TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 05/16 pp2381–2398
DOI: 10.26599/TST.2024.9010240
Volume 31, Number 5, October 2026

Research Article

# Bilateral Personalized Quality Centric Service Recommendation

Fugui He, Huiying Liu, and Yiwen Zhang*

**Abstract:** The widespread adoption of service-oriented architecture in software engineering has fueled the rapid growth of web and cloud services, as well as service-based systems. With the proliferation of numerous functionally-equivalent services, each offering varying quality levels, finding the appropriate service has become increasingly challenging and essential. This challenge has made service recommendation a critical area of research and practical interest. However, existing methods, such as those relying on utility functions or skyline techniques, failed to address a fundamental issue: recommending services that align with users' specific quality preferences, such as response time or failure rate. This problem involves two main aspects: (1) identifying appropriate services for user requests, and (2) identifying suitable users for new services. This paper proposes a set of approaches for bilateral personalized quality centric service recommendation, integrating k-nearest neighbors, dynamic skyline, and reverse dynamic skyline techniques. Our methods address the shortcomings of existing solutions by identifying both qualified and representative services and users. Extensive experiments on a dataset of 2507 real-world web services validate the effectiveness and efficiency of our approaches.

**Key words:** service recommendation; quality; dynamic skyline (DSL); reverse skyline (RSL); k-nearest neighbor (KNN)

## 1 Introduction

The progression of service-oriented architecture (SOA) has led to the rise of service-oriented software engineering, which facilitates the creation of complex systems by combining loosely connected web and cloud services[1, 2] (collectively referred to as services). These services are executed by a system engine, such as a BPEL engine[3], collaboratively delivering the overall functionality of a service-based system (SBS), often provided as software-as-a-service (SaaS) in cloud environments.

Figure 1 illustrates the creation of a sample travel booking SBS, which requires four services to handle tasks like flight booking, hotel reservations, car rentals, and insurance quotes. The process is divided into two stages. In the first stage, services are recommended by identifying suitable candidates from each set and presenting them to the system engineer[4, 5]. In the second stage, the system engineer chooses one service from each group of recommendations to assemble the SBS, ensuring the system meets various quality requirements such as response time and reliability, while optimizing overall performance. This challenge, known as quality-aware service selection[1, 2, 6–9], is an non-deterministic polynomial (NP)-complete problem.

To tackle this challenge, considerable research has focused on simplifying quality-aware service selection

• Fugui He is with School of Electronic and Information Engineering, West Anhui University, Lu'an 237012, China. E-mail: fuguihe@wxc.edu.cn.
• Huiying Liu and Yiwen Zhang are with School of Computer Science and Technology, Anhui University, Hefei 230601, China. E-mail: liuhuiying.ahu@hotmail.com; zhangyiwen@ahu.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2024-09-26; revised: 2024-11-11; accepted: 2024-12-08

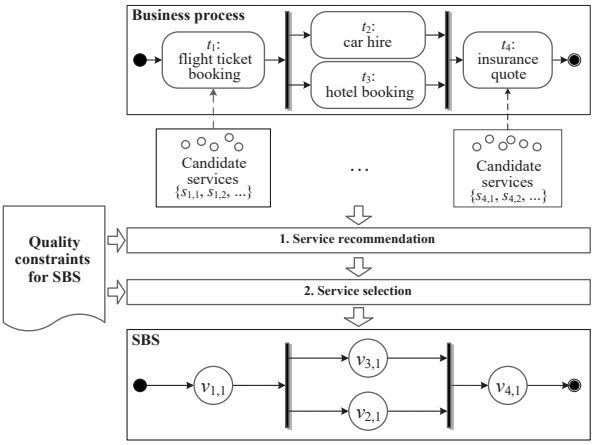



**Fig. 1 Process for service-oriented software engineering.**

through service recommendation techniques[4, 5, 10–16]. By recommending suitable services, the search space for the NP-complete problem is reduced, sparing system engineers (hereafter referred to as users) from evaluating all possible combinations of candidate services.

The key challenge is identifying the most appropriate services that align with the user's quality requirements. This issue, referred to as service identification, is the first aspect of personalized quality centric service recommendation. It also occurs when a user simply needs to select one or a few appropriate services from a group of candidate services based on their quality.

In addition to service identification, there is another aspect of personalized quality centric service recommendation—to find appropriate users for a new service. In the context of this research, a new service refers to either a new service that has just become available or an existing service with updated quality. Upon the appearance of a new service, the service provider wants to find out which users are interested in this new service so that it can be recommended to those users. This aspect is referred to as user identification.

The problem involving the two aspects discussed above is called bilateral personalized quality centric service recommendation. Existing approaches have not effectively solved this problem. Service recommendation methods fall into four main categories: utility-based[7, 15], skyline-based[4, 5], collaborative filtering (CF)[16–21], and matrix factorization (MF)[17, 22]. Utility-based and skyline-based methods failed to adequately consider users' quality preferences. CF-based and MF-based approaches recommended services in a very different way. Given a user, these approaches first predicted the quality of the services unknown to the user by examining the quality perceived by users with similar preferences. This collaborative aspect allows for a more personalized recommendation, leveraging the collective experiences of similar users to enhance the relevance of suggestions. Then, they selected the optimal service to recommend to the user. While CF-based and MF-based service recommendation approaches provide valuable insights into quality prediction, particularly by uncovering hidden patterns in user preferences, they also have significant limitations. For instance, they often neglect the specific quality preferences of users, which can lead to misalignments between recommended services and user expectations. Moreover, their reliance on historical user data may not effectively capture dynamic changes in service quality or shifts in user preferences, potentially resulting in outdated recommendations. Thus, although labeled as a recommendation approach in many related studies[16, 17, 23–25], CF-based and MF-based service recommendation approaches, in fact, focus on a quality-of-service prediction. In addition, they do not take into account users' quality preferences.

In this paper, we introduce four methods for service identification: two fundamental approaches and two hybrid ones. The first fundamental method utilizes the k-nearest neighbor (KNN) technique, while the second employs the dynamic skyline (DSL) technique[26]. The KNN-based method, referred to as KNN, frames the service identification task as a nearest neighbor search problem. It identifies $k$ services that align with a user's quality preferences based on their attributes. The DSL approach models the service identification problem as a DSL query, aiming to find representative services that are not dominated by others according to the user's preferences. The two hybrid approaches, namely KNN-DSL and DSL-KNN, combine the KNN and DSL techniques to overcome their limitations by finding services that are both qualified and representative.

For user identification, two basic approaches and two hybrid ones are also discussed. Similar to service identification, the first basic approach is also based on KNN. The second one utilizes the reverse skyline (RSL) technique[27]. Given a new service, KNN and the RSL-based approach (referred to as RSL hereafter) model and solve the user identification problem as a nearest neighbor search problem and a reverse skyline

query problem, respectively. KNN finds qualified users, whose quality preferences are the most similar to the new service. RSL finds reverse skyline users whose DSLs contain the new service. This way, the new service is not dominated by any services with respect to (wrt) those users' quality preferences. The two hybrid approaches, namely KNN-RSL and RSL-KNN, combine the KNN and RSL to overcome their limitations by finding users that are both qualified and representative.

Extending from its preliminary version[28], this article includes the following major contributions:

(1) This is the first effort to systematically model and address the issue of bilateral personalized quality centric service recommendation, which has two aspects: service identification and user identification.

(2) For service identification, two basic approaches and two hybrid approaches are included. The hybrid approaches, i.e., KNN-DSL and DSL-KNN, address the shortcomings of the basic approaches by finding services that are both qualified and represensitive.

(3) For user identification, two basic approaches and two hybrid approaches are outlined. The hybrid approaches, i.e., KNN-RSL and RSL-KNN, overcome the limitations of the basic approaches by finding users that are both qualified and represensitive.

(4) We conduct extensive experiments to assess the effectiveness and efficiency of the proposed approaches using a dataset with quality information from 2507 real-world web services.

## 2 Related Work

Quality-aware service recommendation is crucial in service-oriented software engineering. Two popular approaches are utility-based recommendation[1, 4, 7, 15, 29–32] and skyline-based recommendation[4].

**Utility-based recommendation is simple:** For a given set of services $S = \{s_1, s_2, \ldots, s_n\}$, each service's utility value reflects how well its $p$-dimensional quality compares with others in $S$, a higher utility value indicates better quality. The utility of service $s_i$ is determined in two steps.

The utility-based service recommendation approach selects services with the highest utility values, helping users identify generally high-quality services. This method is widely applied in service-oriented software engineering[1, 4, 7, 15, 32]. Ardagna and Pernici[1] proposed a new approach to Web service selection, modeling it as a mixed integer linear programming problem. Their method handles large processes and QoS constraints, using loops peeling and negotiation techniques to find feasible solutions. Trummer et al.[7] introduced an approximation scheme for multi-objective service selection, offering a balance between efficiency and precision. Their method significantly reduces optimization time while maintaining low approximation error, outperforming exact and randomized algorithms.

**Skyline-based service recommendation**, first introduced by Alrifai et al.[4], selects services that are not dominated by others. This approach has evolved to address more complex environments. Benouaret et al.[10] proposed the alpha-dominant service skyline, improving fairness in handling services with uneven quality and enhancing calculation efficiency. They also extended it to deal with probabilistic quality values[11]. Zhao et al.[33] incorporated the weighted Tchebycheff distance into skyline techniques to support non-linear utility functions and better account for user constraints across quality dimensions.

A major drawback of both utility-based and skyline-based recommendation methods is their failure to account for users' quality preferences, which are crucial in service-oriented computing[1, 2, 7, 12, 15, 34–37] and skyline-based service composition[33, 38–40]. This oversight makes these methods outdated, as their recommendations are neither qualified nor representative, as shown in Section 5.4. While many approaches focus on service quality prediction using collaborative filtering techniques[16, 17, 23–25, 41], they do not specifically address service recommendation.

