On the Use of Fuzzy Dominance for Computing Service Skyline Based on QoS

Karim Benouaret, Djamal Benslimane
LIRIS, Lyon 1 University
Villeurbanne, France
{*karim.benouaret, djamal.benslimane*}*@liris.cnrs.fr*

Allel Hadjali
IRISA/ENSSAT, Rennes 1 University
Lannion, France
allel.hadjali@enssat.fr

***Abstract*—Nowadays, the exploding number of functionally similar Web services has led to a new challenge of selecting the most relevant services using quality of service (QoS) aspects. Traditionally, the relevance of a service is determined by computing an overall score that aggregates individual QoS values. Users are required to assign weights to QoS attributes. This is a rather demanding task and an imprecise specification of the weights could result in missing some user desired services. Recent approaches focus on computing service skyline over a set of QoS aspects. This can completely free users from assigning weights to QoS attributes. However, two main drawbacks characterize such approaches. First, the service skyline often privileges services with a bad compromise between different QoS attributes. Second, as the size of the service skyline may be quite large, users will be overwhelmed during the service selection process. In this paper, we introduce a new concept, called alpha-dominant service skyline, to address the above issues and we develop a suitable algorithm for computing it efficiently. Experimental evaluation conducted on synthetically generated datasets, demonstrates both the effectiveness of the introduced concept and the efficiency of the proposed algorithm.**

***Keywords*-service selection; quality of service; fuzzy dominance; skyline analysis; optimization;**

I. Introduction

Web services are software entities that can be published, located and invoked across the Web using a set of standards such as SOAP, WSDL and UDDI [1][2]. Web services are designed to perform a specific task. Typical examples include stateful services, altering the world state, such as on-line shopping or booking and stateless services, returning information to the user, such as weather or news. Formally, a Web service is described by the names and types of input and output parameters, constraints, preconditions and postconditions (effects), as well as Quality of Service (QoS) attributes, such as execution time, price, reputation and security.

As Web services and service providers proliferate, there may be multiple service providers competing for fulfilling a desired task. Therefore, there is a need to be able to select among the functionally similar Web services using a set of well-defined QoS aspects.

Example: Consider the common example in the literature concerning a set of Web services that provide hotel search and on-line reservation. For each Web service, we set its execution time and its price (see Figure 1).

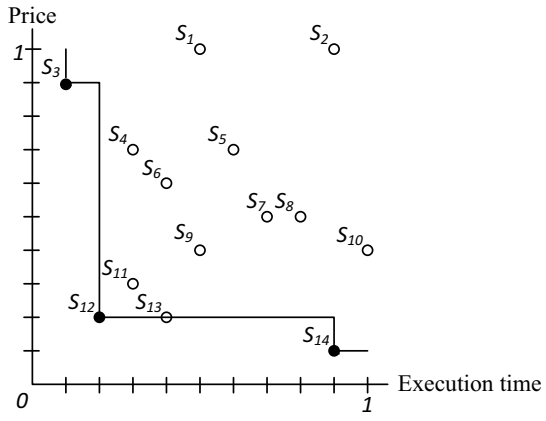

Figure 1. Example of functionally similar Web services along with their execution time and price (where values of both attributes are normalized)

To find satisfactory hotels, users need to go through several trial-run processes. This would be very painstaking and the selected services are not necessarily among the most interesting ones. Therefore, optimization strategies are required for finding the best services with respect to a set of QoS aspects desired by the users.

Currently, most approaches that deal with Web service selection based on QoS, compute a global QoS value for each service as an aggregate of the individual QoS values. Various approaches for combining QoS exist. One direction is to assign weights to individual QoS attributes. However, the users may not know enough how to make tradeoffs between different quality aspects using numbers. They thus lose the flexibility to select their desired Web services.

Computing the service skyline [3][4][5] comes as a popular solution that overcomes this limitation. The service skyline consists of a set of services ($\{S_3, S_{12}, S_{14}\}$ in our example) which are not dominated by any other one w.r.t. all attributes. A service S_i dominates another service S_j if and only if S_i is at least as good as S_j in all QoS attributes and (strictly) better than S_j in at least one QoS attribute. For instance, the Web service S_3 in Figure 1 is better than the Web services S_1 and S_2 since it is faster and cheaper.

