Personalized Quality Centric Service Recommendation

Yiwen Zhang[1], Xiaofei Ai[1], Qiang He[2](✉), Xuyun Zhang[3],
Wanchun Dou[4], Feifei Chen[5], Liang Chen[6], and Yun Yang[2]

[1] Anhui University, Hefei, China
zhangyiwen@ahu.edu.cn, aixiaofeiliujiajia@outlook.com
[2] Swiburne University of Technology, Melbourne, Australia
{qhe, yyang}@swin.edu.au
[3] University of Auckland, Auckland, New Zealand
xuyun.zhang@auckland.ac.nz
[4] Nanjing University, Nanjing, China
douwc@nju.edu.cn
[5] Federation University Australia, Ballarat, Australia
feifei.chen@federation.edu.au
[6] Sun Yat-Sen University, Guangzhou, China
chenliang6@mail.sysu.edu.cn

Abstract. The broad application of service-oriented architecture (SOA) has fueled the rapid growth of web and cloud services and service-based systems (SBSs). Tremendous web and cloud services have been deployed all over the world. Finding the right services becomes difficult and critical. Thus, service recommendation has become of paramount research and practical importance. Existing web service recommendation approaches employ utility functions or skyline techniques. However, those approaches have not addressed a critical and fundamental problem: how to recommend services according to a system engineer's quality constraints, e.g., response time, failure rate, etc. To address this issue, we first propose two basic personalized quality centric approaches for service recommendation, which employ the k-nearest neighbors and the dynamic skyline techniques respectively. To overcome the respective limitations of the two basic approaches, we propose two hybrid approaches, namely KNN-DSL and DSL-KNN. Extensive experiments are conducted on a real-world dataset to demonstrate the effectiveness and efficiency of our approaches.

Keywords: Service recommendation · QoS · Dynamic skyline · KNN

1 Introduction

The service-oriented architecture (SOA) allows complex software systems to be built by composing loosely coupled web services [1, 2]. The component services of such a service-based system (SBS) collectively realize the system functionality which is often offered as SaaS (Software-as-a-Service) in the cloud environment.

Figure 1 shows the process for building an example travel booking SBS that requires four services to perform four system tasks. As depicted, the process consists of

M. Maximilien et al. (Eds.): ICSOC 2017, LNCS 10601, pp. 528–544, 2017.
https://doi.org/10.1007/978-3-319-69035-3_39

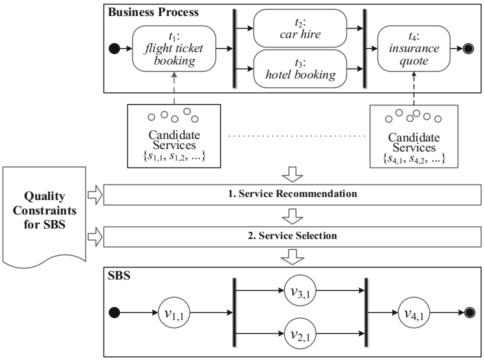

Business Process
t_1: flight ticket booking
t_2: car hire
t_3: hotel booking
t_4: insurance quote
Candidate Services $\{s_{1,1}, s_{1,2}, \ldots\}$
Candidate Services $\{s_{4,1}, s_{4,2}, \ldots\}$
Quality Constraints for SBS
1. Service Recommendation
2. Service Selection
SBS
$v_{1,1}$
$v_{3,1}$
$v_{2,1}$
$v_{4,1}$

Fig. 1. Process for building an SBS

two phases. The first phase is service recommendation where representative services are identified from the candidate services and recommended to the system engineer [3, 4]. The second phase is service selection where the system engineer selects one service from each set of recommended services to build the target SBS that fulfils the multi-dimensional constraints for the system quality, e.g., response time, failure rate, etc., and in the meantime achieves the optimization goal for the system quality. This is an NP-complete problem often referred to as quality-aware service selection [1, 2, 5, 6].

The development and popularity of e-business, e-commerce, especially the pay-as-you-go business model promoted by cloud computing have fueled the rapid growth of services, indicated by the statistics published by programmableweb.com, an online web service repository. This makes the quality-aware service selection problem intractable. In recent years, a lot of efforts have been devoted to reducing the complexity of the problem of quality-aware service selection through service recommendation [3, 4, 7–9]. Recommending appropriate services reduces the search space of the NP-complete service selection problem. In this way, a system engineer does not have to enumerate all candidate services and their possible combinations. **The key issue here is to identify appropriate candidate services that are most likely to fulfil the system engineer's quality constraints**. Unfortunately, none of the existing web service recommendation approaches has properly addressed this issue. There are three major categories of service recommendation approaches, utility-based, skyline-based [3, 4] and collaborative filtering (CF) based [10, 11]. Utility-based and skyline-based approaches do not properly take into account system engineers' personalized quality constraints. CF-based service recommendation aims to predict the quality of web services. Thus, although labeled as a recommendation approach [10–14], CF-based service recommendation is in fact a prediction approach, not a recommendation approach.

In this paper, we first propose two basic approaches for **personalized quality centric service recommendation**, one based on the KNN (k-nearest neighbors) technique and the other based on the DSL (dynamic skyline) technique. The KNN approach models the service recommendation problem as a nearest neighbor search problem. Given a system engineer's quality constraints, it finds a set of *k suitable services* whose quality values are most similar to the quality constraints. The DSL-based approach models the service recommendation problem as a dynamic skyline query problem. It attempts to find *representative services* that are not dominated by any other services with respect to system engineer's quality constraints.

The main contributions of this research are:

1. It is the first attempt to model and solve the critical problem of personalized quality centric service recommendation. The resolution of this problem naturally complements the existing quality-aware service selection approaches by recommending appropriate and representative services to system engineers.
2. Two basic approaches are proposed for personalized quality centric service recommendation, one finding suitable services while the other finding representative services.
3. Two hybrid approaches, KNN-DSL and DSL-KNN, are proposed to overcome the limitations of the two basic approaches.
4. Extensive experiments are conducted to evaluate the effectiveness and efficiency of the proposed approaches using a dataset that contains the quality information about 2,507 real-world web services.