Our methods overcome this by integrating users' quality preferences with KNN, DSL, and RSL techniques, enabling more effective recommendations that are both qualified and representative.

## 3 Motivating Example

This section illustrates the motivation for this research with several examples. Given a set of candidate services and a user, a simple approach to service identification is to model it as a nearest neighbor search problem, as shown in Fig. 2. In Fig. 2, points $s_1$ to $s_8$ represent eight candidate services, and point $u_r$ represents the user's quality preferences for response

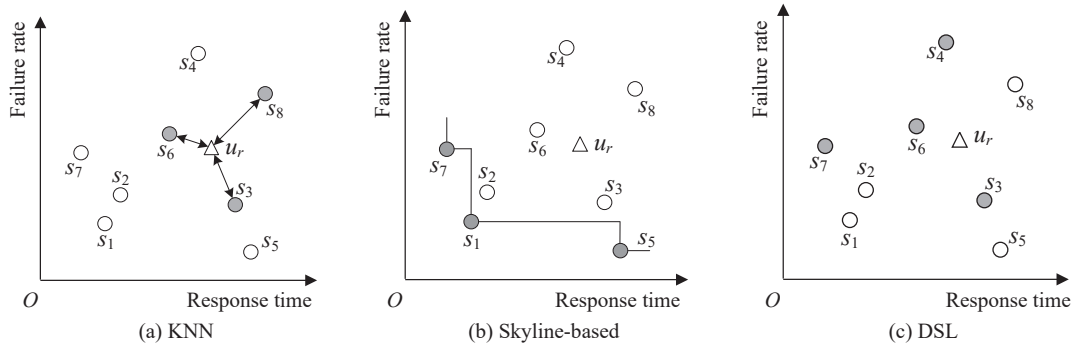



**Fig. 2 Approaches to service identification.**

time and failure rate. As shown in Fig. 2a, services closer to $u_r$ in both dimensions, such as $s_3$, $s_6$, and $s_8$, are more suitable than those farther away, like $s_1$, $s_2$, $s_4$, $s_5$, and $s_7$. For Service $s_1$, it has better response time and lower failure rate, but high-quality services are usually expensive, and $s_1$ may exceed the user's cost limit. In a three-dimensional space that includes cost, $s_1$ might be considered qualified, but in Fig. 2a, it is not suitable based on $u_r$.

We can identify the $k$ nearest neighbor services to uru_rur across all quality dimensions using the KNN approach. However, this method has a limitation: the selected services might not represent all dimensions. For instance, in Fig. 2a, while $s_3$ and $s_6$ are representative in failure rate and response time, respectively, $s_8$ is not representative in either dimension. According to the skyline concept[4, 42], $s_8$ is dominated by $s_3$ and $s_6$ because it does not excel in any dimension.

To address the non-representativeness issue, skyline calculation[4, 42], shown in Fig. 2b, identifies services that are not dominated by others. In Fig. 2b, $s_1$, $s_5$, and $s_7$ are skyline services (light grey circles), superior in response time and failure rate but farther from $u_r$ due to using the origin $O$ as a reference. DSL services[26], illustrated in Fig. 2c, use $u_r$ as the reference point to identify services not dominated in terms of distance from $u_r$ . In Fig. 2c, the DSL services are $s_3$, $s_4$, $s_6$, and $s_7$ (dark circles), with $s_4$ and $s_7$ being most representative in response time and failure rate, respectively. Although DSL improves relevance, it is not perfect; for instance, $s_7$ is still somewhat distant from $u_r$ .

Figure 3 illustrates user identification, where $s_1$ to $s_8$ represent services, $u_1$ to $u_5$ represent users' quality preferences, and $s_n$ represents a new service. If $s_n$ might interest certain users, it should be recommended to them. Using $s_n$ , KNN identifies $u_2$, $u_3$, and $u_4$ as closer to $s_n$ (Fig. 3a). However, KNN has limitations; for example, $s_n$ is dynamically dominated by $s_6$ for $u_4$, making it irrelevant to $u_4$. This issue is addressed by RSL, shown in Fig. 3b, where $\{u_1, u_2, u_3\}$ are identified as RSL with respect to $s_n$ , indicating $s_n$ is not dynamically dominated by other services for these users. Thus, $s_n$ should be recommended to $u_1$, $u_2$, and $u_3$, who are more likely to find it relevant. However, users like $u_1$ with significantly different preferences may still not be interested.

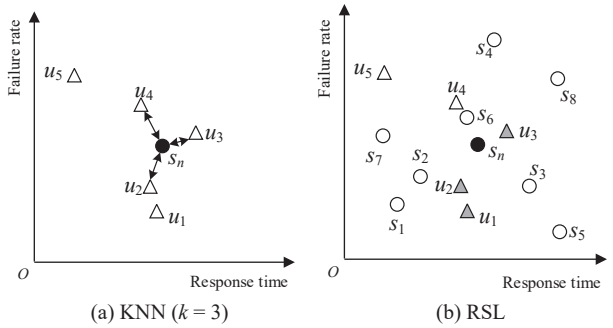



**Fig. 3 Approaches to user identification.**

This section discusses response time and failure rate, but other quality constraints can also be considered, expanding the two-dimensional space into a multi-dimensional one.

## 4 Service Identification

This section presents our basic approaches, i.e., KNN and DSL, and hybrid approaches, i.e., KNN-DSL and DSL-KNN, to service identification.

### 4.1 Service KNN

Given a set of candidate services $S = \{s_1, s_2, \ldots, s_n\}$, each with $p$-dimensional quality values, and a user's $p$-dimensional quality reference $u_r$, KNN seeks to find $k$ qualified services based on $u_r$. The services and $u_r$ are

mapped to a $p$-dimensional space, with each dimension representing a quality metric. For numerical dimensions (e.g., response time, failure rate, reliability), this mapping is simple. For non-numerical dimensions (e.g., reputation), the method from[34] is used, assigning numerical values to levels in a hierarchical structure (e.g., 3 for high, 2 for medium, 1 for low).

KNN then identifies $k$ services based on similarity to $u_r$ in the $p$-dimensional space. To compute similarity, the quality values of each service $s_i \in S$, $1 \leqslant i \leqslant n$ and $u_r$ are normalized using the min-max normalization technique, widely applied by other researchers[15, 43],

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

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

After normalization, the similarity between a candidate service $s_i \in S$ and $u_r$ is measured by the Euclidean distance between them,

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

Based on Eq. (2), KNN employs Algorithm 1 to identify $k$ qualified services based on $u_r$. The algorithm starts by initializing result as a container for the output (Line 2). It then iterates through each candidate service $s$ in $S$, computes its Euclidean distance to the reference $u_r$, and assigns this value to the attribute $s$.distance (Lines 3–5). Subsequently, the function $S$.sortByDistance(·) arranges the services based on these calculated distances (Line 6), and $S$.takeTop($k$) extracts the final set of $k$ qualified services to be returned (Line 7).

The $k$ value must be pre-specified and varies by application and dataset, as different contexts require different optimal $k$ values. A small $k$ may miss similar services, while a large $k$ can include dissimilar ones, reducing accuracy. The $k$ value should be determined through domain-specific experience or experimentation. In Section 6, we conduct experiments to examine how $k$ influences recommendation accuracy using a real-world dataset.

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

**Algorithm 1 KNN**

**Input:** $S$, set of candidate services; $u_r$, reference service; $k$, number of services needed
**Output:** results, $k$ qualified services wrt $u_r$
1: **begin**
2: results ← null;
3: **for** each $s$ in $S$ **do**
4: $s$.distance ← $d(s, u_r)$;
5: **end for**
6: $S$.sortByDistance(·);
7: results ← $S$.takeTop($k$);
8: **return** results
9: **end**

## 4.2 DSL

In this section, we first provide a formal introduction to the concepts of skyline and DSL, followed by a presentation of DSL.

In a $p$-dimensional space, skyline calculation identifies points that are not dominated by others. A point $s_i$ dominates $s_j$ if it is equal to or better in all dimensions and strictly better in at least one. In this study, the dominance relationship between two services is characterized by their $p$-dimensional quality values.

Based on Definition 1, we define skyline services as follows:

**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 $s_i \triangleright s_j$, if $s_i$ is as good as or better than $s_j$ in all quality dimensions and better in at least one quality dimension, i.e., $\forall k \in [1, p]$: $q_p(s_i) \leqslant q_p(s_j)$ and $\exists k \in [1, p]$: $q_p(s_i) < q_p(s_j)$.

**Definition 2. Skyline services:** The skyline of a set $S$, denoted as $S_{\text{SL}}$, includes all services in $S$ that are not dominated by any other services, i.e., $S_{\text{SL}} = \{s_i \in S \mid \neg\exists s_j : s_j \triangleright s_i\}$, where the symbol $\neg\exists$ denotes that there does not exist a service $s_j$ that dominates $s_i$. These services are called skyline services.

Skyline services generally have superior quality in one or more dimensions based on their absolute quality values. However, as mentioned in Section 3, given a reference service $u_r$, DSL needs to find the DSL services in $S$. This can be achieved in a new $p$-dimensional space based on the original space. First,

each service $s \in S$ is mapped to a service $s' = (f_1(s), f_2(s), \ldots, f_p(s))$, where $f_j(s) = |q_j(u_r) - q_j(s)|$, $1 \leqslant j \leqslant p$. Subsequently, the DSL of $S$ concerning functions $f_1, f_2, \ldots, f_p$, is determined by calculating the ordinary skyline in the transformed $p$-dimensional space, using $u_r$ as the origin. Thus, dynamic dominance is defined as follows:

**Definition 3. Dynamic dominance:** For two services, $s_i$, $s_j \in S$, characterized by their $p$-dimensional quality values, and a user's quality preferences $u_r$ as a reference, $s_i$ dynamically dominates $s_j$ wrt $u_r$, denoted by $s_i \rhd s_j$, if $\forall k \in [1, p] : |q_p(u_r) - q_p(s_i)| \leqslant |q_p(u_r) - q_p(s_j)|$ and $\exists k \in [1, p] : |q_p(u_r) - q_p(s_i)| < |q_p(u_r) - q_p(s_j)|$.

