Selecting Services for Multiple Users: Let's Be Democratic

Karim Benouaret, Dimitris Sacharidis, Djamal Benslimane and Allel Hadjali

Abstract—Service selection is a challenging task, and a lot of effort has been devoted to tools that assist the user in choosing the service whose non-functional parameters better match her/his preferences. In many practical situations, the responsibility to decide which is the appropriate service is shared among multiple parties. A standard approach to this service selection problem is to discard services that are unanimously considered inappropriate and focus on the rest. However, as the involved parties may have colliding interests, only a few services may be eliminated. This work addresses this shortcoming and enables users to reach a "democratic" decision by means of a majority vote: a service is eliminated if the majority of the parties find it inappropriate. We formulate the problem using dominance relationships, and propose algorithms that return an appropriate subset of services for the parties, while being more efficient than standard techniques. Moreover, we consider the problem of defining an appropriate ranking for the non eliminated services, and formulate it as an instance of a group recommendation problem. Finally, we demonstrate the effectiveness and the efficiency of our approach through extensive experimental evaluation on real-based and synthetic datasets.

Index Terms—Service Selection, Preferences, Pareto Dominance, Group Recommendation

1 Introduction

Service Oriented Computing [1] and Cloud Computing [2] are dominant technologies in software and Internet-based applications, which present distinct advantages to end users, whether they are individuals, private or public organizations [3]. As this market sees high demand, service providers compete with each other and offer services at different price and performance levels [4]. Consequently, end users are often faced with a huge number of candidate services for fulfilling a desired task. For instance, a popular service directory[1] classifies almost 20,000 application programming interfaces, with the most popular categories containing a few thousands of entries. Therefore, *non-functional* properties of services, such as quality of service (QoS), constitute an important differentiating factor [5].

When the number of services providing equivalent functionality is very large, browsing all competing services to find the most interesting service is impractical, time consuming, and costly. Therefore, service selection has become important for helping users identify desirable services according to their preferences on non-functional parameters. Several approaches have been proposed for the problem of selecting services according to users' preferences; see [6], [7] for a survey on them. While most of the proposed studies seek to satisfy the preferences of a single user, in many practical situations the responsibility to decide which is the appropriate service is shared among multiple parties, e.g., among departments in an organization. A similar problem arises when the same service is to be used in multiple use-cases with differing requirements, e.g., by different applications [8]. In such settings, service selection for multiple users is required.

TABLE 1: User Preferences

User	Response Time (ms)	Redundancy
u_1	≤ 400	$2\times$
u_2	≤ 350	$4\times$
u_3	≤ 300	$3\times$

TABLE 2: Relevant Services

User	Response Time (ms)	Redundancy
s_1	$[150, 450]$	$3\times$
s_2	$[150, 300]$	$2\times$
s_3	$[200, 450]$	$4\times$
s_4	$[300, 800]$	$2\times$
s_5	$[100, 400]$	$2\times$
s_6	$[300, 800]$	$2\times$
s_7	$[250, 800]$	$2\times$
s_8	$[300, 600]$	$1\times$
s_9	$[400, 800]$	$1\times$

Example 1. As a running example, consider an organization, consisting of three departments, that wishes to purchase cloud storage service license among several options. The services are described by their response time (in ms) and level of data redundancy they offer. The users, in this case the department chairs, have different preferences with respect to the service parameters, as depicted in Table 1. For instance, user u_1 prefers a service with response time no more than 400ms, offering at least $2\times$ redundancy.

Preference-based service selection is a two-phase process. First, given the users' preferences on service description attributes, the degrees of match between a requested and available services are computed; see e.g., [9], [10], [11], [12].

- K. Benouaret and D. Benslimane are with Univ Lyon, Université Claude Bernard Lyon 1, CNRS, LIRIS, F-69622, Villeurbanne, France
E-mail: karim.benouaret@liris.cnrs.fr; djamal.benslimane@liris.cnrs.fr
- D. Sacharidis is with TU Wien, E-Commerce Research Division, A-1040 Vienna, Austria
E-mail: dimitris@ec.tuwien.ac.at
- A. Hadjali is with Ensma, LAAS, 86360, Chasseneuil-du-Poitou, France
E-mail: allel.hadjali@ensma.fr

1. www.programmableweb.com [Accessed Jun. 2018]

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

Service	User		
	u_1	u_2	u_3
s_1	$(0.83, 1)$	$(0.67, 0)$	$(0.50, 1)$
s_2	$(1.00, 1)$	$(1.00, 0)$	$(1.00, 0)$
s_3	$(0.80, 1)$	$(0.60, 1)$	$(0.40, 1)$
s_4	$(0.20, 1)$	$(0.10, 0)$	$(0.00, 0)$
s_5	$(1.00, 1)$	$(0.83, 0)$	$(0.67, 0)$
s_6	$(0.20, 1)$	$(0.10, 0)$	$(0.00, 0)$
s_7	$(0.27, 1)$	$(0.18, 0)$	$(0.09, 0)$
s_8	$(0.33, 0)$	$(0.17, 0)$	$(0.00, 0)$
s_9	$(0.00, 0)$	$(0.00, 0)$	$(0.00, 0)$

Example 2. Consider the set of relevant services depicted on Table 2. Each service is shown along with its non-functional parameters. For instance, service s_1 exhibits a response time in the range of $[150, 450]$ ms and offers up to $3\times$ redundancy.
Based on the set of relevant services in Table 2 and the users' preferences in Table 1, the service selection process computes the matching degrees between each user's specified preference and the corresponding service characteristic. Assume that matching degrees for response time are computed as the probability that the service will match the desired user preference. Let $s_i.p_1$ represent the range of response times for service s_i, and $u_x.p_1$ the desired response time of user u_x; then the matching degree is computed as $\mu_i^x.p_1 = \frac{|s_i.p_1 \cap [0, u_x.p_1]|}{|s_i.p_1|}$, i.e., the ratio of the service's response time range that is desirable. The matching degrees for redundancy are boolean and simply indicate whether the service satisfies the user's preference. All matching degrees are shown in Table 3.

The second phase of preference-based service selection is to identify the most interesting services with respect to users' preferences. Considering a user independently, 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., [10], [13], [14], [15].

When all users are taken into consideration, applying a similar method, as done in [16], [17], enforces an additional level of aggregation, the first being across attributes, and the second across users. This can obscure and blur the individual preferences per attribute of each user. In addition, as the number of involved parties increases, it becomes more difficult to make tradeoffs between different weights as conflicting preferences are more likely to appear.

To overcome these limitations, the natural option is the use of the skyline operator to determine the Pareto optimal set of services [18]. 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 matching degrees as good as or better than the latter regarding all users' preferences, and better on at least one user's preference.

Example 3. Service s_1 unanimously dominates services s_4, s_6, s_7, s_8 and s_9. Likewise, service s_2 unanimously dominates service s_5. Note that services s_1, s_2, and s_3 are not unanimously dominated, hence they comprise the unanimous service skyline.

The unanimous service skyline eliminates services which all users agree they are not interesting. Nonetheless, when a large number of parties is involved, the number of services in the skyline becomes very large and no longer offers any interesting insights. As the number of users' preferences increases, for any pair of services, it becomes more likely that they are incomparable, being better than each other at different matching degrees. In such settings, it becomes imperative to further reduce the number of returned services.

To address this drawback, we proposed in [19] to relax the requirement for unanimity, and follow the majority rule. Informally, a service *majority dominates* another, if the former has matching degrees as good or better than the latter regarding the majority of users. Then, we naturally define the *majority service skyline*, as the set of services which are not majority dominated. Thereby, we allow users to make a "democratic" decision on which services are not appropriate, so as to exclude them.

To compute the majority service skyline, we make the observation that conventional skyline computation algorithms, with the exception of the methods proposed in [20], cannot be adapted, due to the intransitivity of the majority-dominance relationship (see Section 4). Motivated by this fact, in [19] we adapted the algorithms in [20] to form the baseline (BA) solutions to our problem. Moreover, we proposed a novel method, termed Sort-Based Algorithm (SBA), that features additional pruning criteria to optimize the extraction of the majority service skyline.

In this paper, we go one step forward and propose a novel method, termed Bounds-Based Algorithm (BBA), that computes bounds on the matching vectors and employs a new dominance check. Based on these bounds, BBA is able to perform fewer comparisons between services, and also check for an early termination condition, so as to avoid examining certain services that are definitely majority dominated.

We then turn our attention to a related problem, that of providing a ranking among services. This is an important presentation task, because users need an effective way to examine the results even if they are much fewer than those returned by conventional (not majority-based) methods. For this task, one can apply the current state-of-the-art in group recommendation techniques [21], [22], which try to construct an optimal ranking that satisfies all group members at the same time. Our contribution consists of fusing the concept of majority service skyline, which by itself does not induce any relative order, with current group recommendation techniques, resulting in a method that produces more effective ranking of services compared to state-of-the-art group recommenders.

The main contributions of our line of work are: We introduce a new concept for service selection when multiple users with different preferences are involved, based on majority rule, and called *majority service skyline*. We present two baseline methods by adapting prior work, and also propose two novel algorithms to efficiently compute the majority service skyline. We show how to address the problem of returning a ranked list of services that satisfies all parties involved. We evaluate the effectiveness of the majority service skyline and our ranking mechanism using real-based semi-synthetic datasets. Specifically, we find that filtering with majority dominance consistently improves the quality of the ranking list w.r.t. the ground truth, when compared to lists produced without filtering or filtering with unanimous dominance. We evaluate the efficiency and scalability of all proposed algorithms through a comprehensive experimental study on synthetic datasets. We find that BBA is up to 34% faster than the current state-of-the-art SBA, and up to 2 orders of magnitude more efficient than the baselines.

The remainder of this paper is structured as follows. Section 2 reviews the related work. Section 3 formally defines the problem of majority service skyline. Section 4 describes the majority service skyline computation algorithms. Then, Section 5 introduces our methodology for ranking services. Section 6 presents our experimental study. Section 7 concludes the paper and supplies some future work directions.

2 Background and Related Work

Web services are an established technology for enabling applications to exchange data and integrate with one another. The description of a service enables users and machines to identify the most appropriate service for a particular task. Such a description should comprise two main parts [23]. The functional description describes the operational characteristics of the service, while the non-functional description focuses on the supply of the non-functional capabilities of the service through the supply of respective constraints on corresponding *Non-Functional Parameters* (NFP), including *Quality of Service* (QoS) aspects, such as response time, reliability, availability. The process of identifying services that satisfy the functional requirements of a task is called *functional matchmaking* (e.g., [24]), and involves examining the functional description of services. The process of identifying the most appropriate service among functionally equivalent or similar services is called *service selection*, and involves examining the non-functional descriptions of services. Service selection can either correspond to the *local* problem of selecting a single service from a set of candidate functionally similar candidate services, or to the *global* problem of selecting appropriate services to compose so as to satisfy the requirements of the whole application [25]. Our work concerns the local service selection problem. In the following, we review relevant work.