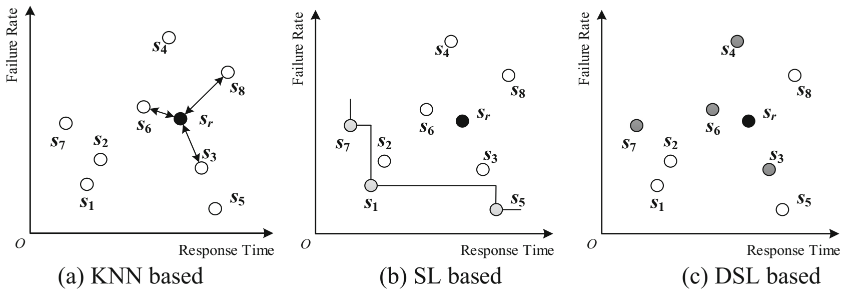
The rest of this paper is organized as follows. Section 2 analyzes the research problem. Section 3 describes the basic and the hybrid approaches. Section 4 evaluates the proposed approaches. Section 5 reviews the related work. Section 6 concludes the paper and points out future work.

2 Problem Analysis

This section analyzes the research problem with the travel booking SBS presented in Fig. 1. For each of the four system tasks, i.e., t_1, t_2, t_3 and t_4, there is a set of candidate services that can perform the system task but with potentially different quality values, e.g., response time and failure rate. The system engineer needs to select one service from each set of candidate services for building the SBS, i.e., Phase 2. The selected services must collectively fulfil the quality constraints for the system. This problem is commonly known as NP-complete. As the scenario scales up, it becomes intractable. A promising approach for simplifying this problem is, from each set of candidate services, to identify services that are more likely to fulfil the quality constraints for the system, to recommend to the system engineer for selection, i.e., Phase 1. The search space for the problem can be significantly reduced.

To facilitate service recommendation, the global quality constraints for the system must be decomposed into local quality references for individual system tasks. Alrifai et al. proposed an approach in [15] that is widely employed to decompose global quality constraints for a system into local quality references. Given a set of quality constraints for the system, e.g., 800 ms for response time and 1% for failure rate (at least 99 out of 100 service requests must be handled properly), Alrifai et al.'s approach can decompose it into four sets of local quality constraints, one for each of the four tasks, e.g., 120 ms for the response time of performing t_1 and 0.9% for its failure rate.

According to the local quality references for each system task, appropriate services can be identified and recommended to the system engineer. A straightforward approach is to model this problem as a nearest neighbors search problem, as presented in Fig. 2. In Fig. 2, points s_1, …, s_8 represent the eight candidate services for t_1 and **point s_r represents the *dummy reference service*** with quality references for the response time and failure rate of this task. As indicated in Fig. 2(a), those services that are close



(a) KNN based (b) SL based (c) DSL based

Fig. 2. Approaches for service recommendation

(similar) to s_r in both dimensions, e.g., s_3, s_6 and s_8, are more suitable than those far away from s_r, e.g., s_1, s_2, s_4, s_5 and s_7. Service s_1, in particular, has lower response time and failure rate than most services. Services with such outstanding quality advantages are usually very expensive. The selection of s_1 will most likely violate the cost constraint for the system. In a 3-dimensional space with cost as the third dimension, s_1 might be one of the services most similar to s_r. However, in Fig. 2(a), s_1 is not considered suitable with respect to s_r.

Therefore, we can identify a set of k nearest neighbor services that are most similar to s_r in all quality dimensions. However, there is an inherent limitation to this approach in certain scenarios – the services identified by this approach might not be representative in all quality dimensions. Take Fig. 2(a) for example. Suppose that s_3, s_6 and s_8 are identified as the services most similar to s_r. Services s_6 has the lowest response time and is thus considered the most representative service in terms of low response time among s_3, s_6 and s_8. Service s_3 has the lowest failure rate and is thus considered the most representative in terms of low failure rate among s_3, s_6 and s_8. Now we take a look at s_8. It is not representative in either low response time or low failure rate. Therefore, considering only response time and failure rate, s_3 and s_6, are more appropriate than s_8. According to the definition of skyline [3], s_8 is dominated by s_3 and s_6 because s_8 is no better than s_3 and s_6 in any quality dimensions.

An approach for solving the above non-representativeness problem is to calculate the skyline [3], as presented in Fig. 2(b). Given a set of services, the skyline calculation identifies the set of services that are not dominated by any other services. In the case of Fig. 2(b), the skyline services include s_1, s_5 and s_7, which are superior to the other services in terms of both low response time and low failure rate. They are represented by light grey circles. Hence, they are considered more representative than the other services. However, s_1, s_5 and s_7 are not close to s_r at all. In fact, they are further away from s_r compared to the other services. This is because the skyline calculation uses the origin O as the reference point. As a result, the skyline services are not suitable with respect to s_r. To address this issue, we need to identify the *dynamic skyline services* [16], as presented in Fig. 2(c). The dynamic skyline service calculation uses a given point rather than the origin as the reference point and identifies the dynamic skyline services with respect to the distances to the reference point. In Fig. 2(c), given s_r as the reference point, the dynamic skyline services include s_3, s_4, s_6 and s_7, represented by heavy dark circles. Among s_3, s_4, s_6 and s_7, s_4 is the most representative in terms of response time with respective to s_r because its response time is the most similar to s_r's. Similarly, s_7 is the most representative in terms of failure rate. Services s_3 and s_6 are two tradeoff services between s_4 and s_7 because they have a response time more similar to s_r than s_4 and failure rate more similar to s_r than s_7.

The KNN-based approach (referred to as KNN in short hereafter) and the DSL-based approach (referred to as DSL in short hereafter) attempt to identify services with respect to s_r from two different perspectives. The former identifies suitable services that are most similar to s_r and the latter finds the representative services that are not dominated by any other services with respect to s_r.

