Efficient Query of Quality Correlation for Service Composition

Yiwen Zhang, Guangming Cui, Shuiguang Deng, *Senior Member, IEEE*, Feifei Chen, Yan Wang, *Senior Member*, *IEEE*, Qiang He, *Member, IEEE*

Abstract— As enterprises around the globe embrace globalization, strategic alliances among enterprises have become an important means to gain competitive advantages. Enterprises cooperate to improve the quality or lower the prices of their services, which introduce quality correlations, i.e., the quality of a service is associated with other services. Existing approaches for service composition have not fully and systematically considered the quality correlations between services. In this paper, we propose a novel approach named Q^{2}C (**Q**uery of **Q**uality **C**orrelation) to systematically model quality correlations and enable efficient queries of quality correlations for service compositions. Given a service composition and a set of candidate services, Q^{2}C first preprocesses the quality correlations among the candidate services and then constructs a quality correlation index graph to enable efficient queries for quality correlations. Extensive experiments are conducted on a real-world web service dataset to demonstrate the effectiveness and efficiency of Q^{2}C.

Index Terms—Quality Correlation, Service Composition, Quality of Service, Aggregation Algorithm, Index Graph, Heuristic Integer Programming.

1 Introduction

THE service-oriented architecture (SOA) has become a major framework for engineering software systems that are composed of services locally or remotely accessed by an execution engine (e.g., a BPEL engine [6]). Through service composition, a system engineer can compose existing services in the form of business process to build a new service-based systems (SBS) that fulfills specific quality constraints, e.g., response time, throughput and availability, and achieves an optimization goal, e.g., least system cost or highest system utility [4]. The development and popularity of e-business, e-commerce, especially the pay-as-you-go business model promoted by cloud computing have fueled the growth of web services [13]. The statistics published by ProgrammableWeb[1], an online web service directory indicates a rapid growth in the number of published web services in the past few years. The popularity of web services and SOA enables the engineering of various SBSs that fulfill different organizations' increasingly sophisticated business needs [7].

Embracing globalization, more and more enterprises have established strategic alliances and cooperated to improve their competitiveness in the market. In recent years, the number of enterprise alliances has continued to increase at an unprecedented rate - around 9,000 strategic alliances worldwide per year [23]. Not only does the number of strategic alliances continue to grow, but also their significance to the allied enterprises. A study conducted by Accenture reports that about 25% of enterprises' executives confirmed that strategic alliances account for at least 15% of their market value [14]. The CEOs of 82% of the Fortune 1000 companies believe alliances will be responsible for more than 26% of their companies' revenues [14]. In the open and competitive service-oriented cloud environment, service providers in a strategic alliance cooperate to improve the quality or lower the prices of their services, which introduces *quality correlations* (referred to as service complementarity [19]), i.e., the phenomenon that the quality of a service is correlated with other services [12, 19].

Take the online book shopping SBS shown in Fig. 1 that consists of four tasks, *Book Search*, *Logistics*, *Insurance* and *Payment*, as an example. In order to build this SBS, the system engineer needs to select one book search service from a set of candidate services CS_1 = {*Tmall, JD, Amazon, Taobao, Dangdang*}, one logistics service from CS_2 = {*DHL, ePacket EMS, STO*}, one insurance service from CS_3 = {*Allianz, AXA, Aviva, Cigna*} and one payment service from CS_4 = {*CreditCard, Alipay, JD Finance, Paypal, WeChat*}. Very often, discounts are offered to bundled services from CS_1 and CS_3. For example, a 5% discount is offered if the system engineer selects the both the *JD* book search service and the *JD Finance* payment service. Such quality correlations can also be found between services from CS_1 and CS_4

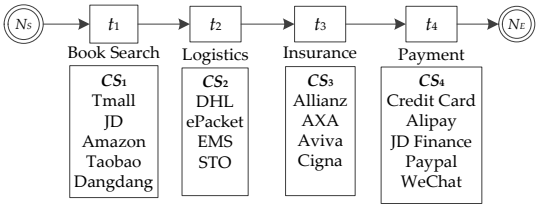


Fig. 1. Online book shopping SBS.

- *Y. Zhang and G. Cui are with Key Laboratory of Intelligent Computing & Signal Processing; Ministry of Education, Anhui University, Hefei, Anhui 230031, China. E-mail: zhangyiwen@ahu.edu.cn; cgm133123@163.com.*
- *S. Deng is with College of Computer Science and Technology, Zhejiang University, Hangzhou, China. Email: dengsg@zju.edu.cn.*
- *Feifei Chen is with the School of Information Technology, Deakin University, Melbourne, Australia. E-mail: feifei.chen@deakin.edu.au.*
- *Y. Wang is with the Department of Computing, Macquarie University, Sydney, Australia. Email: yan.wang@mq.edu.au.*
- *Q. He is with School of Software and Electrical Engineering, Swiburne University of Technology, Melbourne, Australia. Email: qhe@swin.edu.au.*

[1] http://www.programmableweb.com

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2830773, IEEE Transactions on Services Computing

in a different quality dimension other than price. For example, a logistics company may offer a shorter delivery time for books purchased through its allied online book search service.

Existing methods for quality-aware service composition have not fully and systematically considered the quality correlations among services. The lack of consideration in quality correlations will impact the quality of the target SBSs. For example, the SBS built based on the process presented in Fig. 1 will not be able to offer the lowest total price or shortest delivery time without taking into account the quality correlations discussed above. An approach has been proposed in [12] to handle quality correlations among services based on the skyline techniques. However, that approach suffers from poor efficiency and cannot handle large-scale scenarios.

In order to address the above issue, this paper presents Q²C, a novel approach for efficient **q**ueries of **q**uality **c**orrelations. Q²C complements existing service composition approaches by preprocessing quality correlations. Before a service composition starts, Q²C preprocesses the quality correlations among the candidate services and constructs a quality correlation index graph based on a systematic quality correlation model. Service composition approaches [3, 22, 24, 31] can then query for applicable quality correlations while attempting to find a solution for building the target SBS. Q²C simplifies the application of SOA in open and complex service-oriented environment with quality correlations. The major contributions of this paper are as follows:

- A systematic quality correlation model is proposed that describes three different types of quality correlations between services, including adjacent, non-adjacent and hybrid quality correlations.
- A method is proposed to build an index graph to enable efficient queries of quality correlations without losing the correctness in the answers.
- Extensive experiments were conducted on QWS, a published dataset that contains the functional and quality information about over 2,500 real-world web services, to evaluate the effectiveness and efficiency of the proposed approach.

The rest of the paper is organized as follows. Section 2 discusses the composition quality model used in this research. Section 3 describes the quality correlation model. Section 4 presents the method for building the index graph for quality correlations. Section 5 evaluates the effectiveness and efficiency of Q²C. Section 6 reviews the related work. Finally, Section 7 concludes this paper and discusses future directions.

2 Composition Quality Model

Quality evaluation is the basis for quality-aware service composition. In this section, we first present the compositional structures adopted in this research for representing the business processes and service compositions of SBSs. Based on the adopted compositional structures, we introduce the utility and quality evaluation methods for SBSs.

2.1 Composition Structures

Composition structures describe the order in which the tasks (and services) are performed in the business process (and the service composition) of SBSs. Similarly to other work [4, 15, 17, 18, 29], in this research, we consider four types of basic compositional structures, i.e., sequence, conditional branch, loop and parallel:

- **Sequence.** In a sequence structure, the services are executed one by one.
- **Conditional Branch.** In a condition branch structure, only one branch is selected for execution. For every set of branches $\{cb_1, \ldots, cb_n\}$, the execution probability distribution $\{prob(cb_1), \ldots, prob\ (cb_n)\}$, $(0 \le prob(cb_i) \le 1, \sum_{i=1}^{n} prob(cb_i) = 1)$ is specified, where $prob(cb_i)$ is the probability that branch i is selected for execution.
- **Loop.** In a loop structure, the loop is executed for n ($n \ge 0$) times. For every loop, the probability distribution $\{prob_0, \ldots, prob_{MNI}\}$, $(0 \le prob(cb_i) \le 1, \sum_{i=1}^{n} prob(cb_i) = 1$ is specified, where $prob_i$ is the probability that the loop iterates for i times and MNI is the expected maximum number of iterations for the loop.
- **Parallel.** In a parallel structure, all the branches are executed at the same time.

The probabilities, $prob(cb_i)$, $prob_i$, and the maximum number of iterations can be evaluated based on the SBS's past executions or can be empirically specified by the system engineer [4, 15, 17-19, 29]. Similar, we require that for every loop, the MNI must be determined. Otherwise, if an upper bound for the number of iterations for a loop does not exist, the quality of the SBS cannot be calculated because the loop can iterate infinitely. In addition, if $prob(cb_i)$ and $prob_i$ are unknown, an average value will be assigned to each of the branches in conditional branch and loop structures. For example, for a conditional branch that consists of four branches, cb_1, cb_2, cb_3 and cb_4, there is $prob(cb_1) = prob(cb_2) = prob(cb_3) = prob(cb_4) = 0.25$.

In this research, we represent the business processes and service compositions of SBSs using directed acyclic graphs (DAGs), where nodes represent tasks (and services) and edges represent the control flow. We assume that the business process of an SBS is characterized by only one entry point and one exit point (e.g., N_S and N_E in Fig. 1), and it only includes *structured loops* with only one entry point and one exit point. If an SBS includes loops,

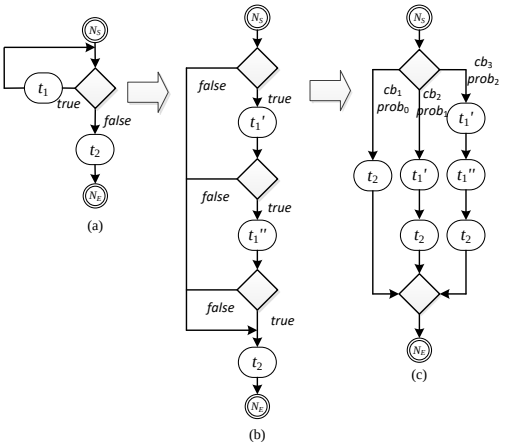


Fig. 2. The loop peeling process.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2830773, IEEE Transactions on Services Computing

TABLE 1
QUALITY AGGREGATION FUNCTIONS

Quality Dimension	Sequence	Condition branch	Parallel	Loop
Availability (*ava*)	$\prod_{i=1}^{n} ava_i$	$\prod_{i=1}^{n} ava_i \times prob(cb_i)$	$\prod_{i=1}^{n} ava_i$	ava^{MNI}
Response Time (*rt*)	$\sum_{i=1}^{n} rt_i$	$\sum_{i=1}^{n} rt_i \times prob(cb_i)$	$max(rt_i), i \in [1,n]$	$rt \times MNI$
Cost (*cost*)	$\sum_{i=1}^{n} cost_i$	$\sum_{i=1}^{n} cost_i \times prob(cb_i)$	$\sum_{i=1}^{n} cost_i$	$cost \times MNI$
Reputation (*rep*)	$\prod_{i=1}^{n} rep_i$	$\prod_{i=1}^{n} rep_i \times prob(cb_i)$	$\prod_{i=1}^{n} rep_i$	rep^{MNI}

we peel the loops by representing loop iterations as a set of branches with certain execution probabilities as in [4]. Fig. 2 gives an example of peeling a loop structure (*MNI* = 2) by transforming it into a conditional branch structure that contains three conditional branches cb_1, cb_2 and cb_3, where p_0, p_1 and p_2 are the probabilities that cb_1, cb_2 and cb_3 are selected for execution respectively.