QoS-based Service Selection. Once functionally equivalent services are identified, the next step differentiates among them using non-functional descriptions, such as QoS.

For the local problem, [26] proposes an extensible QoS computation model distinguishing generic quality criteria and domain related criteria so that new specific criteria can be added and used to evaluate the QoS of web services without changing the computation model. The work in [27] introduces QoS-based selection of semantic services, presenting a QoS ontology and selection method using an optimum normalization algorithm. In [10], a QoS-based service contracting framework is proposed. The work in [14] presents a model where users are allowed to specify their QoS requirements on each QoS parameter as a range of acceptable values along with an importance weight and uncertainty, rather than a single value indicating the required QoS. In [28], a model for service selection using the QoS history is proposed. Specifically, the QoS history is partitioned into several time slots and for each of these slots a service selection decision is made. Then, all decisions are aggregated to determine the overall optimal service.

For the global problem, [29] proposes a selection model, based on linear programming, to find the optimal selection of component services. The work in [30] considers an extended linear programming model that is able to fulfil constraints at runtime through adaptive reoptimization under varying QoS characteristics. In [31], the authors propose two models for the QoS-based service composition problem, a combinatorial model and a graph model, and introduce a heuristic algorithm for each. In [32], the authors propose a hybrid approach that combines global optimization with local selection so as to find a close-to-optimal selection efficiently. First, the authors use mixed integer programming to find the optimal decomposition of global into these local QoS constraints. Second, they use distributed local selection to find the best web services that satisfy these local constraints.

Estimating QoS for Services. A related line of work deals with determining or estimating QoS values for services. The idea is to use historical QoS values from other services and other users in order to predict the expected QoS values for a target service and user. Therefore, collaborative filtering techniques that exploit similarities between services and users are employed. In [33], the authors propose a measure which identifies similar users (or web services) more accurately and leads to better QoS value prediction. [34] proposes a localization-based approach assuming that users in the same geographic area will have the same QoS values to predict the best quality and recommend services to the user. [35] predicts the QoS ranking instead of predicting the QoS values. We note that our methodology, similar to these methods, borrows ideas from recommender systems, but uses a different technique (aggregation of preferences) and applies it to a different problem (preference-based service selection for multiple users).

In contrast to the previous methods, where the goal is to predict the expected QoS a user will experience from a given service, another line of work tries to determine a single objective QoS value (or description) for a given service. Therefore, methods for reaching a consensus are employed. Lin et al. provide in [36] a clustering-based approach for QoS consensus decision making, while allowing consumers to express fuzzy opinions. In [37], the authors propose to use the power of crowdsourcing to assess the QoS of candidate services and facilitate the process of service selection. They adopt a group decision making technique to guarantee that the assessment does not suffer from subjective and dishonest evaluations. In [38], the authors use interval-valued intuitionistic fuzzy numbers for modeling the subjectivity and imprecision of the assessment, and develop an algorithm based on the TOPSIS method and the Choquet integral operator for evaluating cloud services. We note that these methods share similar ideas with our approach (namely, preference aggregation and multi-objective analysis), but are focused on a different problem.

Preference-based Service Selection. Another stream of work focuses on modeling richer user preferences on the non-functional aspects of services. Once preferences are expressed, they are matched to functionally similar services and a degree of match (utility) is determined. Then, services are ranked in decreasing order of their utility.

For the local selection problem, [13] models service configurations and preferences 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. [39] uses a qualitative graphical representation of preference, CP-nets, to deal with services selection in terms of user preferences. This approach can reason over a user's incomplete and constrained preferences. [40] proposes a system for conducting qualitative service selection in the presence of incomplete or conflicting user preferences, using CP-nets to model user preferences. The system utilizes the history of users to amend the preferences of active users, thus improving the results of the service selection.

For global selection, [41] proposes an approach for an automated selection of services for service composition, where preferences are modeled as fuzzy if-then rules. A fuzzy rule describes which combination of attribute values a user is willing to accept and to which degree, where attribute values and degree of acceptance are fuzzy sets. [12] proposes 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 relevance degrees of discovered services w.r.t. user's preferences. To select the most relevant services, a fuzzy dominance relationship is proposed to rank-order services. The selected services are then used to find the top-k service compositions. A method to improve the diversity of returned compositions is also proposed.

Skyline-based Service Selection. Some preference-based service selection methods employ methods that are based on the concept of *skyline*, a.k.a. Pareto optimality, and its variants. A service is in the skyline if there is no other service that dominates it, i.e., be at least as good on all attributes of interest, and strictly better on one. The concept has been heavily studied in the data management community, where efficient methods have been proposed, e.g., [42], [43]. In the context of service descriptions, the attributes of interest are NFP values (typically QoS). Our work borrows ideas from this line of work, but differs in that: (1) the attributes of interest are the degrees of match of user preferences to NFP values (instead of the NFP values directly), and (2) multiple users with distinct preferences are considered.

For the local selection problem, the number of services that belong to a QoS-based skyline can be quite large. Several approaches attempt to control or reduce the number of returned services. [44] uses the concept of representative skyline [45] to select services based on their QoS; briefly, a skyline service is representative if it is similar to a large number of other skyline services. We note that a similar skyline-based method is adopted in [11], but for the problem of service matchmaking according to functional descriptions. [46] and [47] use the k-dominance relationship of [20] to filter services; briefly, an service is said to k-dominate another if there are k dimensions on which dominance holds. These approaches are similar to ours, in that they also employ relaxed dominance relationships. However, in our work, we apply similar ideas to a different problem, that of selecting services for multiple users. Moreover, we note that the algorithms we present here are more efficient than the adaptation of the methods in [20].

For the global selection problem, [48] computes the skyline service execution plans. They propose indexing service operations to compute the skyline more efficiently. In [49], the authors propose a preference order based on a set of fuzzy linguistic predicates. Then, they present a weighting procedure for transforming the preference relations into numerical weights, which is used to identify preferred skyline solutions. In [50], the authors develop strategies to select the skyline composite services efficiently. They show that it is sufficient to compute the local service skylines without generating all possible service compositions. The work in [51] applies the MapReduce computation model for parallel skyline service selection. Specifically, they employ an angle-based data space partitioning approach to deliver services to different nodes. The work in [52] focuses on computing the composite service skyline in the presence of QoS correlations. Different pruning techniques are investigated to accelerate the computing process.

Service Selection for Multiple Users. The problem is to identify services that are appropriate to a set of users, each expressing one's own preferences in terms of NFPs. In [17], the authors refer to the AHP (Analytical Hierarchy Process) approach to transform both user qualitative preferences and users' priorities into user weights, which are then used to rank services. In [53], the authors propose an approach for resolving conflicting service requests using Euclidean distance with weights to calculate the matching degree between a request and a web service, a global optimal web service selection model has been developed based on 0-1 integral programming. Wang et al. [54] first predicts the missing multi-QoS values according to the historical QoS experience from different users, and then selects the global optimal solution for multi-user. In [16], services are first ranked individually per user, and user weights are then used to merge the ranked lists. Similarly, [8] propose to merge the ranked lists adopting a consensus-based approach.

These approaches resort to performing a weight-based aggregation of services. We argue that this approach obscures and blurs the individual user preferences per NFPs. In our previous work [19], we propose to discard services that are majority dominated, i.e., a majority of users agree that there are better alternatives. This way, individual preferences on what constitutes non-desirable services, are not ignored by the aggregation mechanism, ensuring thus a level of fairness across all users. In this work, we go further than [19], in that we consider the problem of ranking services for multiple users, and show how our majority dominance-based approach can be integrated with existing ranking approaches.

3 Problem Description

In this section, we supply the basic notions used in this paper, and formalize the notion of majority service skyline. Table 4 summarizes the frequently used symbols and their description.

Given a set of functionally equivalent services $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$, where each is defined over a set of NFPs

TABLE 4: Notation

Symbol	Description
$\mathcal{S}$, s_i	Set of services, a specific service
$\mathcal{P}$, p_a	Set of NFPs, a specific NFP
n, m, d	Number of services, users, NFPs
$s_i.p_a$	Value of p_a for s_i
$\mathcal{U}$, u_x	Set of users, a specific user
$u_x.p_a$	Preference of u_x on p_a
μ_i^x	Matching vector of s_i to u_x
$\mu_i^x.p_a$	Matching degree of s_i to u_x w.r.t. p_a
$\mu_i^-.p_a$	Lower bound of matching degrees of s_i on p_a
$\mu_i^+.p_a$	Upper bound of matching degrees of s_i on p_a
μ_i^-	Lower bound of the matching vectors of s_i
μ_i^+	Upper bound of the matching vectors of s_i
$s_i \succeq^x s_j$	s_i weakly dominates s_j w.r.t. u_x
$s_i \succ^x s_j$	s_i dominates s_j w.r.t. u_x
$s_i \succ_{\mathtt{U}} s_j$	s_i unanimously dominates s_j
$s_i \succ_{\mathtt{M}} s_j$	s_i majority-dominates s_j
$s_i \succ_{\mathtt{W}} s_j$	s_i Widely dominates s_j
USS	Unanimous service skyline
MSS	Majority service skyline

$\mathcal{P} = \{p_1, p_2, \ldots, p_d\}$, we use $s_i.p_a$ to denote the value of parameter p_a for service s_i. Further, assume a set of users $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$, where each specifies her/his preferences on the set of parameters $\mathcal{P}$, and we use $u_x.p_a$ to denote the preference of u_x on parameter p_a.

Given a service s_i and a user u_x, the matching vector of s_i to u_x, denoted as μ_i^x, is a d-dimensional point in $[0,1]^d$, where its a-th coordinate is the matching degree with respect to parameter p_a, i.e., $\mu_i^x = (\mu_i^x.p_1, \mu_i^x.p_2, \ldots, \mu_i^x.p_d)$. The matching degree $\mu_i^x.p_a$ is defined by a matching function $\mu : 2^{dom(p_a)} \rightarrow [0,1]$ that specifies to which extent the service's NFP value (or range, or set of values) p_a satisfies the users' preference u_x. We emphasize that the mechanism of the matching function is orthogonal to our problem. For example, the matching degree can be a utility function that only depends on the specific user and service, or a collaborative filtering mechanism that considers past interactions of all users with all services.

Example 4. The matching degree of service s_1 to user u_1 with respect to response time is given by the probability that this service satisfies the user's preference, computed as $\frac{|[150,450] \cap (0,400]|}{|[150,450]|} = \frac{|[150,400]|}{|[150,450]|} = 250/300 = 0.83$. With respect to redundancy, the matching degree is 1, indicating that the level of the service's data redundancy satisfies the user. Thus, the matching vector of service s_1 to user u_1 is $\mu_1^1 = (0.83, 1)$. All matching vectors of our example are shown in Table 3.

We now introduce the notion of majority service skyline.

Definition 1 (Weak Dominance). Given a user u_x, we say that a service s_i *weakly dominates* another service s_j w.r.t. u_x, denoted as $s_i \succeq^x s_j$, iff s_i has better or equal matching degrees than s_j on all specified preference parameters. i.e., $s_i \succeq^x s_j \Leftrightarrow \forall p_a \in \mathcal{P} : \mu_i^x.p_a \geq \mu_j^x.p_a$.