In this paper, we use response time and failure rate in the discussion. More quality constraints can be taken into account in a similar manner, which will transform the two-dimensional space into a multi-dimensional one.

3 Recommendation Approaches

3.1 Basic Approaches

In this section, we present the basic KNN and DSL approaches.

KNN Approach. Given a set of candidate services $S = \{s_1, \ldots, s_n\}$, each with p-dimensional quality values, and a reference service s_r with p-dimensional quality references, the candidate services and s_r are first mapped to a p-dimensional space, one dimension for each of the p quality dimensions. For numerical quality dimensions, such as response time, failure rate, reliability, etc., the mapping process is straightforward. To accommodate non-numerical quality dimensions, such as reputation that are expressed by a rating selected from {high, medium, low}, the method discussed in [17] is adopted in this research. Based on a pre-defined hierarchical structure of all possible values, each level of the hierarchy is associated with a numerical value, for example, 3 for high, 2 for medium and 1 for low.

Given a k value, KNN identifies the top k services from S that are most similar to s_r, based on a measure of similarity in the p-dimensional space. To evaluate the similarity between each of the candidate services, $s_i \in S$, $1 \leq i \leq n$, and s_r, we first normalize the quality values of s_i, $1 \leq i \leq n$, as well as s_r, with the min-max normalization technique, which has also been employed by many other researchers [17, 18]:

$$\tilde{q}_p(s_i) = \begin{cases} \frac{q_p^{\max}(S) - q_p(s_i)}{q_p^{\max}(S) - q_p^{\min}(S_i)} & \text{if } q_p^{\max}(S) \neq q_p^{\min}(S) \\ 1 & \text{if } q_p^{\max}(S) = q_p^{\min}(S) \end{cases} \tag{1}$$

where $q_p(s_i)$ is the p^{th} dimensional quality value of s_i, $q_p^{\max}(S)$ and $q_p^{\min}(S)$ are the maximum and minimum values, respectively, for the p^{th} quality dimension among all services in S.

After the normalization, the similarity between a candidate service $s_i \in S$ and the reference service s_r can be evaluated by the Euclidean distance between s_i and s_r:

$$d(s_i, s_r) = \sqrt{\sum\nolimits_{j=1}^{p} \left(q_j(s_i) - q_j(s_r)\right)^2} \tag{2}$$

Based on Eq. (2), KNN employs Algorithm 1 to identify k services that are most similar to s_r. Algorithm 1 first calculates the Euclidean distance between each service in S and s_r (lines 3–5). It then sorts the services by their distances to s_r in a descending order (line 6). Finally, the algorithm returns the top k services from S as the recommendation results.

KNN requires the k value to be pre-specified, which is a domain-specific parameter. Different applications usually have their own characteristics, and hence inherit different optimal k values. On one hand, an overly small k value cannot ensure that adequate similar services be identified for recommendation. On the other hand, an overly large k value will include dissimilar services in the final recommendation results and will consequently decrease the recommendation accuracy. Therefore, the k value should be set domain-specifically based on experiences and/or experiments. In Sect. 4, we experimentally study the impact of k on recommendation accuracy.

The complexity of Algorithm 1 relies on the employed sorting algorithm. Here, we use the complexity of comparison sort algorithms in the worst-case scenario, i.e., $O(nlogn)$. Thus, Algorithm 1 runs in $O(np + nlogn)$.

DSL Approach. Given a set of points in a p-dimensional space, the skyline calculation is to find the points that are not dominated by any other points. A point s_i dominates another point s_j, if s_i is better than or equal to s_j in all dimensions and strictly better in at least one dimension. In the context of this research, the dominance relations between two services is defined based on their p-dimensional quality values:

Definition 1. Dominance: Given two services, $s_i, s_j \in S$, characterized by p-dimensional quality values, s_i dominates s_j, denoted by $s_i \rhd s_j$, iff s_i is as good as or better than s_j in all quality dimensions and better in least one quality dimension, i.e., $\forall p \in [1, n] : q_p(s_i) \leq q_p(s_j)$ and $\exists p \in [1, n] : q_p(s_i) < q_p(s_j)$.

Based on Definition 1, we formally define the concept of skyline services:

Definition 2. Skyline services: The skyline of S, denoted by S_{SL}, consists of the set of services in S that are not dominated by any other services in S, i.e., $S_{SL} = \left\{ s_i \in S \mid \neg \exists s_j : s_i \rhd s_j \right\}$. The services in S_{SL} are referred to as skyline services.

Generally, the skyline services have the best quality according to their absolute quality values in each quality dimension. However, as discussed in Sect. 2, given a reference service s_r, DSL needs to identify the dynamic skyline services in S. This can be achieved in a new p-dimensional space based on the original space.

Algorithm 1 . KNN

Input:
S - Set of candidate services
s_r - Reference service
k - number of services needed
Output:
results - k services most similar to s_r

```
1   Begin
2     result ← null;
3     for each s in S do
4       | s.distance ← d(s, s_r);
5     end for
6     S.sortByDistance();
7     results ← S.takeTop(k);
8     return results;
9   End
```

First, each service $s \in S$ is mapped to a service $s' = \left(f_1(s), \ldots, f_p(s)\right)$, where $f_j(s) = \left|q_j(s_r) - q_j(s)\right|$, $1 \le i \le n$. Then, the dynamic skyline of S with respect to functions $f_1, \ldots, f_p$, is obtained by calculating the ordinary skyline in the transformed p-dimensional space with s_r as the origin. Accordingly, dynamic dominance is defined as:

Definition 3. Dynamic dominance: Given two services, $s_i, s_j \in S$, characterized by p-dimensional quality values, and a reference service s_r, s_i dynamically dominates s_j with respect to s_r, denoted by $s_i \succ s_j$, iff $\forall p \in [1, n] : \left|q_p(s_r) - q_p(s_i)\right| \le \left|q_p(s_r) - q_p\left(s_j\right)\right|$ and $\exists p \in [1, n] : \left|q_p(s_r) - q_p(s_i)\right| < \left|q_p(s_r) - q_p\left(s_j\right)\right|$.

