Majority-Rule-Based Web Service Selection

Karim Benouaret[1], Dimitris Sacharidis[2], Djamal Benslimane[1], and Allel Hadjali[3]

[1] Claude Bernard Lyon1 University, LIRIS, 69622 Villeurbanne, France
{karim.benouaret, djamal.benslimane}@liris.cnrs.fr
[2] IMIS, Athena Research Center, Marousi 15125, Greece
dsachar@imis.athena-innovation.gr
[3] Enssat, University of Rennes 1, IRISA, 22305 Lannion, France
allel.hadjali@enssat.fr

Abstract. Determining the appropriate service for a user request is a two step process. Initially, the available services whose description agrees with that of the request service are discovered. Then, the service selection process assists users in choosing the service that better matches their intention. In many practical situations, the responsibility to decide which is the appropriate service is shared among multiple parties, e.g., among the department heads of a university. The standard approach to such a service selection problem, is to discard services which are unanimously inappropriate, and return the rest. However, as the involved parties may have conflicting interests, it is possible that only few services are eliminated, and thus almost all discovered services need to be considered. This work addresses this shortcoming, by enforcing the *majority rule*: a service is discarded if the majority of the parties find it inappropriate. We formulate the majority-rule-based service selection problem based on the notions of dominance relationship and skyline. Furthermore, we propose an algorithm that (1) returns a more manageable set of services, eliminating many inappropriate ones, and (2) is more efficient that standard skyline techniques.

1 Introduction

Web data services, as a key technology for the development, deployment and management of Web services-based access to information systems, promise to enable maximal mashup, reuse, and sharing of structured data (e.g., relational tables), semi-structured information (e.g., XML documents) and unstructured information (e.g., commercial data from online business sources). Thereby, enabling users to perform several operations, e.g., data analysis, searches, purchases.

Consequently, it becomes apparent that the Web services paradigm rapidly gains popularity constituting an integral part of many real-world applications. For these reasons, several techniques for discovering Web services have been recently proposed. However, as Web data services (or services for short) and service providers proliferate, there will be a large number of candidate, most likely

competing, services for fulfilling a desired task. Thus, *service selection* is becoming important for helping users to identify desirable services. User preferences play a key role during the selection process [1–3]. However, in many practical situations, the responsibility to decide which is the appropriate service is shared among multiple parties, e.g., among the department heads of a university.

The following running example illustrates such a scenario, where a university decides to obtain a software license of a cloud-based data analytics service.

Table 1: User Preferences

User	Budget	Processes	Redundancy	Nodes
u_1	$[7000, 10000]$	$[5, 10]$	—	—
u_2	—	—	$[3, 5]$	—
u_3	—	$[8, 12]$	—	$[80, 100]$

Example 1. Consider a set of cloud-based data analytics Web services, and assume that several departments within a university wish to buy a license for one of them. The services are described by their annual *Cost*, the number of allowed simultaneous *Processes*, the level of data *Redundancy*, and the number of computing *Nodes*.

The users, in this case the department heads, have different preferences with respect to the service descriptions, as depicted in Table 1. User u_1, has a budget of $[7000, 10000]$ and expects to run simultaneously $[5, 10]$ processes; user u_2 cares much about data redundancy and expects a redundancy level of $[3, 5]$; user u_3 expects to run simultaneously $[8, 12]$ processes requiring $[80, 100]$ computing nodes. □

The service selection process follows two phases. In the first, given the user's preferences on service description attributes, the degrees of match between a requested and an available service (see e.g., [4–6]) are computed. In this work, we assume the Jaccard coefficient for matching service descriptions. If I_1, I_2 are two intervals, their Jaccard coefficient is $J(I_1, I_2) = \frac{|I_1 \cap I_2|}{|I_1 \cup I_2|}$, where $|I|$ measures the length of the interval [7].

Table 2: Discovered Services

Service	Cost	Processes	Redundancy	Nodes
s_1	$[7000, 11000]$	$[7, 12]$	$[3.5, 5.5]$	$[60, 110]$
s_2	$[5000, 10000]$	$[5, 11]$	$[4, 6]$	$[70, 115]$
s_3	$[6000, 12000]$	$[1, 10]$	$[4, 6]$	$[70, 110]$
s_4	$[8000, 12000]$	$[2, 12]$	$[3.5, 5]$	$[75, 130]$
s_5	$[9000, 15000]$	$[9, 12]$	$[4, 7]$	$[90, 130]$

Example 2. Returning to our example, consider that the set of relevant Web services are the ones depicted on Table 2. Each service is shown along with their description attributes. For instance service s_1 offers license plans that cost

$[7000, 11000]$ per year, allows $[7, 12]$ simultaneous processes, offers a redundancy level of $[7:2, 11:2]$, and allocates $[60, 110]$ computing nodes.