Definition 2 (Dominance). Given a user u_x, we say that a service s_i *dominates* another service s_j w.r.t. u_x, denoted as $s_i \succ^x s_j$, iff s_i has better or equal matching degrees than s_j on all specified preference parameters, and strictly better matching degree on at least one. i.e., $s_i \succ^x s_j \Leftrightarrow \forall p_a \in \mathcal{P} : \mu_i^x.p_a \geq \mu_j^x.p_a \wedge \exists p_b \in \mathcal{P} : \mu_i^x.p_b > \mu_j^x.p_b$.

Definition 3 (Unanimous Dominance). Given a set of users $\mathcal{U}$, we say that a service s_i *unanimously dominates* another service s_j, denoted as $s_i \succ_{\mathtt{U}} s_j$, iff s_i weakly dominates s_j w.r.t. all users, and there exists at least one user u_y, for which s_i dominates s_j. i.e., $s_i \succ_{\mathtt{U}} s_j \Leftrightarrow \forall u_x \in \mathcal{U} : s_i \succeq^x s_j \wedge \exists u_y \in \mathcal{U} : s_i \succ^y s_j$.

Definition 4 (Unanimous Service Skyline). Given a set of services $\mathcal{S}$ and a set of users $\mathcal{U}$, the *unanimous service skyline USS* comprises the set of services that are not unanimously dominated by any other service. i.e., *USS* $= \{s_i \in \mathcal{S} \mid \nexists s_j \in \mathcal{S} : s_j \succ_{\mathtt{U}} s_i\}$.

Definition 5 (Majority Dominance). Given a set of users $\mathcal{U}$, we say that a service s_i *majority-dominates* another service s_j, denoted as $s_i \succ_{\mathtt{M}} s_j$, 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, and (2) there exists a user u_y for which s_i dominates s_j. i.e., $s_i \succ_{\mathtt{M}} s_j \Leftrightarrow (\exists \mathcal{U}' \subseteq \mathcal{U} : |\mathcal{U}'| > \lfloor |\mathcal{U}|/2 \rfloor \wedge \forall u_x \in \mathcal{U}' \; s_i \succeq^x s_j) \wedge \exists u_y \in \mathcal{U}' : s_i \succ^y s_j$.

Definition 6 (Majority Service Skyline). Given a set of services $\mathcal{S}$ and a set of users $\mathcal{U}$, the *majority service skyline MSS* comprises the set of services that are not majority-dominated by any other service. i.e., *MSS* $= \{s_i \in \mathcal{S} \mid \nexists s_j \in \mathcal{S} : s_j \succ_{\mathtt{M}} s_i\}$.

We note that is possible to enforce a *super-majority*, e.g., require two-thirds of the users to agree. The necessary change is in the first requirement of majority dominance: ensure that the user subset has the desired cardinality.

Example 5. Service s_1 majority dominates service s_3 according to the majority of u_1, u_3. Similarly, service s_2 majority dominates s_1 according to the majority of u_1, u_2. Service s_2 is not majority dominated by any other service, and thus belongs to the majority service skyline.
Recall that the unanimous service skyline comprises services s_1, s_2, and s_3 (see Example 3), and observe that the majority service skyline has smaller cardinality than the unanimous service skyline. This is formally expressed as follows.

Lemma 1. If service s_i unanimously dominates service s_j, then s_i majority-dominates s_j. i.e., $s_i \succ_{\mathtt{U}} s_j \Rightarrow s_i \succ_{\mathtt{M}} s_j$.

Proof: Follows from Definition 3 and Definition 5, setting $\mathcal{U}' = \mathcal{U}$. □

Theorem 1. The majority service skyline is a subset of the unanimous service skyline. i.e., *MSS* $\subseteq$ *USS*.

Proof: Assume that there exists a service s_i, such that $s_i \in$ *MSS* and $s_i \notin$ *USS*. Since $s_i \notin$ *USS*, there must exist a service s_j, such that $s_j \succ_{\mathtt{U}} s_i$. Thus, by Lemma 1, we have $s_j \succ_{\mathtt{M}} s_i$. Which leads to a contradiction, as $s_i \in$ *MSS*. □

We now provide the formal definition for the majority rule-based multiple users service selection problem.

Problem statement. Given a set of functionally similar services $\mathcal{S}$ defined over a set of NFPs $\mathcal{P}$, and a set of users $\mathcal{U}$ along with their preferences over each parameter in $\mathcal{P}$, compute the *majority service skyline*.

4 Computing the Majority Service Skyline

In this section, we first show how an adaptation of existing algorithms can be used to compute the majority service skyline, and we then present our proposed algorithms.

4.1 The Baseline Algorithm

Observe thereafter that unlike the unanimous dominance relationship, the majority dominance relationship does not maintain the transitive property, i.e., the majority dominance relationship is not transitive.

Theorem 2. The majority dominance relationship is not transitive. i.e., $\exists s_i, s_j, s_k \in \mathcal{S} : s_i \succ_{\mathtt{M}} s_j \wedge s_j \succ_{\mathtt{M}} s_k \wedge s_i \not\succ_{\mathtt{M}} s_k$.

Proof: Consider services s_1, s_2, s_3 from our example. Observe that $s_2 \succ_{\mathtt{M}} s_1$ and $s_1 \succ_{\mathtt{M}} s_3$, but $s_2 \not\succ_{\mathtt{M}} s_3$. □

The above theorem shows that the majority dominance relationship shares the intransitivity property of the k-dominance relationship introduced in [20]. Therefore, even if a majority-dominated service cannot be a result, it cannot be completely disregarded as it still might eliminate other services. In our running example, s_1 is the only service that can eliminate s_3. This observation justifies why the existing algorithms for computing the conventional skyline are not applicable for computing the majority service skyline. However, the one scan algorithm (OSA) and two scan algorithm (TSA) of [20], can be adapted to compute the majority service skyline, by exchanging k-dominance checks for majority dominance checks (or equivalently setting $k = \lfloor m/2 \rfloor + 1$).

This *Baseline Algorithm* first computes the matching vector μ_i^x of each service s_i in $\mathcal{S}$ with respect to each user u_x in $\mathcal{U}$. Then, the majority service skyline *MSS* is computed using the adaptation of OSA or TSA. Finally, *MSS* is returned.

Computational Complexity. The computational cost of BA is the sum of two stages. The first is computing the matching degrees, which takes $\mathcal{O}(d \cdot m \cdot n)$ time. The second is computing the majority service skyline, which involves checking all $\mathcal{O}(n^2)$ pairs of services in the worst case. Each dominance check is over m users and d non-functional parameters. So, the total cost of the second stage is $\mathcal{O}(d \cdot m \cdot n^2)$ in the worst case. Thus, BA takes in total $\mathcal{O}(d \cdot m \cdot n^2)$ time.

4.2 The Sort-Based Algorithm

Hereafter, we present the *Sort-Based Algorithm* (SBA), which improves on BA by employing a number of observations; SBA was introduced as MSA in [19]. The main idea is to sort the services according to a monotonic function that preserves the preferences of all users, so that the number of dominance checks is reduced. Specifically, SBA is based on Lemma 1 and the next two lemmas.

Lemma 2. If service s_i unanimously dominates service s_j and s_j majority-dominates service s_k, then s_i majority-dominates s_k. i.e., $s_i \succ_{\mathtt{U}} s_j \wedge s_j \succ_{\mathtt{M}} s_k \Rightarrow s_i \succ_{\mathtt{M}} s_k$.

Proof: Since s_j majority-dominates s_k, there exists a set $\mathcal{U}'$ users with $|\mathcal{U}'| > |\mathcal{U}|/2$ such that s_j weakly dominates s_k according to them, and there also exists a user $u_y \in \mathcal{U}'$ for which s_j dominates s_k. Since s_i unanimously dominates s_j, it holds that for the subset $\mathcal{U}'$ of users s_i weakly dominates s_j. Then, since weak dominance is a transitive relationship, we derive that for all users in $\mathcal{U}'$, s_i weakly dominates s_k, and also according to $u_y \in \mathcal{U}'$ s_i dominates s_k. Therefore, by definition, s_i majority dominates s_k. □

Lemma 3. Let $f : \mathcal{S} \rightarrow \mathbb{R}^+$ be a monotone function aggregating the matching degrees of each service for all users. If a service s_i unanimously dominates another service s_j, then $f(s_i) \geq f(s_j)$. i.e., $s_i \succ_{\mathtt{U}} s_j \Rightarrow f(s_i) \geq f(s_j)$.

Algorithm 1: SBA

```
   Input: set of services S; set of users U;
   Output: majority service skyline MSS;
1  begin
2      MSS ← ∅;
3      USS′ ← ∅;
4      foreach s_i ∈ S do
5          foreach u_x ∈ U do
6              compute μ_i^x;
7      sort S according to f;
8      foreach s_i ∈ S do
9          inUSS ← true;
10         foreach s_j ∈ MSS ∪ USS′ do
11             if s_j ≻_U s_i then
12                 inUSS ← false;
13                 break;
14         if inUSS then
15             foreach s_j ∈ MSS do
16                 if s_i ≻_M s_j then
17                     move s_j from MSS to USS′;
18             inMSS ← true;
19             foreach s_j ∈ MSS ∪ USS′ do
20                 if s_j ≻_M s_i then
21                     inMSS ← false;
22                     break;
23             if inMSS then
24                 insert s_i into MSS;
25             else
26                 insert s_i into USS′;
27     return MSS;
```

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

Lemma 1 and Lemma 2, suggest 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 dominance checks (comparisons). Specifically, if a service s_i is unanimously 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 helps further reduce unnecessary comparisons. To exploit this property, we sort the services in non-ascending order of the sum of their matching degrees. i.e., for a service s_i, $f(s_i) = \sum_{u_x \in \mathcal{U}} \sum_{p_a \in \mathcal{P}} \mu_i^x.p_a$. The implication is that a service s_i can only be unanimously dominated by a service that has appeared before s_i in the examined order. Thus, checks for unanimous dominance can be reduced. This is the idea behind the SFS algorithm [55], which we apply for cyclic dominance relationships.