Based on Definition 3, we formally define DSL services:

**Definition 4. DSL:** The DSL of $S$, represented as $S_{\mathrm{DSL}}(u_r)$, comprises the services in $S$ that are not dynamically dominated by any other services in $S$, wrt a given user's quality preferences ur as a reference, i.e., $S_{\mathrm{DSL}}(u_r) = \{s_i \in S | \neg \exists s_j : s_j \rhd s_i\}$. The services in $S_{\mathrm{DSL}}(u_r)$ are referred to as DSL services.

Figure 4 shows the DSL calculation based on Fig. 2c. Initially, the original space is transformed with $u_r$ as the new origin, using absolute distances to $u_r$ as the mapping functions. Services $s_1$, $s_2$, $s_3$, $s_4$, $s_5$, $s_6$, and $s_7$ are mapped into the new space as by $S'_1$, $S'_2$, $S'_3$, $S'_4$, $S'_5$, $S'_6$, and $S'_7$ while $s_8$ remains in the first quadrant, so $S'_8$ is omitted in Fig. 4 due to identical location. After mapping all services into the new space $S'$, the computation of $S_{\mathrm{DSL}}(u_r)$ is equivalent to the calculation of $S'_{\mathrm{SL}}$ in the new space.

DSL uses Algorithm 2 to compute the service skyline $S_{\mathrm{SL}}$ for a given set of candidate services $S$. The algorithm iterates through all services in $S$ (Line 4), checking whether any service dominates another $s$ (Lines 5–11). If no service dominates $s$, it is added to

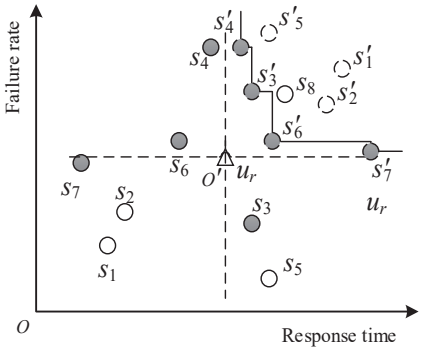



**Fig. 4 Identification of DSL services.**

**Algorithm 2 SL**

**Input:** $S$, set of candidate services

**Output:** $S_{\mathrm{SL}}$, skyline of $S$

```
1: begin
2:   S_SL ← null;
3:   skyline ← true;
4:   for each s_i in S do
5:     skyline ← true;
6:     for each s_j in S do
7:       if s_j ▷ s_i then
8:         skyline ← false;
9:       break
10:      end if
11:    end for
12:    if skyline = true
13:      S_SL ← S_SL + s_i;
14:    end if
15:  end for
16:  return S_SL
17: end
```

$S_{\mathrm{SL}}$ (Lines 12–14). After processing all services, the algorithm returns $S_{\mathrm{SL}}$, containing the skyline services. As shown in Fig. 4, the algorithm returns $S'_{\mathrm{SL}} = \{S'_3, S'_4, S'_6, S'_7\}$, which corresponds to $S_{\mathrm{DSL}}(u_r) = \{s_3, s_4, s_6, s_7\}$.

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

As discussed and demonstrated in Section 2, KNN and DSL have respective limitations. In Sections 3.3 and 3.4, two hybrid approaches, KNN-DSL and DSL-KNN, are introduced to overcome those limitations.

### 4.3 KNN-DSL

Given a reference service $u_r$, KNN identifies qualified services while DSL focuses on representative ones. DSL ensures service representativeness but compromises similarity to $u_r$. For example, in Fig. 4, the DSL $S_{\mathrm{DSL}}(u_r) = \{s_3, s_4, s_6, s_7\}$. While $s_3$ and $s_6$ are closest to $u_r$ based on Euclidean distance and are included in KNN's results when $k \geqslant 2$ (Fig. 2a), $s_4$ and $s_7$ are farther from $u_r$. In fact, $s_8$ and $s_2$ are closer to $u_r$ than $s_4$, suggesting that some DSL services may not be suitable recommendations due to significant quality differences.

KNN-DSL combines the strengths of KNN and DSL to address their limitations. Given $S$ and $u_r$, it first uses KNN to identify $k$ qualified services, denoted as $S_{\mathrm{KNN}}$.

Then, it applies DSL technique to calculate the DSL of $S_{KNN}$, resulting in $S_{KNN\text{-}DSL}$. This method ensures that the identified services are both qualified and representative in terms of their quality relative to $u_r$. For example, in Fig. 5 (based on Fig. 2a), with $k = 3$, KNN identifies $s_3$, $s_6$, and $s_8$ as qualified services. From these, DSL selects $s_3$ and $s_6$ as the DSL services, i.e., the most representative ones. However, this approach can result in fewer than $k$ services, with $|S_{KNN\text{-}DSL}| \leqslant k$, and in the worst case, $S_{KNN\text{-}DSL}$ may be 0, meaning KNN-DSL does not guarantee a fixed number of services in its recommendations.

### 4.4 DSL-KNN

To address the limitation of KNN-DSL, this section introduces a hybrid approach called DSL-KNN. It first applies DSL and then KNN to identify qualified and representative services. Given a set $S$ and a reference $u_r$, DSL-KNN begins by identifying the DSL services $S_{DSL}(u_r)$. From this set, it selects $k$ qualified services, denoted as $S_{DSL\text{-}KNN}$. If $|S_{DSL\text{-}KNN}| < k$, the method selects additional services from $S_{DSL}(u_r)$ to ensure $k$ services are recommended. This ensures that DSL services are prioritized for representativeness, while filling any remaining slots with the most qualified services. Figures 6a and 6b demonstrate this approach with $k = 3$ and $k = 5$ for KNN, respectively. In Fig. 6,

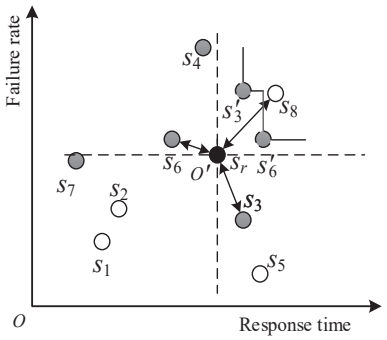
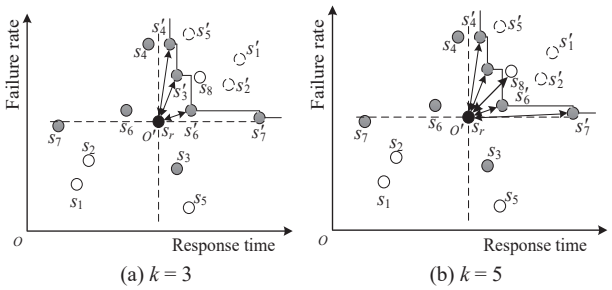



**Fig. 5 Recommendation with KNN-DSL.**




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

**Fig. 6 Recommendation with DSL-KNN.**

DSL-KNN first identifies $S_{DSL} = \{s_3, s_4, s_6, s_7\}$. Then, given $k = 3$, it identifies $s_3$, $s_4$, and $s_6$ from $S_{DSL}$ as the qualified services wrt $u_r$. Given $k = 5$, the method initially select all the services in $S_{DSL}$, and then includes $s_8$ as the fifth service because it is the highest-quality option regarding $u_r$ among the remaining options.

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

## 5 User Identification

This section presents our basic approaches, i.e., KNN, RSL, and hybrid approaches, i.e., KNN-RSL, RSL-KNN, to user identification.

### 5.1 User KNN

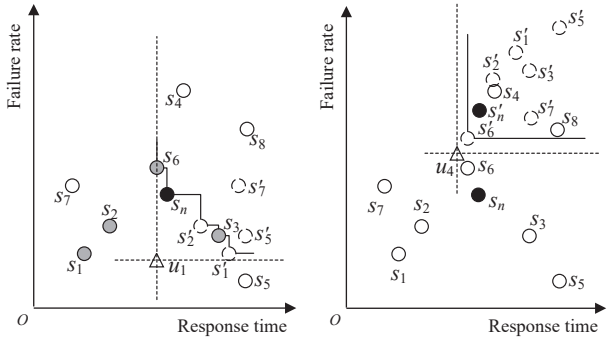
Given a set of users $U = \{u_1, u_2, \ldots, u_m\}$ and a new service $s_n$, KNN attempts to find $k$ qualified users whose quality preferences are the most similar to $s_n$'s, denoted by $U_{KNN}(s_n)$. An example has been given and discussed in Fig. 3a. Algorithmatically, it is very similar to Algorithm 1. Thus, we do not discuss it in detail except its inherent limitation. Suppose that $u_1$ is one of the users in $U_{KNN}(s_n)$. It is possible that $s_n$ is dynamically dominated by other services in $S$ wrt $u_1$. If so, $s_n$ should not be recommended to $u_1$ because it is not representative.

### 5.2 RSL

RSL can tackle the limitation of KNN discussed above. Given a new service $s_n$, we need to find appropriate users that might be interested in $s_n$. As discussed in Section 2, a user is interested in $s_n$ if $s_n$ belongs to the user's DSL because $s_n$ is not dynamically dominated by any other services. Such users are referred to as RSL users, formally defined as follows:

**Definition 5. RSL user:** Given a set of candidate services $S$, a set of users $U$, and a new service $s_n$, the RSL of $s_n$, denoted by $U_{RSL}(s_n)$, is the set of users in $U$ whose DSLs contain $s_n$, i.e., $U_{RSL}(s_n) = \{u_j \in U \mid s_n \in S_{DSL}(u_j)\}$. The users in $U_{RSL}(s_n)$ are referred to as the RSL users of $s_n$.