2.2 Quality Aggregation

The quality of the selected services for building an SBS must be aggregated to evaluate the overall system quality. As examples, Table 1 presents the aggregation functions for four common quality dimensions, i.e., availability, response time, cost and reputation, based on the basic compositional structures introduced in Section 2.1 [15]. In this paper, examples and relevant discussions are based on price (or cost), which has also been used by many researchers for quality discussion and evaluation [4, 26]. Q²C can also handle other quality dimensions with corresponding aggregation functions. For example, given two services, s_1 with a 0.99 availability and s_2 with a 0.95 availability, the availability of the SBS that is composed using s_1 and s_2 is 0.99 × 0.95 = 0.9405. This allows Q²C to accommodate new quality dimensions easily and flexibly.

3 QUALITY CORRELATION MODEL

Similar to system quality, quality correlations can be calculated among services with proper aggregation functions based on the corresponding composition structures. According to the structural relevance between tasks, quality correlations can be grouped into three categories: adjacent quality correlations, nonadjacent quality correlations and hybrid quality correlations. Using the SBS example illustrated in Fig. 1, this section introduces our quality correlation model which extends the one introduced in [12]. To generalize the discussion, Fig. 3 presents the DAG representation of the business process of the SBS shown in Fig. 1. In Fig. 3, the ellipses denote tasks, the circles denote candidate services and the rectangles denote classes of candidate services. Similar to other research efforts [2-4, 16], we assume that alternative functionally-equivalent candidate services are available and can be categorized into classes of candidate services based on their functionalities.

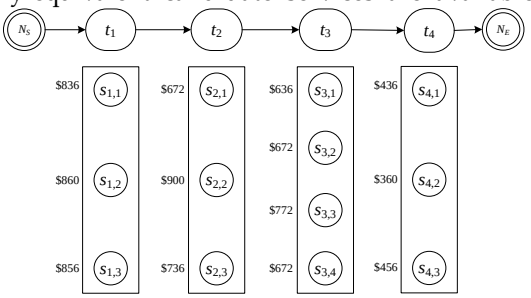


Fig. 3. DAG representation of online book shopping SBS.

3.1 Quality Correlation Notations

A *p*-dimensional quality correlation among a bundle of services $S_i = \{s_1, s_2, \ldots, s_n\}$ is denoted by $qc_{cost,\ rt,\ ava,\ \ldots}(S_i) = qc_{cost,\ rt,\ ava,\ \ldots}(s_1, s_2, \ldots, s_n) = (qc_1, qc_2, \ldots, qc_p)$, where qc_p is the *quality premium* in the p^{th} dimensional quality. For example, $qc_{cost,\ rt}(s_i, s_j)$ = (-$100, -500*ms*) specifies that a discount of $100 and a decrease of 500ms are applicable to the total cost and the total response time of the bundle of s_i and s_j. As discussed in Section 2.2, we use cost (or price) in relevant examples and discussions in this paper. Thus, we use $qc(s_i, s_j)$ = -$100 and the like as the simplified representation of quality correlations.

3.2 Adjacent Quality Correlation

An adjacent quality correlation is a quality correlation between adjacent candidate services, i.e., candidate services performing two neighboring tasks in the business process of the SBS, where one of them directly precedes or succeeds the other. Think of a double-layer encryption SBS composed of two services that use different encryption schemes. When the two services are provided by a single provider rather than two individual providers, a shorter response time is expected from the SBS because both encryption operations are performed by the service provider in-house without having to transmit the intermediate data across organisational boundaries.

Fig. 4 presents several adjacent quality correlations as an example. In Fig. 4, an arc denotes the quality correlation between the services in a bundle and the value on the arc denotes the corresponding discount for the total price of the service bundle. In Fig. 4, we can identify three quality correlations: $qc(s_{1,1}, s_{2,2})$ = -$314, $qc(s_{2,3}, s_{3,4})$ = -$112 and $qc(s_{3,2}, s_{4,3})$ = -$141.

Based on Fig. 4, the optimal solution for the SBS in terms of total price is analyzed as follows, with and without consideration of the quality correlations (i.e., price correlations in this case). Please note that the other quality dimensions are omitted for simplicity. With consideration of the price correlations, the optimal solution is $\{s_{1,1}, s_{2,2}, s_{3,2}, s_{4,3}\}$ with a total price of $2409 = $836 + $900 - $314 +

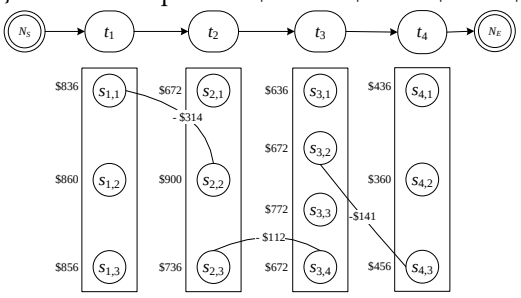


Fig. 4. Adjacent quality correlations.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2830773, IEEE Transactions on Services Computing

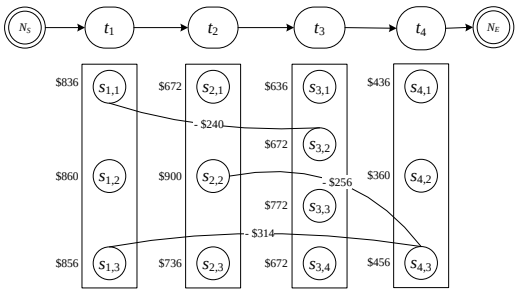


Fig. 5. Nonadjacent Quality Correlations

$672 + $456 - $141. Without consideration of the price correlations, the optimal solution is {$s_{1,1}$, $s_{2,1}$, $s_{3,1}$, $s_{4,2}$} with a total price of $2504 = $836 + $672 + $636 + $360. Apparently, the first solution is better than the second one by $95 = $2504 - $2409.

3.3 Nonadjacent Quality Correlation

Quality correlations can also be specified among nonadjacent services, e.g., price and availability. A nonadjacent quality correlation is a quality correlation between two nonadjacent candidate services, i.e., services perform nonadjacent tasks in the business process of the SBS. Fig. 5 presents three nonadjacent quality correlations as an example: $qc(s_{1,1}, s_{3,2})$ = -$240, $qc(s_{1,3}, s_{4,3})$ = -$314, $qc(s_{2,2}, s_{4,3})$ = -$256.

With consideration of the price correlations, the optimal solution is {$s_{1,3}$, $s_{2,2}$, $s_{3,1}$, $s_{4,3}$} with a total price of $2,278 = $856 + $900 + $636 + $456 - $256 - $314. Without consideration of the price correlations, the optimal solution is {$s_{1,1}$, $s_{2,1}$, $s_{3,1}$, $s_{4,2}$} with a total price of $2,504 = $836 + $672 + $636 + $360. The advantage of the first solution over the second one is $226 = $2,504 - $2,278.

3.4 Hybrid Quality Correlation

A hybrid quality correlation is a quality correlation among services that include both adjacent candidate services and nonadjacent candidate services. This allows more than two services to be included in a quality correlation. Fig. 6 presents three hybrid quality correlations as an example: $qc(s_{1,2}, s_{2,2}, s_{4,3})$ = -$124, $qc(s_{1,3}, s_{2,3}, s_{3,4}, s_{4,3})$ = -$241, $qc(s_{2,1}, s_{3,1}, s_{4,1})$ = -$324.

With consideration of the price correlations, the optimal solution is {$s_{1,1}$, $s_{2,1}$, $s_{3,1}$, $s_{4,1}$} with a total price of $2,256 = $836 + $672 + $636 + $436 - $324. Without consideration of the price correlations, the optimal solution is {$s_{1,1}$, $s_{2,1}$, $s_{3,1}$, $s_{4,2}$} with a total price of $2,504 = $836 + $672 + $636 + $360. The advantage of the first solution over the second one is $248 = $2,504 - $2,256.

3.5 Applicable Quality Dimensions

The aggregation of some quality values must take into account the system structure, e.g., response time. If there are multiple execution paths from the entry service to the exit service of the system, the one with maximum execution time determines the overall response time of the system. On the other hand, some quality dimensions are independent of the system structure, e.g., cost and availability. The total cost of an SBS is the summation of the costs of all its component services. The overall availability of an SBS is the multiplication of the availability of all its component services. Accordingly, the quality premium applicable to a service bundle relies on the structural relevance between the services. For adjacent services, quality premiums are applicable in all quality dimensions. For nonadjacent services, quality premiums are applicable only in quality dimensions that are independent of the system structure, e.g., cost and availability.

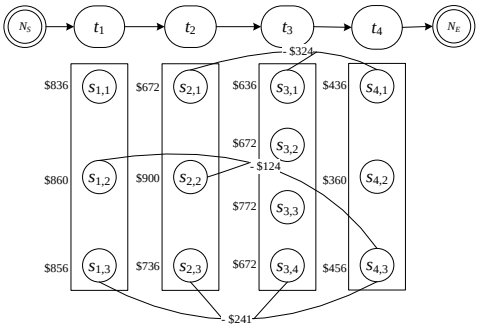


Fig. 6. Hybrid quality correlations.

4 Quality Correlation Index Graph

In a service composition scenario, the three different types of quality correlations often coexist, especially in a large-scale scenario with a large number of candidate services offered by service providers in different strategic alliances. Table 2 presents an example based on Fig. 3. The consideration of quality correlations further complicates the NP-hard problem of quality-aware service composition [2] by significantly increasing the search space of the problem. To address this issue, Q²C builds a quality correlation index graph that enables efficient queries for quality correlations. The process consists two steps: quality correlation aggregation and index graph construction.

