up to 20 dollars in total, and the throughput cannot be less than 2 Mbps".

The correlation, i.e., trade-off, between cost and throughput, as well as the constraint on each dimension, can be identified from the above QoS requirement, which can be represented explicitly with a function, named *QoS correlation function* in this paper. For a QoS correlation with d ($d \in N^+, d \geq 2$) dimensions, the function is defined as follows:

$$q_p(u) = f(q_1(u), ..., q_{p-1}(u), q_{p+1}(u), ..., q_d(u)) \mid for\ \forall i \in [1,d],\ q_i(u) \in [q_i^{min}, q_i^{max}] \quad (1)$$

where $q_i(u)$ is the variable that represents the i^{th} dimension of QoS requirement. q_i^{min} and q_i^{max} are the minimum and maximum constraint on the i^{th} QoS dimension respectively. Then the correlated QoS requirement above-mentioned can be represented with the following function:

$$q_{ct}(u) = q_{tp}(u) \mid q_{tp}(u) \geq 2, q_{ct}(u) \leq 20 \quad (2)$$

where $q_{ct}(u)$ and $q_{tp}(u)$ are the variables that represent the QoS values of cost and throughput respectively.

Similar to many other research works [2][5][8], in this paper, we only focus on the QoS dimensions with numerical values. Thus a QoS correlation function that involves d QoS dimensions can be represented with a graph, e.g., a straight or curved line, in a d-dimensional Euclidean space. With cost and throughput, Fig. 2 shows three example QoS correlation functions including (2). Because the QoS evaluation of throughput increases along with the increase of its QoS value, a higher throughput incurs higher cost. Thus, as demonstrated in Fig. 2, the cost monotonically increases with throughput.

The QoS correlation functions can be provided and formalized empirically by the system engineers of the SBS through the formulation of SLAs. However, the formalization of QoS correlation functions are domain-specific, which should be determined on a case-by-case basis.

In order to eliminate the impact of different measurement units of QoS dimensions in QoS correlation, we normalize the QoS value of each dimension of the requirement into the

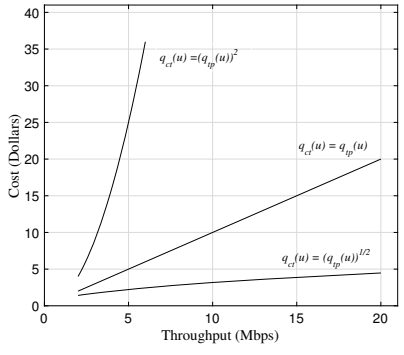

Figure 2. Example QoS correlation functions.

same range [0, 1]. For a given $q_p(U)$, we can normalize it as follows with widely used Min-Max normalization technique [5].

$$Q_p(u)=\begin{cases}\dfrac{q_p(u)-q_p^{min}(S)}{q_p^{max}(S)-q_p^{min}(S)} & \text{if } q_p^{max}(S)\neq q_p^{min}(S)\\ 1 & \text{if } q_p^{max}(S)=q_p^{min}(S)\end{cases} \quad (3)$$

where $q_p(u)$ is the variable that represents the QoS value of the p^{th} dimension of QoS requirements, $q_p^{max}(S)$ and $q_p^{min}(S)$ are the p^{th} maximum and minimum QoS values respectively that the SBS S can offer based on the available candidate services, $q_p(u) \in [q_p^{min}(S), q_p^{max}(S)]$, and $Q_p(u)$ represents the normalized p^{th} QoS value of the requirement. Then with (3) the following equation holds:

$$q_p(u)=(q_p^{max}(S)-q_p^{min}(S)) \times Q_p(u) + q_p^{min}(S) \quad (4)$$

With (1) and (3), we can obtain the normalized QoS correlation function as follows:

$$Q_p(u)=f(Q_1(u),...,Q_{p-1}(u), Q_{p+1}(u), ..., Q_d(u)) \mid for\ \forall i \in [1,\ d],\ Q_i(u) \in [Q_i^{min},\ Q_i^{max}] \quad (5)$$

where Q_i^{min} and Q_i^{max} are the normalized minimum and maximum constraints on the i^{th} QoS dimension respectively.