Figure 7 demonstrates how to find $u_1$ as one of the RSL users and $u_4$ as one of the non RSL users wrt $s_n$ based on the example in Fig. 3b. Figure 7a shows that $DSL(u_1) = \{s_1, s_2, s_3, s_6, s_n\}$ and there is $s_n \in DSL(u_1)$. According to Definition 5, $u_1$ is a RSL user wrt $s_n$. Figure 7b shows that $s_n \notin DSL(u_4)$, and there is $u_4 \notin$




(a) RSL user $u_1$ (b) non-RSL user $u_4$

**Fig. 7 Recommendation with RSL.**

$U_{\text{RSL}}(s_n)$. Thus, $s_n$ should be recommended to $u_1$ but not $u_4$.

A straightforward solution to finding $U_{\text{RSL}}(s_n)$ is to iterate through all the users and calculate their DSLs. In practice, usually, before $s_n$ appears, users' DSLs have already been calculated as the solution to service identification—the first aspect of the bilateral service recommendation discussed in Section 3. Therefore, upon $s_n$, we only need to inspect whether any of the users' DSL services dynamically dominates $s_n$. Algorithm 3 shows the pseudo code for calculating $U_{\text{RSL}}(s_n)$. Algorithm 3 iterates through all users in $U$ (Lines 3−14). During each iteration, it inspects whether any of user $u$'s DSL services dynamically dominates $s_n$ (Lines 5−10). If no such services can be found, $u$ is included in $U_{\text{RSL}}(s_n)$ (Lines 11−13). After all users are inspected, $U_{\text{RSL}}(s_n)$ is returned as the results.

**Algorithm 3 RSL**

**Input:** $U$, set of users; $s_n$, new service

**Output:** $U_{\text{RSL}}(s_n)$, reverse skyline of $s_n$

```
1: begin
2:   U_RSL(s_n) ← null;
3:   for each u ∈ U do
4:     dominated ← false;
5:     for each s ∈ DSL(u) do
6:       if s ▷ s_n then
7:         dominated ← true;
8:       break
9:       end if
10:    end for
11:    if dominated = false
12:      U_RSL(s_n) ← U_RSL(s_n) + u;
13:    end if
14:  end for
15:  return U_RSL(s_n)
16: end
```

Algorithm 3 contains two loops, one nested within the other. Let $n$ represent the number of candidate services in $S$ and $m$ denote the number of users in $U$. The time complexity of Algorithm 3 in the worst-case scenario is $O(mn)$.

### 5.3 KNN-RSL

RSL shares the same inherent limitation with DSL—they both sacrifice similarity in favor of representativeness in their recommendation results. Figure 7a shows that RSL will recommend $s_n$ to $u_1$ because $u_1 \in U_{\text{RSL}}(s_n)$. However, $s_n$ is not quite close to $u_1$, indicating that it is in fact very unlikely to suit the needs of $u_1$. To address this issue, this section presents KNN-RSL, a hybrid method that leverages the strengths of both KNN and RSL. Given a set of candidate services $S$, a set of users $U$, and a new service $s_n$, KNN-RSL first employs KNN to identify $k$ qualified users wrt $s_n$ from $U$, denoted by $U_{\text{KNN}}(s_n)$. Then, it employs RSL technique to find the RSL users from $U_{\text{KNN}}(s_n)$ wrt $s_n$, denoted by $U_{\text{KNN-RSL}}(s_n)$. In this way, KNN-RSL ensures both similarity and representativeness in the recommendation results with similarity prioritized over representativeness.

Figure 3a demonstrates how KNN finds $U_{\text{KNN}}(s_n) = \{u_2, u_3, u_4\}$ with $k = 3$. KNN-RSL takes a step further by looking for $U_{\text{KNN-RSL}}(s_n)$. Figure 3b shows that $U_{\text{RSL}} = \{u_1, u_2, u_3\}$. Thus, there is $U_{\text{KNN-RSL}} = \{u_2, u_3\}$.

Similar to KNN-DSL, KNN-RSL does not ensure a specific number of users in its recommendation results because there is always $|U_{\text{KNN-RSL}}(s_n)| \leqslant |U_{\text{KNN}}(s_n)| \leqslant k$. In fact, in the worst-case scenario, none of the users in $U_{\text{KNN}}(s_n)$ is RSL users wrt $s_n$. In such scenarios, KNN-DSL cannot recommend $s_n$ to any users.

### 5.4 RSL-KNN

To tackle the limitation of KNN-RSL, this section presents another hybrid approach, named RSL-KNN. Given a set of candidate services $S$, a set of users $U$, and a new service $s_n$, RSL-KNN first employs RSL to identify the RSL users wrt $s_n$, denoted by $U_{\text{RSL}}(s_n)$. Then, from $U_{\text{RSL}}(s_n)$, it finds $k$ users that are closest to $s_n$, denoted by $U_{\text{RSL-KNN}}(s_n)$. If $|U_{\text{RSL-KNN}}(s_n)| < k$, RSL-KNN finds $k - |U_{\text{RSL-KNN}}(s_n)|$ users from $U - U_{\text{RSL}}(s_n)$ that are closest to $s_n$ to ensure a total of $k$ users in its recommendation results. In this way, the RSL users are always selected first to ensure the representativeness of the recommendation results.

Figures 8a and 8b demonstrate KNN-RSL based on Fig. 3b with $k = 2$ and $k = 4$ for KNN, respectively. RSL-KNN first finds $U_{\text{RSL}}(s_n) = \{u_1, u_2, u_3\}$, as demonstrated in Fig. 3b. Then, from $U_{\text{RSL}}(s_n)$, it finds $U_{\text{RSL-KNN}}(s_n) = \{u_2, u_3\}$ when $k = 2$, or $U_{\text{RSL-KNN}}(s_n) = \{u_1, u_2, u_3, u_4\}$ when $k = 4$. In Fig. 8b, $u_1$, $u_2$, and $u_3$ are identified as the RSL services. Because $k = 4$, KNN-RSL adds $u_4$ to $U_{\text{RSL-KNN}}(s_n)$.

Both KNN-RSL and RSL-KNN employ KNN and RSL. Thus, their complexity are both $O(mn + np + n\log n) = O(mn + np)$.

## 6 Experimental Evaluation

We conduct two series of experiments: Series I for service identification and Series II for user identification. These experiments evaluate the effectiveness and efficiency of the proposed methods by comparing them to existing representative approaches based on recommendation accuracy.

### 6.1 Comparing approaches

#### 6.1.1 Service identification

For service identification, we compare KNN, DSL, KNN-DSL, and DSL-KNN with three existing representative approaches, as follows:

- **RS**: This method randomly picks $k$ services from the candidates and serves as the baseline for comparison.
- **UF**: This method selects $k$ services with the highest utility values, indicating the overall quality of a service relative to others. The utility is computed using a commonly used function[1, 4, 7, 15, 29].
- **SL**: This approach employs the skyline technique[4, 44] to identify the skyline services according to Definitions 1 and 2.

#### 6.1.2 User identification

For user identification, we compare KNN, RSL, KNN-RSL, and RSL-KNN with two representative methods, as follows:

- **RS**: This approach randomly selects $k$ users. It is employed as the baseline approach in the comparison.
- **DSL**: This approach employs the DSL technique to find the users that are not dominated by any other users wrt $s_n$.

According to the discussion in Section 2, SL is not suitable for user identification. The UF-based approach is not suitable either because the utility is not a proper metric for evaluating users wrt $s_n$.

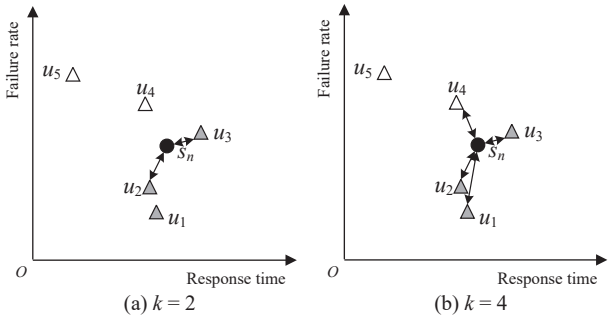



**Fig. 8 Recommendation with RSL-KNN.**

### 6.2 Metrics

This section describes the metrics used in the evaluation.

**Effectiveness.** The effectiveness of the comparing approaches is measured with three accuracy metrics, mean absolute error (MAE), root mean squared error (RMSE), and non-dominance rate (NDR).

MAE is defined as

$$\text{MAE} = \frac{\sum_{i=1}^{|R|} \sqrt{\sum_{j=1}^{p} \left(q_j(r_i) - q_j(r_r)\right)^2}}{|R|} \tag{3}$$

where $R$ is the results returned by the recommendation approach, $q_j(r_i)$ is the $j$-th dimensional quality value of $r_i \in R$, and $q_j(r_r)$ is the $j$-th dimensional quality value of $u_r$ or $s_n$. MAE reflects the average discrepancy between the recommended results and $u_r$ or $s_n$ in their $p$-dimensional quality. All the individual differences are weighted equally during the calculation of MAE. A low MAE signifies high recommendation accuracy.

RMSE is defined as

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{|R|} \sum_{j=1}^{p} \left(q_j(r_i) - q_j(r_r)\right)^2}{|R|}} \tag{4}$$

For RMSE, the differences between recommended results and $u_r$ or $s_n$ are squared, averaged over $R$, and the square root of the average is taken. Like MAE, a lower RMSE reflects higher accuracy.

NDR is defined as

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

For service identification, $R_{\text{rep}}$ is the set of DSL services in $R$ wrt $u_r$. For user identification, $R_{\text{rep}}$ is the set of RSL users in $R$. NDR measures the percentage of the recommendation results that are representative in at least one quality dimension wrt to $u_r$ or $s_n$. A high NDR

indicates high recommendation accuracy.

**Efficiency**. In order to evaluate the efficiency of the proposed approaches, we measure the computational overheads of all approaches.