978-0-7695-4463-2/11 $26.00 © 2011 IEEE
DOI 10.1109/ICWS.2011.93

However, there are some inherent issues of applying the service skyline approach. The first issue is related to the nature of retrieved services in the service skyline that privileges services with some very good and very bad QoS values like the services S_3 and S_{14} in Figure 1. Such services are referred to as services with a *bad compromise* (between QoS attributes). Whereas, users usually prefer services that are (moderately) good in all QoS attributes like the services S_{12} and S_{11} in Figure 1, where the last one is unfortunately not returned by the service skyline approach. The services of this type are referred to as services with a *good compromise* (between QoS attributes). Clearly, the Web service S_{12} is better than S_{11}. Furthermore, S_{12} may be currently unavailable or temporarily deprived of a functionality (e.g., on-line reservation). Users thus have to choose between the services S_3 and S_{14} while several services like S_{11} and S_{13} may be more appropriate. The second issue concerns the fact that the service skyline approach does not allow users to control the size of the returned set of services. With the presence of a possibly large number of skyline services, the full service skyline may be less informative. Thus, it may be hard for users to make a good, quick selection by scanning the entire skyline that consists of too many services.

Contribution: In this paper, we address the above mentioned issues by considering a fuzzy dominance relationship between Web services based on their QoS attributes. Our main contributions can be summarized as follows:

- We introduce a novel concept, called α-*dominant* service skyline, to tackle the problem of QoS-based web service selection.
- We develop a suitable algorithm, which leverages pruning techniques to efficiently compute the α-*dominant* service skyline.
- We evaluate both the effectiveness of the proposed concept and the efficiency of the algorithm through a set of experiments.

Outline: The rest of the paper is structured as follows. Section II provides the formal definition and analysis of the α-*dominant* service skyline. Section III describes the α-*dominant* service skyline computation algorithm. Section IV presents our experimental study. Section V reviews related work. Finally, Section VI concludes the paper.

II. Definitions and Analysis

In this section, we introduce our terminology and notation. Then, we formalize our concept, called α-*dominant* service skyline, based on a dominance relationship defined in a fuzzy way. To motivate and justify our formulation, we also discuss some related notions, namely Pareto-dominance and service skyline, showing that our concept is more adequate for Web service selection.

Given a set $\mathcal{S} = \{S_1, \cdots, S_n\}$ of functionally similar Web services and a set $Q = \{q_1, \cdots, q_d\}$ of QoS attributes. Each service S_i is characterized by a vector $Q(S_i) = (q_1(S_i), \cdots, q_d(S_i))$ where $q_\imath(S_i)$ denotes the value of the $\imath$-th QoS attribute of S_i. We consider quantitative QoS attributes (e.g., execution time, price, reputation, etc). To allow for a uniform measurement of services qualities, we also assume that QoS values are normalized on the range $[0, 1]$ (i.e., $\forall S_i \in \mathcal{S}, \forall q_\imath \in Q, q_\imath(S_i) \in [0, 1]$) such that the smaller values they have, the better they are. Note that the issue of normalizing the QoS values is out of the scope of our current study.

A. Fuzzy dominance vs Pareto dominance

We start by defining the Pareto dominance, then discuss the reasons that motivate to make it fuzzy.

Definition 1: (Pareto dominance)
Given two services $S_i, S_j \in \mathcal{S}$, we say that S_i dominates S_j, denoted by $S_i \prec S_j$, iff S_i is as good or better than S_j in all attributes in Q and better in at least one attribute in Q. i.e., $\forall \imath \in [1, d], q_\imath(S_i) \leq q_\imath(S_j) \wedge \exists \jmath \in [1, d], q_\jmath(S_i) < q_\jmath(S_j)$.

Pareto dominance does not allow for discriminating between Web services with bad compromise and those with good compromise. To illustrate this issue, let $Q(S_3) = (q_1(S_3), q_2(S_3)) = (0.1, 0.9)$ and $Q(S_{12}) = (q_1(S_{12}), q_2(S_{12})) = (0.2, 0.2)$ be the QoS vectors of S_3 and S_{12}, respectively. i.e., q_1 and q_2 represent respectively the execution time and the price (see Figure 1). With Pareto order, we have neither $S_3 \prec S_{12}$ nor $S_{12} \prec S_3$, i.e., the services S_3 and S_{12} are incomparable. However, one can consider that S_{12} is better than S_3 since $q_2(S_{12}) = 0.2$ is too much preferred than $q_2(S_3) = 0.9$, contrariwise $q_1(S_3) = 0.1$ is almost close to $q_1(S_{12}) = 0.2$. For this purpose, it is interesting to fuzzify the Pareto dominance in order to express the extent to which a Web service (more or less) dominates another one [6].

We define below a fuzzy dominance relationship that relies on particular comparison function expressing a graded inequality of the type *strongly smaller* than.