SBA is depicted in Algorithm 1. Based on Lemma 1 and Lemma 2, SBA maintains two sets *MSS* and *USS*$'$, containing respectively the set of intermediate majority skyline services and the set of intermediate unanimous skyline services that are not in *MSS*. Thus, *MSS*$\cup$*USS*$'$ constitutes the intermediate unanimous service skyline. Initially, both sets *MSS* and *USS*$'$ are empty (lines 2–3). Then, the matching vector μ_i^x of each service s_i in $\mathcal{S}$ with respect to each user u_x in $\mathcal{U}$ is computed (lines 4-6). After that, the services are sorted in the descending order of f (line 7). Afterwards, the algorithm iterates over the sorted services (loop in line 8). At each iteration, a service s_i from $\mathcal{S}$ is compared against services in *MSS* $\cup$ *USS*$'$ (loop

TABLE 5: Lower and Upper Bounds of the Matching Vectors

Service	Lower Bound	Upper Bound
s_1	$(0.50, 0)$	$(0.83, 1)$
s_2	$(1.00, 0)$	$(1.00, 1)$
s_3	$(0.40, 1)$	$(0.80, 1)$
s_4	$(0.00, 0)$	$(0.20, 1)$
s_5	$(0.67, 0)$	$(1.00, 1)$
s_6	$(0.00, 0)$	$(0.20, 1)$
s_7	$(0.09, 0)$	$(0.27, 1)$
s_8	$(0.00, 0)$	$(0.33, 0)$
s_9	$(0.00, 0)$	$(0.00, 0)$

in line 10), i.e., the set of services that may unanimously dominate s_i (as the other services cannot dominate s_i from Lemma 3), to check if s_i is part (or not) of the unanimous service skyline. If s_i is unanimously dominated by any service in *MSS* $\cup$ *USS*$'$, then SBA breaks out of inner for-loop (i.e., loop in line 10); in other words, service s_i is discarded since it is not part of the majority service skyline (Lemma 1), and it is unnecessary for discarding other services (Lemma 2). Otherwise, i.e., service s_i is a unanimous skyline service, if s_i majority-dominates any service s_j in *MSS* (i.e., s_j is not part of the majority service skyline), then s_j is moved from *MSS* to *USS*$'$, as it is a unanimous skyline service, thus useful for eliminating other services (lines 15–17). For the same reason, if s_i is majority-dominated by any service in *MSS* $\cup$ *USS*$'$, it is inserted into *USS*$'$. Else, s_i is an intermediate majority skyline service and is inserted into *MSS* (lines 19–26). Once all services in $\mathcal{S}$ are examined, *MSS* is returned (line 27).

Example 6. For SBA, the services are sorted, in the format $\langle s_i, f(s_i)\rangle$, as follows: $\langle s_3, 4.8\rangle$, $\langle s_1, 4.0\rangle$, $\langle s_2, 4.0\rangle$, $\langle s_5, 3.5\rangle$, $\langle s_7, 1.5\rangle$, $\langle s_4, 1.3\rangle$, $\langle s_6, 1.3\rangle$, $\langle s_8, 0.5\rangle$, $\langle s_9, 0.0\rangle$. Then, service s_2 is inserted into *MSS* as it is not majority-dominated, while, services s_1 and s_3 are inserted into *USS*$'$ since they are both majority-dominated, but they are unanimous skyline services. On the other hand, all other services are discarded since they are unanimously dominated. Thus, the SBA correctly returns service s_2 as the majority service skyline.

Computational Complexity. Compared to BA, SBA takes an additional step of sorting the services. As this takes $\mathcal{O}(n \cdot \log n)$ time, the worst-case time complexity of SBA is identical to BA, $\mathcal{O}(d \cdot m \cdot n^2)$.

4.3 The Bounds-Based Algorithm

In the following, we present our new algorithm, termed *Bounds-Based Algorithm* (BBA), for efficiently computing the majority service skyline. The key idea of BBA is to establish lower and upper bounds for the matching vectors of services in order to: (1) reduce the cost of dominance checks; and (2) minimize the number of dominance checks.

Given a service $s_i \in \mathcal{S}$ and a parameter $p_a \in \mathcal{P}$, let $\mu_i^-.p_a$, and $\mu_i^+.p_a$ be respectively the lower bound and upper bound of the matching degrees of s_i on p_a, i.e., $\mu_i^-.p_a = \min_{u_x \in \mathcal{U}} \mu_i^x.p_a$, and $\mu_i^+.p_a = \max_{u_x \in \mathcal{U}} \mu_i^x.p_a$. Then, the lower bound and the upper bound of the matching vectors of s_i are $\mu_i^- = (\mu_i^-.p_1, \mu_i^-.p_2, \ldots, \mu_i^-.p_d)$, and $\mu_i^+ = (\mu_i^+.p_1, \mu_i^+.p_2, \ldots, \mu_i^+.p_d)$, respectively.

Example 7. Table 5 shows the lower bounds and the upper bounds of the matching vectors of each service.

Moreover, our algorithm leverages an important concept called wide dominance, which offers a key property that can be used to quickly discard inappropriate services. We define the wide dominance relationship as follows.

Definition 7 (Wide Dominance). A service s_i *widely dominates* another service s_j, denoted as $s_i \succ_{\mathtt{W}} s_j$, iff the lower bounds of the matching degrees of s_i are better than or equal to the upper bounds of the matching degrees of s_j on all parameters, and strictly better on at least one. i.e., $s_i \succ_{\mathtt{W}} s_j \Leftrightarrow \forall p_a \in \mathcal{P} : \mu_i^-.p_a \geq \mu_j^+.p_a \wedge \exists p_b \in \mathcal{P} : \mu_i^-.p_b > \mu_j^+.p_b$.

Example 8. Service s_3 widely dominates service s_4, as the lower bounds of the former $(0.40, 1)$ are better than the upper bounds of the latter $(0.20, 1)$ (see Table 5).

To compute the majority service skyline efficiently, BBA exploits the following properties, in addition to those previously defined.

Lemma 4. If service s_i widely dominates service s_j, then s_i unanimously dominates s_j. i.e., $s_i \succ_{\mathtt{W}} s_j \Rightarrow s_i \succ_{\mathtt{U}} s_j$.

Proof: Assume that $s_i \succ_{\mathtt{W}} s_j$ and $s_i \not\succ_{\mathtt{U}} s_j$. Given that $s_i \not\succ_{\mathtt{U}} s_j$, there must exist a user u_x and a parameter p_a such that $\mu_j^x.p_a > \mu_i^x.p_a$. This leads to a contradiction as $s_i \succ_{\mathtt{W}} s_j$ means that the lower bounds of the matching degrees of s_i are better or equal than the upper bounds of the matching degrees of s_j on all parameters, and strictly better on at least one. □

Note that the inverse direction does not hold, i.e, if s_i unanimously dominates s_j, then the former might not widely dominate the latter. This is exhibited in our example: s_1 unanimously dominates, but does not widely dominate, s_4.

Lemma 5. Let $f : \mathcal{S} \rightarrow \mathbb{R}^+$ be a monotone function aggregating the matching degrees of each service for all users such that $f(s_i) \geq \max_{p_a \in \mathcal{P}} \mu_i^+.p_a$. Given two services s_i and s_j, if $\min_{p_a \in \mathcal{P}} \mu_i^-.p_a > f(s_j)$, then s_i unanimously dominates s_j. i.e., $\min_{p_a \in \mathcal{P}} \mu_i^-.p_a > f(s_j) \Rightarrow s_i \succ_{\mathtt{U}} s_j$.

Proof: As $f(s_j) \geq \max_{p_a \in \mathcal{P}} \mu_j^+.p_a$, $\min_{p_a \in \mathcal{P}} \mu_i^-.p_a > f(s_j)$ implies that $\min_{p_a \in \mathcal{P}} \mu_i^-.p_a > \max_{p_a \in \mathcal{P}} \mu_j^+.p_a$. Thus, $s_i \succ_{\mathtt{W}} s_j$. Hence, by Lemma 4, $s_i \succ_{\mathtt{U}} s_j$. □

Similar to SBA, BBA exploits Lemmas 1 and 2 to compare each service against only the unanimous skyline services.

Moreover, as in SBA, BBA employs Lemma 3 to avoid unnecessary comparisons. To exploit this property, we sort the services in non-ascending order considering for each of them the sum of the upper bounds of their matching degrees. We denote this function as f, and thus for a service s_i, $f(s_i) = \sum_{p_a \in \mathcal{P}} \mu_i^+.p_a$. In case of ties, the sum of their matching degrees for all users is used. We denote this function as g, i.e., for a service s_i, $g(s_i) = \sum_{u_x \in \mathcal{U}} \sum_{p_a \in \mathcal{P}} \mu_i^x.p_a$. Therefore, given a service s_i, searching for services by which s_i is unanimously dominated can be limited to the part of the service before s_i. Note that function f satisfies the requirement of Lemma 5.

Lemma 4 allows us to avoid iterating over all users when checking if a service s_i unanimously dominates another service s_j by first comparing their corresponding lower bound and the upper bound of their matching vectors, i.e., μ_i^- and μ_j^+. Thereby, reducing the cost of a number of unanimous dominance checks from $\mathcal{O}(d \cdot m)$ to $\mathcal{O}(d)$.

Algorithm 2: BBA

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

1 **begin**
2 $\quad$ $MSS \leftarrow \emptyset$;
3 $\quad$ $USS' \leftarrow \emptyset$;
4 $\quad$ $\mu_{stop} \leftarrow 0$;
5 $\quad$ **foreach** $s_i \in \mathcal{S}$ **do**
6 $\quad\quad$ **foreach** $u_x \in \mathcal{U}$ **do**
7 $\quad\quad\quad$ compute μ_i^x;
8 $\quad\quad$ compute μ_i^-;
9 $\quad\quad$ compute μ_i^+;
10 $\quad$ sort $\mathcal{S}$ according to f, then g in case of ties;
11 $\quad$ **foreach** $s_i \in \mathcal{S}$ **do**
12 $\quad\quad$ **if** $\mu_{stop} > f(s_i)$ **then**
13 $\quad\quad\quad$ $break$;
14 $\quad\quad$ **else**
15 $\quad\quad\quad$ $inUSS \leftarrow true$;
16 $\quad\quad\quad$ **foreach** $s_j \in MSS \cup USS'$ **do**
17 $\quad\quad\quad\quad$ **if** $s_j \succ_{\mathtt{W}} s_i$ **then**
18 $\quad\quad\quad\quad\quad$ $inUSS \leftarrow false$;
19 $\quad\quad\quad\quad\quad$ $break$;
20 $\quad\quad\quad\quad$ **else**
21 $\quad\quad\quad\quad\quad$ **if** $s_j \succ_{\mathtt{U}} s_i$ **then**
22 $\quad\quad\quad\quad\quad\quad$ $inUSS \leftarrow false$;
23 $\quad\quad\quad\quad\quad\quad$ $break$;
24 $\quad\quad\quad$ **if** $inUSS$ **then**
25 $\quad\quad\quad\quad$ $\mu_{stop} \leftarrow max(\mu_{stop}, \min_{p_a \in \mathcal{P}} \mu_i^- .p_a)$;
26 $\quad\quad\quad\quad$ **foreach** $s_j \in MSS$ **do**
27 $\quad\quad\quad\quad\quad$ **if** $s_i \succ_{\mathtt{M}} s_j$ **then**
28 $\quad\quad\quad\quad\quad\quad$ remove s_j from *MSS* to *USS′*;
29 $\quad\quad\quad\quad$ $inMSS \leftarrow true$;
30 $\quad\quad\quad\quad$ **foreach** $s_j \in MSS \cup USS'$ **do**
31 $\quad\quad\quad\quad\quad$ **if** $s_j \succ_{\mathtt{M}} s_i$ **then**
32 $\quad\quad\quad\quad\quad\quad$ $inMSS \leftarrow false$;
33 $\quad\quad\quad\quad\quad\quad$ $break$;
34 $\quad\quad\quad\quad$ **if** $inMSS$ **then**
35 $\quad\quad\quad\quad\quad$ insert s_i into *MSS*;
36 $\quad\quad\quad\quad$ **else**
37 $\quad\quad\quad\quad\quad$ insert s_i into *USS′*;
38 $\quad$ **return** *MSS*;

Furthermore, Lemma 5 provides a termination condition. Specifically, given two services s_i and s_j, if $\min_{p_a \in \mathcal{P}} \mu_i^- .p_a > f(s_j) = \sum_{p_a \in \mathcal{P}} \mu_j^+ .p_a$ then s_i unanimously dominates s_j as well as all services after s_j (since services are sorted in non-ascending order, $f(s_i) > f(s_k)$ for any service s_k after s_j).