### 6.3 Experiment setup

The experiments are carried out using the publicly available QWS dataset, which includes functional and quality information for over 2500 real-world web services[44]. This dataset has also been widely used in many other research on service-oriented software engineering[7, 12, 15, 34]. The nine-dimensional quality information in QWS is collected from public registries, search engines, and service portals. More details on the dataset can be found in Refs. [44, 45].

To evaluate the proposed approaches and the impacts of four parameters, i.e., the number of candidate services ($n$), the number of users ($m$), the number of recommendations ($k$), and the number of quality dimensions ($q$), on those approaches, we conduct three sets of experiments in Series I, i.e., I.A–I.C, and four sets in Series II, i.e., II.A–II.D. In each set of experiments, we change one parameter while fixing the other parameters to create different recommendation scenarios. Table 1 outlines the parameter settings. In each experiment, we randomly select $n$ services from the QWS dataset as the candidate services, and another $m$ services from the remaining services in QWS to represent different users' quality preferences. Then, we run all the recommendation approaches. Each experiment is repeated 100 times, and the results are averaged.

All approaches are implemented in Java using JDK 1.8. All experiments are conducted on a machine with Intel i7-4790 CPU 3.60 GHz and 16 GB RAM, running Windows 10 x64 Professional.

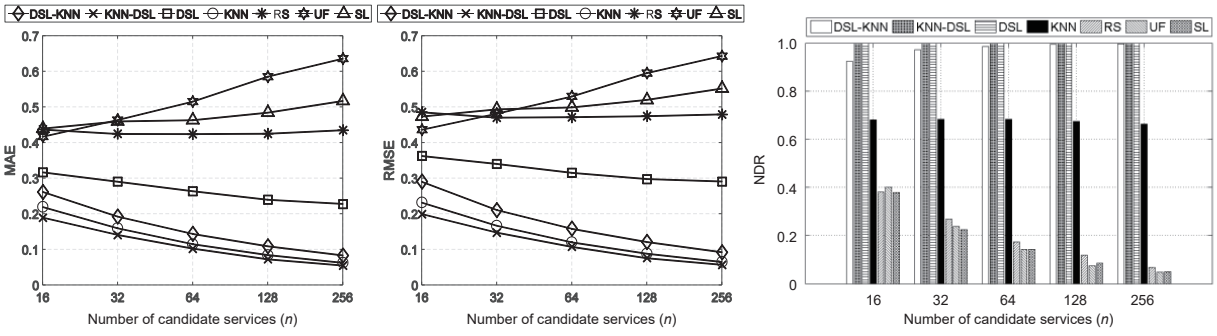
### 6.4 Effectiveness evaluation

In this section, the experiments aim to evaluate the effectiveness of different recommendation approaches under varying parameter settings, such as $n$, $k$, and $q$. These parameters are adjusted to assess their impact on recommendation accuracy and representativeness, helping to identify key factors influencing the performance of each method. Figures 9–11 show the impacts of $n$, $k$, and $q$ on the effectiveness of the recommendation approaches in experiment series I.

Figures 9a and 9b demonstrate that KNN-DSL, KNN, and DSL-KNN result in the highest MAE and RMSE values, emphasizing the importance of considering users' quality preferences in service recommendation. RS, SL, and UF fail to account for these preferences, leading to less qualified recommendations. Notably, UF exhibits the worst accuracy by recommending services with high overall quality but misaligned with user preferences, as reflected in its high MAE and RMSE values. As $n$ increases, KNN-DSL and DSL-KNN show improved

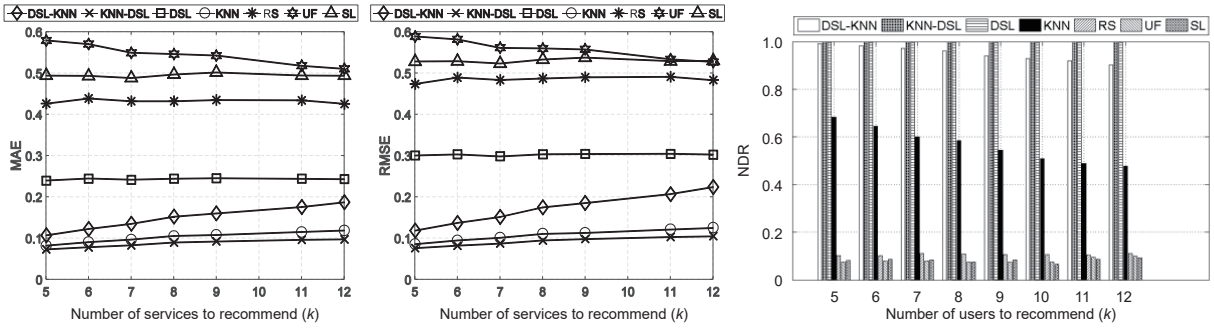
**Table 1 Experiment parameters and settings. NA denotes not applicable.**

| Parameter | Setting | | | | | | |
|---|---|---|---|---|---|---|---|
| | I.A | I.B | I.C | II.A | II.B | II.C | II.D |
| Number of users ($m$) | NA | NA | NA | 10–100 | 50 | 50 | 50 |
| Number of candidate services ($n$) | 16–256 | 128 | 128 | 128 | 16–256 | 128 | 128 |
| Number of services to recommend ($k$) | 5 | 5–12 | 5 | 5 | 5 | 5–12 | 5 |
| Number of quality dimensions ($q$) | 4 | 4 | 2–9 | 4 | 4 | 4 | 4–9 |



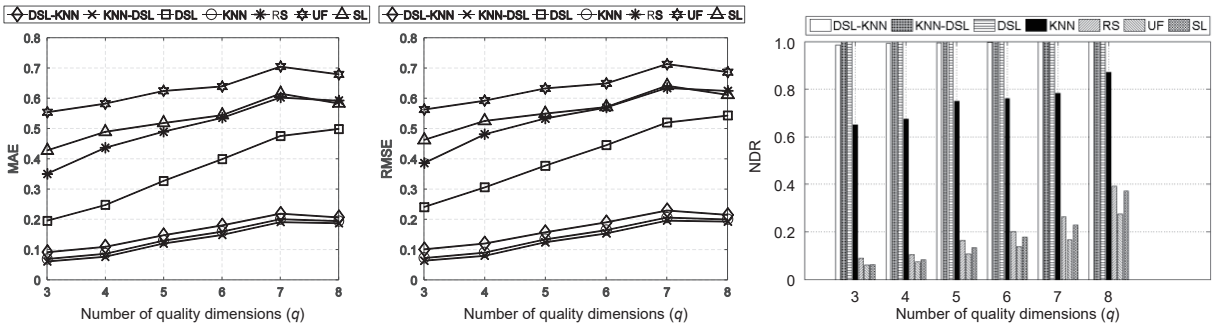

(a) MAE (b) RMSE (c) NDR

**Fig. 9 Impact of parameter $n$ on effectiveness (experiment series I.A).**




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

**Fig. 10 Impact of parameter *k* on effectiveness (experiment series I.B).**




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

**Fig. 11 Impact of number of quality constraints (*q*) on effectiveness (experiment series I.C).**

recommendation accuracy, as the larger number of candidate services increases the chance of finding a match that meets user preferences (Figs. 9a and 9b). Figure 9c demonstrates that KNN-DSL and DSL provide the most representative services, slightly outperforming DSL-KNN. This emphasizes the effectiveness of the DSL operator in selecting services that align closely with users' preferences, compared to methods like KNN, which provide significantly less representative recommendations.

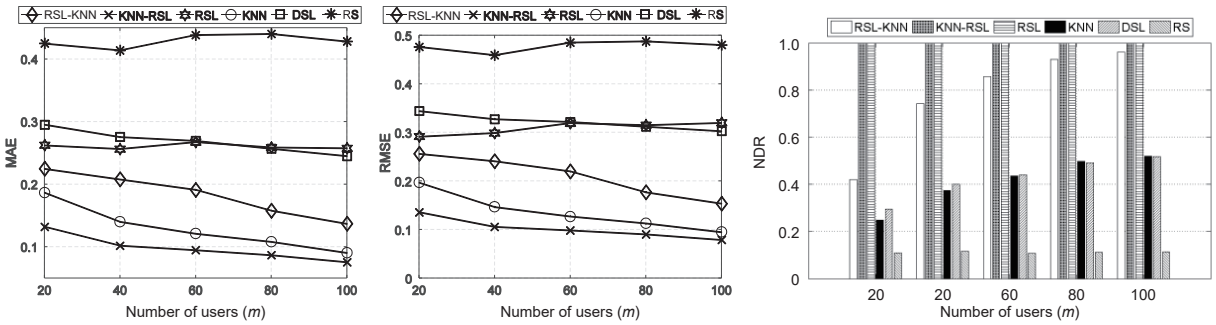
Figures 10a and 10b illustrate that KNN-DSL delivers the best recommendation accuracy, followed by KNN, DSL-KNN, and DSL. KNN-DSL outperforms due to its two-step process: first identifying similar services using the KNN operator and then refining the selection with the DSL operator to prune dissimilar services. KNN ranks second, as its inclusion of non-DSL services reduces its accuracy. DSL-KNN, also a two-stage approach, starts by selecting DSL services, which are representative but not always the most qualified. As a result, its accuracy (MAE and RMSE) is lower than KNN-DSL and KNN. RS, UF, and SL perform significantly worse, as reflected in their higher MAE and RMSE values. Figures 10a and 10b indicate that increasing the value of $k$ slightly raises the MAE and RMSE values for KNN-DSL, DSL-KNN, and SL, as additional services less similar to $u_r$ are included. The increase in $k$ does not affect DSL, as it only recommends DSL services, which are not influenced by $k$. Figure 10c highlights that both KNN-DSL and DSL produce highly representative recommendations, thanks to the DSL operator, which filters out non-representative services. DSL-KNN ranks third in representativeness, but as $k$ increases, it must include non-DSL services, reducing the overall representativeness of its recommendations.