Taking the QoS correlation function (2) for example. Given the maximum and minimum throughput in Fig. 2 are 21 and 1 Mbps respectively, and the maximum and minimum cost of the SBS are 41 and 1 dollars respectively, we can obtain the normalized QoS correlation function according to (3) and (4) as follows:

$$Q_{ct}(u) = \frac{1}{2}Q_{tp}(u) \mid Q_{tp}(u) \geq 0.2, Q_{ct}(u) \leq 0.475 \quad (6)$$

Its function graph is shown in Fig. 3.

The maximum and minimum QoS values that an SBS S can offer can be acquired with greedy methods. For example, for each task in the SBS, we select the candidate service with the highest cost, and then we can obtain the maximum cost of the SBS by calculating its overall cost with the corresponding QoS aggregation function, which has been introduced in many works [2][3].

There may be multiple QoS correlations in a user's requirement, which can be handled in the same manner. After the QoS correlation functions are normalized, service selection can be performed based on the functions and the candidate services, which is introduced in detail in the following two subsections.

B. Service Selection with User's Optimization Goal

In the service selection of an SBS S, only the service compositions that satisfy the user's QoS requirements in all dimensions can be regarded as satisfactory solutions. As shown in Fig. 4, the grey area is enclosed by the line of the example QoS correlation function (6) that represents the user's correlated QoS requirements and the dotted lines that indicate the user's constraints on both dimensions. Only the service compositions with normalized QoS values that fall in this area are satisfactory to the user, such as s_1 to s_4.

When the users propose their QoS requirements for the SBS, their own optimization goals may be also included, such as minimized cost, maximized throughput, etc. In such context, service selection is to select proper services to create service compositions to fulfill users' QoS requirements, while achieving their optimization goals. As shown in Fig. 4, service composition s_1 is the best solution if the user's goal is to minimize the cost, while s_4 is definitely better than others when the user is after maximized throughput.

In this paper, we model this problem as a Constraint Optimization Problem (COP). Considering an SBS S with n tasks, and each of them is associated with a service class containing m candidate services, we create a set of 0-1 integer variables $x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\}$ for the i^{th} service class. $x_{i,j}$ with value of 1 means that the j^{th} candidate service in the i^{th} service class is selected for the

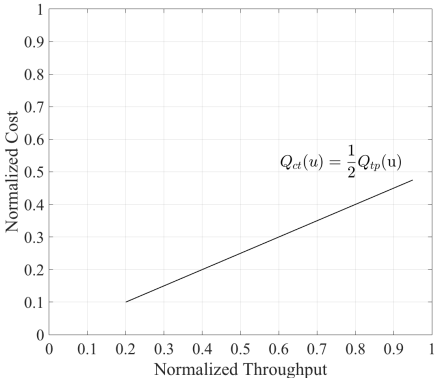

Figure 3. Example normalized QoS correlation function.

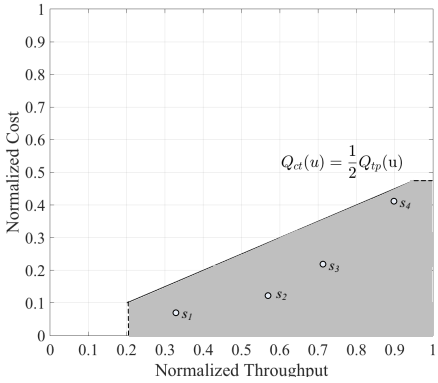

Figure 4. Service compositions with satisfactory QoS.