Based on the set of relevant service in Table 2 and the user requirements in Table 1, the service selection process computes the matching degrees between each user's specified preference and the corresponding service characteristic. For instance, the matching degree of service s_1 with respect to the Cost and Processes requirements of user u_1 are $\frac{|[7000,10000]|}{|[7000,11000]|} = 0.75$ and $\frac{|[7,10]|}{|[5,12]|} = 0.43$, respectively. □

Table 3: Matching Degrees of Services w.r.t. Users

Service	u_1: (Cost, Processes)	u_2: Redundancy	u_3: (Processes, Nodes)
s_1	$(0.75, 0.43)$	0.62	$(0.83, 0.41)$
s_2	$(0.67, 0.86)$	0.35	$(0.57, 0.47)$
s_3	$(0.57, 0.54)$	0.35	$(0.23, 0.51)$
s_4	$(0.50, 0.54)$	0.76	$(0.45, 0.38)$
s_5	$(0.57, 0.25)$	0.27	$(0.80, 0.27)$

The second phase of service selection is to identify the most interesting services w.r.t. users preferences. Most of service selection approaches focus on computing a score for each service as an aggregate of its individual matching degrees. Various approaches for aggregating the matching degrees exist. A common direction is to assign weights over different preference attributes; e.g., [8]. However, when multiple users are involved, it would be difficult to make tradeoffs between different weights. The natural option is to use the skyline operator [9–14] to determine an objectively good set of services. We refer to this set as the *unanimous service skyline*, and it contains all services which are not unanimously dominated. A service *unanimously dominates* another, if the former has higher matching degrees than the latter in all users' preferences.

Example 3. In our running example, service s_1 unanimously dominates service s_5, as s_1's matching degrees are higher. On the other hand, no other service is unanimously dominated. Hence, the skyline comprises services s_1, s_2, s_3 and s_4. □

Computing the unanimous service skyline frees users from assigning relative importance over different preference attributes. However, a major drawback is that, when multiple parties are involved, the number of services in the skyline becomes very large and no longer offers any interesting insights. The reason is that as the number of users and preferences increase, for any services s_i, s_j, it is more likely that s_i and s_j are incomparable, i.e., better than each other in different matching degree. It is thus crucial to further reduce the size of the service skyline.

The core of the above drawback is in the definition of dominance, which requires a unanimous verdict. To mitigate this, we choose to follow the majority rule. Informally, a service *majority-dominates* another, if the former has higher

matching degrees than the latter in the majority of users' preferences. Then, we naturally define the *majority service skyline*, as the services which are not majority-dominated.

To compute the majority service skyline, we make the observation that conventional skyline computation algorithms, with the exception of [15], cannot be adapted, due to the intransitivity of the majority-dominance relationship. Therefore, an extension of the algorithms in [15] can be used to compute the majority service skyline. However, we propose a novel algorithm for the service selection problem and show that it most cases it outperforms the extended algorithms.

Our main contributions are summarized as follows:

- We introduce a new concept for service selection when multiple preferences are involved, which is based on the majority rule, and is called the *majority service skyline*.
- We extend existing algorithms and propose a novel algorithm to efficiently compute the majority service skyline.
- We evaluate both the effectiveness of the proposed concept and the efficiency of our algorithm through a comprehensive experimental study.

The rest of the paper is structured as follows. Section 2 reviews related work. Section 3 introduces the problem of majority service skyline. Section 4 describes the majority service skyline computation algorithm. Section 5 presents our experimental study. Finally, Section 6 concludes the paper.

2 Related Work

In this section, we discus related work in the areas of preference-based service selection and skylines. We also highlight our contribution in these areas.