Figures 11a and 11b illustrate that KNN-DSL consistently achieves the highest recommendation accuracy, with KNN and DSL-KNN following, and DSL in fourth place, aligning with previous experiment outcomes. Increasing $q$ decreases MAE and RMSE, as a larger $q$ value leads to a broader set of skyline and DSL services, reducing the dominance of any single service. Nonetheless, KNN-DSL remains superior due to its KNN operator, which ensures the selected services closely match $u_r$. Even as the DSL operator identifies more DSL services with a larger $q$, they remain closely aligned with $u_r$. KNN secures the second position as it also prioritizes similarity with $u_r$, whereas DSL-KNN ranks third, as its DSL operator

might exclude eligible services that are dominated by others. Figure 11c reveals that KNN-DSL and DSL consistently provide highly representative recommendations. DSL-KNN generally performs well, except when $q = 2$, where the limited DSL services compel the KNN operator to include eligible but less representative services, thus reducing overall representativeness.

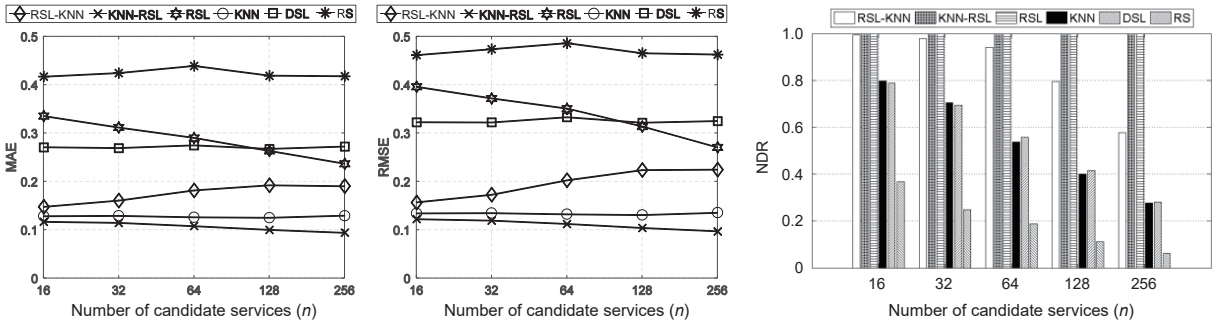
Figures 12–15 show the impacts of $n$, $m$, $k$, and $q$ on the effectiveness of the recommendation approaches in experiment series II. Overall, KNN-RSL, KNN, and RSL-KNN achieve the highest MAE and RMSE values, while KNN-RSL and RSL achieve the highest NDR values. This shows that they can ensure 100% representativeness in their recommendation results with their RSL operators.

Figures 12a and 12b show RSL obtains the fourth best accuracy of recommendation and RSL the fifth. This confirms that RSL alone is not able to ensure the similarity between the recommendation results and $s_n$, as discussed in Section 4.3. Figures 12a and 12b also show that the increase in $m$ improves the MAE and RMSEs achieved by all approaches except RS. As $m$ increases, there are more users to choose from. This increases the possibility of finding qualified users. Figure 12c highlights that as $m$ increases, KNN-RSL and RSL become more effective at finding users for whom the recommendation is representative. This



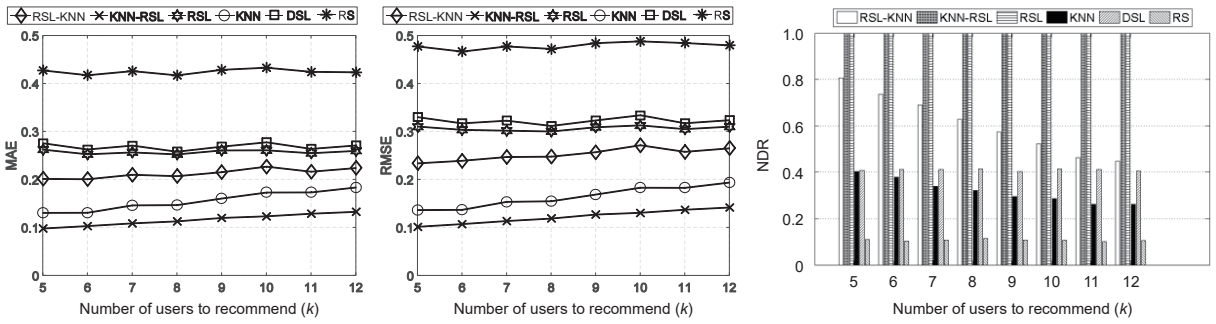

(a) MAE (b) RMSE (c) NDR

**Fig. 12 Impact of parameter $m$ on effectiveness (experiment series II.A).**



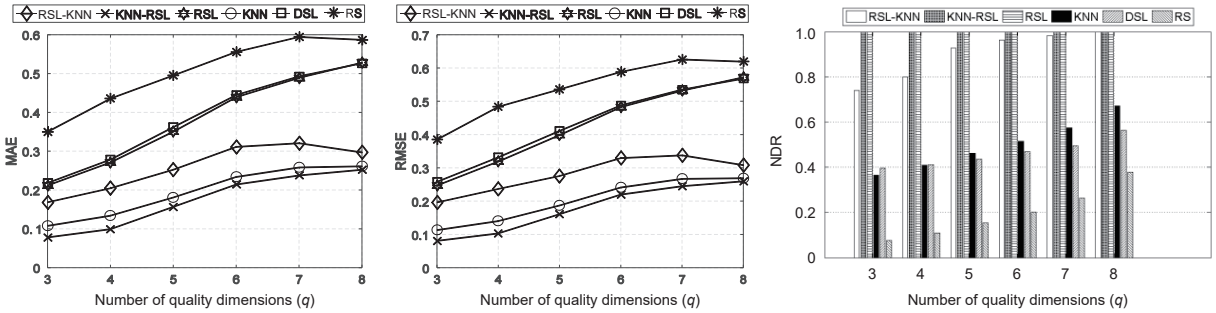

(a) MAE (b) RMSE (c) NDR

**Fig. 13 Impact of parameter $n$ on effectiveness (experiment series II.B).**




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

**Fig. 14 Impact of parameter $k$ on effectiveness (experiment series II.C).**




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

**Fig. 15 Impact of number of quality constraints ($q$) on effectiveness (experiment series II.D).**

improvement is more significant as $m$ grows, showing the advantage of having a larger pool of users from which to select.

Figures 13a and 13b demonstrate that a rise in $n$ leads to an increase in MAE and RMSE values of KNN-RSL and RSL while decreases the those of RSL-KNN. A larger $n$ is more likely to place more services around each user, makes it harder for the RSL operator to find RSL users. Only the users that are very close to $s_n$ will find it in their DSL, which increases in the similarity between $s_n$ and the users recommended by KNN-RSL and RSL. However, the RSL operator of RSL-KNN finds fewer users with a larger $n$. Its KNN operator must include some non RSL users into the recommendation results. Those users are not as close to $s_n$, and thus increase the average MAE and RMSE values of RSL-KNN. This is also the reason why the representativeness in the recommendation results achieved by RSL-KNN decreases, as indicated by its declining NDR values in Fig. 13c.

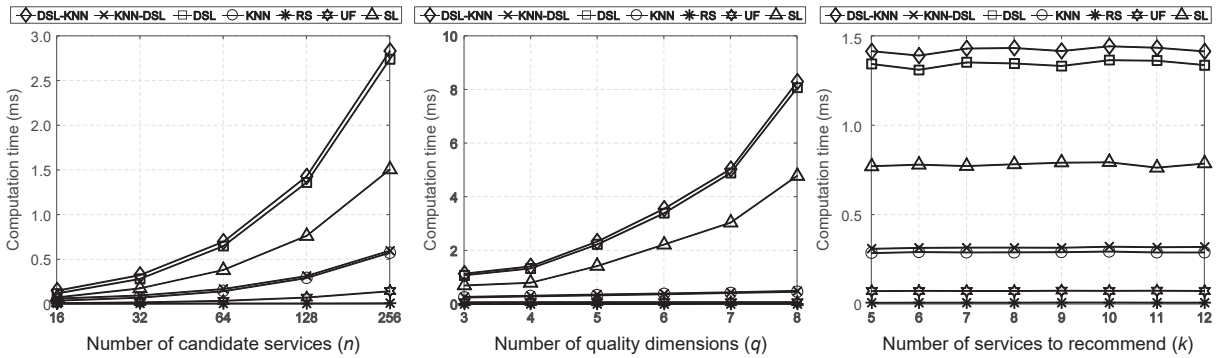
Figure 14 shows that the increase in $k$ lowers the recommendation accuracies of KNN-RSL, RSL-KNN, and KNN. It is because a larger $k$ requires more users to be included in the recommendation results. On average, these users' quality preferences tend to be dissimilar to $s_n$ and not representative. Parameter $n$ does not impact the other approaches because they do not rely on $k$.

Figure 15 shows that the increase in $q$ increases recommendation accuracies of all approaches. According to Eqs. (3) and (4), MAE and RMSE accumulate the differences between the recommendation results and $u_r$ or $s_n$ across all quality dimensions. Thus, a larger $q$ directly leads to higher MAE and RMSE values. A larger $q$ will allow the RSL operator to find more RSL users. This increases the representativeness of the recommendation results of RSL-KNN, as indicated its rising NDR values, because its KNN operator does not have to find as many non RSL users.

In summary, the experiments demonstrate that KNN-DSL and KNN-RSL generally outperform other methods in terms of both accuracy and representativeness. The results also show that increasing $n$, $k$, and $q$ has a varying impact on different methods. These findings suggest that careful tuning of the parameters is crucial for optimizing recommendation performance.