BBA, shown in Algorithm 2, leverages the observations made above to compute efficiently the majority service skyline. Based on Lemma 1 and Lemma 2, BBA maintains two sets *MSS* and *USS′*, containing respectively the set of intermediate majority skyline services and the set of intermediate unanimous skyline services that are not in *MSS*. Thus, $MSS \cup USS'$ constitutes the intermediate unanimous service skyline. Also, based on Lemma 5, BBA uses variable μ_{stop}, which maintains the maximin matching degrees of the examined services, i.e., $\max_{s_i \in MSS \cup USS'} \min_{p_a \in \mathcal{P}} \mu_i^- .p_a$; observe that the maximin strategy offers the earliest termination position. Initially both sets *MSS* and *USS′* are empty (lines 2–3), and μ_{stop} is set to 0 (line 4); since no service is examined up to now. Then, the matching vector μ_i^x of each service s_i in $\mathcal{S}$ with respect to each user u_x in $\mathcal{U}$ is computed, and the lower bound μ_i^- and the upper bound μ_i^+ of the matching vectors of each service s_i is deduced (lines 5–9). After that, the services are sorted in non-ascending order of f, then g in case of ties (line 10). Afterwards, the algorithm iterates over the services (loop in line 11). Each time a new service s_i from $\mathcal{S}$ is examined. If $\mu_{stop} > f(s_i)$ (line 12), i.e., the algorithm has reached the sufficient condition to conclude that no additional service in $\mathcal{S}$ can be part of the majority service skyline (Lemma 5), then BBA breaks out of for-loop and the result *MSS* is returned (line 38). Otherwise, service s_i is compared against services in $MSS \cup USS'$ (loop in line 16), i.e., the set of services that may unanimously dominate s_i (as the other services cannot dominate s_i from Lemma 3), to check if s_i is part (or not) of the unanimous service skyline. From Lemma 4, BBA first checks if service s_i is widely dominated, then if it is unanimously dominated. If s_i is widely dominated or unanimously dominated by any service in $MSS \cup USS'$ then BBA breaks out of inner for-loop (i.e., loop in line 16); in other words, service s_i is discarded since it is not part of the majority service skyline (Lemma 1), and it is unnecessary for discarding other services (Lemma 2). Otherwise, i.e., service s_i is a unanimous skyline service, μ_{stop} is updated (line 25), and if s_i majority-dominates any service s_j in *MSS* (i.e., s_j is not part of the majority service skyline), then s_j is removed from *MSS* to *USS′*, as it is a unanimous skyline service, thus useful for eliminating other services (lines 26–28). For the same reason, if s_i is majority-dominated by any service in $MSS \cup USS'$, it is inserted into *USS′*. Else, s_i is an intermediate majority skyline service and is thus inserted into *MSS* (lines 30–37). In the case that all services in $\mathcal{S}$ are examined, this means that the termination condition (lines 12–13) is not reached, the result *MSS* is returned (line 38).

Example 9. For BBA, the services are sorted, in the format $\langle s_i, f(s_i) \rangle$, as follows: $\langle s_2, 2.0 \rangle$, $\langle s_5, 2.0 \rangle$, $\langle s_1, 1.8 \rangle$, $\langle s_3, 1.8 \rangle$, $\langle s_7, 1.3 \rangle$, $\langle s_4, 1.2 \rangle$, $\langle s_6, 1.2 \rangle$, $\langle s_8, 0.3 \rangle$, $\langle s_9, 0.0 \rangle$. Then, service s_2 is inserted into *MSS* as it is not majority-dominated, while, services s_1 and s_3 are inserted into *USS′* since they are both majority-dominated, but they are unanimous skyline services. On the other hand, service s_5 is discarded since it is unanimously dominated by service s_2. Also, service s_4 is discarded as it is widely dominated by service s_3. Moreover, services s_8 and s_9 are not considered since BBA will reach the termination condition ($\mu_{stop} = 0.4 > f(s_8) > f(s_9)$). BBA correctly returns service s_2 as the majority service skyline.

Computational Complexity. The cost of BBA is the sum of three stages. The first is computing the matching degrees, and lower/upper bounds of the matching vectors at $\mathcal{O}(d \cdot m \cdot n)$ time. The second is sorting the services at $\mathcal{O}(n \cdot \log(n))$ time. The last is computing the majority service skyline, which performs $\mathcal{O}(d \cdot m \cdot n^2)$ comparisons in the worst case. Thus, BBA takes in total $\mathcal{O}(d \cdot m \cdot n^2)$ time.

5 Ranking Services for Multiple Users

The motivation for computing the majority service skyline is to reduce the number of services returned by keeping only the most interesting ones. However, as the number of users and the number of NFPs increase, it becomes more likely that the users express very different and conflicting preferences, leading to a large cardinality of the majority service skyline. Hence, the task of identifying an appropriate service may still be cumbersome. In this section, we address the problem of providing a ranking among services in the majority skyline such that it takes into account users' preferences.

It is possible to pose the service ranking problem as a group recommender task: given available services and a group of users, where for each user we know her/his individual preferences, what is an appropriate ranking of services? As we explicitly know the preferences of individual users as well as the characteristics of services, we can already compute matching degrees per user as in Section 3. Therefore, for each user individually we can compile a ranked list of recommended services, e.g., by ranking based on the average matching degrees across all service parameters. The challenge is to come up with a single ranked list satisfying all users.

The literature on group recommenders is rich (see [21], [22]). The most appropriate line of work is the combination of individual recommendations for group members. Specifically, in *rating aggregation*, an item is explicitly assigned a group rating determined by an aggregation over the predicted member ratings. The aggregation strategies are mostly inspired by social choice theory, which deals with techniques for combining individual preferences (see [56] for an overview).

The additive strategy (Add) adds the individual matching degrees and produces an overall desirability among users. The multiplicative strategy (Mult) takes the product of individual matching degrees. The minimum strategy (Min) considers the minimum of individual matching degrees and reflects thereby the fact of not strongly displeasing any user (least misery principle). Under these two last aggregation rules, unsatisfied users have a higher impact on the final decision than satisfied ones. The maximum strategy (Max) considers the maximum of individual matching degrees for offering the maximum pleasure among users.

Translated to our problem, the baseline approach following standard practice from group recommenders entails the following steps. For each service and user, compute their total (i.e., average) matching degree. Then, for each service compute the *group matching degree*, by applying an aggregation strategy, Add, Mult, Min or Max. Then recommend to the group a ranked list of services, ordered decreasingly by their group matching degree.

Observe that the aforementioned procedure can be applied to all services, or to one of their subsets. For instance, one can select the services in the unanimous service skyline, or those in the majority service skyline. Our proposal is to compute group matching degrees only for services that are in the majority skyline, independently of the aggregation strategy adopted. As a final remark, let us note that several impossibility results have been proven in social choice theory with respect to optimal rankings, e.g., Arrow's impossibility theorem [57]. Essentially, they suggest that no ranking mechanism can be appropriate in all cases. Hence, we do not advocate one strategy over another. In the experimental evaluation we have investigated several ranking mechanisms, and the consistent finding is that majority-based filtering leads to more informative rankings.

6 Experimental Evaluation

In this section, we first describe the experiment setup and then we present and discuss the respective results.

6.1 Evaluation Setup

We first discuss the objectives of the evaluation, and then present the datasets used and the methodology followed.

6.1.1 Research Questions

Our evaluation answers the following questions.

Q1: Is MSS more effective than USS in identifying relevant services?

Q2: Is MSS more effective than USS in ranking services?

Q3: How do MSS algorithms scale?

6.1.2 Datasets

To answer our research questions, we use three datasets. The first two are based on real datasets, while the third is synthetic. Note that we were unable to find a real dataset that contains all necessary ingredients for our evaluation, i.e., service non-functional parameter values, user preferences, and ground truth services for the users.

CLOUD. We manually compile a list of 70 services offering cloud storage, and which are described by two parameters, (monthly) Cost and Storage Size. As the ground truth, we use three lists, denoted as CLOUD List A, B, C, each containing the top-10 cloud storage services as evaluated by different websites. The lists and all services are included in the supplementary material.

Based on each list, we generate a set of user preferences. Specifically, for each user, we assign a preferred value for the two service non-functional parameters, Cost and Storage Size, by selecting uniformly at random among the values of services in the top-10 list. The matching degrees between users' preferences and services' parameters are computed using the Jaccard coefficient [58].

QWS. We use the publicly available dataset QWS[2], with measurements of 9 QoS parameters for 2507 real-world web services. To allow for a uniform measurement of service qualities independent of units, we normalize the QoS values to $[0, 1]$, where 0 indicates the worst value and 1 the best.

Users' preferences on each QoS parameter are generated uniformly at random taking a value between the 50$^{\text{th}}$ overall best normalized QoS value and the maximum possible normalized QoS value (1), meaning that users require the best QoS values. The matching degrees between users' preferences and services' parameters are computed as follows. If the value of a given QoS parameter is greater than or equal to that of the user's preference then the matching degree is 1, meaning that the QoS parameter is completely satisfied; otherwise, the matching degree is penalized by how much the preference deviates from the service's parameter value. Concretely, for service s_i, user u_x, and QoS parameter p_a, the degree of match is computed as $\mu_i^x.p_a = 1 - \max\{u_x.p_a - s_i.p_a, 0\}$.

The ground truth contains the top-250 services ($\sim$10% of dataset) according to the average normalized QoS value.

SYNTH. We synthetically generate datasets to greatly vary the problem parameters, so as to study their effect on the efficiency of the majority service skyline algorithms, as well on the effectiveness of the various approaches. In particular, the services and users' preferences are generated following two distributions: (1) *similar*, where users' preferences are almost similar, i.e., a good match of a given service to some user increases the possibility of its good match to the other users; (2) *conflicting*, where users' preferences are diversified,

2. www.uoguelph.ca/~qmahmoud/qws/

i.e., for a given service, good matches (or bad matches) to all users are less likely to occur.