4.1 Quality Correlation Aggregation

There are various approaches for solving the problem of quality-aware service composition. In this section, we employ the brute force service composition approach (referred to as *composition approach* in the remainder of this paper) to explain and demonstrate the construction of the quality correlation index graph as it is the most basic and straightforward approach. Other composition approaches can also query our quality correlation index graph for quality correlations. The composition approach inspects all possible service composition instances and selects the one with the optimal quality as the optimal solution. A service composition instance is composed of multiple

TABLE 2
Quality Correlation Table

ID #	Quality Correlations
1	$qc(s_{1,1}, s_{2,2})$ = -$314
2	$qc(s_{1,1}, s_{3,2})$ = -$240
3	$qc(s_{1,2}, s_{2,2}, s_{4,3})$ = -$124
4	$qc(s_{1,3}, s_{2,3}, s_{3,4}, s_{4,3})$ = -$241
5	$qc(s_{1,3}, s_{4,3})$ = -$314
6	$qc(s_{2,1}, s_{3,1}, s_{4,1})$ = -$324
7	$qc(s_{2,2}, s_{4,3})$ = -$256
8	$qc(s_{2,3}, s_{3,4})$ = -$112
9	$qc(s_{3,2}, s_{4,3})$ = -$141

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2830773, IEEE Transactions on Services Computing

component services, one from each class of candidate services. Take the SBS presented in Fig. 3 for example, there are a total of 108 (3 × 3 × 4 × 3) possible service composition instances, including $sc(S_1) = \{s_{1,1}, s_{2,1}, s_{3,1}, s_{4,1}\}$, $sc(S_2) = \{s_{1,1}, s_{2,1}, s_{3,1}, s_{4,2}\}$, …, $sc(S_{108}) = \{s_{1,3}, s_{2,3}, s_{3,4}, s_{4,3}\}$. Among all those service composition instances, the optimal solution is the one with the lowest total cost.

When inspecting a service composition instance $sc(S_i)$, the composition approach needs to query the quality correlation table for all *applicable quality correlations* that involve any of $sc(S_i)$'s component services. Here we formally define the concept of applicable quality correlations:

DEFINITION 1. Applicable Quality Correlation: Given a service composition instance $sc(S_i)$, a quality correlation $qc(S_j)$ is applicable to $sc(S_i)$ if $S_j \subseteq S_i$.

Take $sc(S_1) = \{s_{1,1}, s_{2,1}, s_{3,1}, s_{4,1}\}$ for example, the composition approach queries Table 2 and obtains three quality correlations, ie.., $qc\#1 = qc(\boldsymbol{s_{1,1}}, s_{2,2})$, $qc\#2 = qc(\boldsymbol{s_{1,1}}, s_{3,2})$ and $qc\#6 = (\boldsymbol{s_{2,1}}, s_{3,1}, \boldsymbol{s_{4,1}})$, that are applicable to $sc(S_1)$ because they involve $sc(S_1)$'s component services. Given a total of f possible service composition instances, each composed of k component services, and g quality correlations, to build an SBS, the composition approach needs to inspect a total of $f \times k \times g$ quality correlations to obtain all applicable quality correlations in the worst-case scenario. Take the SBS presented in Fig. 3 for example, a total of 3,888 (108 × 4 × 9) entries in Table 2 will be inspected. This method for quality correlation query runs in $O(fkg)$ and is very inefficient, especially in large-scale scenarios. However, it has the lowest space complexity, requiring only a total of g quality correlations to be stored.

To improve the efficiency of quality correlation query, some of the quality correlations can be aggregated. Take Table 2 for example, $qc\#1 = qc(s_{1,1}, s_{2,2})$ = -$314 and $qc\#2 = qc(s_{1,1}, s_{3,2})$ = -$240 can be aggregated to create a new quality correlation $qc\#10 = qc(s_{1,1}, s_{2,2}, s_{3,2})$ = -$554. For a service composition instance that involves $s_{1,1}$, $s_{2,2}$ and $s_{3,2}$, the application of both qc#1 and qc#2 is equivalent to the application of qc#10. Thus, the composition approach only needs to find qc#10 instead of qc#1 and qc#2 to find out the quality correlations between $s_{1,1}$, $s_{2,2}$, and $s_{3,2}$. Quality correlation qc#10 can be further aggregated with *qc#7* to create a new quality correlation $qc(s_{1,1}, s_{2,2}, s_{3,2}, s_{4,3})$, which can be inspected to find out the quality correlations between $s_{1,1}$, $s_{2,2}$, $s_{3,2}$, $s_{4,3}$. Quality correlations that can be aggregated are referred to as *compatible quality correlations*, as defined below:

DEFINITION 2. Compatible Quality Correlations: Two quality correlations are compatible if they do not involve two different services that belong to the same class of candidate services.

Take Table 2 for example. Quality correlations qc#1 and qc#2 are compatible. However, qc#1 and qc#3 are incompatible because $s_{1,1}$ and $s_{1,2}$ belong to the same class of candidate services S_1.

Q²C employs Algorithm 1 to aggregate compatible quality correlations. Taking a quality correlation table QCT_1 as input, it first enumerates all pairs of quality correlations in the table to create new quality correlations (lines 2 - 10). Algorithm 1 first checks whether qc#1 and qc#2 are compatible. If they are, it creates a new quality correlation by aggregating qc#1 and qc#2 with the $\oplus$ operation and adds it to the quality correlation table. Then, it proceeds to check qc#1 and qc#3, then qc#1 and qc#4, etc. After enumerating all pairs of the original quality correlations in the table, Algorithm 1 also processes the newly created quality correlations until no new quality correlations are created. Fig. 7 demonstrates how Algorithm 1 processes the quality correlations presented in Table 2. At Step 1, it pairs qc#1 with every other quality correlation and creates three new quality correlations, i.e., qc#10, qc#11 and qc#12, by aggregating qc#1 and three quality correlations compatible with qc#1, i.e., qc#2, qc#7 and qc#9 respectively. At this stage, a newly created quality correlation still contains the corresponding pair of compatible quality correlations to preserve the information needed for further processing. After processing qc#1, the algorithm pairs qc#2 and every other quality correlation except qc#1 at Step 2 because the pair of qc#1 and qc#2 has already been processed at Step 1. The algorithm continues until no new quality correlations can be created. After the creation of all new quality correlations, the algorithm optimizes the quality correlation table to guarantee the uniqueness of each quality correlation (lines 11 - 19). If there are multiple applicable quality correlations that involve the same service bundle, the one with the optimal quality premium, e.g., maximum discount, is reserved and the others are removed from the quality correlation table (lines 13 - 17). Take the quality correlation table at Step n in Fig. 7 for example, qc#12 and qc#k are both applicable to the same service bundle $\{s_{1,1}, s_{2,2}, s_{3,2}, s_{4,3}\}$. Quality correlation qc#k with a total discount of $951 ($314 + $240 + $256 + $141) is better than qc#12 with $455 ($314+$141). Thus, Algorithm 1 will reserve qc#k and re-

Algorithm 1: Quality Correlation Aggregation

```
Input: Quality correlation table QCT1
Output: Optimized quality correlation table QCT2
 1: QCT2 ← QCT1
 2: for each qc(Si) ∈ QCT2 do
 3:   for each qc(Sj) ∈ QCT1 do
 4:     if qc(Si) and qc(Sj) are compatible
 5:       k ← QCT2.Length + 1
 6:       qc(Sk) ← qc(Si) ⊕ qc(Sj)
 7:       QCT2.add(qc(Sk))
 8:     end if
 9:   end for
10: end for
11: for each qc(Si) ∈ QCT2 do
12:   for each qc(Sj) ∈ QCT2 ∧ i != j do
13:     if Si = Sj then
14:       if qc(Si) > qc(Sj) then
15:         QCT2.remove(qc(Si))
16:       else
17:         QCT2.remove(qc(Sj))
18:       end if
19:     end if
20:   end for
21: end for
22: return QCT2
```

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2830773, IEEE Transactions on Services Computing

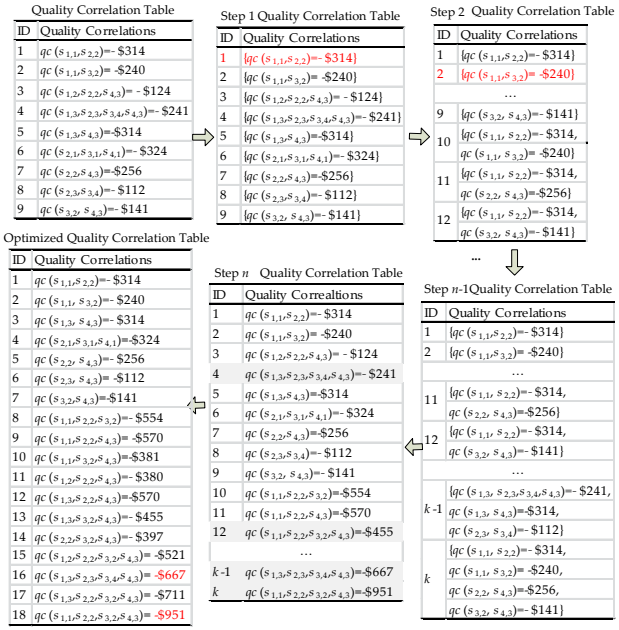


Fig. 7. The process of quality correlation aggregation.

move *qc*#12. In this research, we assume no *pricing error* phenomena, where the discount for a set of services S_i is worse than the total discount for the combination of two subsets of S_i. For example, the discount for $\{s_{1,1}, s_{2,1}, s_{3,1}, s_{4,1}\}$ is never lower than the total discount for $\{s_{1,1}, s_{2,1}\}$ and $\{s_{3,1}, s_{4,1}\}$. This way, the uniqueness of the quality correlations, defined below as Property 1, can be guaranteed.

PROPERTY 1. Uniqueness: Each quality correlation is unique, i.e., for any service composition instance $sc(S_i)$, there are no quality correlations that involve any same service bundle that belongs to S_i.

The uniqueness of the quality correlations ensures that, for each service instance $sc(S_i)$, the composition approach only needs to find the quality correlation that involves the maximum number of $sc(S_i)$'s component services rather than having to find multiple individual quality correlations that involve different subsets of S_i.