### 6.5 Efficiency evaluation

Figure 16 illustrates the effects of various parameters on computation times for different recommendation approaches in experiment series I. The approaches can be divided into two groups: slow approaches (SL, DSL, and DSL-KNN) and fast approaches (RS, KNN, KNN-DSL, and UF). The slow approaches take more time because they identify skyline or DSL services from a large number of candidate services, whereas the fast approaches bypass this step. Despite KNN-DSL using a DSL operator like DSL and DSL-KNN, it is faster because its KNN operator pre-selects only $k$ services for further processing, making the DSL calculation quicker due to the smaller set size. The slow approaches take significantly more time than the fast ones but remain relatively efficient. Figure 16a shows that even in the worst-case scenarios, they complete in under 3 ms. In Fig. 16b, the impact of $q$ is minimal for the fast approaches, which complete in under 1 ms on average, demonstrating their scalability. The slow approaches take 60 to 100 ms but remain fast enough for most real-world applications. Figure 16c shows that $k$ has no significant effect on the efficiency, as selecting $k$ services involves simple operations with a complexity of $O(1)$.




(a) Computation time vs $n$ (b) Computation time vs $q$ (c) Computation time vs $k$

**Fig. 16 Impact of parameters on efficiency (experiment series I).**

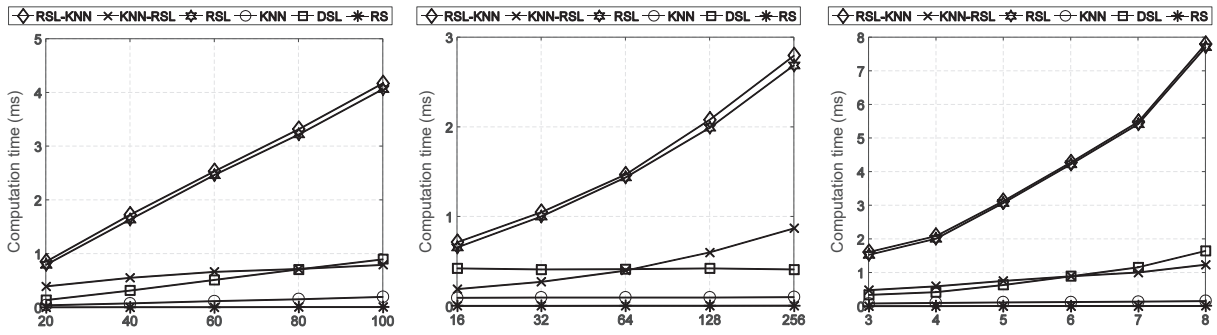
Figure 17 compares computation times for approaches in experiment series II, categorizing them into two groups based on scalability: slow approaches (RSL-KNN and RSL) and fast approaches (KNN-RSL, KNN, DSL, and RS). The slow approaches take longer because they identify RSL users using an RSL operator from a large user set. KNN-RSL also uses an RSL operator but processes fewer users due to pre-selection by its KNN operator, making it faster than RSL- KNN or RSL. Fast approaches complete within 2 ms on average, while slow approaches take only a few more milliseconds. Figure 17a shows that even slow approaches scale linearly with $m$, indicating they can handle large user sets. Figure 17b confirms similar scaling behavior for $m$ across slow and fast approaches. Figure 17c shows $q$ slightly impacts the fast approaches' computation times, but they still scale almost linearly. The effect of $k$ is not presented, as its impact is negligible, similar to the findings in Fig. 16c.

### 6.6 Discussion

KNN-DSL and KNN-RSL outperform other approaches in terms of MAE, RMSE, and NDR, making their recommendations both the most qualified and the most representative. Thus, in most real-world applications, KNN-DSL and KNN-RSL are the best approaches for service identification and user identification, respectively. However, they do not guarantee a specific number of results, potentially returning fewer than $k$. If $k$ results are mandatory, DSL-KNN, RSL-KNN, or KNN are better options. KNN shows the second lowest MAE and RMSE values, suggesting good recommendation quality. However, its NDR values are considerably lower than those of DSL-KNN and RSL-KNN, which affects its representativeness. Therefore, DSL-KNN and RSL-KNN are generally better when $k$ results are required. While DSL and RSL achieve the highest NDR values, their higher MAE and RMSE, and lack of result guarantees, make them less ideal for practical applications.

## 7 Threat to Validity

**Threats to construct validity.** A key threat to the construct validity of our evaluation arises from comparing recommendation accuracy against




(a) Computation time vs $m$ (b) Computation time vs $n$ (c) Computation time vs $q$

**Fig. 17 Impact of parameters on efficiency (experiment series II).**

existing methods, such as random, utility-based, and skyline-based approaches, with the random method serving as the baseline. Utility-based and skyline approaches, commonly used to evaluate and select services[1, 4, 7, 15, 29], do not consider users' quality preferences, often yielding less qualified and representative results compared to ours. The main threat is whether the chosen comparison adequately demonstrates our approaches' effectiveness. To mitigate this, we also compare our hybrid approaches with the basic ones (KNN, DSL, and RSL) to show their advantages objectively. Additionally, we varied four parameters that could influence effectiveness for a thorough evaluation.

**Threats to external validity.** The primary concern is the representativeness of the dataset, which may not accurately reflect all real-world applications. However, this threat is minimal. The QWS dataset consists of genuine data on real-world services[44] and has been widely used in research related to service-oriented software engineering[7, 15, 34]. Additionally, the crucial aspect influencing our evaluation is the distribution of the quality values, rather than their absolute values. To address this, we vary the number of candidate services and quality constraints to simulate a range of application characteristics, allowing us to assess the proposed approaches across different scenarios. The MAE and RMSE values obtained by the proposed approaches in real-world applications might be different from what is presented in the evaluation. However, their advantages and disadvantages compared with the other approaches are similar in general.

**Threats to internal validity.** The primary threat to internal validity in our evaluation arises from the limited comprehensiveness of the experiments. Due to space constraints, we cannot present results for all parameter settings, such as additional combinations of values. However, we consider this threat to be minimal. Although the exact values, such as MAE and RMSE, obtained from experiments with different parameter settings may vary, the advantages of our approaches compared to others remain consistent with those discussed.

**Threats to conclusion validity.** A key threat to conclusion validity in our evaluation is the absence of statistical tests, such as chi-square tests. Although these tests are used to draw more formal conclusions, we conduct each experiment 100 times with varying parameters and averaged the results. This large number of test cases will likely lead to small $p$-values, reducing the practical significance of the test results[36]. Despite this, the number of runs is much smaller than the sample size concerns raised by Lin et al.[46], so the threat, while present, is not substantial.

## 8 Conclusion and Future Work

This paper presents a suite of approaches for personalized quality centric bilateral service recommendation, including two basic and two hybrid approaches for both service indentification and user identification. By combining KNN, DSL, and RSL techniques, the hybrid approaches significantly outperform both the basic methods and existing state-of-the-art approaches. Extensive experiments conducted on a real-world dataset with over 2500 web services validate the superior performance of the hybrid approaches, demonstrating their effectiveness and efficiency in generating highly relevant recommendations. The results highlight the potential of these approaches to address real-world challenges in personalized service recommendation, providing both theoretical and practical insights into the design of recommendation systems.

The theoretical significance of this work lies in its exploration of hybrid methods that integrate multiple recommendation techniques to enhance the overall quality and relevance of recommendations. This contribution not only advances the state-of-the-art in the field but also offers a framework for future research on hybrid recommendation systems. Practically, the proposed methods can be applied in various domains, such as e-commerce, healthcare, and digital services, offering users more accurate and personalized service recommendations.

In future work, we plan to explore the diversity of recommendation results produced by the proposed methods, examining how to balance accuracy with diversity to improve user satisfaction. Additionally, we will investigate the scalability of the hybrid approaches in larger-scale service environments and explore their integration with other emerging techniques, such as deep learning and reinforcement learning, to further enhance recommendation performance.

**Acknowledgment**

This work was supported by the Open Fund of

Information Materials and Intelligent Sensing Laboratory of Anhui Province (No. IMIS202010), the Excellent Young Talents Support Program in Universities of Anhui Province (No. 2022AH020091), the Outstanding Youth Talent Support Program in Universities of Anhui Province (No. gxyqZD2021128), the Major Project of Anhui Education Department (No. KJ2021ZD0116), the Research Foundation of High-Level Talent of West Anhui University (Nos. WGKQ202001006 and WGKQ2025011), the Scientific Research Foundation of the Higher Education Institutions of Anhui Province (No. KJ2019A0630), the University Key Research Project of Department of Education Anhui Province (No. 2022AH051683), and the University Innovation Team Project of Department of Education Anhui Province (No. 2023AH010078).

## References

[1] D. Ardagna and B. Pernici, Adaptive service composition in flexible processes, *IEEE Trans. Softw. Eng.*, vol. 33, no. 6, pp. 369–384, 2007.

[2] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, QoS-aware middleware for web services composition, *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, 2004.

[3] L. Baresi and S. Guinea, Self-supervising BPEL processes, *IEEE Trans. Softw. Eng.*, vol. 37, no. 2, pp. 247–263, 2011.

[4] M. Alrifai, D. Skoutas, and T. Risse, Selecting skyline services for QoS-based web service composition, in *Proc. 19th Int. Conf. World Wide Web*, Raleigh, NC, USA, 2010, pp. 11–20.

[5] Q. Yu and A. Bouguettaya, Efficient service skyline computation for composite service selection, *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 776–789, 2013.

[6] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli, Dynamic QoS management and optimization in service-based systems, *IEEE Trans. Softw. Eng.*, vol. 37, no. 3, pp. 387–409, 2011.

[7] I. Trummer, B. Faltings, and W. Binder, Multi-objective quality-driven service selection—a fully polynomial time approximation scheme, *IEEE Trans. Softw. Eng.*, vol. 40, no. 2, pp. 167–191, 2014.

[8] W. Xiong, B. Li, L. He, M. Chen, and J. Chen, Collaborative web service QoS prediction on unbalanced data distribution, in *Proc. 2014 IEEE Int. Conf. Web Services*, Anchorage, AK, USA, 2014, pp. 377–384.

[9] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, A break in the clouds: Towards a cloud definition, *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, 2009.