composition of SBS S. The COP is formulated as follows:

$$Objective: User's\ Optimization\ Goal \tag{7}$$

$$Q_i(S) \prec C_i, i \in [1, d] \tag{8}$$

$$Q_p(S) \prec f(Q_1(S), ..., Q_{p-1}(S), Q_{p+1}(S), ..., Q_d(S)) \tag{9}$$

$$\sum_{j=1}^{m} x_{i,j} = 1, i \in [1, n], j \in [1, m] \tag{10}$$

where the symbol $\prec$ means "as good or better than". Constraint family (8) ensures that user's constraint on every single QoS dimension can be fulfilled, $Q_i(S)$ is the normalized i^{th} QoS value of S, and C_i is the user's normalized i^{th} QoS constraint for S. Constraint family (9) guarantees that the QoS correlation in the user's requirement can be satisfied. Constraint family (10) ensures that only one service in each service class can be selected to compose S.

After the COP model is created, the Integer Programming (IP) technique [5] is employed to find the optimal solution.

C. Service Selection without User's Optimization Goal

It is not always the case that users can propose their own optimization goals when subscribing an SBS. The system engineers can find the optimal solutions of service selection according to certain criteria. In this paper, we use *QoS Satisfaction Degree* to represent the optimality of a satisfactory solution. Given a service composition s, f is the point with shortest Euclidean distance to s in terms of QoS values on the graph that represents the QoS correlation function. The QoS satisfaction degree (denoted by QSD) of s is defined as follows:

$$QSD(s) = w_1 \times \|\Delta Q_1\| + ... + w_d \times \|\Delta Q_d\| \tag{11}$$

where

$$\Delta Q_i = Q_i(s) - Q_i(f), i \in [1, d] \tag{12}$$

and

$$w_i = \frac{\frac{\partial Q_p(u)}{\partial Q_i(u)}}{\frac{\partial Q_p(u)}{\partial Q_1(u)} + ... + \frac{\partial Q_p(u)}{\partial Q_d(u)}} \mid u = f, \sum_{i=1}^{d} w_i = 1 \tag{13}$$

ΔQ_i is the difference in the i^{th} normalized QoS value between f and s. w_i represents the user's preference on the i^{th} QoS dimension, which is calculated based on the partial derivatives of the QoS correlation function with respect to each $Q_i(u)$ at f. All the points on the graph of QoS correlation function have the same QoS satisfaction degree with value of 0. Fig. 5 shows an example with cost and throughput, where f is the perpendicular foot of s to the line segment that represents the QoS correlation function.

Then we can model the problem of service selection as a COP even though the user's optimization goal is not available. We use maximizing the QoS satisfaction degree as the objective of the COP. We also create a set of 0-1 integer

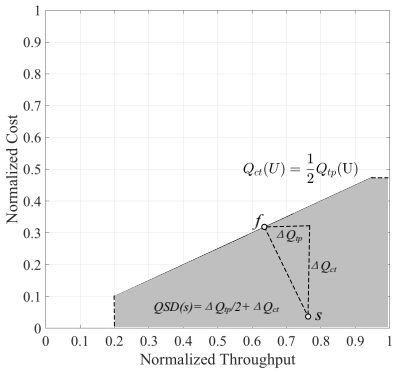

Figure 5. An example of QoS satisfaction degree calculation.

variables $x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\}$ for the i^{th} service class. The COP is formulated as follows:

$$Objective: Maximizing\ QSD(S) \tag{14}$$

$$Q_i(S) \prec C_i, i \in [1, d] \tag{15}$$

$$Q_p(S) \prec f(Q_1(S), ..., Q_{p-1}(S), Q_{p+1}(S), ..., Q_d(S)) \tag{16}$$

$$\sum_{j=1}^{m} x_{i,j} = 1, i \in [1, n], j \in [1, m] \tag{17}$$

where the constraint families (15), (16) and (17) are the same as (8), (9) and (10) respectively. Again, we employ IP technique to find the optimal solution of the COP. The method for finding a point f with shortest Euclidean distance to a given service composition in terms of QoS is domain-specific, which depends on the QoS correlation function. Take the QoS correlation functions in Fig. 5 for example, f can be found in many ways, such as IP techniques, Lagrange multiplier methods, etc.