2.1 Preference-Based Service Selection

During the last years, the problem of preference-based service selection has received a lot of attention. The main objective is to provide users with the most relevant services, i.e., those that better satisfy their preferences, among the services retrieved by service discovery. Agarwal and Lamparter [16] propose an approach for an automated selection of services for service composition. Service compositions can be compared with each other and ranked according to the user preferences, where preferences are modeled as fuzzy IF-THEN rules. The IF part contains fuzzy descriptions of the various properties of a service, while the THEN part is one of the fuzzy characterizations of a special concept called Rank. A fuzzy rule describes which combination of attribute values a user is willing to accept to which degree, where attribute values and degree of acceptance are fuzzy sets. In [8], the authors indicate that they model service configurations and preferences more compactly using utility function policies, which allows drawing from multi-attribute decision theory methods to develop

an algorithm for optimal service selection. The authors also present the OWL ontology for the specification of configurable service offers and requests, and a flexible and extensible framework for optimal service selection that combines declarative logic-based matching rules with optimization methods, such as linear programming. In [2], the authors use a qualitative graphical representation of preferences, CP-nets, to deal with services selection in terms of user preferences. This approach can reason about users incomplete and constrained preferences. In [17], the authors propose a system for conducting qualitative service selection in the presence of incomplete or conflicting user preferences. The paradigm of CP-nets is used to model user preferences. The system utilizes the history of users to amend the preferences of active users, thus improving the results of service selection. In our recent work [1, 3], we propose an approach to automatically compose services, while taking into account the user preferences. User preferences are modeled using fuzzy sets. Different methods are investigated to compute the degrees of relevance of discovered services w.r.t. user's preferences. In order to select the most relevant services, a multi-criteria fuzzy dominance relationship is proposed to rank-order services. The selected services are then used to find the top-k service compositions. We propose also a method to improve the diversity of returned service compositions.

In the above line of work, the problem of reconciling multiple users preferences is not addressed. It is the focus of this work to propose a service selection solution that is both effective, i.e., it does not overwhelm the users, and efficient, i.e., it outperforms straightforward extensions of existing algorithms.

2.2 Skyline Computation

Skyline computation is related to problems from various fields, including the contour problem [18], maximal vectors [19] and convex hull [20]. The concept was re-introduced in the data management community by Börzsönyi et al. [21], where three external memory algorithms are presented, BNL, D&C and B-tree. Since then, several algorithms have been developed to compute the skyline. They can be categorized into non-index-based algorithms, e.g., SFS [22], LESS [23], SaLSa [24] and OSP [25], and index-based algorithms, e.g., Index [26], NN [27], BBS [28] and ZSearch [29].

As the size of the skyline may become too large, several variations have been investigated. Papadias et al. [30] propose the concept of k-dominating query, which retrieves the k points that dominate the largest number of other points. However, a k-dominating query does not always return skyline points. To resolve this, Lin et al. propose in [31] the top-k representative skyline, so that the k skyline points with the maximal number of dominated points can be produced. However, this approach often return similar points [32]. For diversifying the result, Tao et al. propose in [32] the distance based representative skyline. A similar approach is adapted in [10] for selecting services based on QoS. However, these approaches are only useful with anti-correlated datasets [10]. In [33], the authors propose the top-k skyline frequency. The skyline frequency of a point p is the number of subspaces where p is a skyline point. In a recent work [12], we

propose the concept of α-dominant skyline, which gives preference to services with a good compromise between QoS attributes. It also gives users the flexibility to control the size of the skyline. However, the problem of multiple users preferences is not addressed in these works. In [15], the authors relax the notion of dominance to k-dominance, so that more points are dominated. A point p is said to k-dominate another point q iff there are k dimensions on which p dominates q. The k-dominant skyline then consists of the subset of points that are not k-dominated.

The problem of computing the majority service skyline can be solved by an adaptation of the algorithms proposed in [15], since the majority-dominance relationship exhibits similar properties, most particularly intransitivity, with the k-dominance relationship. However, as our experimental evaluation demonstrates, our algorithm is more efficient that such an adaptation.

3 Problem Definition

In this section, we provide the basic notions used throughout this paper, and formalize the notion of *majority service skyline*.

We assume a set of users $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$, and a set of discovered services $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$. We use $s_i.u_k$ to denote the matching degrees of service s_i w.r.t. user u_k. For instance, the matching vector of s_1 w.r.t. u_1 is $s_1.u_1 = (0.75, 0.43)$.