[10] K. Benouaret, D. Benslimane, and A. Hadjali, On the use of fuzzy dominance for computing service skyline based on QoS, in *Proc. 2011 IEEE Int. Conf. Web Services*, Washington, DC, USA, 2011, pp. 540–547.

[11] K. Benouaret, D. Benslimane, and A. Hadjali, Selecting skyline web services from uncertain QoS, in *Proc. 9th IEEE Int. Conf. Services Computing*, Honolulu, HI, USA, 2012, pp. 523–530.

[12] 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, in *Proc. 38th Int. Conf. Software Engineering*, Austin, TX, USA, 2016, pp. 85–95.

[13] W. Dou, W. Tang, X. Wu, L. Qi, X. Xu, X. Zhang, and C. Hu, An insurance theory based optimal cyber-insurance contract against moral hazard, *Inf. Sci.*, vol. 527, pp. 576–589, 2020.

[14] C. Hu, W. Fan, E. Zeng, Z. Hang, F. Wang, L. Qi, and Z. A. Bhuiyan, Digital twin-assisted real-time traffic data prediction method for 5G-enabled internet of vehicles, *IEEE Trans. Industr. Inf.*, vol. 18, no. 4, pp. 2811–2819, 2022.

[15] 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 Trans. Softw. Eng.*, vol. 40, no. 2, pp. 192–215, 2014.

[16] Z. Zheng and M. R. Lyu, Collaborative reliability prediction of service-oriented systems, in *Proc. ACM/IEEE 32nd Int. Conf. Software Engineering*, Cape Town, South Africa, 2010, pp. 35–44.

[17] Z. Zheng and M. R. Lyu, Personalized reliability prediction of web services, *ACM Trans. Softw. Eng. Methodol.* (*TOSEM*), vol. 22, no. 2, p. 12, 2013.

[18] F. Fei, S. Li, H. Dai, C. Hu, W. Dou, and Q. Ni, A K-anonymity based schema for location privacy preservation, *IEEE Trans. Sustain. Comput*., vol. 4, no. 2, pp. 156–167, 2019.

[19] W. Liang, X. Zhou, S. Huang, C. Hu, X. Xu, and Q. Jin, Modeling of cross-disciplinary collaboration for potential field discovery and recommendation based on scholarly big data, *Future Gener. Comput. Syst.*, vol. 87, pp. 591–600, 2018.

[20] Q. Sun, L. Shi, L. Liu, Z. Han, L. Jiang, Y. Wu, and Y. Zhao, A novel recommendation algorithm integrates resource allocation and resource transfer in weighted bipartite network, *Big Data Mining and Analytics*, vol. 7, no. 2, pp. 357–370, 2024.

[21] Z. Han, L. Shi, L. Liu, L. Jiang, J. Fang, F. Lin, J. Zhang, J. Panneerselvam, and N. Antonopoulos, A survey on event tracking in social media data streams, *Big Data Mining and Analytics*, vol. 7, no. 1, pp. 217–243, 2024.

[22] Z. Zheng, H. Ma, M. R. Lyu, and I. King, Collaborative web service QoS prediction via neighborhood integrated matrix factorization, *IEEE Trans. Serv. Comput.*, vol. 6, no. 3, pp. 289–299, 2013.

[23] X. Chen, Z. Zheng, Q. Yu, and M. R. Lyu, Web service recommendation via exploiting location and QoS information, *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1913–1924, 2014.

[24] L. Yao, Q. Z. Sheng, A. H. H. Ngu, J. Yu, and A. Segev, Unified collaborative and content-based web service recommendation, *IEEE Trans. Serv. Comput.*, vol. 8, no.

3, pp. 453–466, 2015.
[25] Z. Zheng, H. Ma, M. R. Lyu, and I. King, QoS-aware web service recommendation by collaborative filtering, *IEEE Trans. Serv. Comput.*, vol. 4, no. 2, pp. 140–152, 2011.
[26] D. Papadias, Y. Tao, G. Fu, and B. Seeger, An optimal and progressive algorithm for skyline queries, in *Proc. 2003 ACM SIGMOD Int. Conf. Management of Data*, San Diego, CA, USA, 2003, pp. 467–478.
[27] E. Dellis and B. Seeger, Efficient computation of reverse skyline queries, in *Proc. 33rd Int. Conf. Very Large Data Bases*, Vienna, Austria, 2007, pp. 291–302.
[28] Y. Zhang, X. Ai, Q. He, X. Zhang, W. Dou, F. Chen, L. Chen, and Y. Yang, Personalized quality centric service recommendation, in *Proc. 15th Int. Conf. Service-Oriented Computing*, Malaga, Spain, 2017, pp. 528–544.
[29] M. Alrifai and T. Risse, Combining global optimization with local selection for efficient QoS-aware service composition, in *Proc. 18th Int. Conf. World Wide Web*, Madrid, Spain, 2009, pp. 881–890.
[30] L. Qi, R. Wang, C. Hu, S. Li, Q. He, and X. Xu, Time-aware distributed service recommendation with privacy-preservation, *Inf. Sci.*, vol. 480, pp. 354–364, 2019.
[31] J. Liu, H. Gao, C. Yang, C. Shi, T. Yang, H. Cheng, Q. Xie, X. Wang, and D. Wang, Heterogeneous spatio-temporal graph contrastive learning for point-of-interest recommendation, *Tsinghua Science and Technology*, vol. 30, no. 1, pp. 186–197, 2025.
[32] D. Ma, Y. Wang, J. Ma, and Q. Jin, SGNR: A social graph neural network based interactive recommendation scheme for e-commerce, *Tsinghua Science and Technology*, vol. 28, no. 4, pp. 786–798, 2023.
[33] X. Zhao, L. W. Shen, X. Peng, and W. Zhao, Finding preferred skyline solutions for SLA-constrained service composition, in *Proc. 20th IEEE Int. Conf. Web Services*, Santa Clara, CA, USA, 2013, pp. 195–202.
[34] Q. He, R. Zhou, X. Zhang, Y. Wang, D. Ye, F. Chen, J. C. Grundy, and Y. Yang, Keyword search for building service-based systems, *IEEE Trans. Softw. Eng.*, vol. 43, no. 7, pp. 658–674, 2017.
[35] Y. Cheng, W. Cao, H. Fang, and S. Zang, A context-aware edge-cloud collaboration framework for QoS prediction, *Tsinghua Science and Technology*, vol. 30, no. 3, pp. 1201–1214, 2025.
[36] W. Lin, M. Zhu, X. Zhou, R. Zhang, X. Zhao, S. Shen, and L. Sun, A deep neural collaborative filtering based service recommendation method with multi-source data for smart cloud-edge collaboration applications, *Tsinghua Science and Technology*, vol. 29, no. 3, pp. 897–910, 2024.
[37] S. Santhosh and N. S. Ramaiah, cloud-based software development lifecycle: A simplified algorithm for cloud service provider evaluation with metric analysis, *Big Data Mining and Analytics*, vol. 6, no. 2, pp. 127–138, 2023.
[38] Y. Du, H. Hu, W. Song, J. Ding, and J. Lü, Efficient computing composite service skyline with QoS correlations, in *Proc. 2015 IEEE Int. Conf. Services Computing*, New York, NY, USA, 2015, pp. 41–48.
[39] F. Zhang, K. Hwang, S. U. Khan, and Q. M. Malluhi, Skyline discovery and composition of multi-cloud mashup services, *IEEE Trans. Serv. Comput.*, vol. 9, no. 1, pp. 72–83, 2016.
[40] S. Zhang, W. Dou, and J. Chen, Selecting top-k composite web services using preference-aware dominance relationship, in *Proc. 20th IEEE Int. Conf. Web Services*, Santa Clara, CA, USA, 2013, pp. 75–82.
[41] K. Taha, P. D. Yoo, C. Yeun, and A. Taha, Empirical and experimental perspectives on big data in recommendation systems: A comprehensive survey, *Big Data Mining and Analytics*, vol. 7, no. 3, pp. 964–1014, 2024.
[42] S. Börzsönyi, D. Kossmann, and K. Stocker, The skyline operator, in *Proc. 17th Int. Conf. Data Engineering*, Heidelberg, Germany, 2001, pp. 421–430.
[43] Z. Zheng, X. Wu, Y. Zhang, M. R. Lyu, and J. Wang, QoS ranking prediction for cloud services, *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1213–1222, 2013.
[44] E. Al-Masri and Q. H. Mahmoud, Investigating web services on the world wide web, in *Proc. 17th Int. Conf. World Wide Web*, Beijing, China, 2008, pp. 795–804.
[45] C. W. Hang and M. P. Singh, From quality to utility: Adaptive service selection framework, in *Proc. 8th Int. Conf. Service-Oriented Computing*, San Francisco, CA, USA, 2010, pp. 456–470.
[46] M. Lin, H. C. Lucas Jr, and G. Shmueli, Research commentary—Too big to fail: Large samples and the *p*-value problem, *Inf. Syst. Res.*, vol. 24, no. 4, pp. 906–917, 2013.

**Fugui He** received the PhD degree in computing application from Anhui University, China, in 2011. He is currently an associate professor at West Anhui University, China. His research interests include intelligent computing, positioning technology, wireless communication, Internet of Things technology, etc.

**Huiying Liu** received the bachelor degree from Guangxi University, China, in 2022. Now she is a master student at School of Computer Science and Technology, Anhui University, China. Her current research interests include cloud computing, recommendation systems, machine learning, and artificial intelligence.

**Yiwen Zhang** received the PhD degree in management science and engineering from Hefei University of Technology, China, in 2013. He is currently a full professor at School of Computer Science and Technology, Anhui University, China. He has published more than 70 papers in highly regarded journals and conferences, including *IEEE TKDE, IEEE TMC, IEEE TSC, ACM TOIS, IEEE TPDS, IEEE TNNLS, ACM TKDD,* SIGIR, ICSOC, ICWS, etc. His research interests include service computing, cloud computing, and big data analytics.