IV. Evaluation

We have conducted a range of experiments in a simulated environment to evaluate SCORE-QoS in terms of effectiveness and efficiency by comparing it with other representative approaches. The effectiveness is measured by two quantitative metrics: success rate and system optimality.

1) Success rate of service selection is defined by the percentage of test instances where a satisfactory solution for service selection is found.
2) System optimality is evaluated based on the objectives of the COPs, which are defined in (7) and (14). The objective values of the solutions found by different approaches are investigated and compared.

The efficiency of all approaches is evaluated based on the computational overhead, which is the average computation time needed for service selection for an SBS.

Section IV-A introduces the setup of the experiments. Sections IV-B and IV-C evaluate the effectiveness and efficiency of SCORE-QoS in two scenarios: service selection with and without users' optimization goals.

A. Experimental Setup

SCORE-QoS is implemented in Java with JDK 1.6.0. IBM CPLEX v12.6, a widely used linear programming tool, is used to solve the COPs.

In the first scenario of service selection, we use minimized cost as an example of users' optimization goals. Similar to [3][11], we have implemented three representative approaches for comparison in the context of this research. These approaches, namely Cost-Greedy, Utility-Greedy and Random, are described as follows:

1) **Cost-Greedy**: A greedy approach that creates a service composition by selecting the candidate service with lowest cost in each service class.
2) **Utility-Greedy**: Also a greedy approach that selects the candidate service with highest utility in each service class for service composition. The utility of a service is calculated with the same method employed in [2] based on average user preference for each QoS dimension.
3) **Random**: A naive and non-optimal approach, which selects a candidate service randomly from each service class for service composition.

In the second scenario where users' optimization goals are not available, two representative approaches, similar to [12][11], are implemented for comparison as follows:

1) **Utility-Optimal**: A COP is created for service selection where the objective is to maximize the system utility. Average user preference for each QoS dimension is adopted.
2) **Random**: Same as the Random approach in the first scenario.

For the demonstration purpose, in this paper we use throughput and cost as two representative QoS dimensions. The correlations between them in the QoS requirements are linear. Other QoS dimensions and correlations can be handled in the similar manner. The services used in the experiments are generated based on QWS [13], which is a public dataset that consists of more than 2500 real-world Web services with multi-dimensional QoS. For each service, the cost is generated randomly by considering the trade-off between cost and throughput of the service.

The QoS correlation and QoS constraints determine the difficulty level of fulfilling a user's QoS requirements. The works in [2][3] have demonstrated that the difficulty levels of users' QoS requirements have significant impacts on the performance of service selection, especially the success rates. Thus, in the experiments, we generate users' QoS constraints with different difficulty levels: easy, medium and severe, which are combined with different QoS correlation functions to simulate the QoS requirements in different difficulty levels. Fig. 6 shows an example based on cost and throughput, where the QoS requirement represented by F_3 is more difficult to satisfy than that represented by F_1 due to a much small solution space.

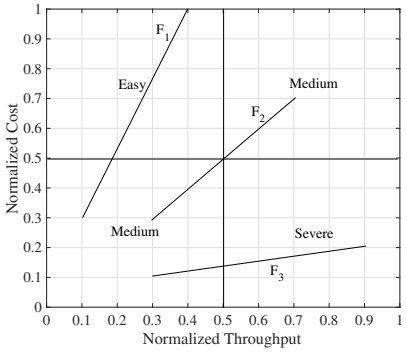

Figure 6. Difficulty levels of users' QoS requirements.

In order to compare SCORE-QoS with other approaches more comprehensively, we have mimicked SBSs with different numbers of tasks n (up to 50) and different numbers of candidate services in each service class m (up to 100). The findings are consistent with those from the experiments on the motivating example introduce in Section II.

All the experiments were conducted on a machine running Windows 7x64 Enterprise with Intel(R) Core (TM) i5-4570 3.2 GHz CPU and 8 GB RAM. The experimental results are collected, averaged and compared from 500 test instances.