Now we analyze the time complexity of querying the processed quality correlation table. Given a total of f service composition instances, each composed of k component services, and g quality correlations, the composition approach needs to inspect a total of $f \times g$ quality correlations to obtain all applicable quality correlations in the worst-case scenario. This is much fewer than before the quality correlation table is processed, which is a total of $f \times k \times g$ quality correlations as discussed in Section 4.1. This improvement comes at the price of space complexity caused by the new quality correlations. Take Fig. 7 for example. At Step 1, a maximum of k - 2 new quality correlations can be created based on each of the initial g quality correlations. Thus, there are a total of $g + g\ (k - 2) / 2 = g \cdot k / 2$ quality correlations in the table at the end of Step 1. Similarly, at the end of Step 2, there are a maximum of $g \cdot k/2 + g \cdot k/2 \cdot (k - 2) / 2 = g \cdot (k / 2)\ \hat{}\ 2$ quality correlations in the table. The same applies to Steps 3, 4, …, and Step k – 2, which is the final step. At the end of Step k – 2, in the worst-case scenario, there are a maximum of $g \cdot (k / 2)\ \hat{}\ (k – 2)$ quality correlations in the table.

During the entire quality correlation aggregation process, there is no information loss. Take Fig. 7 for example. The original 9 quality correlations, after being processed, are still in the optimized quality correlation table. Thus, it is guaranteed that Q²C does not jeopardize service composition approaches' chances of finding optimal solutions.

4.2 Quality Correlation Index Graph Construction

The creation of new quality correlations increases the total number of quality correlations, potentially leading to a very large quality correlation table. Given n classes of candidate services, each containing m candidate services, in the worst-case scenario, there will be a quality correlation applicable for every two candidate services from two different classes of candidate services - a total of $C_n^2 \times m^2$ such quality correlations. There will also be a quality correlation applicable for every three candidate services from three different classes of candidate services - a total of $C_n^3 \times m^3$ such quality correlations. The same goes to every four candidate services, every five, six, etc. In total, there are $\sum_2^n C_n^i \times m^i$ unique quality correlations in the quality correlation table. These massive quality correlations can significantly slow down the queries for quality correlations.

To address this issue, Q²C constructs an index graph that enables efficient queries for quality correlations. In this index graph, a node is a quality correlation – terms "quality correlation" and "node" are interchangeable in the discussion of the index graph – and a directed edge indicates the *containing relation* between two quality correlations, which is defined as follows:

DEFINITION 3. Containing Relation: Given two quality correlations $qc_1(S_1)$ and $qc_2(S_2)$, where S_1 and S_2 are two sets of services, $qc_1(S_1)$ is contained in $qc_2(S_2)$ or $qc_2(S_2)$ contains $qc_1(S_1)$, if $S_1 \subset S_2$, indicated as $qc_1(S_1) \subset qc_2(S_2)$ or $qc_2(S_2) \supset qc_1(S_1)$.

The index graph is constructed according to the containing relations among the quality correlations. Suppose four quality correlations, $qc\#a = qc(s_{1,x}, s_{2,y}, s_{3,z})$, $qc\#b = qc(s_{1,x}, s_{2,y})$, $qc\#c = qc(s_{1,x}, s_{3,z})$ and $qc\#d = qc(s_{2,y}, s_{3,z})$. According to Definition 3, $qc\#a$ contains $qc\#b$, $qc\#c$ and $qc\#d$. The containing relation indicates that, for a service composition instance $SC_i = \{s_{1,x}, s_{2,y}, s_{3,z}\}$, the composition approach applies $qc\#a$ instead of $qc\#b$, $qc\#c$ and $qc\#d$ because $qc\#a$ contains the most information about the quality correlations between $s_{1,x}$, $s_{2,y}$, $s_{3,z}$. The application of $qc\#a$ instead of any two of $qc\#b$, $qc\#c$ and $qc\#d$ does not trap the composition approach in a local optimum. The reason is that Algorithm 1 has already combined every two of $qc\#b$, $qc\#c$ and $qc\#d$ to create new quality correlations that involve $s_{1,x}$, $s_{2,y}$, $s_{3,z}$ and has reserved only the quality correlation with the optimal quality premium. Take the optimized quality correlation table in Fig. 7 for example. Quality correlation $qc\#8 = qc(s_{1,1}, s_{2,2}, s_{3,2})$ contains $qc\#1 = qc(s_{1,1}, s_{2,2})$ and $qc\#2 = qc(s_{1,1}, s_{3,2})$. The containing relation indicates that, for a service composition instance involving $s_{1,1}$, $s_{2,2}$ and $s_{3,2}$, the composition approach applies $qc\#8$ instead of $qc\#1$ and $qc\#2$ because $qc\#8$ contains more information about the quality correlations among $s_{1,1}$, $s_{2,2}$, $s_{3,2}$ than $qc\#1$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2830773, IEEE Transactions on Services Computing

ID	Quality Correlations
1	$qc_{cost}(s_{1,1},s_{2,2})$=- \$314
2	$qc_{cost}(s_{1,1}, s_{3,2})$=- \$240
3	$qc_{cost}(s_{1,3}, s_{4,3})$=- \$314
4	$qc_{cost}(s_{2,1},s_{3,1},s_{4,1})$=-\$324
5	$qc_{cost}(s_{2,2}, s_{4,3})$=- \$256
6	$qc_{cost}(s_{2,3}, s_{4,3})$= -\$112
7	$qc_{cost}(s_{3,2},s_{4,3})$=-\$141
8	$qc_{cost}(s_{1,1},s_{2,2},s_{3,2})$=- \$554
9	$qc_{cost}(s_{1,1},s_{2,2},s_{4,3})$= -\$570
10	$qc_{cost}(s_{1,1},s_{3,2},s_{4,3})$=-\$381
11	$qc_{cost}(s_{1,2},s_{2,2},s_{4,3})$=- \$380
12	$qc_{cost}(s_{1,3},s_{2,2},s_{4,3})$=-\$570
13	$qc_{cost}(s_{1,3},s_{3,2},s_{4,3})$=- \$455
14	$qc_{cost}(s_{2,2},s_{3,2},s_{4,3})$=- \$397
15	$qc_{cost}(s_{1,3},s_{2,3},s_{3,4},s_{4,3})$= -\$667
16	$qc_{cost}(s_{1,2},s_{2,2},s_{3,2},s_{4,3})$= -\$521
17	$qc_{cost}(s_{1,3},s_{2,2},s_{3,2},s_{4,3})$= -\$711
18	$qc_{cost}(s_{1,1},s_{2,2},s_{3,2},s_{4,3})$= -\$951

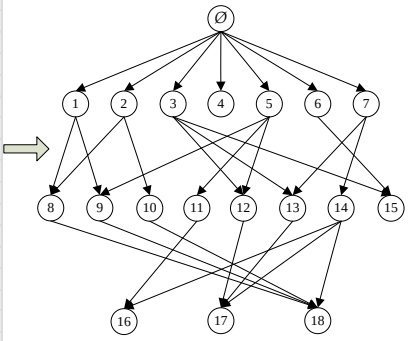


Fig. 8. Example of aggregation relation index graph.

and qc#2.

According to the containing relations among the quality correlations, Q²C employs Algorithm 2 to construct the quality correlation index graph. The principal is that *a parent node in the index graph is contained in its children nodes*. Take the optimized quality correlation table in Fig. 7. for example, there is qc#1 $\subset$ qc#9 $\subset$ qc#15 because $\{s_{1,1}, s_{2,2}\} \subset \{s_{1,1}, s_{2,2}, s_{4,3}\} \subset \{s_{1,1}, s_{2,2}, s_{3,2}, s_{4,3}\}$. Accordingly, qc#9 will be inserted as one of qc#1's children nodes and qc#15 as one of qc#9's.

Algorithm 2 takes a quality correlation table as the input. After inserting a Ø quality correlation as the only entry node of the index graph, Algorithm 2 enumerates all the quality correlations in the table and inserts them into the index graph as nodes according to their containing relations. For each of the remaining quality correlations, denoted by $qc(S_i)$, Algorithm 2 employs a breadth-first algorithm to find the first quality correlation in the graph, denoted by $qc(S_j)$, where $qc(S_j) \subset qc(S_i)$, and inserts $qc(S_i)$ as one of $qc(S_j)$'s children nodes. After all the quality correlations are inserted into the index graph, the index graph is returned. A quality correlation that is not contained by any other quality correlations will be inserted as a child node of the Ø node. Fig. 8 demonstrates the index graph constructed from the optimized quality correlation table from Fig. 7. Quality correlations qc#1, qc#2, …, and qc#7 are first inserted into the index graph as children nodes of the Ø node because none of them are compatible with each other. Next, qc#8 is inspected and inserted as one of qc#1's children nodes because qc#1 $\subset$ qc#8. The next quality correlation, qc#9, is also inserted a child node of qc#1 because qc#1 $\subset$ qc#9. Algorithm 2 completes when the last quality correlation, i.e., qc#18 is inspected and properly inserted into the index graph.

In this index graph, some of the nodes have multiple parent nodes. Take qc#13 in Fig. 8 for example, there are qc#3 $\subset$ qc#13 and qc#7 $\subset$ qc#13. Accordingly, qc#13 has two parent nodes in the index graph, i.e., qc#3 and qc#7. Thus, we have Theorem 1:

Algorithm 2: Quality Correlation Index Graph Construction

Input: Quality correlation relation table *QCT*
Output: Quality correlation index graph *iGraph*
1: *iGraph.add*(Ø)
2: **for** each $qc(S_i) \in QCT$ **do**
3: | find $qc(S_j)$ where $qc(S_j) \subset qc(S_i)$
4: | iGraph.$qc(S_j)$.*addChild*($qc(S_i)$)
5: **end for**
6: **return** *iGraph*

THEOREM 1. The quality correlation index graph is a direct acyclic graph (DAG), i.e., a directed graph that contains no cycles.

PROOF. This can be proven by contradiction. If there exist three nodes qc#1, qc#2 and qc#3 in the index graph forming a cycle, where qc#1 points to qc#2, qc#2 points to qc#3 and qc#3 points to qc#1. Accordingly, there are qc#1 $\subset$ qc#2, qc#2 $\subset$ qc#3 and qc#3 $\subset$ qc#1. According to set theory, the containing relation defined before is transitive, i.e., given qc#1 $\subset$ qc#2 and qc#2 $\subset$ qc#3, there is qc#1 $\subset$ qc#3. Now given qc#3 $\subset$ qc#1 and qc#1 $\subset$ qc#3, there is qc#1 = qc#3, which contradicts the uniqueness property.

Theorem 2 indicates the depth of the index graph, i.e., the maximum distance from the Ø node to the end nodes (i.e., nodes without outgoing edges):

THEOREM 2. Given n classes of candidate services, i.e., n tasks in the SBS, the depth of the quality correlation index graph is at most n.

PROOF. According to Property 1 and Definition 3, each node in the index graph, except the Ø node, involves at least one more service than any of its parent nodes. In the worse-case scenario, each node on the longest path from the Ø node to the end node that involves n services involves exactly one more service than any of its parent nodes. Thus, the distance from the Ø node to the end node on that path is n.