6.1.3 Methodology

To answer **Q1**, we consider two ways. First, we examine the *size* of MSS and USS. Fewer services means that the users will decide among fewer alternatives, which is desired. We would like to see how much smaller is the MSS compared to USS. Second, when we have a ground truth, we quantify the quality of the returned services with respect to it. Specifically, we measure *precision*, *recall*, and *F1 score*, which is the harmonic average of the first two. These three metrics take values in the range $[0, 1]$, with higher values being better.

To answer **Q2**, we use the ground truth to quantify the quality of a ranked list containing all services (All), the majority service skyline (MSS), and the unanimous service skyline (USS).

The way services are ranked is orthogonal to our approach. Therefore, we consider various aggregation strategies, namely, the four mentioned in Section 5, namely, additive (Add), maximum (Max), multiplicative (Mult), and minimum (Min), as well as the consensus-based approach (Cons) described in [8]. Each aggregation strategy is paired with a set of services to rank, either All, MSS, or USS, for a total of 15 distinct rankings.

The quality of a ranked list with respect to the ground truth is measured in terms of its *normalized discounted cumulative gain* at rank k (NDCG@k). Let σ denote the ranking returned by an approach, with $\sigma[i]$ representing the service at the i-th rank. Moreover, let $score(\sigma[i])$ denote the Borda score of service $\sigma[i]$ according to the ground truth. Then, the discounted cumulative gain (DCG) at rank k is defined as: DCG@k $= \sum_{i=1}^{k} \frac{2^{score(\sigma[i])}-1}{\log(i+1)}$.

The ideal discounted cumulative gain (IDCG) is defined as the DCG achieved when the relevant items are ranked as in the ground truth. The normalized discounted cumulative gain at position k is then computed as the ratio of DCG over IDCG: NDCG@k $= \frac{\text{DCG@k}}{\text{IDCG@k}}$. NDCG take values in the range $[0, 1]$, with higher values being better.

To answer **Q3**, we investigate, the performance of four methods: BA-OSA and BA-TSA which are the two baseline variants (Section 4.1), SBA (Section 4.2), and BBA (Section 4.3). We measure performance by the amount of time required to produce the MSS on SYNTH, and investigate different problem settings by varying: (1) the number n of services available to choose from, (2) the number m of users to satisfy, and (3) the number d of parameters describing a service. The involved parameters and their examined values are summarized in Table 6. In all experimental setups, we investigate the effects of one parameter, while we set the remaining ones to their default values. All experiments were conducted on a 2.5 GHz Intel Core i7 processor with 16 GB 1600 MHz DDR3 Memory, running macOS Sierra. Reported metric values are averages over one thousand instances.

TABLE 6: Parameters and Examined Values of SYNTH

Parameter	Values	Default
Number of services (n)	10K, 50K, 100K, 500K, 1M	100K
Number of users (m)	4, 7, 10, 13, 16	10
Number of parameters (d)	2, 3, 4, 5, 6	4

6.2 Q1: MSS vs. USS in Identifying Relevant Services

Size of the Results Set. In the first set of experiments, we compare the size of the majority service skyline with that of the unanimous service skyline. Having a more manageable result size is beneficial, as it reduces the effort required to manually examine services in order to select the most appropriate. Here, we are using all datasets.

Figure 1 depicts the result set size for the three instances of CLOUD, and the QWS datasets, as we vary the number of users. The results across all datasets are similar. Observe that the size of MSS is significantly smaller than USS. Moreover, as the number of users increases, the size of USS increases, while that of MSS decreases slightly. For 16 users, MSS eliminates up to 85% services from the USS. Fig. 2 repeats this experiment on the SYNTH datasets. The result set size is the smallest in Similar and largest in Conflicting. Again, the size of USS increases with the number of users, while that of MSS remains relatively constant.

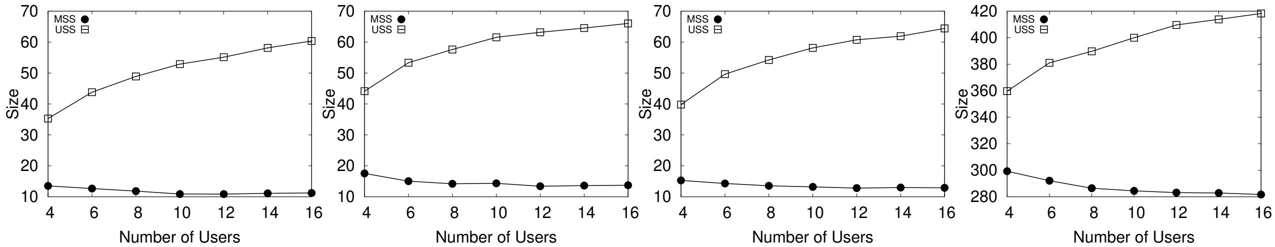
The reason for the difference in the sizes of MSS and USS is the following. A service is unanimously dominated if all m users agree. Clearly, increasing m means that it becomes harder for all users to agree, hence we see an increase in the size of USS. On the other hand, a service is majority dominated if *any* $\lfloor m/2 \rfloor + 1$ users say so. The key observation is that there exist $O(m^2)$ possible subsets of $\lfloor m/2 \rfloor + 1$ users, and it suffices that *only one* of them agrees that a service should be dominated. Therefore, when m increases, two opposing phenomena occur: it becomes harder for a service to be dominated by a particular subset of $\lfloor m/2 \rfloor + 1$ users, and at the same time there exist many more such subsets and thus more opportunities for a service to be dominated.

Next in Fig. 3, we investigate the effect of the number of services on the result set size, using the SYNTH dataset. In all cases, the result set size increases, but at a smaller rate for MSS. Increasing n means more services have a chance of not being dominated. A similar trend appears when we increase the number of non-functional parameters in Fig. 4. Increasing d means it becomes harder for a service to be dominated.

Precision and Recall. We now investigate how good the returned services are with respect to the ground truth; hence, only CLOUD and QWS datasets are used. Fig. 5 shows the precision of MSS and USS, as we vary the number of users. A general observation is that precision of USS reduces with the number of users. This is because the size of the returned results increases, and thus more unfavorable services have the chance to be included in the result. However, precision of MSS is rather stable, as it always returns a similar result set size with equally good services.

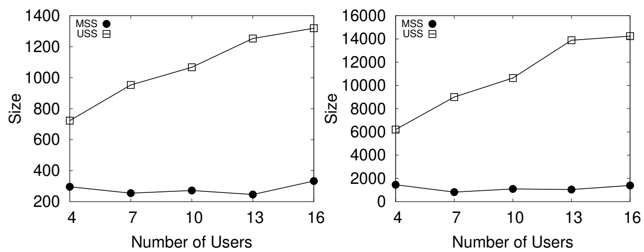
On the other hand, Fig. 6 shows that USS has better recall than MSS. Clearly, returning more results can increase recall, at the expense of precision. Indeed, any approach that randomly selects a large number of services is able to achieve a good recall. Thus, recall alone is not a good indication of effectiveness in this case.

A more appropriate measure is the F1 score, which balances recall and precision. Fig. 7 depicts the F1 score, where MSS clearly outperforms USS, with the gap widening as m increases.



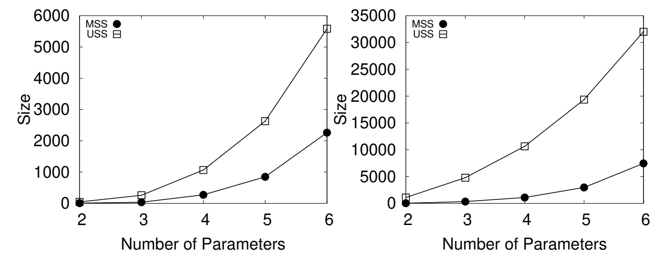
(a) CLOUD List A (b) CLOUD List B (c) CLOUD List C (d) QWS

Fig. 1: Size of the Result Set vs. Number of Users (m) on CLOUD and QWS



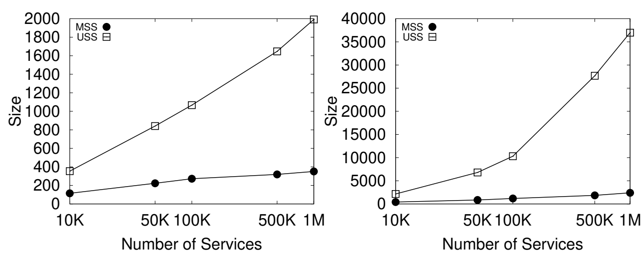
(a) Similar (b) Conflicting

Fig. 2: Size of the Result Set vs. Number of Users (m) on SYNTH



(a) Similar (b) Conflicting

Fig. 4: Size of Result Set vs. Number of Parameters (d) on SYNTH



(a) Similar (b) Conflicting

Fig. 3: Size of Result Set vs. Number of Services (n) on SYNTH

6.3 Q2: MSS vs. USS in Ranking Services

This research question investigates whether it is beneficial to filter by USS and MSS before ranking services. Therefore, we study the rankings produced by five methods, without any filtering and with dominance-based (USS or MSS) filtering. The evaluation metric is the quality of the ranked list with respect to the ground truth, measured by NDCG.

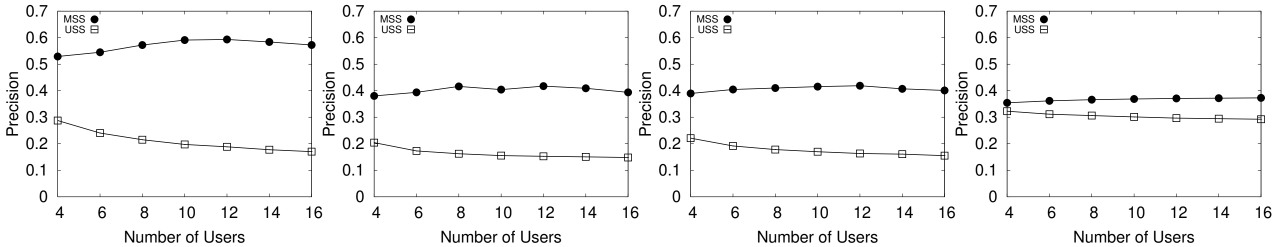
Figure 8 shows NDCG varying the number of users. Regarding the ranking methods, observe that the effectiveness improves with m for Add, Mul, and Cons, but decreases for Max and Min. Add is the best ranking method followed by Cons. Such results are consistent with empirical studies of group recommender systems [22]. The most important observation though is that, independent of the ranking method, MSS-based filtering results in better ranked lists than USS-based filtering, which only gives a smaller improvement over no filtering. A similar observation holds when we measure at NDCG at different ranks, as shown in Fig. 9.

Let us investigate this phenomenon. First, let us explain why the ranked lists across All, USS, and MSS can differ even in the first ranks for the various aggregation strategies (i.e., excluding Cons). At the first few ranks in All, there are services that have very good matching degrees for all parameters and all users. Note that these top services in All are not likely to be unanimously dominated, as they have high matching degrees to at least one parameter and user. Hence, we observe that the first few ranks in All and USS are occupied by the exact same services. On the other hand, it is possible that these top services in All can be majority dominated by other services, even by services ranked lower. The reason is that there might exist a service which is highly preferable for the majority of users, but not so for the rest, hence ranked low. Still it is possible that this service majority dominates a top service in All, meaning that the latter will not appear in the USS ranking.