B. Service Selection with User's Optimization Goal

In this section, we focus on the scenario where users have their optimization goals. We use minimized cost as an example objective of service selection in experiments. The effectiveness and efficiency of SCORE-QoS is evaluated by comparing it with other three approaches using three metrics as follows:

1) Comparison in Success Rates: First, we compare the success rates of the four approaches in finding the satisfactory solutions of service selection with QoS requirements of different difficulty levels. Similar results are observed with different number of tasks n and number of candidate services per service class m. As a representative case, we fix n at 5, m at 10, and vary the difficulty level of users' QoS requirements. The success rates of service selection obtained by the four approaches are shown in Fig. 7. SCORE-QoS outperforms other approaches across all test cases with an average success rate of 95.4%. Utility-Greedy and Cost-Greedy show average success rates of 86% and 30.7% respectively. Random obtains the lowest average success rate as expected, which is 22.9%. When the difficulty level

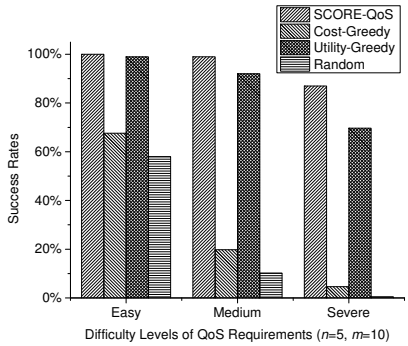

Figure 7. Success rates at different difficulty levels of users' QoS requirements.

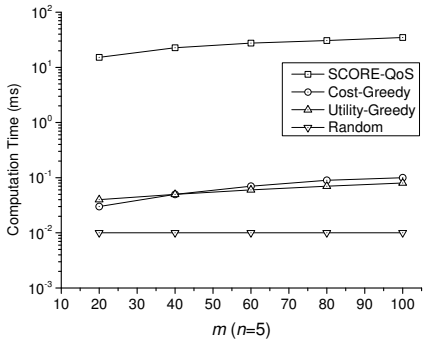

Figure 8. Computation time vs the number of candidate services per service class.

change from easy to severe, the success rates achieved by the four approaches all decrease. However, Cost-Greedy and Random show more dramatical decreases in success rates, which are from 67.6% to 4.6% and from 58% to 0.4% respectively, and the success rates obtained by SCORE-QoS and Utility-Greedy decrease by 13% and 29.3% to 87% and 69.7% respectively. The results demonstrate that in this set of experiments, SCORE-QoS can maintain higher success rates than other approaches even the users' QoS requirements are difficult to satisfy.

2) Comparison in Optimality: Then we evaluate the optimality achieved by the four approaches in minimizing the cost of the SBS. With the same experimental setup as the comparison in success rates, we collect and average the cost of the SBS obtained by the four approaches from the test cases that all approaches succeed in finding a solution of service selection. Since the QoS values have been normalized into the range of [0, 1], we evaluate the system optimality of an approach by comparing the cost it obtains with the optimal cost, which is the cost obtained by SCORE-QoS when an optimal solution to the COP can be found. The formula used and the experimental results are shown in Table I. SCORE-QoS can obtain the optimal cost across all scenarios, and Cost-Greedy shows the same performances when a solution to the COP exists. Utility-Greedy achieves a system optimality of approximate 0.98 on average despite different optimization goals adopted. Random, as expected, obtains the lowest system optimality among all approaches, which is 0.69 on average. The increase in system optimality obtained by Random is because only the solutions with higher optimalities can be found successfully when users' QoS requirements are more difficult to satisfy.