Once constructed, the index graph can be updated to accommodate dynamic changes in quality of services or quality correlations among services without reconstruction. When such a change occurs, Q²C only needs to find and update the corresponding quality correlation(s).

4.3 Quality Correlation Queries

The index graph has one entry node, i.e., the Ø node, and many other nodes. Each node is a quality correlation that matches a specific set of services. Given a service composition instance $sc(S_i)$, a quality correlation query aims to find the node in the index graph *iGraph* that matches the maximum number of services in S_i. Q²C employs Algorithm 3 to answer a query It first finds the node from the Ø node's children nodes that matches S_i with the maximum number of services. If there are no such nodes, it returns the Ø node. If there are multiple such nodes, it randomly selects one of them and checks if any of the selected node's children nodes matches S_i with more services (lines 3 - 8). This process iterates (lines 2 - 9) until a node is found, none of whose children nodes are applicable to $sc(S_i)$ (line 9). That node is returned as the optimal quality correlation applicable to $sc(S_i)$. Take $sc(S_1) = \{s_{1,3}, s_{2,3}, s_{3,4}, s_{4,3}\}$ in Fig. 8 for example, Algorithm 3 first finds that qc#3 is applicable to $sc(S_1)$ because it matches $sc(S_1)$ with two services, i.e., $s_{1,3}$ and $s_{4,3}$. Then, it inspects qc#3's children nodes, i.e., qc#12, qc#13 and qc#15, and finds that only qc#15 is applicable to $sc(S_1)$. Node qc#15 has no child node and thus is returned as the optimal quality correlation applicable to $sc(S_1)$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2830773, IEEE Transactions on Services Computing

Algorithm 3: Index Graph Query

Input:
Service composition instance $sc(S_i)$
Quality correlation index graph *iGraph*
Output:
Optimal quality correlation *QC* applicable to $sc(S_i)$

```
1: Initialize QC ← iGraph.root
2: do
3:   find qc ∈ QC.childrenNodes that matches sc(Si)
     with maximum services
4:   if qc ≠ Ø then
5:     QC ← qc
6:   else
7:     return QC
8:   end if
9: while QC.childrenNodes ≠ Ø
10: end
```

Algorithm 3 does not always find a quality correlation that matches all the services of a service composition instance. For example, given a service composition instance $sc(S_2) = \{s_{1,3}, s_{2,4}, s_{3,2}, s_{4,3}\}$, Algorithm 3 first finds *qc*#3 and *qc*#7 that match $sc(S_2)$. Because *qc*#3 and *qc*#7 both match $sc(S_2)$ with two services, it randomly selects one of them to proceed. In this example, we assume that it selects *qc*#3. From *qc*#3's children nodes, i.e., *qc*#12, *qc*#13 and *qc*#15, the algorithm finds that only *qc*#13 is applicable to $sc(S_2)$. It selects *qc*#13. Node *qc*#13 has only one child node, i.e., *qc*#17, which is not applicable to $sc(S_2)$. Thus, the algorithm returns *qc*#13 as the quality correlation applicable to $sc(S_2)$.

According to Theorem 1, the index graph is a DAG. Thus, a query always completes, with either the Ø node or an applicable node as the result. The query performance is dependent of the depth on the index graph, as shown by Theorem 3:

THEOREM 3. Given a service composition instance $sc(S_i)$, where $|S_i| = k$, suppose an end node in the index graph involves all services in S_i, Algorithm 3 takes a maximum of k iterations to reach from the Ø node to that end node.

PROOF. Except for the Ø node, each node selected by Algorithm 3 involves at least one more service than the node selected at the previous iteration. In the worse-case scenario, the node selected by Algorithm 3 at each iteration involves exactly one more service than the node selected at the previous iteration. Thus, it takes Algorithm 3 a maximum of k iterations to find the end node forming the longest path from the Ø node that determines the depth of the index graph.

The time complexity of querying for quality correlations based on the index graph is analyzed as follows. Given a total of f service composition instances, each composed of k component services, and g quality correlations, the composition approach needs to inspect a total of $f \times k$ quality correlations to obtain all applicable quality correlations in the worst-case scenario. This is much fewer than before the quality correlation index graph in constructed, which is a total of $f \times g$ quality correlations because there is $k \ll g$ in most cases. The space complexity is the same as before the construction of the index graph, which is $g \cdot (k / 2)\ \hat{}\ (k - 2)$. No new quality correlations are created during the construction of the index graph. Thus, there are a maximum of $g \cdot (k / 2)\ \hat{}\ (k - 2)$ nodes in the index graph.

5 Experimental Evaluation

This section evaluates the effectiveness of Q²C through the comparison between a composition approach with and without the support of Q²C, where the effectiveness is measured by the success rate of finding a solution to the service composition problem and the system optimality. This section also evaluates the efficiency of Q²C, measured by its computation time for preprocessing quality correlations and the computation time taken by the composition approach, through a comparison with the state-of-the-art approach.

5.1 Prototype Implementation

We developed a prototype of Q²C in C# on Visual Studio 2010. Based on the quality correlation model introduced in Section 3, it implements the algorithms introduced in Section 4. Given the quality correlations among the candidate services, the prototype processes the quality correlations and constructs the quality correlation index graph.

Based on the quality correlation index graph, a service composition approach queries for quality correlations when searching for the solution to the problem of quality correlation aware service composition. In the past decade, the most popular optimal service composition approaches are based on integer programming (IP) [4, 19, 24]. However, in most, if not all, large-scale scenarios, finding a sub-optimal solution efficiently is more important and practical than finding the optimal solution. Many heuristics approaches have been proposed to serve this purpose [3, 22, 24, 31]. Such approaches heuristically inspect different service composition instances for the solution. The essential difference between those approaches is the adopted heuristics. Q²C supports all the approaches that fall into this category in the same way and is not dependent on any specific heuristics. To evaluate Q²C in a generic manner, we have employed a simple heuristic service composition method similar to [3]. First of all, the candidate services in each class of candidate services are ranked and sorted by a descending order of their utility values. The service composition method then iterates to, from all possible service composition instances, find the one that fulfills all the quality constraints and achieves the optimization goal for the SBS. In the i^{th} ($i \geqslant 1$) iteration, the method takes 2^{i-1} more candidate services from each set of candidates (1 in the first iteration, 2 in the second, 4 in the third, etc.) and inserts them into the search space to increase the chances of finding a solution. When inspecting a possible service composition instance, the service composition method queries the quality correlation index graph for the quality correlation applicable to the service composition instance. This Q²C-based service composition approach is referred to as Q²CO hereafter.

5.2 Comparing Approaches

To evaluate the Q²CO, We have implemented the following two approaches for comparison with Q²CO:

- nQ²CO: This approach does not consider the quality correlations among candidate services. It employs the heuristic service composition method introduced in Section 5.1 to search for solutions based on only the original quality of the candidate services.
- uQ²CO: This approach queries for quality correlations by looking up a table that contains the original quality correlations instead of using the quality correlation index graph introduced in Section 4.

The ability to query for and apply quality correlations makes it easier and takes fewer iterations for Q²CO and uQ²CO to find a solution than nQ²CO. This advantage comes at a price – the extra computation time needed for Q²C to preprocess the quality correlations. Thus, we have also implemented the following approach to compare its efficiency in preprocessing quality correlations with Q²C:

- CASP [12]. This approach employs the skyline technique [8] to preprocess the quality correlations by removing the non-skyline quality correlations.

5.3 Experiment Setup

We conducted the experiments on QWS, a widely used public dataset that contains the functional and quality information on over 2500 real-world web services [1]. The evaluation process mimicked the SBS presented in Fig. 1. Web services were randomly selected from QWS to form different sets of candidate services, one for each task of the SBS. In the experiments, hybrid quality correlations, which have the characteristics of both adjacent and non-adjacent quality correlations, were randomly generated and applied to a certain proportion of the candidate services according to a quality correlation ratio – the percentage of candidate services involved in at least one quality correlation - between 5% to 30%. Quality premiums in different quality dimensions were randomly generated from the range between 10% and 30%.

As indicated in many literatures, the difficulty of the quality constraints has a significant impact on the success rate and system optimality achieved by the service composition approaches [15, 17, 19, 20]. Thus, we have also simulated three different difficulty levels in the quality constraints for the target SBS:

- **Simple**. This type of quality constraints is relatively easy to be satisfied in all quality dimensions.
- **Medium**. This type of quality constraints is more difficult to be satisfied than the "simple" level as the constraints for some quality dimensions are demanding.
- **Severe**. This type of quality constraints is the most difficult to be satisfied as the constraints imposed on all quality dimensions are demanding.

There are three main parameters that influence the effectiveness and efficiency of Q²C: 1) the number of candidate services per task; 2) the quality correlation ratio; and 3) the number of quality dimensions. Accordingly, in each experiment series, we have conducted three sets of experiments. In Set #1, we increased the number of candidate services per task from 10 to 90 in steps of 10 while fixing the quality correlation ratio at 30% and the number of quality dimensions at 2. In Set #2, we increased the quality correlation ratio from 5% to 30% in steps of 5% while fixing the number of candidate services per task at 90 and the number of quality dimensions at 2. In Set #3, we increased the number of quality dimensions (i.e., the number of quality constraints) from 1 to 9 in steps of 1 while fixing the number of candidate services per task at 90 and the quality correlation ratio at 30%. Under each parameter setting, we changed the difficulty level of quality constraints, creating three subsets of experiments in each set of experiments, i.e., "simple", "medium" and "severe". In each subset of experiments, 100 experiment instances were run and the collected results were averaged.

For effectiveness evaluation, we compare the *success rates* achieved by Q²CO, nQ²CO and uQ²CO, i.e., the percentage of runs where a solution was found. We also compare the system optimality achieved by Q²CO, nQ²CO and uQ²CO, indicated by *system cost* because we employed minimum system cost as their optimization goal. The QWS dataset does not contain cost information. Thus, we employed the following method to calculate a cost for the web services in QWS. First, we normalized the quality values of all the web services in each quality dimension with the min-max normalized technique, which has been widely in a lot of research [4, 19]. The cost of a web service s can then be calculated by $cost(s) = \sum_{i=1}^{p} q_i(s) / p$, $p \leqslant 9$, where $q_i(s)$ is the normalized i^{th}-dimensional quality value of s. This way, a service with a high overall utility has a high cost, and vice versa, which is realistic in real-world scenarios. The total cost of a system $\mathbb{S} = \{s_1, \ldots, s_r\}$ is the summation of the costs of all its component services: $cost(\mathbb{S}) = \sum_{i=1}^{r} cost(s_i)$. The system cost achieved by a service composition indicates its ability to achieve the optimization goal – the lower, the better.