Based on Definition 3, we formally define the concept of dynamic skyline services:

Definition 4. Dynamic skyline services: The dynamic skyline of S, denoted by S_{DSL}, consists the services that are not dynamically dominated by any other services, with respect to a given reference service s_r, i.e., $S_{DSL} = \left\{s_i \in S \mid \neg\exists s_j : s_j \triangleright s_i\right\}$. The services in S_{DSL} are referred to as dynamic skyline services.

Figure 3 illustrates the calculation of the dynamic skyline based on Fig. 2(c). First, the original space is transformed into a new one with s_r as the new origin and the absolute distances to s_r as the mapping functions. Then, $s_1, s_2, s_3, s_4, s_5, s_6, s_7$ are mapped into the new space where they are denoted by s_1', s_2', s_3', s_4', s_5', s_6' and s_7'. Service s_8 is already in the first quadrant of the new space. After the mapping, the location of $s_8^{'}$ is exactly the same as s_8 and thus is omitted in Fig. 3. Having mapped all the services into the new space, where they are collectively referred to as *S'*, the calculation of S_{DSL} is equivalent to the calculation of $S_{SL}^{'}$ in the new space. DSL employs Algorithm 2 to calculate the service skyline S_{SL} of a set of candidate services S. It iterates through all services in S (line 4). In each iteration, it selects one service s from S and checks if any other services in S dominate s (lines 5–11). If none, the algorithm includes s in the service skyline S_{SL} (lines 12–14). After processing all the services in S, it returns S_{SL}, i.e., the service skyline that consists of all the skyline services. As presented in Fig. 3, the algorithm returns $S_{SL}' = \left\{s_3', s_4', s_6', s_7'\right\}$ as the

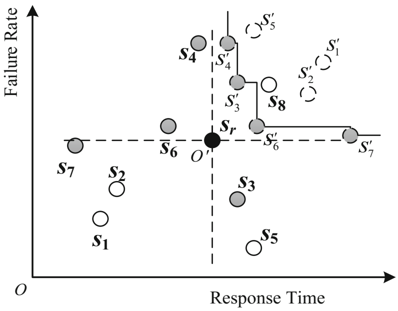


Fig. 3. Identification of dynamic skyline services.

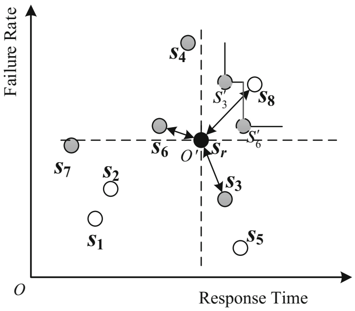


Fig. 4. Recommendation with KNN-DSL

skyline services in the new space. Accordingly, we can determine that $S_{DSL} = \{s_3, s_4, s_6, s_7\}$. As discussed in Sect. 2, s_3 and s_7 are closest to s_r in terms of response time and failure rate respectively. In the meantime, s_4 and s_6 are considered tradeoffs between s_3 and s_7.

Algorithm 2 contains two loops, one nested in the other. Let n be the number of nodes in S. The time complexity of Algorithm 2 is $O(n^2)$.

3.2 Hybrid Approaches

KNN and DSL have respective limitations. This section presents two hybrid approaches, KNN-DSL and DSL-KNN, that overcome those limitations.

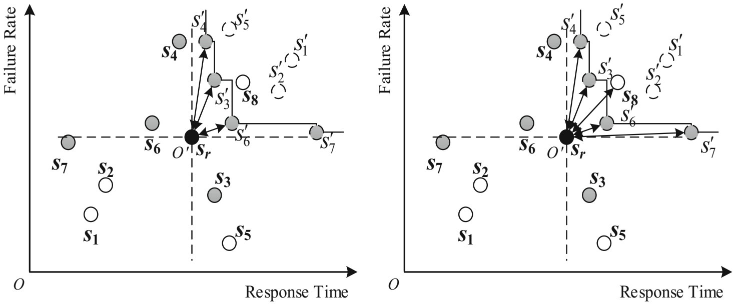
KNN-DSL. Given a reference service s_r, DSL identifies representative services. However, it sacrifices the similarity between the identified services and s_r. Take Fig. 3 for example, where $S_{DSL} = \{s_3, s_4, s_6, s_7\}$. In terms of standardized Euclidean distance, s_3 and s_6 are the closest to s_r. In addition, they both belong to the results returned by KNN when $k \geq 2$, as shown in Fig. 2(a). Services s_4 and s_7, on the other hand, are not the next services that are closest to s_r. Compared with s_4 and s_7, s_8 is closer to s_r. In addition, s_2 is closer to s_r than s_4. Thus, some dynamic skyline services should not be recommended as they are dissimilar to s_r.

To address this issue, we propose KNN-DSL, an approach that combines the advantages of KNN and DSL. Given a set of candidate services S and a reference service s_r, it first employs KNN to identify k services most similar to s_r, denoted by S_{KNN}. Then, it calculates the dynamic skyline of S_{KNN}, denoted by $S_{KNN\text{-}DSL}$, using DSL. In this way, KNN-DSL identifies those services that are similar to s_r and, in the meantime, representative with respect to s_r. Figure 4 shows an example based on Fig. 2(a). Suppose $k = 3$ for KNN. First, s_3, s_6 and s_8 are identified as the three services that are most similar to s_r. Then, from s_3, s_6 and s_8, s_3 and s_6 are identified as the dynamic skyline services. The limitation to this approach is that the number of services eventually identified is lower than or equals to k, i.e., $|S_{KNN\text{-}DSL}| \leq k$. Thus, KNN-DSL does not ensure a specific number of services in its recommendation results.