Definition 1. *(Weak Dominance)*
Given a user u_k, we say that service s_i weakly dominates *s_j w.r.t. u_k, denoted as $s_i.u_k \succeq s_j.u_k$, iff s_i has better matching degrees than s_j on all specified preference attributes.*

Definition 2. *(Dominance)*
Given a user u_k, we say that service s_i dominates *s_j w.r.t. u_k, denoted as $s_i.u_k \succ s_j.u_k$, iff s_i has better matching degrees than s_j on all specified preference attributes, and strictly better matching degree on at least one.*

Definition 3. *(Unanimous Dominance)*
Given a set of users $\mathcal{U}$, we say that service s_i unanimous-dominates *s_j, denoted as $s_i \succ_U s_j$, iff s_i weakly dominates s_j w.r.t. all users, i.e., $\forall u_k \in \mathcal{U}$ $s_i.u_k \succeq s_j.u_k$, and there exists one user, say u'_k, for which s_i dominates s_j, i.e., $\exists u'_k \in \mathcal{U}$ $s_i.u'_k \succ s_j.u'_k$.*

Definition 4. *(Unanimous Service Skyline)*
Given a set of discovered services $\mathcal{S}$ and a set of users $\mathcal{U}$, the unanimous service skyline *$USS(\mathcal{S},\mathcal{U})$ comprises the set of services that are not dominated by any other.*

In the following, we introduce the concept of majority rule in the service selection process and alter the definitions of dominance and skyline.

Definition 5. *(Majority Dominance)*
Given a set of users $\mathcal{U}$, we say that service s_i majority-dominates *s_j, denoted as $s_i \succ_M s_j$, iff (1) there exists a subset $\mathcal{U}' \subseteq \mathcal{U}$ containing more than half of the users such that s_i weakly dominates s_j w.r.t. all users in this subset, i.e., $|\mathcal{U}'| > \lfloor |\mathcal{U}|/2 \rfloor$ and $\forall u_k \in \mathcal{U}'$ $s_i.u_k \succeq s_j.u_k$, and (2) there exists one user, say u'_k, for which s_i dominates s_j, i.e., $\exists u'_k \in \mathcal{U}$ $s_i.u'_k \succ s_j.u'_k$.*

Definition 6. *(Majority Service Skyline)*
Given a set of discovered services $\mathcal{S}$ and a set of users $\mathcal{U}$, the majority service skyline *$MSS(\mathcal{S},\mathcal{U})$ comprises the set of services that are not majority-dominated by any other.*

Example 4. Returning to our example, s_1 majority-dominates services s_3, s_4 and s_5, while, services s_1 and s_2 are not majority-dominated by any other service. Thus, services s_1 and s_2 form the majority service skyline. Recall that the unanimous service skyline comprises services s_1, s_2, s_3 and s_4. Observe that the MSS has smaller cardinality than the USS. □

We now provide the formal definition for the service selection problem for multiple users.

Problem statement: Given a set of users $\mathcal{U}$ and a set of discovered services $\mathcal{S}$, compute the *majority service skyline*.

4 Computing the Majority Service Skyline

In this section, we first introduce some important observations regarding the problem at hand. We then develop an algorithm based on these observations for efficiently computing the majority service skyline.

4.1 Observations

We make some observations regarding the majority dominance relationship.

Lemma 1. *If s_i unanimous-dominates s_j, then s_i majority-dominates s_j. i.e., $s_i \succ_U s_j \Rightarrow s_i \succ_M s_j$.*

Proof. Proof follows from Definition 1 and Definition 3, setting $\mathcal{U}' = \mathcal{U}$.

Theorem 1. *The majority service skyline is a subset of the unanimous service skyline. i.e., $MSS(\mathcal{S},\mathcal{U}) \subseteq USS(\mathcal{S},\mathcal{U})$.*

Proof. Assume that there exists a service s_i, such that $s_i \in MSS(\mathcal{S},\mathcal{U})$ and $s_i \notin USS(\mathcal{S},\mathcal{U})$. Since $s_i \notin USS(\mathcal{S},\mathcal{U})$, there must exist a service s_j, such that $s_j \succ_U s_i$. Thus, by Lemma 1, we have $s_j \succ_M s_i$. Which leads to a contradiction, as $s_i \in MSS(\mathcal{S},\mathcal{U})$.

Table 4: Example of Cyclic Majority Dominance

Service	u_1: (Cost, Processes)	u_2: Redundancy	u_3: (Processes, Nodes)
s_a	$(0.76, 0.69)$	0.74	$(0.58, 0.80)$
s_b	$(0.56, 0.64)$	0.70	$(0.78, 0.86)$
s_c	$(0.80, 0.88)$	0.68	$(0.72, 0.76)$
s_d	$(0.78, 0.86)$	0.61	$(0.75, 0.89)$

Moreover, observe that the majority dominance relationship does not maintain the transitive property, as services can exhibit a *cyclic majority dominance relationship*.