Table I
COMPARISON IN SYSTEM OPTIMALITY $(1\text{-}q_{ct})/(1\text{-}q_{ct}^{optimal})$

Difficulty Level	SCORE-QoS	Cost-Greedy	Utility-Greedy	Random
Easy	1.0	1.0	0.97	0.54
Medium	1.0	1.0	0.98	0.72
Severe	1.0	1.0	0.98	0.79

3) Comparison in Computational Overhead: In order to evaluate the efficiency of SCORE-QoS, we conduct experiments to compare the computational overhead introduced by all approaches, which is represented by the consumed computation time for service selection. We found that the number of tasks n in an SBS and the number of candidate services in each service class m can affect the efficiency of service selection dramatically in a similar manner by changing the scale of the COPs. As an example, we set the difficulty level of QoS requirements at medium and n at 5, vary m from 20 to 100 in steps of 20. The computation time consumed by all approaches are shown in Fig. 8. Because there is no any optimization involved, Random show the least computation time across all test instances, which remains at approximate 0.01ms. Cost-Greedy and Utility-Greedy obtain similar performance, and consume a bit more computation time (up to 0.1ms) than Random because they need to rank the candidate services. SCORE-QoS shows a slightly increase in computation time from 15.2ms to 34.7ms when m increases to 100, which is more than those of other approaches. From this set of experiments, we can see that SCORE-QoS can obtain the optimal solutions to the COPs at the cost of more computation time, which, however, is not a significant overhead in overall terms.

C. Service Selection without User's Optimization Goal

In this section, we evaluate SCORE-QoS in the scenario that users' optimization goals are not available. As discussed in Section III-C, SCORE-QoS is to maximize the QoS satisfaction degree of an SBS. SCORE-QoS is compared with other two representative approaches (Utility-Optimal and Random) using the following metrics:

1) Comparison in Success Rates: With different n and m, similar comparison results in success rates are observed by changing difficulty levels of QoS requirements. As an example, we fix n at 5, m at 10, vary the difficulty level of

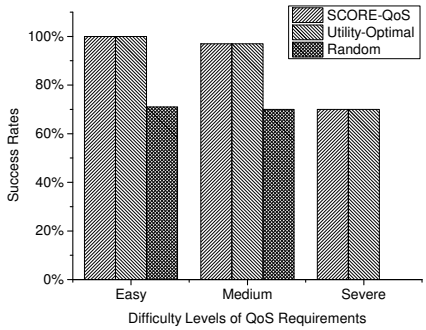

Figure 9. Success rates at different difficulty levels of users' QoS requirements.

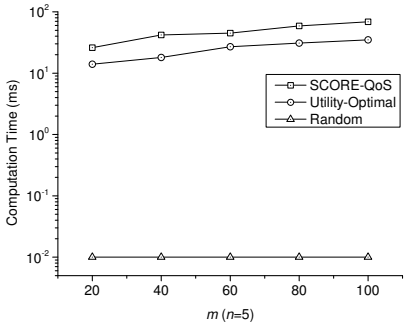

Figure 10. Computation time vs the number of candidate services per service class.

QoS requirements from easy to severe, and investigate the success rates obtained by the three approaches in finding the solutions of service selection. As shown in Fig. 9, SCORE-QoS and Utility-Optimal achieve the same success rates and outperform Random across all scenarios. When difficulty level changes from easy to severe, the success rates of SCORE-QoS and Utility-Optimal decrease from 100% to 70%, while Random shows a dramatical decrease from 71% to 0%.

2) Comparison in Optimality: Then we compare the three approaches in the system optimality with different difficulty levels of QoS requirements. In this set of experiments, system optimality is measured by comparing SCORE-QoS with Utility-Optimal and Random in terms of the QoS satisfaction degree of the SBS. The comparison results are shown in Table II. SCORE-QoS shows the highest system optimality across all scenarios as expected. Utility-Optimal outperforms Random significantly with an average system optimality of 0.89. Random obtains the lowest system optimality, which is even not available due to the success rate of 0% when the difficulty level is severe.