Algorithm 2 . SL

```
Input:
S - Set of candidate services
Output:
$S_{SL}$ - skyline of S
Begin
    $S_{SL}$ ← null;
    skyline ← true;
    for each $s_i$ in S do
        skyline ← true;
        for each $s_j$ in S do
            if $s_j \triangleright s_i$ then
                skyline ← false;
                break;
            end if
        end for
        if skyline = true
            $S_{SL}$ ← $S_{SL}$ + $s_i$;
        end if
    end for
    return $S_{SL}$
End
```

DSL-KNN Approach. To tackle the limitation of KNN-DSL, we propose DSL-KNN, which combines DSL and KNN to recommend services. Given S and s_r, it first identifies the dynamic skyline services, denoted by S_{DSL}. Then, from S_{DSL}, it identifies k services that are the closest to s_r, denoted by $S_{DSL-KNN}$. If $|S_{DSL\text{-}KNN}| < k$, DSL-KNN continues to find k - $|S_{DSL\text{-}KNN}|$ more services that are closest to s_r to ensure a total of



(a) $k = 3$ (b) $k = 5$

Fig. 5. Recommendation with DSL-KNN

k services in its recommendation results. In this way, the dynamic skylines services are always selected first to ensure the representativeness of some of the recommended services. Figure 5(a) and (b) demonstrate this approach based on Fig. 3 with k = 3 and k = 5 for KNN respectively. In Fig. 5, DSL-KNN first identifies the dynamic skyline services with respect to s_r, i.e., $S_{DSL} = \{s_3, s_4, s_6, s_7\}$. Then, given k = 3, it identifies s_3, s_4 and s_6 from S_{DSL} as the services that are the most similar to s_r. Given k = 5, the approach will first select all the services in S_{DSL}, and then select s_8 as the fifth service in addition to S_{DSL} because it is the service that is the closest to s_r among the rest of the services, i.e., s_1, s_2, s_5 and s_8.

Both KNN-DSL and DSL-KNN employ KNN and DSL. Thus, their complexity are both $O(n^2 + np + nlogn) = O(n^2 + np)$.

4 Experimental Evaluation

This section evaluates the proposed approaches through comparison with three existing representative approaches in their effectiveness (measured by recommendation accuracy) and efficiency (measured by computation time).

4.1 Experiment Setup

The experiments were conducted on a publicly available real-world dataset named QWS [19], which has been widely used [6, 9, 17, 20]. We have implemented the four **personalized quality centric approaches** for service recommendation proposed in Sects. 3. For comparison, we have implemented three existing representative **non-personalized quality centric approaches** for service recommendation:

- **RS:** This approach randomly selects k services from the candidate services.
- **UF:** This approach selects k services with the highest utility values, calculated with the widely used utility function [1, 6, 15, 17].
- **SL:** This approach identify skyline services [3, 7, 8].

Metrics for effectiveness. Given a system engineer's quality preferences represented by s_r, personalized quality centric service recommendation aims to find services whose quality is (1) similar to s_r; and (2) representative with respect to s_r. Accordingly, we evaluate the recommendation accuracies of the comparing approaches, which are measured by three metrics, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Non-Dominance Rate (NDR).

MAE is defined as:

$$\mathrm{MAE} = \sum\nolimits_{i=1}^{|R|} \sqrt{\sum\nolimits_{j=1}^{p} \left(q_j(s_i) - q_j(s_r)\right)^2} / |R| \tag{3}$$

where R is the set of services returned by the recommendation approach, $q_j(s_i)$ and $q_j(s_r)$ are the j^{th} dimensional quality value of $s_i \in R$ and s_r. An MAE value indicates the average difference between the recommended services and s_r in their p-dimensional quality. **A low MAE value indicates high recommendation accuracy.**

RMSE is defined as:

$$\mathrm{RMSE} = \sqrt{\sum\nolimits_{i=1}^{|R|} \sum\nolimits_{j=1}^{p} \left(q_j(s_i) - q_j(s_r)\right)^2 / |R|} \tag{4}$$

During the calculation of RMSE, the individual differences between the recommended services and s_r are each squared and then averaged over R. Similar to MAE, **a low RMSE indicates high recommendation accuracy.**

NDR is defined as:

$$NDR = |R_{DSL}| / |R| \tag{5}$$

where R_{DSL} is the set of dynamic skyline services in R. NDR measures the representativeness of the results. **A high NDR indicates high recommendation accuracy.**

Metric for efficiency. In order to evaluate the efficiency of the proposed approaches, we measure their computational overheads.

To simulate different recommendation scenarios, we have conducted three series of experiments, namely series A, B and C. Table 1 presents the parameter settings. In each experiment, we randomly select n services from the QWS dataset as the candidate services, and another one as s_r. Then, we run the comparing approaches to identify the services to recommend. All approaches are implemented in Java using JDK 1.8. All

Table 1. Experiment parameter settings.

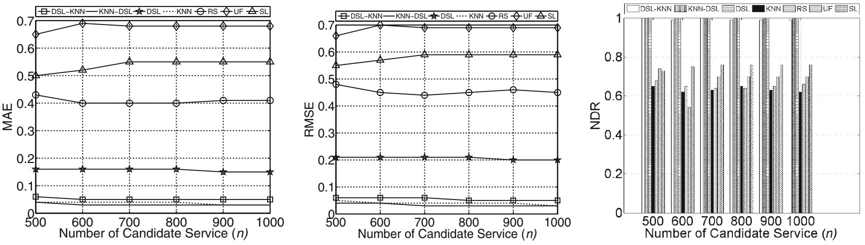
Parameter	Experiment series		
	A	B	C
Number of candidate services (n)	500 to 1000	1000	1000
Number of services to recommend (k)	5	5 to 12	5
Number of quality dimensions (q)	4	4	2 to 9

experiments are conducted on a machine with Intel i7-4790 CPU 3.60 GHz and 16 GB RAM, running Windows 10 x64 Professional.