Definition 2: (Fuzzy dominance)
Given two services $S_i, S_j \in \mathcal{S}$, we define the fuzzy dominance to express the degree to which S_i dominates S_j as:

$$deg_{\mu_{\varepsilon,\lambda}}(S_i \prec S_j) = \frac{\sum_{\imath=1}^{d} \mu_{\varepsilon,\lambda}(q_\imath(S_i), q_\imath(S_j))}{d} \quad (1)$$

Where $\mu_{\varepsilon,\lambda}$ is a monotone comparison function that expresses the extent to which $q_\imath(S_i)$ is more or less (strongly) smaller than $q_\imath(S_j)$. The function $\mu_{\varepsilon,\lambda}$ can be defined in an absolute way as follows:

$$\mu_{\varepsilon,\lambda}(x, y) = \left\{ \begin{array}{ll} 0 & if\ y - x \leq \varepsilon \\ 1 & if\ y - x \geq \lambda + \varepsilon \\ \frac{y-x-\varepsilon}{\lambda} & otherwise \end{array} \right\} \quad (2)$$

Where $\lambda > 0$, i.e., $\mu_{\varepsilon,\lambda}$ is more demanding than the idea of *strictly smaller*. We should also have $\varepsilon \geq 0$ in order to ensure that $\mu_{\varepsilon,\lambda}$ agrees with the idea of *smaller* in the usual sense. Figure 2, shows the graphical representation of the function $\mu_{\varepsilon,\lambda}$ in terms of the difference $y - x$.

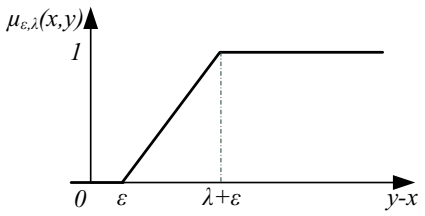

Figure 2. Graphical representation of $\mu_{\varepsilon,\lambda}$ w.r.t. $y - x$

One can interpret the semantics of the function $\mu_{\varepsilon,\lambda}$ as follows:

- if $y - x$ is less than ε, then x is not at all strongly smaller than y.
- if $y - x$ is larger than $\lambda + \varepsilon$, then x is all much smaller then y.
- if $y - x$ is between ε and $\lambda + \varepsilon$, then x is much smaller than y to some extent.

Let us now reconsider the previous Web services S_3 and S_{12}, with $\varepsilon = 0.1$ and $\lambda = 0.2$ we have $deg_{\mu_{0.1,0.2}}(S_3 \prec S_{12}) = 0$ and $deg_{\mu_{0.1,0.2}}(S_{12} \prec S_3) = 0.5$. This is more significant than S_3 and S_{12} are incomparable (provided by Pareto dominance). As can be seen, the fuzzy dominance introduced favors Web services with a good compromise.

B. α-dominant service skyline vs service skyline

We first formally define the service skyline and the α-*dominant* service skyline, we then investigate the difference between them showing that the latter is more robust.

Definition 3: (Service skyline)
The service skyline of $\mathcal{S}$, denoted by $sky^{\mathcal{S}}$ comprises the set of services in $\mathcal{S}$ that are not dominated by any other service, i.e., $sky^{\mathcal{S}} = \{S_i \in \mathcal{S} \mid \nexists S_j \in \mathcal{S}; S_j \prec S_i\}$.

In contrast to the service skyline which relies on Pareto dominance, the α-*dominant* service skyline leverages a notion called α-*dominance*. Below, we define both of the α-*dominance* and the α-*dominant* service skyline.

Definition 4: (α-*dominance*)
Given two services $S_i, S_j \in \mathcal{S}$ and $\alpha \in [0, 1]$, we say that S_i α-*dominates* S_j (or S_i dominates S_j at a degree α) in the context of $\mu_{\varepsilon,\lambda}$, denoted by $S_i \prec^{\alpha}_{\mu_{\varepsilon,\lambda}} S_j$, iff $deg_{\mu_{\varepsilon,\lambda}}(S_i \prec S_j) \geq \alpha$.

For instances, in the context of $\mu_{0.1,0.2}$ the service S_{12} 0.7-*dominates* both of S_5 and S_6. In the same context, S_{12} 0.8-*dominates* S_5 but does not 0.8-*dominates* S_6 as $deg_{\mu_{0.1,0.2}}(S_{12} \prec S_6) = 0.75 < 0.8$.

Definition 5: (α-*dominant* service skyline)
The α-*dominant* service skyline of $\mathcal{S}$ w.r.t. $\mu_{\varepsilon,\lambda}$, denoted by α-$sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$, comprises the set of services in $\mathcal{S}$ that are not α-*dominated* by any other service in the context of $\mu_{\varepsilon,\lambda}$, i.e., α-$sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}} = \{S_i \in \mathcal{S} \mid \nexists S_j \in \mathcal{S}; S_j \prec^{\alpha}_{\mu_{\varepsilon,\lambda}} S_i\}$.