Table II
COMPARISON IN SYSTEM OPTIMALITY ($QSD(S))/d^{optimal}$)

Difficult Level	SCORE-QoS	Utility-Optimal	Random
Easy	1.0	0.80	0.28
Medium	1.0	0.90	0.58
Severe	1.0	0.93	N/A

3) Comparison in Computational Overhead: We conduct a range of experiments to evaluate the efficiency of SCORE-QoS in this scenario. We change the number of tasks n, the number of candidate services per service class m, different QoS correlation functions, etc., and compare the computation times consumed by all approaches. The results are consistent, which are demonstrated by an example in Fig. 10. We fix n at 5, difficulty level of QoS requirements at medium, and vary m from 20 to 100 in steps of 20. Random shows the highest efficiency as expected with a average computation time of 0.01ms. SCORE-QoS, outperformed by Utility-Optimal slightly by 23ms on average, shows an increase in computation time from 26ms to 69ms when m increases to 100. The results show again that compared with other approaches, SCORE-QoS can find the optimal solution to the COP with acceptable computational overhead in most, if not all, scenarios.

V. RELATED WORK

With the proliferation of Service-Oriented Architecture (SOA), Service-based Systems (SBSs) have attracted unprecedented attention over the past decade. Many approaches for QoS-aware service selection have been proposed. Just name a few, the authors in [5] propose a middle-ware platform named Age-Flow for QoS-aware Web service composition, which is modeled as an Constraint Optimization Problem and Integer Programming (IP) techniques are used to find the solutions. In [6] the authors model the problem of QoS-aware service selection as a 0-1 knapsack problem (MMKP) in the combinatorial model and a multi-constrained optimal path (MCOP) problem in the graph model respectively. The near optimal solutions can be found by proposed heuristic algorithms. In [14] the authors propose an approach for service selection for services with probabilistic QoS. However, most of the existing works assume that the users' QoS requirements are deterministic, which cannot handle correlated QoS requirements.

Correlations between QoS of different services have received considerable attention in recent years. The authors in [2] propose CASS, a model that selects service based on iterative multi-attribute combinatorial auction. The complementarities between services are taken into account. In [9], the authors propose an alliance-aware service composition approach, which considers the Alliance Relation (AR) between services when formulating service composition. In [8], the authors propose CASP, a correlation-aware service pruning method for service selection. The work in [15] proposes an approach for computing the composite service skyline in the presence of QoS correlations. Pruning techniques

are used to accelerate the computation. However, none of the above works and the like take into consideration the correlation between different QoS attributes of a user's QoS requirements.

Fuzzy logic can be used to handle users' imprecise QoS requirements. For example, in [16], the authors model the service selection as a fuzzy constraint satisfaction problem. The constraint levels on each QoS attribute are represented with multiple fuzzy sets. The authors in [17] present a fuzzy model for ranking real-world Web services. A ranking algorithm is proposed, which is based on the objective weighting technique that leverages the distance correlation metrics between QoS attributes. The authors in [18] propose an approach for service selection based on fuzzy logic that considers users' personalized trade-off strategies. However, the service selection and ranking approaches based on fuzzy logic mostly assume that the QoS requirements can be represented with a set of fuzzy expressions based on users' QoS trade-off preferences, which is insufficient to handle the dynamically varied and correlated QoS requirements.

In order to address above issues, we propose SCORE-QoS in this paper. It takes into consideration the correlations between different QoS dimensions of requirements, and can select appropriate services to create a service composition to fulfill user's correlated QoS requirements.

VI. Conclusion

In this paper, we propose a novel approach named SCORE-QoS for service selection based on correlated QoS requirements. The correlations between different QoS dimensions of requirements are formalized with QoS correlation functions and integrated into the COP that models the service selection problem. Two types of QoS requirements are considered: with or without users' optimization goals. The experimental results show that, compared with other representative approaches, SCORE-QoS can find the optimal solutions for service selection with high success rates, high system optimality, and acceptable computational overhead.

As our future work, we will investigate more types of QoS correlations. More comparison with other approaches such as those based on fuzzy logic will also be carried out.

Acknowledgment

This work is partly supported by Australian Research Council Discovery Project DP150101775. Yun Yang is the corresponding author of this paper.

References

[1] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli, "Dynamic QoS Management and Optimization in Service-Based Systems," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 387–409, 2011.