4.2 Experimental Results

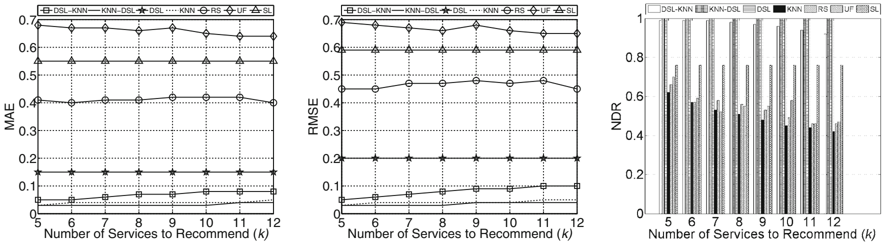
Effectiveness. Figure 6 shows the impact of the number of services in S (denoted by n) on the recommendation accuracies obtained by the approaches. Figure 6(a) and (b) show that KNN-DSL, KNN and DSL-KNN obtain the best recommendation accuracies, measured by their MAE and RMSE values which are much lower than the other four approaches. DSL obtains the fourth best recommendation accuracy overall. This indicates the importance of considering s_r during service recommendation. Interestingly, we observe that UF achieves the worst recommendation accuracy. It finds the services with the best overall quality, which however, are not necessarily preferable to the system engineer, indicated by its extremely high MAE values. Figure 6(a) and (b) also show that the increase in n increases the recommendation accuracies of our approaches, i.e., DSL-KNN, KNN-DSL, KNN and DSL. As n increases, there are more candidate services for the approaches to choose from, increasing the possibility of finding suitable services. Figure 6(c) shows that the services recommended by KNN-DSL and DSL are the most representative, with slight advantages over DSL-KNN. The services recommended by the other four approaches, including KNN, are significantly less representative. It clearly shows the effectiveness of the DSL operator in finding representative services.

Figure 7 demonstrates the impact of the number of services to recommend (denoted by k). Figure 7(a) and (b) show that KNN-DSL, again, achieves the best recommendation results overall in experiment series B, KNN the second, DSL-KNN the third and DSL the fourth. KNN-DSL is the winner because its KNN operator ensures the similarity between the selected k services and s_r with its KNN operator, and then further prunes some of the k services whose quality are relatively dissimilar to s_r with its DSL operator. KNN seconds to KNN-DSL because the non-dynamic-skyline services in its recommendation results lower its recommendation accuracy. Similar to KNN-DSL, DSL-KNN also employs two operators, DSL then KNN. Its DSL operator selects the dynamic skyline services, which are representative however not necessarily very similar to s_r. Thus, its MAE values are not as low as KNN and DSL-KNN. RS, UF and



(a) MAE (b) RMSE (c) NDR

Fig. 6. Impact of parameter n on accuracy (experiment series A)

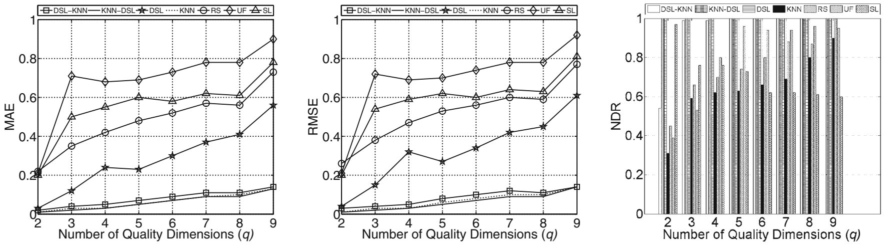


(a) MAE (b) RMSE (c) NDR

Fig. 7. Impact of parameter *k* on accuracy (experiment series B)

SL are again no match for our approaches, demonstrated by their significantly higher MAE and RMSE values. Figure 7(a) and (b) also show that the increase in the *k* value leads to slight increases in the MAE and RMSE values obtained by KNN-DSL, DSL-KNN and SL. As *k* increases, services that are not quite similar to s_r are also included in the recommendation results, which lowers the recommendation accuracy. The increase in *k* has no impact on DSL because DSL recommends only the dynamic skyline services which are irrelevant to the *k* value. Figure 7(c) shows that the recommendation results of KNN-DSL and DSL are all representative. This is because they both employ the DSL operator to ensure the representativeness of the recommendation results. DSL-KNN achieves the third highest representativeness in the recommendation results. Its DSL operator selects only the representative services, i.e., the dynamic skyline services. However, as *k* increases and exceeds the number of dynamic skyline services identified by its DSL operator, its KNN operator has to include some non-dynamic-skyline services in the recommendation results. These services reduce the overall representativeness of its recommendation results.

Figure 8 shows the impact of the number of quality dimensions (denoted by *q*). Figure 8(a) and (b) show that KNN-DSL, for the third time, achieves the best recommendation accuracy, outperforming KNN and DSL-KNN, which achieve the second best and third best recommendation accuracy respectively. DSL achieves the fourth highest - however significantly lower - recommendation accuracy. We can also observe that the increase in *q* decreases the recommendation accuracies achieved by our approaches, indicated by their increasing MAE and RMSE values. The increase in



(a) MAE (b) RMSE (c) NDR

Fig. 8. Impact of number of quality constraints (*q*) on accuracy (experiment series C)

q mainly impacts the skyline calculation and dynamic skyline calculation. A large q makes it harder for one service to dominate or dynamically dominate the other, resulting in a large number of skyline services and dynamic skyline services. As q increases, KNN-DSL manages to maintain its slight advantage over KNN and DSL-KNN in most cases. Its KNN operator ensures that the quality of the selected k services are the closest to s_r. KNN achieves the second best recommendation accuracy because it also ensures the similarity between the recommendation results and s_r. DSL-KNN loses to KNN-DSL and KNN on average because some services that are similar to s_r are pruned by its DSL operator for being dominated by other services. Figure 8(c) shows that KNN-DSL and DSL consistently obtain highly representative recommendation results. DSL-KNN also obtains representative recommendation results except when $q = 2$. We investigated this interesting phenomenon and found out that when $q = 2$, its DSL identified only a few dynamic skyline services, requiring its KNN operator to find some services that are similar to s_r but are not representative. This lowers the overall representativeness of its recommendation results.