In the ranking produced by Cons, we observe that even the first few ranks in All and USS are not identical. This is because Cons computes a utility for a service that depends on the set of competing services. Hence when dominance-based filtering is applied, Cons derives different scores for the same service. Overall for Cons, we still observe that MSS is more effective than USS.

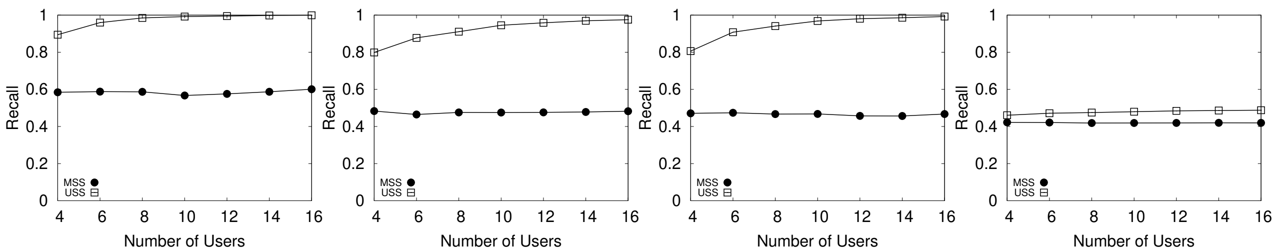
6.4 Q3: Scalability of MSS Algorithms

For this research question, we measure the total execution time for all MSS algorithms using the SYNTH dataset. The results varying n, m, and d are shown respectively in Fig. 10, Fig. 11, and Fig. 12. In Fig. 10, the execution time of the algorithms increases with the increase of n since they perform more dominance checks. As shown in Fig. 11, when m increases, the execution time of the algorithms increases since the cost of dominance checks increases. Observe in Fig. 12 that execution time of the algorithms increases with the increase of d since, on the one hand, the cost of dominance checks increases, and on the other hand, the size of the



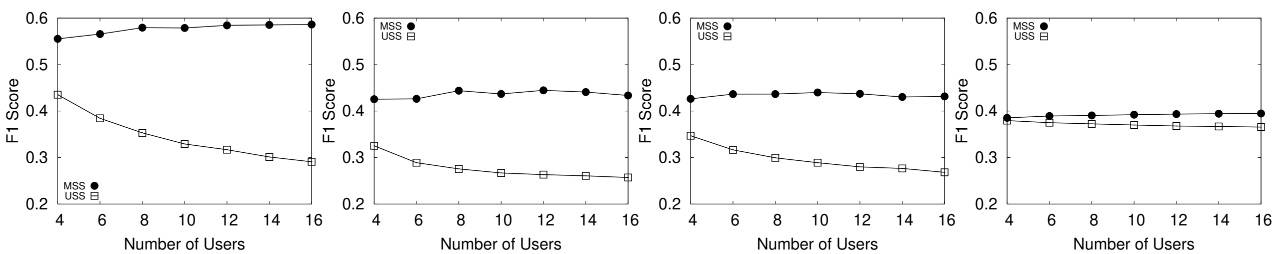
(a) CLOUD List A (b) CLOUD List B (c) CLOUD List C (d) QWS

Fig. 5: Precision vs. Number of Users (m)



(a) CLOUD List A (b) CLOUD List B (c) CLOUD List C (d) QWS

Fig. 6: Recall vs. Number of Users (m)



(a) CLOUD List A (b) CLOUD List B (c) CLOUD List C (d) QWS

Fig. 7: F1 Score vs. Number of Users (m)

majority service skyline becomes larger, thus, less services can be quickly eliminated.

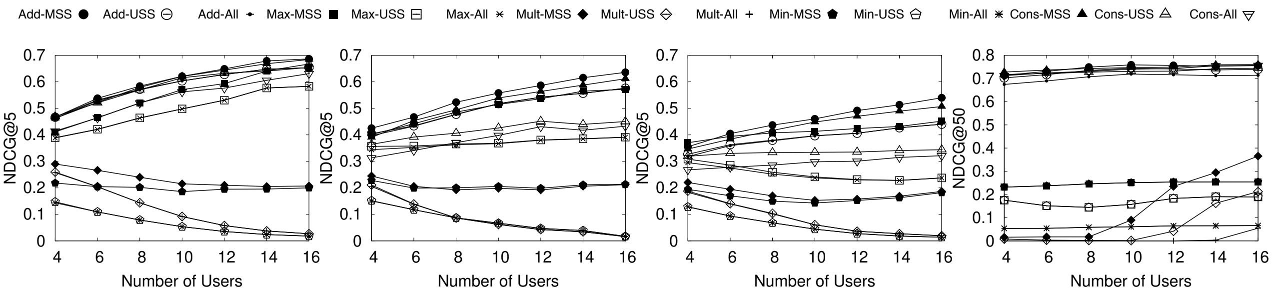
Overall, the results indicate that BBA is consistently faster than SBA (up to 34% better), and up to 2 orders of magnitude more efficient than both BA-OSA and BA-TSA. This indicates that the optimizations of SBA over BA variants, namely Lemmas 2, 3, are highly effective. Improving over SBA is a more difficult task, with Lemmas 4, 5, offering a more modest, but still significant improvement over SBA.

7 Conclusion

In this paper, we studied the problem of preference-based service selection under multiple users' preferences. We first introduced a novel concept for this problem based on the concept of majority. The majority service skyline allows users to make a "democratic" decision on which services are inappropriate, so as to exclude them from further consideration. For this problem, we adapt prior work, and also propose two algorithms that are based on novel problem properties. We then turned our attention to the problem of extracting a ranking of non-eliminated services. We observed the similarity of the task to the group recommendation problem, where it is known that no single ranking can be optimal. Therefore, our proposal is to apply any existing method after the majority dominated services are excluded.

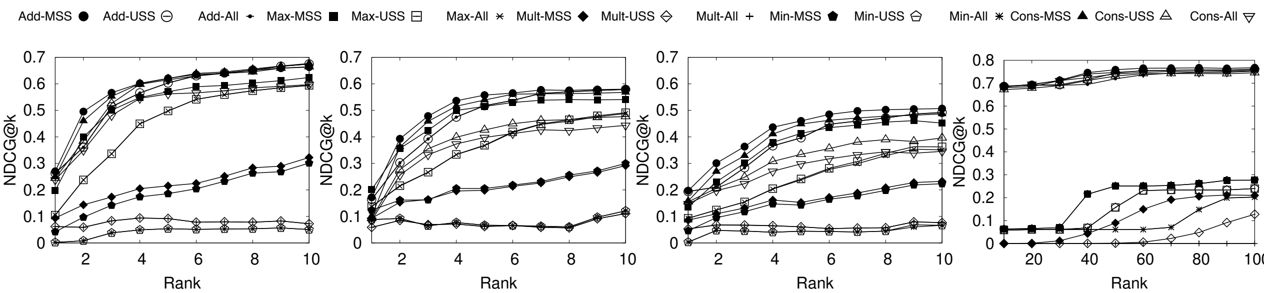
An extensive evaluation using real-based semi-synthetic datasets showed that the majority based dominance can eliminate a large number of services which are considered non relevant. Moreover, we found that applying existing preference-based service ranking methods after the filtering leads to more accurate rankings, and we explained why this interesting phenomenon occurs. We also investigated the scalability of the proposed methods as we increased the number of available services, users in the decision group, or non-functional parameters. We saw that our algorithm is up to 2 orders of magnitude more efficient than baselines. The negative aspect of our approach, is that the running time for extracting the majority skyline may become prohibitively large for cases involving several hundreds of services, tens of users, and with more than 5 NFPs.

As future work, it would be interesting to develop a dialogue-based mechanism for service selection, that can suggest to users changes to their preferences so that better group decisions can be reached. A different direction is to



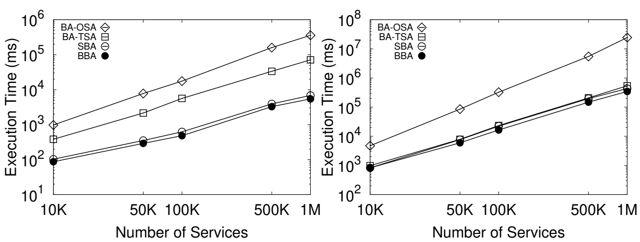
(a) CLOUD List A (k=5) (b) CLOUD List B (k=5) (c) CLOUD List C (k=5) (d) QWS (k=50)

Fig. 8: Normalized Discounted Cumulative Gain (NDCG) vs. Number of Users (m)



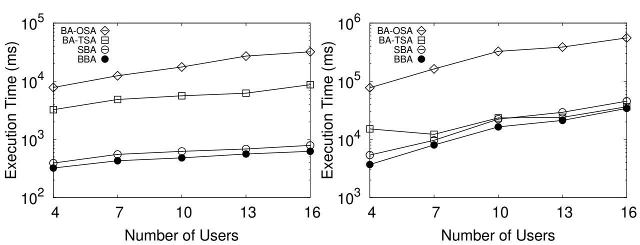
(a) CLOUD List A (b) CLOUD List B (c) CLOUD List C (d) QWS

Fig. 9: Normalized Discounted Cumulative Gain (NDCG) at different ranks (k) ($m = 10$ users)



(a) Similar (b) Conflicting

Fig. 10: Execution Time vs. Number of Services (n)



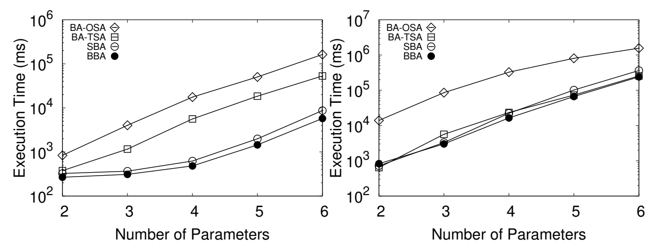
(a) Similar (b) Conflicting

Fig. 11: Execution Time vs. Number of Users (m)

consider the global service selection problem under multiple parties. We believe that our dominance-based framework could bring useful insights to this problem.

References

[1] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *Computer*, vol. 40, no. 11, 2007.



(a) Similar (b) Conflicting

Fig. 12: Execution Time vs. Number of Parameters (d)

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] Y. Wei and M. B. Blake, "Service-oriented computing and cloud computing: Challenges and opportunities," *IEEE Internet Computing*, vol. 14, no. 6, pp. 72–75, 2010.

[4] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing—the business perspective," *Decision support systems*, vol. 51, no. 1, pp. 176–189, 2011.

[5] S. Ran, "A model for web services discovery with qos," *ACM Sigecom exchanges*, vol. 4, no. 1, pp. 1–10, 2003.

[6] L. Sun, H. Dong, F. K. Hussain, O. K. Hussain, and E. Chang, "Cloud service selection: State-of-the-art and future research directions," *Journal of Network and Computer Applications*, vol. 45, pp. 134–150, 2014.