[2] Q. He, J. Yan, H. Jin, and Y. Yang, "Quality-aware Service Selection for Service-based Systems based on Iterative Multi-attribute Combinatorial Auction," *IEEE Transactions on Software Engineering*, vol. 40, no. 2, pp. 192–215, 2014.

[3] Y. Wang, Q. He, and Y. Yang, "QoS-Aware Service Recommendation for Multi-Tenant SaaS on the Cloud," in *Proceedings of the 12th IEEE International Conference on Service Computing (SCC2015)*, 2015, pp. 178–185.

[4] P. A. Bonatti and P. Festa, "On Optimal Service Selection," in *Proceedings of the 14th International Conference on World Wide Web (WWW2005)*, 2005, pp. 530–538.

[5] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware Middleware for Web Services Composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311–327, 2004.

[6] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient Algorithms for Web Services Selection with End-to-end QoS Constraints," *ACM Transactions on the Web*, vol. 1, no. 1, pp. 1–26, 2007.

[7] M. Alrifai and T. Risse, "Combining Global Optimization with Local Selection for Efficient QoS-aware Service Composition," in *Proceedings of the 18th International Conference on World Wide Web (WWW2009)*, 2009, pp. 881–890.

[8] S. Deng, H. Wu, D. Hu, and J. L. Zhao, "Service Selection for Composition with QoS Correlations," *IEEE Transactions on Services Computing*, vol. 9, no. 2, pp. 291–303, 2016.

[9] Y. Zhang, G. Cui, S. Deng, and Q. He, "Alliance-Aware Service Composition Based on Quotient Space," in *Proceedings of the 23rd IEEE International Conference on Web Services (ICWS2016)*, 2016, pp. 340–347.

[10] M. C. Jaeger, G. Muhl, and S. Golze, "QoS-aware Composition of Web Services: A Look at Selection Algorithms," in *Proceedings of the 12th IEEE International Conference on Web Services (ICWS2005)*, 2005, pp. 807–808.

[11] Y. Wang, Q. He, D. Ye, and Y. Yang, "Formulating Criticality-Based Cost-Effective Fault Tolerance Strategies for Multi-Tenant Service-Based Systems," *IEEE Transactions on Software Engineering*, 2017, doi:10.1109/TSE.2017.2681667.

[12] Q. He, J. Han, Y. Yang, J. Grundy, and H. Jin, "QoS-driven Service Selection for Multi-tenant SaaS," in *Proceedings of the 5th IEEE International Conference on Cloud computing (CLOUD2012)*, 2012, pp. 566–573.

[13] E. Al-Masri and Q. H. Mahmoud, "Discovering the Best Web Service," in *Proceedings of the 16th International Conference on World Wide Web (WWW2007)*, 2007, pp. 1257–1258.

[14] S.-Y. Hwang, C.-C. Hsu, and C.-H. Lee, "Service Selection for Web Services with Probabilistic QoS," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 467–480, 2015.

[15] Y. Du, H. Hu, W. Song, J. Ding, and J. Lü, "Efficient Computing Composite Service Skyline with QoS Correlations," in *Proceedings of the 12th IEEE International Conference on Services Computing (SCC2015)*, 2015, pp. 41–48.

[16] M. Lin, J. Xie, H. Guo, and H. Wang, "Solving QoS-Driven Web Service Dynamic Composition as Fuzzy Constraint Satisfaction," in *Proceedings of IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE2005)*, 2005, pp. 9–14.

[17] M. Almulla, K. Almatori, and H. Yahyaoui, "A QoS-based Fuzzy Model for Ranking Real World Web Services," in *Proceedings of the 9th IEEE International Conference on Web Services (ICWS2011)*, 2011, pp. 203–210.

[18] K. K. Fletcher, X. F. Liu, and M. Tang, "Elastic Personalized Nonfunctional Attribute Preference and Trade-off based Service Selection," *ACM Transactions on the Web (TWEB)*, vol. 9, no. 1, pp. 1–26, 2015.