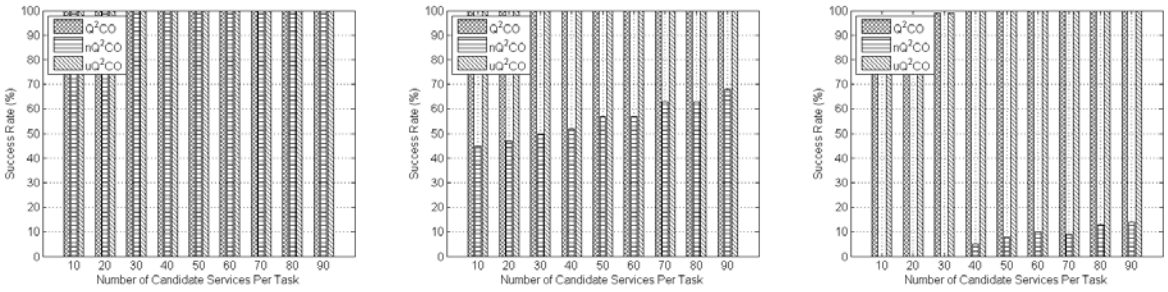
For efficiency evaluation, we first compare the computation time taken for Q²C to preprocess the quality correlations with CASP [12]. We then compare the computation times taken by Q²CO, nQ²CO and uQ²CO respectively to complete under different parameter settings. The comparison between Q²CO and uQ²CO is aimed to demonstrate the usefulness of the quality correlation index tree discussed in Section 4. CASP is not included in this comparison because it cannot handle the hybrid quality correlations in the experiments.

All experiments were conducted on a machine with Intel Dual Core i7 3.6GHZ CPU, 16G RAM, running Windows 7 x64 Enterprise.

5.4 Effectiveness Evaluation

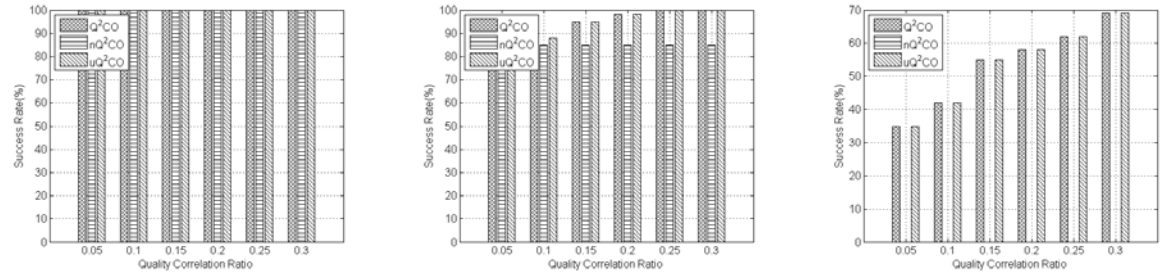
Success Rate. Figs. 9-11 present the success rates achieved by nQ²CO, Q²CO and uQ²CO under different parameter settings. As demonstrated, Q²C and uQ²C always achieve the same success rate under the same parameter setting. This indicates that the conversion from the original quality correlations to the quality correlation index graph introduced in Section 4 does not sacrifice the correctness in the queries of applicable quality correlations. In the remaining discussion in this section, we focus on the comparison between Q²CO and nQ²CO. Figs. 9-11 illustrate that Q²CO significantly outperforms nQ²CO, 92.22% versus 58.54% on average across all experiments. As the dif-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2830773, IEEE Transactions on Services Computing



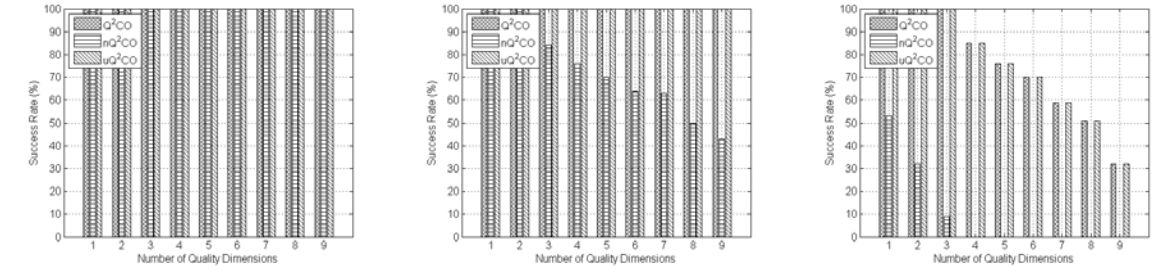
(a) Simple (b) Medium (c) Severe

Fig. 9. Set #1: success rate versus the number of candidate services



(a) Simple (b) Medium (c) Severe

Fig. 10. Set #2: success rate versus the quality correlation ratio



(a) Simple (b) Medium (c) Severe

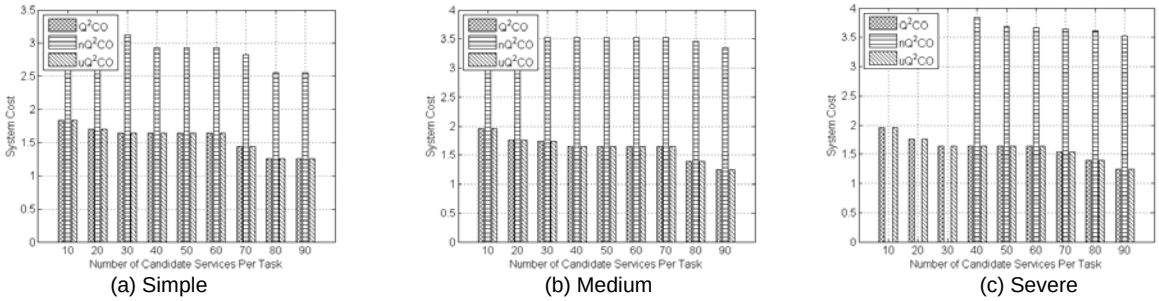
Fig. 11. Set #3: success rate versus the number of quality dimensions

ficulty level increases from "simple" to "severe", the success rate of nQ²CO decreases significantly in all three sets of experiments while Q²CO consistently maintains significantly higher success rates compared to nQ²CO. In particular, in the "severe" subsets of experiments, the success rate of nQ²CO drops to zero in most cases while Q²CO is still able to find a solution in certain percentages of those "severe" cases. Specifically, the average success rates of Q²CO are 97.78%, 53.5% and 74.78% in the "severe" scenarios of experiment Sets #1, #2 and #3 respectively, 75.35% on average, versus 6.56%, 0.00% and 10.44% achieved by nQ²CO. In 5 of the 9 subsets of experiments, including Sets #1-Simple, #1-Medium, #2-Simple, #3-Simple and #3-Medium, where not all quality constraints are demanding, Q²CO always finds a solution, as illustrated by Figs. 9(a), 9(b), 10(a), 11(a) and 11(b) respectively. Even in Set#2-Medium, an exception where Q²CO could not find a solution in some cases, its success rate was still above 85.00%, as shown by Fig. 10(b).

In Sets #1-Severe and #2-Severe, the increase in the number of candidate services per task and the quality correlation ratio both lead to an increase in the success rate achieved by Q²CO, from 89.00% to 100.00% on average in Set #1-Severe and from 35.00% to 69.00% in Set #2-Severe. The reason for the increase in the success rate in Set #1-Severe is that the increase in the number of candidate services per task offers Q²CO more choices for each task, making it more possible to find a solution. Similarly, in Set #2-Severe, a higher quality correlation ratio creates more (and possibly higher) quality premiums which also increases the possibility of finding a solution. In Set #3-Severe, the increase in the number of quality dimensions from 1 to 9 makes it harder to find a solution because Q²CO needs to find a solution that fulfills the quality dimensions in more quality dimensions. Consequently, the success rate achieved by Q²CO decreases significantly from 100.00% to 32.00%.

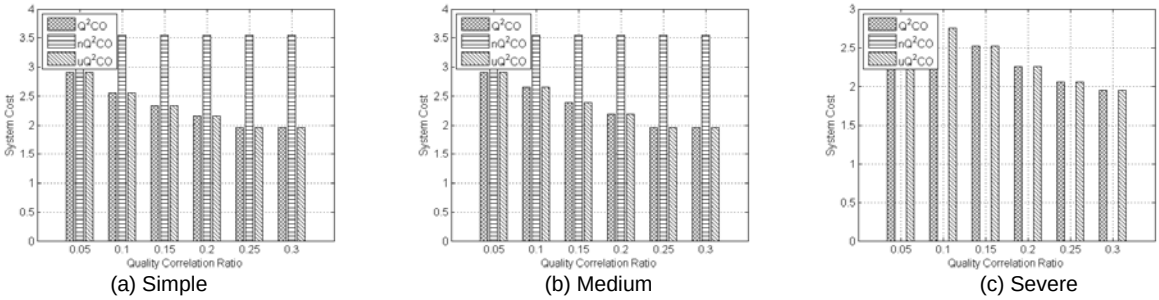
System Cost. Figs. 12 – 14 compare the average system cost obtained by nQ²CO and Q²CO. Similar to Figs. 9-11, Figs. 12-14 also demonstrate that Q²C and uQ²C always achieve the same system cost under the same parameter settings. This, again, confirms the ensured correctness during the construction of the quality correlation index graph discussed in Section 4. Thus, in the following discussion, we focus on the comparison between Q²CO and nQ²CO. Please note that the success rates achieved by nQ²CO are missing from Figs. 12(c), 13(c) and 14(c). This indicates that nQ²CO could not find a solution in those cases, as shown earlier in Figs. 9(c), 10(c) and 11(c). Across all successful cases where a solution was found, the system cost achieved by Q²CO is 1.905 versus 2.991 achieved by nQ²CO. Specifically, Q²CO outperforms nQ²CO by 108% (1.598 versus 3.333) in Set#1, by 51% (2.353 versus 3.554) in Set #2 and 72.2% (1.914 versus 3.296) in Set #3. As

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2830773, IEEE Transactions on Services Computing