For example, with $\varepsilon = 0.1, \lambda = 0.2$ and $\alpha = 0.7$ we have 0.7-$sky^{\mathcal{S}}_{\mu_{0.1,0.2}} = \{S_3, S_4, S_{11}, S_{12}, S_{13}, S_{14}\}$.

One can observe that, in contrast to the service skyline, the α-*dominant* service skyline privileges Web services with a good compromise. Now if the Web service S_{12} fails, users can choose between a good deal of Web services with a good compromise (e.g., S_{11} and S_{13}).

Further, the following theorem provides another key property of the α-*dominant* service skyline: a*ll Web services selected by the service skyline can be also selected by the* α-*dominant service skyline*.

Theorem 1: If $\alpha > \frac{d-1}{d}$, then the service skyline is a subset of the α-*dominant* service skyline for any comparison function $\mu_{\varepsilon,\lambda}$, i.e., $\alpha > \frac{d-1}{d} \Longrightarrow sky^{\mathcal{S}} \subset \alpha$-$sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}} (\forall \varepsilon \geq 0, \forall \lambda > 0)$.

Proof: Assume that $\alpha > \frac{d-1}{d}$ and prove that for any comparison function $\mu_{\varepsilon,\lambda}$, $sky^{\mathcal{S}} \subset \alpha$-$sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$.
Let $S_i \in sky^{\mathcal{S}}$. According to Definition 3, $\nexists S_j \in \mathcal{S}; S_j \prec S_i$. i.e., $\forall S_j \in \mathcal{S}, \exists k \in [1, d]\,; q_k(S_i) < q_k(S_j)$. Therefore, for any comparison function $\mu_{\varepsilon,\lambda}$ we will have: $\forall S_j \in \mathcal{S}, \exists k \in [1, d]\,; \mu_{\varepsilon,\lambda}(q_k(S_j), q_k(S_i)) = 0$. Thus $\forall S_j \in \mathcal{S}; deg_{\mu_{\varepsilon,\lambda}}(S_j \prec S_i) \leq \frac{d-1}{d}$ as S_i is better at least on the dimension j. Then $\forall S_j \in \mathcal{S}, deg_{\mu_{\varepsilon,\lambda}}(S_j \prec S_i) < \alpha$, since $\alpha > \frac{d-1}{d}$. This means that S_i is not α-*dominated* by any other service S_j in $\mathcal{S}$, i.e., $\nexists S_j \in \mathcal{S}; S_j \prec^{\alpha}_{\mu_{\varepsilon,\lambda}} S_i$. Thus, $S_i \in \alpha$-$sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$.
Hence, $sky^{\mathcal{S}} \subset \alpha$-$sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}} (\forall \varepsilon \geq 0, \forall \lambda > 0)$. ■

Theorem 1 shows that the α-*dominant* service skyline is appropriate for all types of users. In other words, if a user prefers a Web service with a bad compromise (e.g., he/she prefers a fast service although it is very expensive), the α-*dominant* service skyline can include such kind of services.

In addition to the above observations, the α-*dominant* service skyline allows users to control the size of the returned services by making changes on the parameter α, and possibly on ε and λ (whereas the service skyline's size does not change for the same set of services and the same query). For example, if users find that the size of 0.7-$sky^{\mathcal{S}}_{\mu_{0.1,0.2}}$ is quite large (resp. small), they can reduce (resp. expand) it by decreasing (resp. increasing) the value of α (e.g., $\alpha = 0.5$ or $\alpha = 0.8$). They will thus have 0.5-$sky^{\mathcal{S}}_{\mu_{0.1,0.2}} = \{S_3, S_{11}\}$ or 0.8-$sky^{\mathcal{S}}_{\mu_{0.1,0.2}} = \{S_3, S_4, S_6, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}\}$. Roughly speaking, the α-*dominant* service skyline allows for taking the feedback of users into account. We show formally this behavior below:

Lemma 1: If $\alpha' < \alpha$, then the α'*-dominant* service skyline w.r.t. $\mu_{\varepsilon,\lambda}$ is a subset of the α*-dominant* service skyline w.r.t. $\mu_{\varepsilon,\lambda}$, i.e., $\alpha' < \alpha \Longrightarrow \alpha'\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}} \subset \alpha\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$.