[7] R. R. Kayastha and J. Baria, "A survey on web service selection and ranking methods," *Intl. Journal of Computer Applications*, vol. 117, no. 16, pp. 5–8, May 2015, full text available.

[8] A. Arman, S. Foresti, G. Livraga, and P. Samarati, "A consensus-based approach for selecting cloud plans," in *IEEE Intl. Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow*, 2016, pp. 1–6.

[9] X. Dong, A. Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Simlarity search for web services," in *Intl. Conf. on Very Large Data Bases*, 2004, pp. 372–383.

[10] M. Comuzzi and B. Pernici, "A framework for qos-based web service contracting," *ACM Trans. on the Web*, vol. 3, no. 3, 2009.

[11] D. Skoutas, D. Sacharidis, A. Simitsis, and T. K. Sellis, "Ranking and clustering web services using multicriteria dominance relationships," *IEEE Trans. on Services Computing*, vol. 3, no. 3, pp. 163–177, 2010.

[12] K. Benouaret, D. Benslimane, A. Hadjali, M. Barhamgi, Z. Maamar, and Q. Z. Sheng, "Web service compositions with fuzzy preferences: A graded dominance relationship-based approach," *ACM Trans. on Internet Technology*, vol. 13, no. 4, pp. 12:1–12:33, 2014.

[13] S. Lamparter, A. Ankolekar, R. Studer, and S. Grimm, "Preference-based selection of highly configurable web services," in *Intl. World Wide Web Conf.*, 2007, pp. 1013–1022.

[14] S. S. Yau and Y. Yin, "Qos-based service ranking and selection for service-based systems," in *IEEE Intl. Conf. on Services Computing*, 2011, pp. 56–63.

[15] H. Wang, C. Yu, L. Wang, and Q. Yu, "Effective bigdata-space service selection over trust and heterogeneous qos preferences," *IEEE Trans. on Services Computing*, vol. 11, no. 4, pp. 644–657, 2018.

[16] H. Wang, J. Zhang, C. Wan, S. Shao, and R. Cohen, "Qualitative preference-based service selection for multiple agents," *Web Intelligence And Agent Systems: An Intl. Journal*, vol. 11, no. 3, pp. 263–282, 2013.

[17] W. Dou, C. Lv, X. Zhang, and J. Chen, "A qos-aware service evaluation method for co-selecting a shared service," in *IEEE Intl. Conf. on Web Services*, 2011, pp. 145–152.

[18] M. Moradi and S. Emadi, "A review of service skyline algorithms," *Journal of Soft Computing and Decision Support Systems*, vol. 3, no. 3, pp. 55–58, 2016.

[19] K. Benouaret, D. Sacharidis, D. Benslimane, and A. HadjAli, "Majority-rule-based web service selection," in *Intl. Conf. on Web Information Systems Engineering*, 2012, pp. 689–695.

[20] C. Y. Chan, H. V. Jagadish, K. Tan, A. K. H. Tung, and Z. Zhang, "Finding k-dominant skylines in high dimensional space," in *ACM SIGMOD Intl. Conf. on Management of Data*, 2006, pp. 503–514.

[21] A. Jameson and B. Smyth, "Recommendation to groups," in *The Adaptive Web, Methods and Strategies of Web Personalization*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Springer, 2007, pp. 596–627.

[22] J. Masthoff, "Group recommender systems: Aggregation, satisfaction and group attributes," in *Recommender Systems Handbook*, 2015, pp. 743–776.

[23] M. P. Papazoglou, *Web Services - Principles and Technology*. Prentice Hall, 2008.

[24] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic matching of web services capabilities," in *Intl. Semantic Web Conf.* Springer, 2002, pp. 333–347.

[25] D. Ardagna and B. Pernici, "Global and local qos constraints guarantee in web service selection," in *IEEE Intl. Conf. on Web Services*, 2005.

[26] Y. Liu, A. H. H. Ngu, and L. Zeng, "Qos computation and policing in dynamic web service selection," in *Intl. World Wide Web Conf.*, 2004, pp. 66–73.

[27] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma, "A qos-aware selection model for semantic web services," in *Intl. Conf. on Service Oriented Computing*, 2006, pp. 390–401.

[28] Z. ur Rehman, O. K. Hussain, and F. K. Hussain, "Parallel cloud service selection and ranking based on qos history," *Intl. Journal of Parallel Programming*, vol. 42, no. 5, pp. 820–852, 2014.

[29] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Trans. on Software Engineering*, vol. 30, no. 5, pp. 311–327, 2004.

[30] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Trans. on Software Engineering*, vol. 33, no. 6, pp. 369–384, 2007.

[31] T. Yu, Y. Zhang, and K. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM Trans. on the Web*, vol. 1, no. 1, 2007.

[32] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *Intl. World Wide Web Conf.*, 2009, pp. 881–890.

[33] H. Sun, Z. Zheng, J. Chen, and M. R. Lyu, "Personalized web service recommendation via normal recovery collaborative filtering," *IEEE Trans. on Services Computing*, vol. 6, no. 4, pp. 573–579, 2013.

[34] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, "Personalized qos-aware web service recommendation and visualization," *IEEE Trans. on Services Computing*, vol. 6, no. 1, pp. 35–47, 2013.

[35] Z. Zheng, X. Wu, Y. Zhang, M. R. Lyu, and J. Wang, "Qos ranking prediction for cloud services," *IEEE Trans. on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1213–1222, 2013.

[36] W. Lin, C. Lo, K. Chao, and N. Godwin, "Multi-group qos consensus for web services," *Journal of Computer and System Sciences*, vol. 77, no. 2, pp. 223–243, 2011.

[37] M. Sharifi, A. A. Manaf, A. Memariani, H. Movahednejad, and A. V. Dastjerdi, "Consensus-based service selection using crowdsourcing under fuzzy preferences of users," in *IEEE Intl. Conf. on Services Computing*, 2014, pp. 17–26.

[38] S. Wibowo, H. Deng, and W. Xu, "Evaluation of cloud services: A fuzzy multi-criteria group decision making method," *Algorithms*, vol. 9, no. 4, p. 84, 2016.

[39] H. Wang, J. Xu, and P. Li, "Incomplete preference-driven web service selection," in *IEEE Intl. Conf. on Services Computing*, 2008, pp. 75–82.

[40] H. Wang, S. Shao, X. Zhou, C. Wan, and A. Bouguettaya, "Web service selection with incomplete or inconsistent user preferences," in *Intl. Conf. on Service Oriented Computing: Workshop on ServiceWave*, 2009, pp. 83–98.

[41] S. Agarwal and S. Lamparter, "User preference based automated selection of web service compositions," in *Intl. Conf. on Service Oriented Computing: Workshop on Dynamic Web Processes*, 2005, pp. 1–12.

[42] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *IEEE Intl. Conf. on Data Engineering*, 2001, pp. 421–430.

[43] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," *ACM Trans. on Database Systems*, vol. 30, no. 1, pp. 41–82, 2005.

[44] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for qos-based web service composition," in *Intl. World Wide Web Conf.*, 2010, pp. 11–20.

[45] Y. Tao, L. Ding, X. Lin, and J. Pei, "Distance-based representative skyline," in *IEEE Intl. Conf. on Data Engineering*, 2009, pp. 892–903.

[46] K. Benouaret, D. Benslimane, and A. Hadjali, "On the use of fuzzy dominance for computing service skyline based on qos," in *IEEE Intl. Conf. on Web Services*, 2011, pp. 540–547.

[47] ——, "Ws-sky: An efficient and flexible framework for qos-aware web service selection," in *IEEE Intl. Conf. on Services Computing*, 2012, pp. 146–153.

[48] Q. Yu and A. Bouguettaya, "Multi-attribute optimization in service selection," *The World Wide Web Journal*, vol. 15, no. 1, pp. 1–31, 2012.

[49] X. Zhao, L. Shen, X. Peng, and W. Zhao, "Finding preferred skyline solutions for sla-constrained service composition," in *IEEE Intl. Conf. on Web Services*, 2013, pp. 195–202.

[50] Q. Yu and A. Bouguettaya, "Efficient service skyline computation for composite service selection," *IEEE Trans. on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 776–789, 2013.

[51] J. Wu, L. Chen, Q. Yu, L. Kuang, Y. Wang, and Z. Wu, "Selecting skyline services for qos-aware composition by upgrading mapreduce paradigm," *Cluster Computing*, vol. 16, no. 4, pp. 693–706, 2013.

[52] Y. Du, H. Hu, W. Song, J. Ding, and J. Lu, "Efficient computing composite service skyline with qos correlations," in *IEEE Intl. Conf. on Services Computing*, 2015, pp. 41–48.

[53] G. Kang, J. Liu, M. Tang, X. F. Liu, and K. K. Fletcher, "Web service selection for resolving conflicting service requests," in *IEEE Intl. Conf. on Web Services*, 2011, pp. 387–394.

[54] S. Wang, C. Hsu, Z. Liang, Q. Sun, and F. Yang, "Multi-user web service selection based on multi-qos prediction," *Information Systems Frontiers*, vol. 16, no. 1, pp. 143–152, 2014.

[55] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting: Theory and optimizations," in *Proceedings of the Intl. IIS: IIPWM Conf. on Intelligent Information Processing and Web Mining*, 2005, pp. 595–604.

[56] J. Masthoff, "Group modeling: Selecting a sequence of television items to suit a group of viewers," *User Modeling and User-Adapted Interaction*, vol. 14, no. 1, pp. 37–85, 2004.

[57] K. J. Arrow, *Social choice and individual values*. Yale university press, 2012, vol. 12.

[58] R. O. Duda and P. E. Hart., *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

Karim Benouaret is an Associate Professor at Claude Bernard Lyon 1 University and a member of the Lyon Research Center for Images and Intelligent Information Systems, associated to the French National Center for Scientific Research. Prior to that, he was a teaching and research assistant at Télécom Saint-Étienne, and post-doctoral researcher at Inria Nancy - Grand Est. His research interests include preference queries, recommender systems, and services computing.

Dimitris Sacharidis is an Assistant Professor at the Institute of Information Systems Engineering of Technische Universität Wien. Prior to that, he was a junior researcher at "Athena" Research Center, and a Marie Curie postdoctoral fellow at Hong Kong University of Science and Technology. His research interests include recommender systems, spatio-temporal and social data management and analytics.

Djamal Benslimane is a Full Professor at Claude Bernard Lyon 1 University and a member of the Lyon Research Center for Images and Intelligent Information Systems, associated to the French National Center for Scientific Research. Prior to that, he was an Associate Professor at University of Burgundy. His research interests include distributed information systems, Web services, ontologies, and databases.

Allel Hadhali is a Full Professor at the National Engineering School for Mechanics and Aerotechnics and a member of the Laboratory of Computer Science and Automatic Control for Systems. Prior to that, he was an Associate Professor at University of Rennes 1 and University of Tizi Ouzou. His research interests include computational intelligence in databases, data uncertainty, Web services, and approximate reasoning.