Efficiency. Figure 9 shows the computation times taken by the approaches in experiment series A. As demonstrated, the seven approaches can be categorized into two groups according to their scalability to n, the *slow approaches*, including SL, DSL and DSL-KNN, and the *fast approaches*, including RS, KNN, KNN-DSL and UF. The slow approaches share one thing in common - they have to identify the skyline services or the dynamic skyline services from a large number of candidate services, which is not required for the fast approaches. KNN-DSL, which also employs a DSL operator like DSL and DSL-KNN, takes much less time to complete. It is because its KNN operator selects only k services for its DSL operator to process further. Given that k is usually a small number, its DSL operator does not need take long to finish. The slow approaches take significantly more time than the fast approaches. However, they are in fact not quite slow - they require slightly more than 70 ms to process 1,000 candidate services. In addition, their computation times are roughly linear to n, which indicates high scalability. We believe their efficiency are acceptable in most, if not all, real-world applications.

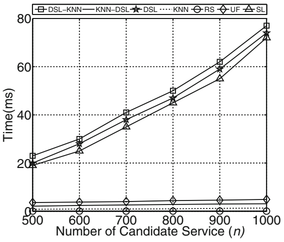


Fig. 9. Impact of parameter n on efficiency (experiment series A)

Figure 10 shows the computation times taken by different recommendation approaches in experiment series C. We observe differences in their computation times similar to Fig. 9 between the slow and fast approaches. The fast approaches take less than 10 ms on average to complete. Their computation times are not significantly impacted by the increase in q, which demonstrates their high

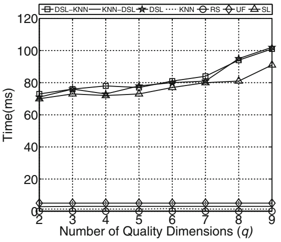


Fig. 10. Impact of parameter q on efficiency (experiment series C)

scalability to q. On the other hand, the slow approaches taking 60 ms to 100 ms to complete, much longer than fast approaches, are still fast enough for most, if not all, real-world applications.

Please note that the computation times of the recommendation approaches in experiment series B, where k varies, are not presented. The k value means to select k services from the processed candidate services. For example, the KNN operator, which employs Algorithm 1 introduced in Sect. 3.1, simply picks the top k services from a set of services sorted by their distance to s_r. UF selects k services in a similar way. Such operations have a complexity of $O(1)$ and do not impact the computation times of the approaches.

4.3 Discussion

Table 2 presents the average MAE, RMSE and NDR values obtained by the recommendation approaches in each experiment series. The lowest and second lowest MAE, RMSE, as well as the highest and the second highest DNR, achieved in each experiment series are highlighted by dark grey and light grey respectively. We can see that on average, KNN-DSL outperforms the other approaches. This indicates that KNN-DSL obtains the most suitable and most representative recommendation results with respect to s_r. In the meantime, KNN-DSL is also highly efficient, as illustrated by Figs. 9 and 10. Thus, KNN-DSL is the best approach for service recommendation in most real-world applications. However, KNN-DSL has a limitation - it does not ensure a certain number of services in its recommendation results. It might eventually finds fewer than k services. Thus, if a number of k services in the recommendation results is mandatory, KNN-DSL is not a proper choice. In those cases, DSL-KNN and KNN are preferable. Table 2 shows that KNN achieves the second lowest MAE and RMSE values, indicating that its recommendation results are close to s_r. However, its advantage over DSL-KNN is only marginal. In addition, its NDR values are much lower than those of DSL-KNN, meaning that its recommendation results are not as representative. Therefore, in most cases, DSL-KNN is a better choice than KNN k services are mandatory. DSL achieves the highest NDR in all three experiment series. However, its MAE and RMSE values are much higher than KNN-DSL, DSL-KNN and KNN. In addition, it does not guarantee the number of services in its recommendation results. As a result, DSL is not the first choice in any envisaged real-world applications.

Table 2. Average performance (MAE/RMSE/NDR)

Experiment series	A			B			C		
KNN-DSL	0.03	0.03	1.00	0.03	0.03	1.00	0.061	0.06	1.00
DSL-KNN	0.05	0.06	0.99	0.07	0.08	0.97	0.08	0.08	0.94
KNN	0.04	0.04	0.06	0.04	0.04	0.50	0.064	0.07	0.65
DSL	0.16	0.21	1.00	0.15	0.20	1.00	0.28	0.33	1.00
RS	0.41	0.46	0.65	0.41	0.47	0.54	0.48	0.52	0.76
UF	0.68	0.69	0.68	0.66	0.67	0.55	0.69	0.69	0.81
SL	0.54	0.58	0.75	0.55	0.59	0.76	0.56	0.58	0.71

5 Related Work

Quality-aware service recommendation is a critical issue in service-oriented computing. Utility-based recommendation [1, 3, 6, 15, 17] and skyline-based recommendation [3] are currently the two most popular approaches for quality-aware service recommendation.

Utility-based recommendation is very straightforward. The utility value of a service indicates how good its overall p-dimensional quality is in comparison with the other candidate services in S - the higher, the better. The utility calculation for a service s_i goes through two phases. First, a utility value is calculated for each of its quality dimensions. Then, the utility of service s_i is calculated by summing its utility values in all p quality dimensions. Given a set of candidate services, utility-based recommendation selects the services with the highest utility values. This approach has been widely employed [1, 3, 6, 15, 17].

The other popular approach for service recommendation is the skyline-based service recommendation. Its process can be found in Sect. 2. It was first employed by Alrifai et al. to select representative services that are not dominated by any other candidate services [3]. Since then, many researchers have attempted to improve the skyline-based service recommendation approach to accommodate more sophisticated environments. To name few, Benouaret et al. propose a concept named alpha-dominant service skyline to address two issues in the approach proposed by Alrifai et al. [7]. First, it treats services with a bad compromise between different quality dimensions in a fairer manner. Second, it improves the efficiency of skyline calculation. Benouaret et al. have also proposed an improved skyline-based approach for service recommendation that handles services' probabilistic quality values [8].