Proof: Let $\mu_{\varepsilon,\lambda}$ be a comparison function. Assume that $\alpha' < \alpha$ and prove that $\alpha'\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}} \subset \alpha\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$.
Let $S_i \in \alpha'\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$. This means that S_i is not α'*-dominated* by any other service S_j in $\mathcal{S}$, i.e., $\nexists S_j \in \mathcal{S}; S_j \prec^{\alpha'}_{\mu_{\varepsilon,\lambda}} S_i$. Thus, there is not a service S_j in $\mathcal{S}$ such as $deg_{\mu_{\varepsilon,\lambda}}(S_j \prec S_i) \geq \alpha'$, i.e., $\forall S_j \in \mathcal{S}; deg_{\mu_{\varepsilon,\lambda}}(S_j \prec S_i) < \alpha'$. Then, $\forall S_j \in \mathcal{S}; deg_{\mu_{\varepsilon,\lambda}}(S_j \prec S_i) < \alpha$, since $\alpha' < \alpha$. Therefore $\nexists S_j \in \mathcal{S}; S_j \prec^{\alpha}_{\mu_{\varepsilon,\lambda}} S_i$. This means that S_i is not α-*dominated* , thus $S_i \in \alpha\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$.
Hence, $\alpha'\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}} \subset \alpha\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$. ■

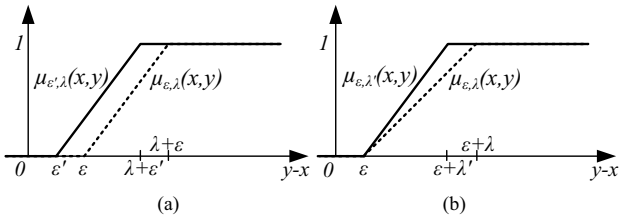

Figure 3. Effects of ε and λ

Property 1: If $\varepsilon' \leq \varepsilon$ and $\lambda' \leq \lambda$, then for any $(x, y) \in [0, 1]$, $\mu_{\varepsilon',\lambda'}(x, y) \geq \mu_{\varepsilon,\lambda}(x, y)$, i.e., $(\varepsilon' \leq \varepsilon \wedge \lambda' \leq \lambda) \Longrightarrow \forall (x, y) \in [0, 1]\,, \mu_{\varepsilon',\lambda'}(x, y) \geq \mu_{\varepsilon,\lambda}(x, y)$.

Proof:
$[\varepsilon' \leq \varepsilon \wedge \lambda' = \lambda]$: see Figure 3 (plot-a). $\ldots (\star)$
$[\varepsilon' = \varepsilon \wedge \lambda' \leq \lambda]$: see Figure 3 (plot-b). $\cdots (*)$
$[\varepsilon' \leq \varepsilon \wedge \lambda' \leq \lambda]$: it is straightforward from $(\star)$ and $(*)$. ■

Lemma 2: If $\mu_{\varepsilon',\lambda'} \geq \mu_{\varepsilon,\lambda}$, then for any $\alpha \in [0, 1]$ the α*-dominant* service skyline w.r.t. $\mu_{\varepsilon',\lambda'}$ is a subset of the α*-dominant* service skyline w.r.t. $\mu_{\varepsilon,\lambda}$, i.e., $\mu_{\varepsilon',\lambda'} \geq \mu_{\varepsilon,\lambda} \Longrightarrow \alpha\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon',\lambda'}} \subset \alpha\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$.

Proof: Let α be a dominance degree. Assume that $\mu_{\varepsilon',\lambda'} \geq \mu_{\varepsilon,\lambda}$ and prove that $\alpha\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon',\lambda'}} \subset \alpha\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$.
Let $S_i \in \alpha\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon_1,\lambda}}$. This means that S_i is not α-*dominated* by any other service S_j in $\mathcal{S}$ in the context of $\mu_{\varepsilon',\lambda'}$, i.e., $\nexists S_j \in \mathcal{S}; S_j \prec^{\alpha}_{\mu_{\varepsilon',\lambda'}} S_i$. Thus, there is not a service S_j in $\mathcal{S}$ such as $deg_{\mu_{\varepsilon',\lambda'}}(S_j \prec S_i) \geq \alpha'$, i.e., $\forall S_j \in \mathcal{S}; deg_{\mu_{\varepsilon',\lambda'}}(S_j \prec S_i) < \alpha'$. Then, $\forall S_j \in \mathcal{S}, \frac{\sum_{i=1}^{d} \mu_{\varepsilon',\lambda'}(q_i(S_j),q_i(S_i))}{d} < \alpha$. Thus, $\forall S_j \in \mathcal{S}, \frac{\sum_{i=1}^{d} \mu_{\varepsilon,\lambda}(q_i(S_j),q_i(S_i))}{d} < \alpha$ (since $\mu_{\varepsilon',\lambda'} \geq \mu_{\varepsilon,\lambda}$). Then, $\forall S_j \in \mathcal{S}, deg_{\mu_{\varepsilon,\lambda}}(S_j \prec S_i) < \alpha$. Thus, $\nexists S_j \in \mathcal{S};\ S_j \prec^{\alpha}_{\mu_{\varepsilon,\lambda}} S_i$. This means that S_i is not α-*dominated* in the context of $\mu_{\varepsilon,\lambda}$, therefore $S_i \in \alpha\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$.
Hence, $\alpha\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon',\lambda'}} \subset \alpha\text{-}sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$. ■