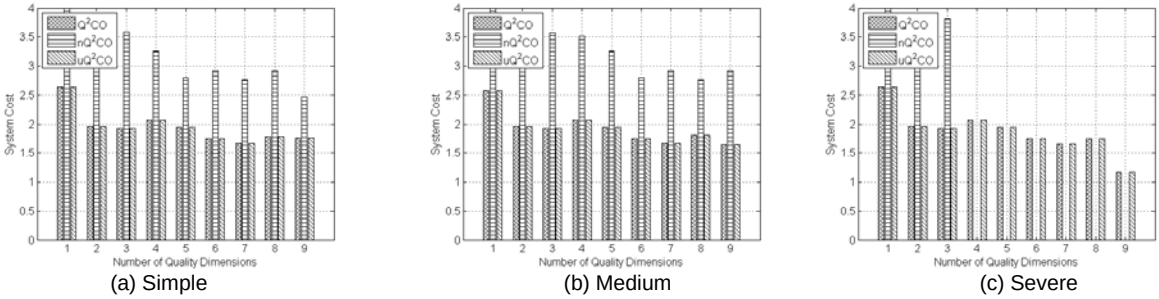
(a) Simple (b) Medium (c) Severe

Fig. 12. Set #1: system cost versus the number of candidate services



(a) Simple (b) Medium (c) Severe

Fig. 13. Set #2: system cost versus the quality correlation ratios



(a) Simple (b) Medium (c) Severe

Fig. 14. Set #3: system cost versus the number of quality dimensions

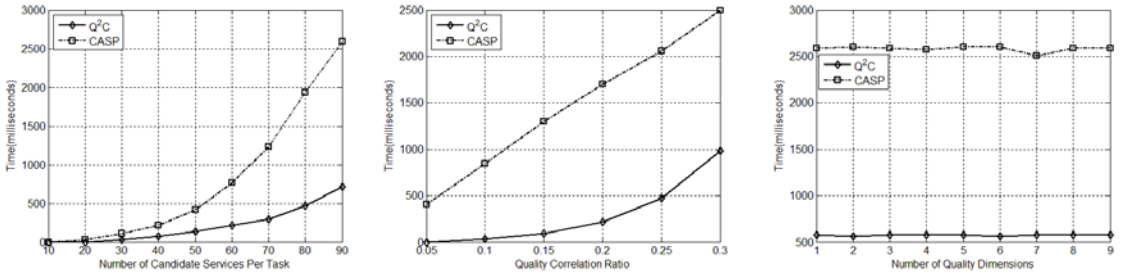
illustrated, the changes in the parameter settings do not significantly impact the system cost achieved by Q²CO. In Fig. 12, we can observe that the increase in the number of candidate services per task from 10 to 90 only leads to a slight decrease from 1.916 to 1.245 in system cost. Similar phenomenon can also be seen in Fig. 13 with a decrease from 2.902 to 1.955 in system cost. Fig. 14 shows that the increase in the difficulty level caused by the increase in the number of quality dimensions does not lead to an obvious increase or decrease in the system cost. Figs. 11 and 14 collectively demonstrate that the increase in the number of quality dimensions impacts the success rate achieved by Q²CO, however, does not compromise the ability of Q²CO to achieve system optimality.

5.5 Efficiency Evaluation

Preprocessing time. Fig. 15 demonstrates the computation time taken by Q²C to preprocess the quality correlations under different parameter settings. We can see that Q²C spends much less time than CASP on preprocessing the same quantity of quality correlations under the same parameter settings. Across all experiments, Q²C takes an average of 370ms, only 22.57% of the average 1639ms taken by CASP. Specifically, Q²C takes an average of 216ms to preprocess all quality correlations versus 810ms taken by CASP in Set #1, 298ms versus 1466ms in Set #2 and 573ms versus 2583ms in Set #3. Fig. 15 also illustrates that two of the experiment parameters significantly impact the preprocessing time taken by both Q²C and CASP: the number of candidate services per task and the quality correlations. This is expectable. The increases in those parameters immediately result in an increase in the total number of quality correlations, which require more time for Q²C and CASP to preprocess. Fig. 15(c) shows that the number of quality dimensions does not impact the preprocessing time of Q²C significantly. This indicates that Q²C can handle high-dimensional quality correlations efficiently. Compared with the search time, which is presented and analyzed next, the preprocessing time is not significant and thus is acceptable in most cases. In particular, quality correlations that are independent of system structure can be preprocessed offline without a specific system structure, e.g., price correlations. This can further reduce the computation time of Q²C at runtime. In extremely large-scale scenarios, Q2C can be paralleized to ensure its efficiency.

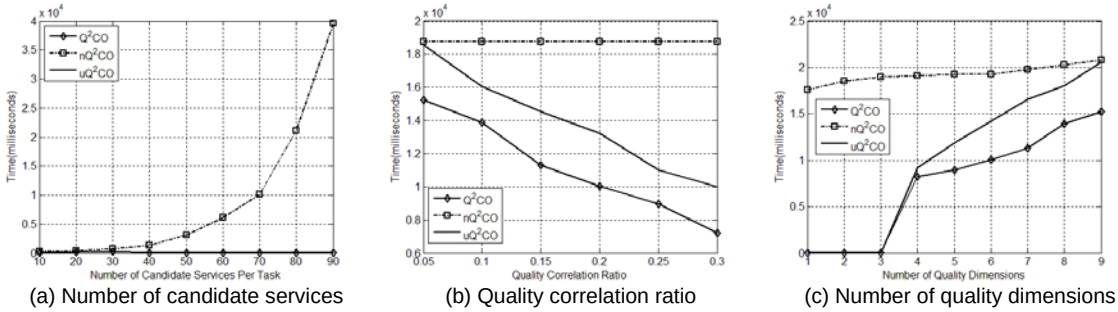
Search time. Fig. 16 demonstrates the search time taken by Q²CO, nQ²CO, and uQ²CO to complete under different parameter settings in the severe cases. Due to the space limit, the simple and the medium cases are not presented as they are similar to Fig. 16. Please note that the search time demonstrated in Fig. 16 is not the average search time to find a solution. It is the average search time required for nQ²CO, Q²CO and uQ²CO to *find a solution or determine that no solution can be found*. This way, we evalu-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2018.2830773, IEEE Transactions on Services Computing



(a) Set #1 (b) Set #2 (c) Set #3

Fig. 15. Quality correlation preprocessing time



(a) Number of candidate services (b) Quality correlation ratio (c) Number of quality dimensions

Fig. 16. Search time versus different parameters

ate the overall efficiency of nQ²CO, Q²CO and uQ²CO. As demonstrated, uQ²CO and Q²CO take much less time overall than nQ²CO to complete, specifically, 3,092ms versus 2,478ms and 9,164ms. This is because of uQ²CO and Q²CO's ability to query for and apply quality correlations to corresponding service composition instances. The application of quality correlations makes it easier and takes fewer iterations for uQ²CO and Q²CO to complete than nQ²CO. Fig. 16(a) demonstrates that the increase in the number of candidate services per task from 10 to 90 significantly increases the search time of nQ²CO from 262.18ms to 39,581.88ms, however not Q²CO and uQ²CO. This indicates that Q²CO and uQ²CO can efficiently handle large-scale scenarios with lots of candidate services. Fig. 16(b) demonstrates that the increase in the quality correlation ratio from 0.05 to 0.3 does not impact the search time of nQ²CO, but leads to decreases in the search times of Q²CO, from 15,236ms to 7,201ms, and uQ²CO from 18,542ms to 9,985ms. A large quality correlation ratio results in more quality correlations and potentially more applicable quality correlations. Thus, it takes few iterations for Q²CO and uQ²CO to find a solution. In Set #3, as illustrated by Fig. 16(c), more quality dimensions increase the difficulty of finding a solution, and thus require more time for all three approaches to complete. The impact of the increase in the number of quality dimensions is more significant on Q²CO and uQ²CO than on nQ²CO. This observation indicates that, as the number of quality dimensions increases, it becomes so hard to find a solution that even the application of quality correlations cannot significantly reduce the search time. In particular, when the number of quality dimensions reaches 9, nQ²CO and uQ²CO take approximately the same amount of time to complete. Note that this does not mean that nQ²CO and uQ²CO take exactly the same number of iterations to find a solution. In fact, uQ²CO still takes fewer iterations than nQ²CO. However, in each iteration, uQ²CO needs to query for applicable quality correlations. As the number of quality dimensions continues to increase, it is possible that uQ²CO (and even Q²CO) might take more time to complete than nQ²C. An important conclusion we can draw from Figs. 11(c), 14(c) and 16(c) is that, in extreme scenarios with a lot of severe quality constraints, nQ²CO and Q²CO have a better chance to find a solution, however, it might take more time than uQ²CO to complete when a solution cannot be found because of the extra time taken to query for applicable quality correlations.

6 Related Work

In the field of service computing, quality-aware service composition has attracted extensive attention in recent years and many approaches have been proposed [2-4, 9, 10, 21, 28, 32]. To name a few most representative ones, Zeng et al. [32] present AgFlow, a middleware platform that enables quality-driven composition of Web services. The selection of component service is performed to meet the users' requirements for the composite service's QoS modeled from multiple dimensions. IP is used to compute the optimal plan for composite service executions from several execution paths represented by Directed Acyclic Graph (DAG). Following the work in [32], in [4], Ardagna and Pernici formulate the quality-aware service selection problem as MIP and adopt loops peeling for optimization. When a feasible solution does not exist, a QoS negotiation algorithm is suggested to enlarge the solution space of the optimization problem. Alrifai and Risse [2] adopt a heuristic distributed method to find the best Web services that meet local QoS constraints generated by decomposing global QoS constraints using, again, IP. They then propose in [3] an approach based on the notion of skyline to reduce the search space for the problem of quality-aware service composition. In [21], Li et al. use a different philosophy from works described above to address the quality-aware service selection problem. They use Service Composition Graph (SCG) to represent the composite service. Then, they employ Dijkstra's shortest-

path algorithm to find the optimal solution to the service composition problem. In [28], Wang et al. find that optimal solutions can be found in polynomial time for some specially structured service compositions that consist of three services. Algorithms are proposed to detect if the optimal solution can be found for a given service composition in polynomial time. Attempting to maintain the optimality of SBSs at runtime, Cardellini et al. [9, 10] propose MOSES (Model-based Self-adaptation of SOA systems), a methodology that models the problem of service selection for SBS adaptation also as IP problems.