Theorem 2. *It is possible to have a set of users $\mathcal{U}$ and a set of discovered services $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ such that s_1 majority-dominates s_2, s_2 majority-dominates s_3, ..., s_{n-1} majority-dominates s_n and s_n majority-dominates s_1, i.e., forming a cyclic majority dominance relationship.*

Proof. The example in Table 4, where $s_a \succ_M s_b$, $s_b \succ_M s_c$, $s_c \succ_M s_d$, and $s_d \succ_M s_a$, proves the claim.

The above theorem shows that the majority dominance relationship shares the cyclic property of the k-dominance relationship introduced in [15]. Therefore, a service cannot be discarded even if it is majority-dominated because it might be needed for excluding other services. This justifies why the existing algorithms for computing the skyline are not applicable for computing the majority service skyline. However, the one scan algorithm (OSA) and two scan algorithm (TSA) of [15], can be adapted to compute the majority service skyline, by exchanging k-dominance checks for majority dominance checks as defined in Section 3. In the following, we denote as OSA and TSA the adaptations of the algorithms in [15] to computing the majority service skyline.

4.2 Majority Service Skyline Algorithm

In this section, we introduce the *Majority Service Skyline Algorithm* (MSA), which improves on OSA by employing the following properties.

Lemma 2. *if s_i unanimous-dominates s_j and s_j majority-dominates s_k, then s_i majority-dominates s_k. i.e., $s_i \succ_U s_j \wedge s_j \succ_M s_k \Rightarrow s_i \succ_M s_k$.*

Proof. As s_j majority-dominates s_k means that s_j dominates s_k w.r.t. more than half of users, and s_i unanimous-dominates s_j means that s_i dominates s_j w.r.t. all users. Thus, s_i dominates s_k w.r.t. more than half of users. Hence, s_i majority-dominates s_k.

Lemma 3. *Let $f : \mathcal{S} \to \mathbb{R}^+$ be a monotone function aggregating the matching degrees of s_i for all users. If s_i unanimous-dominates s_j, then $f(s_i) > f(s_j)$. i.e., $s_i \succ s_j \Rightarrow f(s_i) > f(s_j)$.*

Proof. The fact that s_i unanimous-dominates s_j means that s_i is better than or equal to s_j with respect to all preference attributes of all users. This implies that a monotone aggregate function over the matching degrees of s_i has a greater value than that function over the matching degrees of s_j. Hence, $f(s_i) > f(s_j)$.

From Lemma 1 and Lemma 2, we can see that it is sufficient to compare each service against the unanimous skyline services to detect if it is part (or not) of the majority service skyline. This essentially reduces the number of comparisons. Specifically, *if a service s_i is unanimous-dominated, then discard it as (1) it is not part of the majority service skyline (Lemma 1), and (2) it is unnecessary for eliminating other services (Lemma 2).*

Lemma 3 also helps reduce unnecessary comparisons. In fact, to exploit this property, we sort the services in non-ascending order of the sum of their matching degrees. Then, given a service s_i, searching for services by which s_i is unanimous-dominated can be limited to the part of the service before s_i. This is the idea behind the SFS algorithm [22], which in this context we apply it for cyclic dominance relationships.

The MSA algorithm leverages the observations made above to compute efficiently the majority service skyline. Based on Lemma 1 and Lemma 2, MSA maintains two sets $\mathcal{R}$ and $\mathcal{T}$, containing respectively the set of intermediate majority service skyline services and the set of intermediate unanimous skyline services that are not in $\mathcal{R}$. Thus, $\mathcal{R} \cup \mathcal{T}$ constitutes the intermediate unanimous skyline.

Algorithm 1: MSA

Input: *set of users $\mathcal{U}$; set of discovered services $\mathcal{S}$;*
Output: *majority service skyline $\mathcal{R}$;*

```
1  sort S in a non-ascending order of the sum of services' matching degrees;
2  R ← ∅; T ← ∅;
3  while S is not empty do
4      extract the top service s_i from S;
5      if s_i is unanimous-dominated by any service in R ∪ T then
6          discard s_i;
7      else
8          if s_i majority-dominates any service s_j in R then
9              remove s_j from R to T;
10         if s_i is majority-dominated by any service in R ∪ T then
11             insert s_i into T;
12         else
13             insert s_i into R;
14 return R;
```