Lemma 1 and both Property 1 and Lemma 2 provide appropriate tools in order to adapt (by contracting or expanding) the size of the retrieved services to users needs.

III. Computing the α-Dominant Service Skyline

Index structures are frequently used to reduce search space in large databases. To this end, in our study we make use of R-trees structures [7] due to their popularity and effectiveness in skyline computation. For the sake of illustration, let us use the Web services given in Figure 1. These services can be organized in the R-tree of Figure 4, with node capacity $= 3$. An intermediate entry e_i corresponds to the minimum bounding rectangle (MBR) of a node N_i at the lower level, while a leaf entry corresponds to a Web service. Distances are computed according to L1 norm, i.e., the *mindist* of a point equals the sum of its coordinates and the *mindist* of a MBR (i.e., intermediate entry) equals the *mindist* of its lower-left corner point.

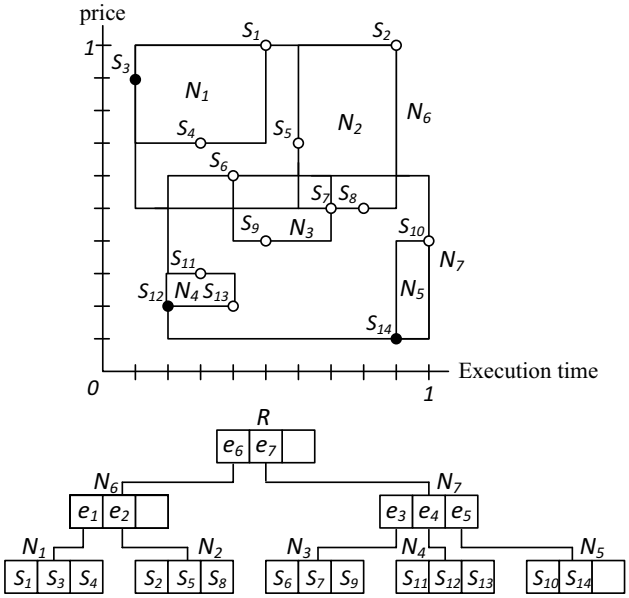

Figure 4. An example of R-trees

A. Efficient computation of the α-dominant service skyline

Intuitively, a straightforward approach to compute the α-*dominant* service skyline is to compare each Web service S_i with every other one. If S_i is not α-*dominated*, then it belongs to the α-*dominant* service skyline. However, this approach results in a high computational cost, as it needs to compare each Web service with every others. It is thus crucial to quickly eliminate Web services that are α-*dominated*.

It is worth to note that contrary to Pareto dominance the α-*dominance* relationship is not asymmetric, for instance in the context of $\mu_{0.1,0.2}$, S_{13} 0.5-*dominates* S_3 and also S_3 0.5-*dominates* S_{13}. Therefore, the pruning process is far from being straightforward since it can lead to erroneous

results. For example, if S_3 is pruned as it is α-*dominated* by S_{13} and there is no other service that α-*dominates* S_{13} (the case of our example given in Figure 1, for $\varepsilon = 0.1$, $\lambda = 0.2$ and $\alpha = 0.5$), then S_{13} will be included in the α-*dominant* service skyline, whereas it is α-*dominated* by S_3. This justifies why the current skyline algorithms are not suitable for computing the α-*dominant* service skyline.

In the following, we provide optimization techniques to address the above mentioned issues. The idea is to prune services that are both α-*dominated* and not needed for pruning other services and to minimize the number of comparisons. The optimization techniques follow an important concept called α-*Pareto-dominance*.