In recent years, many researchers have turned their attention to the problem of quality correlations [5, 11, 12, 19, 25, 27, 30, 33]. F. Wagner et al. [27] propose an approach that takes into account the time and input aspects which affect the quality values of a service. However, their approach does not incorporate quality correlations into the service composition process. F. Tao et al. [25] present a correlation-aware quality model for resource services, and propose a resource composition method based on the particle swarm optimization. However, they completely ignore the issue of efficiency and do not evaluate the efficiency of the proposed approach. Q. Wu et al. [30] propose a business correlation model and attempt to solve the quality constrained service composition problem through an improved genetic algorithm. However, the proposed approach is limited to the optimization of SBS that consists of only two tasks. L. Barakat et al. [5] present a correlation aware service selection approach capable of handling quality dependencies among services and pruning candidate web services. However, their pruning techniques can cause losses in quality correlations between services. In addition, their approach can handle only one quality dimension when solving the problem of quality correlation aware service composition. We have also conducted some research on quality correlations. In [19], we propose an auction-based approach to support service composition, where service providers can propose QoS offers with quality correlations. However, we did not systematically model different types of quality correlations in [19]. In [33], we propose an approach for quality correlation aware service composition based on quotient space and relation granulation quotient space. However, that approach can handle only one quality dimension and cannot handle hybrid quality correlations. In [12], we propose a quality correlation model and a skyline-based technique named CASP for pruning candidate web services. The major limitation to that approach is that it only considers the quality correlations between services provided by the same service provider, and not those between different service providers. As a result, their approach cannot handle service composition scenarios with hybrid quality correlations.

To address the above issue, based on our previous work [12, 33], we propose a systematic quality correlation model, and an approach named Q²C to facilitate efficient **Q**ueries of **Q**uality **C**orrelations without losing the correctness in the answers. Empowered by Q²C, composition approaches can achieve better success rates in finding a solution as well as higher system optimality under different parameter settings. Its efficiency (measured by processing time and search time) also outperforms CASP [12] significantly.

7 Conclusions and Future Work

Strategic alliances formed among enterprises in globalization have made quality correlation a critical issue in research on services. In this paper, we propose Q²C to enable efficient queries for quality correlations. Based on a systematic quality correlation model, Q²C preprocesses quality correlations to build an index graph that allows service composition approaches to query for applicable quality correlations efficiently without losing the correctness in the results. Comprehensive experimental analysis shows the effectiveness and efficiency of Q²C.

In the future, we will investigate the integration of Q²C in integer programming based approaches for service composition. We will also develop an enhanced version of Q²C powered by the Spark distributed computing platform further improve its efficiency.

Acknowledgment

This work is supported by the National Key Research and Development Program of China (No. 2015BAK24B01, 2017YFB1400600), the National Science Foundation of China (No. 61772461), the Key Research and Development Project of Zhejiang Province (No. 2015C01027). Qiang He is the corresponding author of this paper.

References

[1] E. Al-Masri and Q. H. Mahmoud, "Investigating Web Services on the World Wide Web," *Proc. the 17th International Conference on World Wide Web (WWW2008)*, Beijing, China, pp. 795-804, 2008.

[2] M. Alrifai and T. Risse, "Combining Global Optimization with Local Selection for Efficient QoS-Aware Service Composition," *Proc. the 18th International Conference on World Wide Web (WWW2009)*, Madrid, Spain, pp. 881-890, 2009.

[3] M. Alrifai, D. Skoutas, and T. Risse, "Selecting Skyline Services for QoS-based Web Service Composition," *Proc. the 19th International Conference on World Wide Web*, Raleigh, North Carolina, USA, pp. 11-20, 2010.

[4] D. Ardagna and B. Pernici, "Adaptive Service Composition in Flexible Processes," *IEEE Transactions on Software Engineering*, vol. 33, no. 6, pp. 369-384, 2007.

[5] L. Barakat, S. Miles, and M. Luck, "Efficient Correlation-Aware Service Selection," *Proc. the 19th IEEE International Conference on Web Services*, pp. 1-8, 2012.

[6] L. Baresi and S. Guinea, "Self-Supervising BPEL Processes," *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 247-263, 2011.

[7] D. Benslimane, S. Dustdar, and A. Sheth, "Services Mashups: The New Generation of Web Applications," *IEEE Internet Computing*, vol. 12, no. 5, pp. 13-15, 2008.

[8] S. Börzsönyi, D. Kossmann, and K. Stocker, "The Skyline Operator," *Proc. the International Conference on Data Engineering (ICDE2001)*, Washington, DC, USA, pp. 421-430, 2001.

[9] V. Cardellini, E. Casalicchio, V. Grassi, S. Lannucci, F. Lo Presti, and R. Mirandola, "MOSES: A Framework for QoS Driven Runtime Adaptation of Service-Oriented Systems," *IEEE Transactions on*

Software Engineering, vol. 38, no. 5, pp. 1138-1159, 2012.

[10] V. Cardellini, E. Casalicchio, V. Grassi, F. Lo Presti, and R. Mirandola, "QoS-driven Runtime Adaptation of Service Oriented Architectures," *Proc. the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (SEC/SIGSOFT FSE2009)*, Amsterdam, The Netherlands, pp. 131-140, 2009.

[11] S. Deng, L. Huang, D. Hu, J. L. Zhao, and Z. Wu, "Mobility-Enabled Service Selection for Composite Services," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 394-407, 2016.

[12] 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.

[13] L. M. V. Gonzalez, L. Rodero-Merino, C. Juan, and M. A. Lindner, "A Break in the Clouds: Towards A Cloud Definition," *Computer Communication Review*, vol. 39, no. 1, pp. 50-55, 2009.

[14] R. Gulati, M. Sytch, and P. Mehrotra, "Breaking Up Is Never Easy: Planning for Exit in A Strategic Alliance," *California Management Review*, vol. 50, no. 4, pp. 147-163, 2008.

[15] Q. He, J. Han, F. Chen, Y. Wang, R. Vasa, Y. Yang, and H. Jin, "QoS-Aware Service Selection for Customisable Multi-Tenant Service-Based Systems: Maturity and Approaches," *Proc. the 2015 IEEE 8th International Conference on Cloud Computing*, pp. 237-244, 2015.

[16] Q. He, J. Han, Y. Yang, J. Grundy, and H. Jin, "QoS-Driven Service Selection for Multi-tenant SaaS," *Proc. the 2012 IEEE Fifth International Conference on Cloud Computing*, Honolulu, HI, USA, pp. 566-573, 2012.

[17] Q. He, J. Han, Y. Yang, J. Grundy, and H. Jin, "QoS-Driven Service Selection for Multi-Tenant SaaS," *Proc. the 5th IEEE International Conference on Cloud Computing*, pp. 566-573, 2012.

[18] Q. He, J. Han, Y. Yang, H. Jin, J.-G. Schneider, and S. Versteeg, "Formulating Cost-Effective Monitoring Strategies for Service-Based Systems," *IEEE Transactions on Software Engineering (TSE)*, vol. 40, no. 5, pp. 461-482, 2014.

[19] 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.

[20] Q. He, R. Zhou, X. Zhang, Y. Wang, D. Ye, F. Chen, J. Grundy, and Y. Yang, "Keyword Search for Building Service-Based Systems," *IEEE Transactions on Software Engineering*, 2016. DOI: DOI: 10.1109/TSE.2016.2624293

[21] Y. Li, J. Huai, T. Deng, H. Sun, H. Guo, and Z. Du, "QoS-aware Service Composition in Service Overlay Networks," *Proc. the IEEE International Conference on Web Services (ICWS2007)*, Salt Lake City, Utah, USA, pp. 703-710, 2007.

[22] Q. Liang, X. Wu, and H. C. Lau, "Optimizing Service Systems Based on Application-Level QoS," *IEEE Transactions on Services Computing*, vol. 2, no. 2, pp. 108-121, 2009.

[23] M. Martins, "China's Outward Strategic Alliances in European Union," 2016.

[24] T. H. Tan, M. Chen, J. Sun, Y. Liu, É. André, Y. Xue, and J. S. Dong, "Optimizing Selection of Competing Services with Probabilistic Hierarchical Refinement," *Proc. the 38th International Conference on Software Engineering*, pp. 85-95, 2016.

[25] F. Tao, D. Zhao, H. Yefa, and Z. Zhou, "Correlation-Aware Resource Service Composition and Optimal-Selection in Manufacturing Grid," *European Journal of Operational Research*, vol. 201, no. 1, pp. 129-143, 2010.

[26] I. Trummer, B. Faltings, and W. Binder, "Multi-Objective Quality-Driven Service Selection - A Fully Polynomial Time Approximation Scheme," *IEEE Transactions on Software Engineering*, vol. 40, no. 2, pp. 167-191, 2014.

[27] F. Wagner, A. Klein, B. Klöpper, F. Ishikawa, and S. Honiden, "Multi-Objective Service Composition with Time-and Input-Dependent QoS," *Proc. the 19th IEEE International Conference on Web Services*, pp. 234-241, 2012.

[28] J. Wang, J. Wang, B. Chen, and N. Gu, "Minimum Cost Service Composition in Service Overlay Networks," *World Wide Web*, vol. 14, no. 1, pp. 75-103, 2011.

[29] 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

[30] Q. Wu, Q. Zhu, and M. Zhou, "A Correlation-Driven Optimal Service Selection Approach for Virtual Enterprise Establishment," *Journal of Intelligent Manufacturing*, vol. 25, no. 6, pp. 1441-1453, 2014.

[31] 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, 2007.

[32] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality Driven Web Services Composition," *Proc. the 12th International Conference on World Wide Web (WWW2003)*, Budapest, Hungary, pp. 411-421, 2003.

[33] Y. Zhang, G. Cui, S. Deng, and Q. He, "Alliance-Aware Service Composition Based on Quotient Space," *Proc. the 23rd International Conference on Web Services*, pp. 340-347, 2016.

Yiwen Zhang received his PhD degree in management science and engineering in 2013 from the Hefei University of Technology. He is an associate professor in the School of Computer Science and Technology at Anhui University. His research interests include service computing, cloud computing, and e-commerce.

Guangming Cui received his bachelor degree in software engineeringin in 2015 and now is a master student in the School of Computer Scienceand Technology, Anhui University. His current research interests include service computing, cloud computing and evolutionary computation.

Shuiguang Deng is a full professor at the College of Computer Science and Technology in Zhejiang University. He received the BS and PhD both in Computer Science from Zhejiang University in 2002 and 2007, respectively. His research interests include Service Computing, Mobile Computing, and Business Process Management.

Feifei Chen received her PhD degree from Swinburne University of Technology, Australia in 2015. She is a lecturer at Deakin University. Her research interests include software engineering, cloud computing and green computing.

Yan Wang is currently an associate professor in the Department of Computing, Macquarie University, Sydney, NSW, Australia. His research interests cover trust management, services computing and social computing. He serves on the editorial board of several international journals, including the IEEE Transactions on Services Computing.

Qiang He received his first Ph. D. degree from Swinburne University of Technology (SUT), Australia, in 2009 and his second Ph. D. degree in computer science and engineering from Huazhong University of Science and Technology (HUST), China, in 2010. He is a senior lecturer at Swinburne University of Technology. More details about his research can be found at https://sites.google.com/site/heqiang/.