The details of MSA depicted in Algorithm 1 are as follows. First, services in $\mathcal{S}$ are sorted in a non-ascending order of the sum of their matching degrees, and both sets $\mathcal{R}$ and $\mathcal{T}$ are initialized to empty sets. Then, the top service (i.e., the service with the maximum sum of matching degrees), say s_i, is extracted from $\mathcal{S}$. Service s_i is compared against services in $\mathcal{R} \cup \mathcal{T}$, i.e., the set of services that may unanimous-dominate s_i (as the other services cannot dominate s_i from Lemma 3). If s_i is unanimous-dominated, then it is removed from $\mathcal{S}$ as it is not part of the majority service skyline (Lemma 1) and it is unnecessary for eliminating other services (Lemma 2). Otherwise, i.e., when s_i is not unanimous-dominated by any service in $\mathcal{R} \cup \mathcal{T}$, if s_i majority-dominates any service s_j in $\mathcal{R}$ (i.e., s_j is not a service in MSS), then s_j is removed from $\mathcal{R}$ to $\mathcal{T}$, as it is a unanimous skyline service, thus useful for eliminating other services. For the same reason, if s_i is *majority-dominated* by any service in $\mathcal{R} \cup \mathcal{T}$, it is inserted into $\mathcal{T}$ as it is not part of the majority service skyline. Else, s_i is an intermediate MSS service and is thus inserted into $\mathcal{R}$. Once all services in $\mathcal{S}$ have been examined, i.e., $\mathcal{S}$ is empty, services in $\mathcal{R}$ form the majority service skyline, and $\mathcal{R}$ is returned.

Example 5. Applying MSA on our example, services s_1 and s_2 will be inserted into $\mathcal{R}$, while, services s_3 and s_4 will be inserted into $\mathcal{T}$ since they are both *majority-dominated* by service s_1, but they are skyline services. On the other hand, service s_5 is discarded as it is dominated by service s_1. Thus, the algorithm correctly returns services s_1 and s_2 as the majority service skyline. □

5 Experimental Evaluation

In this section, we present an extensive experimental evaluation of our approach. Our objective is to prove the *effectiveness* of the majority service skyline and the *efficiency* of the proposed algorithm. More specifically, we focus on two issues. (1) The size of the majority service skyline (denoted as MSS). To demonstrate that the majority service skyline further reduces the size of the (traditional) skyline, we also compute the size of the unanimous skyline (denoted as USS) to compare how their sizes varies. (2) The performance of our algorithm in terms of elapsed time for computing the majority service skyline. For comparison purposes, we also implemented the adaptations of OSA and TSA [15] for computing the majority service skyline.

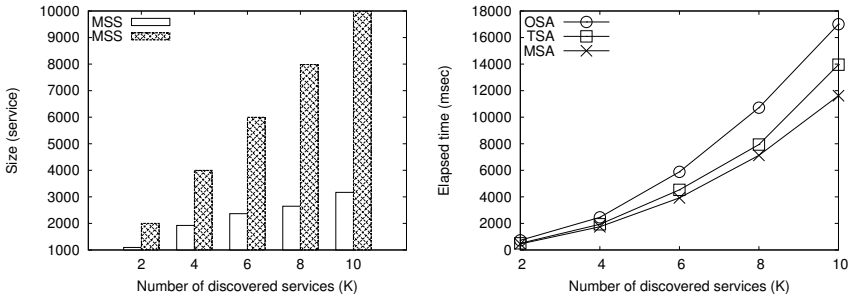
Table 5: Parameters and Examined Values

Parameter	Symbol	Range	Default
Number of discovered services	n	$[2, 10]$K	5K
Number of users	m	$[3, 7]$	5
Number of preferences per user	d	$[3, 7]$	5

5.1 Experimental Setup

It is worth noting that, due to the limited availability of real-world service data, most existing skyline-based service selection approaches, e.g., [9, 11–14], use synthetic datasets for their evaluation. For ease of comparison, we also follow this direction. The service generator we use takes as input a (real-world) model service and its associated constraints, representing the requested service and the multiple users preferences, and produce a set of synthetic services, as well as their associated constraints, representing the set of discovered services. The Jaccard coefficient is used for computing the matching degrees between discovered service' constraints and users preferences. The generation of the sets of synthetic services is controlled by the parameters in Table 5, which displays the parameters under investigation, their corresponding ranges and their default values. In each experimental setup, we investigate the effect of one parameter, while setting the remaining ones to their default values.

The service generator and the algorithms, i.e., MSA, OSA and TSA were implemented in Java, and all experiments were conducted on a 2.3 GHz Intel Core i5 processor.