Definition 6: (α-Pareto-dominance)
Given two services $S_i, S_j \in \mathcal{S}$, we say that S_i α-Pareto-dominates S_j in the context of $\mu_{\varepsilon,\lambda}$, denoted by $S_i \lhd^{\alpha}_{\mu_{\varepsilon,\lambda}} S_j$, iff $S_i \prec S_j \wedge S_i \prec^{\alpha}_{\mu_{\varepsilon,\lambda}} S_j$

Lemma 3: Given two services $S_i, S_j \in \mathcal{S}$, if S_i dominates S_j, then for any $S_k \in \mathcal{S}$ and for any comparison function $\mu_{\varepsilon,\lambda}$: $deg_{\mu_{\varepsilon,\lambda}}(S_i \prec S_k) \geq deg_{\mu_{\varepsilon,\lambda}}(S_j \prec S_k)$, i.e., $S_i \prec S_j \Longrightarrow \forall S_k \in \mathcal{S}, deg_{\mu_{\varepsilon,\lambda}}(S_i \prec S_k) \geq deg_{\mu_{\varepsilon,\lambda}}(S_j \prec S_k)$ $(\forall \varepsilon \geq 0, \forall \lambda > 0)$.

Proof: $S_i \prec S_j \Longleftrightarrow \forall \imath \in [1,d], q_\imath(S_i) \leq q_\imath(S_j) \wedge \exists \jmath \in [1,d], q_\jmath(S_i) < q_\jmath(S_j)$. Taking only the implication $\Longrightarrow$, we will have, $S_i \prec S_j \Longrightarrow \forall \imath \in [1,d], q_\imath(S_i) \leq q_\imath(S_j)$. Thus for any service S_k, $\forall \imath \in [1,d], q_\imath(S_i) - q_\imath(S_k) \leq q_\imath(S_j) - q_\imath(S_k)$. Then, $\forall \imath \in [1,d], q_\imath(S_k) - q_\imath(S_i) \geq q_\imath(S_k) - q_\imath(S_j)$. Therefore, for any comparison function $\mu_{\varepsilon,\lambda}$, we will have, $\forall \imath \in [1,d], \mu_{\varepsilon,\lambda}(q_\imath(S_i), q_\imath(S_k)) \geq \mu_{\varepsilon,\lambda}(q_\imath(S_j), q_\imath(S_k))$. It follows that
$\frac{\sum_{\imath=1}^{d} \mu_{\varepsilon,\lambda}(q_\imath(S_i), q_\imath(S_k))}{d} \geq \frac{\sum_{\imath=1}^{d} \mu_{\varepsilon,\lambda}(q_\imath(S_j), q_\imath(S_k))}{d}$
Hence, $deg_{\mu_{\varepsilon,\lambda}}(S_i \prec S_k) \geq deg_{\mu_{\varepsilon,\lambda}}(S_j \prec S_k)$. ■

Lemma 4: For $\alpha > \frac{d-1}{d}$, if a service S_i is not α-Pareto-dominated by any service in $\mathcal{S}$ in the context of a comparison function $\mu_{\varepsilon,\lambda}$, then $S_i \in sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$

Proof: Assume that S_i is not α-Pareto-dominated. This means that $\forall S_j \in \mathcal{S}, S_j \nprec S_i \vee S_j \nprec^{\alpha}_{\mu_{\varepsilon,\lambda}} S_i$.
$[S_j \nprec^{\alpha}_{\mu_{\varepsilon,\lambda}} S_i]$: the proof is obvious as S_i is not α-dominated.
$[S_j \nprec S_i]$: by adopting the same formality of the proof of theorem 1, we will have $\nexists S_j \in \mathcal{S}; S_j \prec^{\alpha}_{\mu_{\varepsilon,\lambda}} S_i$.
Hence, $S_i \in sky^{\mathcal{S}}_{\mu_{\varepsilon,\lambda}}$. ■

Lemma 4 shows that the skyline services are sufficient to decide if a service is part (or not) of the α-*dominant* service skyline. This essentially reduces the number of comparisons. Also, the combination of Definition 6 and Lemma 4 specifies a key property that can be used to prune services: *if a service S_j is α-Pareto-dominated then prune it as (i) it is not part of the α-dominant service skyline (it is α-dominated) and (ii) it is unnecessary for comparisons (it is dominated)*. In addition, Lemma 5 helps to avoid any comparison after pruning all α-*dominated* services, in the case where $\alpha > \frac{d-1}{d}$.

B. α-dominant service skyline algorithm

The algorithm, hereafter referred to as α-$\mathcal{DSSA}$, leverages the techniques presented above to compute the α-*dominant* service skyline, avoiding an exhaustive comparison of each service with all other ones. More specifically, it proceeds in two steps. The goal of the first step is to prune the α-*Pareto-dominated* services. The remaining services will go into the second step, where the α-*dominant* ones will be selected.