The common and critical limitation of the utility-based and the skyline-based service recommendation is the lack of consideration for system engineers' quality constraints, which have always been a fundamental and critical issue in quality-aware service selection [1, 2, 6, 9, 17, 20], as well as skyline-based service composition [21–23]. This renders the utility-based and skyline-based service recommendation obsolete. As demonstrated in Sect. 4, their recommendation results are neither suitable nor representative.

There is a large body of approaches labeled service recommendation approaches [10–14]. However, aiming to predict the quality values of services, those approaches are not designed for service recommendation.

Our approaches address the limitation of existing recommendation approaches by centering system engineers' quality constraints in the recommendation. By combining the KNN and DSL techniques, our approaches can efficiently recommend suitable and representative services with respect to system engineers' quality preferences.

6 Conclusion and Future Work

In this paper, we first proposed two basic approaches, named KNN and DSL, for personalized quality centric service recommendation, based on k-nearest neighbors and dynamic skyline techniques, respectively. Then, to overcome their limitations, we proposed two hybrid approaches, named KNN-DSL and DSL-KNN. Finally, we presented extensive experiment results to demonstrate their effectiveness and efficiency.

In the future, we will combine the proposed approaches with approaches for service compositions to facilitate an effective and efficient personalized quality centric process for building service-based systems.

Acknowledgment. This work is supported by the National Key Technology R&D Program (No. 2015BAK24B01), the General Research for Humanities and Social Sciences Project of Chinese Ministry of Education (No. 15YJAZH112), the Educational Commission of Anhui Province of China (No. KJ2016A038), and the Australian Research Council Projects DP150101775 and LP130100324. Qiang He is the corresponding author of this paper.

References

1. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Trans. Softw. Eng. **33**(6), 369–384 (2007)
2. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-Aware middleware for web services composition. IEEE Trans. Softw. Eng. **30**(5), 311–327 (2004)
3. Alrifai, M., Skoutas, D., Risse, T.: Selecting skyline services for QoS-based web service composition. In: 19th International Conference on World Wide Web, pp. 11–20 (2010)
4. Yu, Q., Bouguettaya, A.: Efficient service skyline computation for composite service selection. IEEE Trans. Knowl. Data Eng. **25**(4), 776–789 (2013)
5. He, Q., Yan, J., Jin, H., Yang, Y.: Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction. IEEE Trans. Softw. Eng. **40**(2), 192–215 (2014)
6. Trummer, I., Faltings, B., Binder, W.: Multi-objective quality-driven service selection - a fully polynomial time approximation scheme. IEEE Trans. Softw. Eng. **40**(2), 167–191 (2014)
7. Benouaret, K., Benslimane, D., Hadjali, A.: On the use of fuzzy dominance for computing service skyline based on QoS. In: 9th IEEE International Conference on Web Services, pp. 540–547 (2011)
8. Benouaret, K., Benslimane, D., Hadjali, A.: Selecting skyline web services from uncertain QoS. In: 9th IEEE International Conference on Services Computing, pp. 523–530 (2012)
9. Tan, T.H., Chen, M., Sun, J., Liu, Y., André, É., Xue, Y., Dong, J.S.: Optimizing selection of competing services with probabilistic hierarchical refinement. In: 38th International Conference on Software Engineering, pp. 85–95 (2016)
10. Zheng, Z., Lyu, M.R.: Collaborative reliability prediction of service-oriented systems. In: 32nd ACM/IEEE International Conference on Software Engineering, pp. 35–44 (2010)
11. Zheng, Z., Lyu, M.R.: Personalized reliability prediction of web services. ACM Trans. Softw. Eng. Methodol. **22**(2), 12 (2013)
12. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Qos-aware web service recommendation by collaborative filtering. IEEE Trans. Serv. Comput. **4**(2), 140–152 (2011)
13. Chen, X., Zheng, Z., Yu, Q., Lyu, M.R.: Web service recommendation via exploiting location and QoS information. IEEE Trans. Parallel Distrib. Syst. **25**(7), 1913–1924 (2014)
14. Yao, L., Sheng, Q.Z., Ngu, A.H., Yu, J., Segev, A.: Unified collaborative and content-based web service recommendation. IEEE Trans. Serv. Comput. **8**(3), 453–466 (2015)

15. Alrifai, M., Risse, T.: Combining global optimization with local selection for efficient QoS-aware service composition. In: 18th International Conference on World Wide Web, pp. 881–890 (2009)
16. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for skyline queries. In: ACM SIGMOD International Conference on Management of Data, pp. 467–478 (2003)
17. He, Q., Yan, J., Jin, H., Yang, Y.: Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction. IEEE Trans. Softw. Eng. **40**(2), 192–215 (2014)
18. Zheng, Z., Wu, X., Zhang, Y., Lyu, M.R., Wang, J.: QoS ranking prediction for cloud services. IEEE Trans. Parallel Distrib. Syst. **24**(6), 1213–1222 (2013)
19. Al-Masri, E., Mahmoud, Q.H.: Investigating web services on the world wide web. In: 17th International Conference on World Wide Web, pp. 795–804 (2008)
20. He, Q., Zhou, R., Zhang, X., Wang, Y., Ye, D., Chen, F., Grundy, J., Yang, Y.: Keyword search for building service-based systems. IEEE Trans. Softw. Eng. **43**(7), 658–674 (2017)
21. Zhao, X., Shen, L.W., Peng, X., Zhao, W.: Finding preferred skyline solutions for SLA-constrained service composition. In: 20th IEEE International Conference on Web Services, pp. 195–202 (2013)
22. Zhang, S., Dou, W., Chen, J.: Selecting Top-k composite web services using preference-aware dominance relationship. In: 20th IEEE International Conference on Web Services, pp. 75–82 (2013)
23. Zhang, F., Hwang, K., Khan, S.U., Malluhi, Q.M.: Skyline discovery and composition of multi-cloud mashup services. IEEE Trans. Serv. Comput. **9**(1), 72–83 (2016)