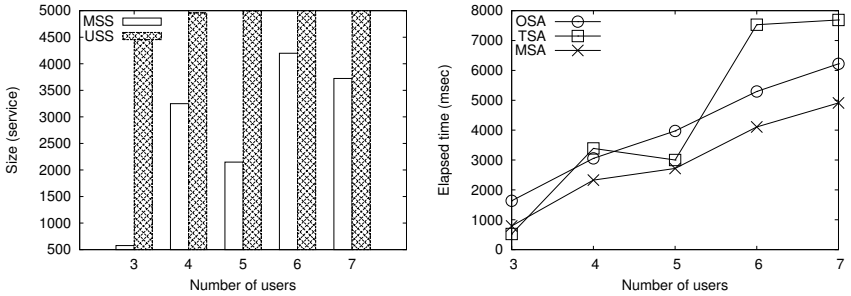


(a) Size (b) Elapsed time

Fig. 1: Effects of n.

5.2 Effect of n

Figure 1 depicts the effect of n. As shown in Figure 1a, the size of the majority service skyline increases slightly with n. This is because as n varies, it is becoming more difficult to find services which are majority-dominated. Figure 1a shows also that the size of the majority service skyline is very smaller then that of the skyline, which is almost equal to the number of discovered services, as the skyline cannot discard all inappropriate services, while the majority service skyline includes only the most interesting ones. As depicted in Figure 1b, the execution time of the algorithms increases with n. However, MSA consistently outperforms OSA and TSA.

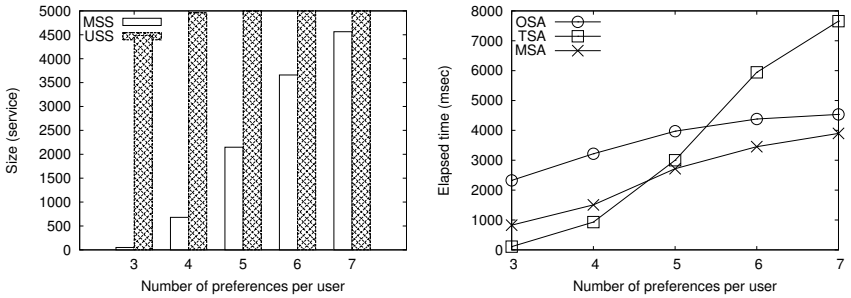


(a) Size (b) Elapsed time

Fig. 2: Effects of m.

5.3 Effect of m

Figure 2 shows the effect of m. Figure 2a shows a fluctuation in the size of the majority service skyline. The fluctuation is related to the definition of the majority dominance relationship (Definition 5). Indeed, we can distinguish two trend. One for the even values of m, and the second for the odd values of m; each trend increases with the increase of m. This is because, if we have an odd value of m, say m_o, and an even value of m, say m_e, such that $m_o = m_e + 1$, then the percentage of most of users for m_e is greater than that of m_o. For example, for $m = 4$, the percentage is $\frac{3}{4} = 0.75\%$, and for $m = 5$ the percentage is $\frac{3}{5} = 0.60\%$. When this percentage is large, small number of services is discarded, and vice versa. Also, note that the size of the majority service skyline is very smaller then that of the unanimous service skyline, which approximates the number of discovered services for $m \geq 4$. As shown in Figure 2b, when m increases, the performance of TSA deteriorates due to the second scan performed. However, the execution time of OSA and MSA increases slightly with m. Still, MSA is better.



(a) Size (b) Elapsed time

Fig. 3: Effects of d.

5.4 Effect of d

Figure 3 shows the effect of d. As depicted in Figure 3a, the size of the majority service skyline increases significantly with the increase of d. This is because as d increases, a service has increased probability not to be dominated in all preference attributes w.r.t. a given user. However, the size of the majority service skyline remains smaller than that of the unanimous service skyline, which approximates the number of discovered services for $d \geq 4$. As shown in Figure 3b, TSA is better than OSA and MSA for $d \leq 4$ since the size of the majority service skyline is small, thus a large number of services can be eliminated in the first scan. However, TSA does not scale with d as the size of the majority service skyline becomes large, thus the second scan is very time consuming. The execution time of OSA and MSA, on the other hand, increases slightly with d. Also, observe that MSA consistently performs better than OSA.

6 Conclusion

In this paper, we deal with the problem of preference-based Web service selection under multiple users preferences. We introduce a novel concept for this problem based on the majority rule. This allows users to make a "democratic" decision on which services are the most appropriate. We develop a suitable algorithm for the majority-rule-based Web selection problem. Our experimental evaluation demonstrates the effectiveness of the concept and the efficiency of the algorithm.

References

1. K. Benouaret, D. Benslimane, A. Hadjali, and M. Barhamgi, "Fudocs: A web service composition system based on fuzzy dominance for preference query answering," *PVLDB*, vol. 4, no. 12, pp. 1430–1433, 2011.
2. H. Wang, J. Xu, and P. Li, "Incomplete preference-driven web service selection," in *IEEE SCC (1)*, 2008, pp. 75–82.
3. K. Benouaret, D. Benslimane, A. Hadjali, and M. Barhamgi, "Top-k web service compositions using fuzzy dominance relationship," in *IEEE SCC*, 2011, pp. 144–151.
4. M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara, "Semantic matching of web services capabilities," in *International Semantic Web Conference*, 2002, pp. 333–347.
5. L. Li and I. Horrocks, "A software framework for matchmaking based on semantic web technology," in *WWW*, 2003, pp. 331–339.
6. X. Dong, A. Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Simlarity search for web services," in *VLDB*, 2004, pp. 372–383.
7. R. O. Duda and P. E. Hard, *Pattern Classifcation and Scene Analysis.* New York: A Wiley-Interscience Publication, 1973.
8. S. Lamparter, A. Ankolekar, R. Studer, and S. Grimm, "Preference-based selection of highly configurable web services," in *WWW*, 2007, pp. 1013–1022.
9. Q. Yu and A. Bouguettaya, "Computing service skyline from uncertain qows," *IEEE T. Services Computing*, vol. 3, no. 1, pp. 16–29, 2010.
10. M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for qos-based web service composition," in *WWW*, 2010, pp. 11–20.

11. Q. Yu and A. Bouguettaya, "Computing service skylines over sets of services," in *ICWS*, 2010, pp. 481–488.
12. K. Benouaret, D. Benslimane, and A. Hadjali, "On the use of fuzzy dominance for computing service skyline based on qos," in *ICWS*, 2011, pp. 540–547.
13. Q. Yu and A. Bouguettaya, "Multi-attribute optimization in service selection," *World Wide Web*, vol. 15, no. 1, pp. 1–31, 2012.
14. Q. Yu and A. Bouguettaya, "Efficient service skyline computation for composite service selection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 99, no. PrePrints, 2011.
15. C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "Finding k-dominant skylines in high dimensional space," in *SIGMOD Conference*, 2006, pp. 503–514.
16. S. Agarwal and S. Lamparter, "User preference based automated selection of web service compositions," in *ICSOC Workshop on Dynamic Web Processes*, K. V. A. S. M. Z. C. Bussler, Ed. Amsterdam, Netherlands: IBM, Dezember 2005, In proceedings, pp. 1–12.
17. H. Wang, S. Shao, X. Zhou, C. Wan, and A. Bouguettaya, "Web service selection with incomplete or inconsistent user preferences," in *ICSOC/ServiceWave*, 2009, pp. 83–98.
18. D. H. McLain, "Drawing contours from arbitrary data points," *Comput. J.*, vol. 17, no. 4, pp. 318–324, 1974.
19. H. T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *J. ACM*, vol. 22, no. 4, pp. 469–476, 1975.
20. F. P. Preparata and M. I. Shamos, *Computational Geometry - An Introduction*. Springer, 1985.
21. S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *ICDE*, 2001, pp. 421–430.
22. J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *ICDE*, 2003, pp. 717–719.
23. P. Godfrey, R. Shipley, and J. Gryz, "Maximal vector computation in large data sets," in *VLDB*, 2005, pp. 229–240.
24. I. Bartolini, P. Ciaccia, and M. Patella, "Efficient sort-based skyline evaluation," *ACM Trans. Database Syst.*, vol. 33, no. 4, 2008.
25. S. Zhang, N. Mamoulis, and D. W. Cheung, "Scalable skyline computation using object-based space partitioning," in *SIGMOD Conference*, 2009, pp. 483–494.
26. K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient progressive skyline computation," in *VLDB*, 2001, pp. 301–310.
27. D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in *VLDB*, 2002, pp. 275–286.
28. D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *SIGMOD Conference*, 2003, pp. 467–478.
29. K. C. K. Lee, B. Zheng, H. Li, and W.-C. Lee, "Approaching the skyline in z order," in *VLDB*, 2007, pp. 279–290.
30. D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 41–82, 2005.
31. X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting stars: The k most representative skyline operator," in *ICDE*, 2007, pp. 86–95.
32. Y. Tao, L. Ding, X. Lin, and J. Pei, "Distance-based representative skyline," in *ICDE*, 2009, pp. 892–903.
33. C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "On high dimensional skylines," in *EDBT*, 2006, pp. 478–495.