Algorithm 1: α-$\mathcal{DSSA}$

Input: *R-tree R; dominance degree α; comparison function $\mu_{\varepsilon,\lambda}$;*
Output: *the α-dominant service skyline DSky;*

```
1  H ← ∅; Sky ← ∅; Dom ← ∅;
2  insert the root entries of R into H;
3  while H ≠ ∅ do
4      extract the top entry e from H;
5      if e is α-Pareto-dominated by some service in Sky then
6          discard e;
7      else
8          if e is an intermediate entry then
9              foreach child e_i of e do
10                 if e_i is not α-Pareto-dominated by some service
                   in Sky then
11                     insert e_i into H;
12         else
13             if e_i is not dominated by some service in Sky then
14                 insert e_i into Sky;
15             else
16                 insert e_i into Dom;
17 if α > (d−1)/d then
18     DSky ← Sky ∪ Dom;
19 else
20     foreach S_i in Sky do
21         if S_i is not α-dominated by some service in Sky then
22             insert S_i into DSky;
23         if S_i α-dominate a service S_j in Dom then
24             discard S_j;
25     DSky ← DSky ∪ Dom;
26 return DSky;
```

1) Finding candidate services (lines 1-16): It stars from the root node of the R-tree R and inserts all its entries into the heap H, sorted in ascending order according to their *mindist*. The top entry e (i.e., the entry with the minimum *mindist*) is extracted. If e is α-*Pareto-dominated* by any service in *Sky* then discard it, since all the services obtained from it (i.e., e) are α-*dominated* and not useful to prune

other entries. Otherwise (i.e., e is not α-*Pareto-dominated*), if e is an intermediate entry, insert all its child entries that are not α-*Pareto-dominated* by any service in Sky into the heap. Else (i.e., e is a service), there are two cases: (i) If e is not dominated by any service in Sky (i.e., e is a skyline service), it is inserted into Sky as it may be part of the α-*dominant* service skyline and it is necessary for pruning other entries. (ii) If e is dominated by any service in Sky, it is inserted into Dom as it may be part of the α-*dominant* service skyline but it is not necessary for pruning other entries. This step proceeds in the same manner until the heap becomes empty.

2) Computing and returning the α-dominant service skyline (lines 17-26): If $\alpha > \frac{d-1}{d}$, then the α-*dominant* service skyline comprises all services of Sky and Dom, according to Lemma 5. Otherwise (i.e., $\alpha \leq \frac{d-1}{d}$), the algorithm proceeds by refining the lists Sky and Dom, keeping only the services that are not α-*dominated* by any services in Sky, as they are also not α-*dominated* by any services in Sky. Finally it provides the user with the α-*dominant* service skyline (line 26).

Note that according to Lemma 2 and Lemma 3, once we compute the α-*dominant* service skyline w.r.t. $\mu_{\varepsilon,\lambda}$, the α'-*dominant* service skyline w.r.t. $\mu_{\varepsilon,\lambda}$ ($\alpha' < \alpha'$) and the α-*dominant* service skyline w.r.t. $\mu_{\varepsilon',\lambda'}$ ($\mu_{\varepsilon',\lambda'} \geq \mu_{\varepsilon,\lambda}$) can be computed by only performing a simple search on the services returned by the α-*dominant* service skyline w.r.t. $\mu_{\varepsilon,\lambda}$ instead of rerun the algorithm.

IV. Experimental Evaluation

In this section, we present an extensive experimental study of our approach. More specifically, we conduct two sets of experiments. First we focus on the size of the α-*dominant* service skyline. We also compute the service skyline (refereed to as $\mathcal{TSS}$) to compare how the size of the the α-*dominant* service skyline (refereed to as α-$\mathcal{DSS}$) varies from that of the traditional service skyline. In the second set of experiments, we study the computational cost of the proposed algorithm. In order to prove the efficiency and the scalability of our algorithm (α-$\mathcal{DSSA}$), we developed also a base line algorithm (referred to as $\mathcal{BLA}$) for comparison purpose.

The algorithms (i.e., α-$\mathcal{DSSA}$ and $\mathcal{BLA}$) were implemented in Java. Datasets are indexed with an R-tree with node capacity=100. The experiments were conducted on a 2.00 GHz Intel dual core CPU and 2 GB of RAM, running Windows.

A. Experimental Setup

In our experimental study we focus on synthetically generated datasets due to the limited availability of real data. The QoS values of services are generated in three different ways: independent (ind), where QoS values are assigned independently to each QoS attribute; correlated (cor), where the QoS values of a service are positively correlated, i.e., a good value in some QoS attribute increases the possibility of a good value in the others; anti-correlated (ant), where the QoS values are negatively correlated, i.e., good values (or bad values) in all QoS attributes are less likely to occur. The involved parameters and their examined values are summarized in Table I. In all experimental setups, we investigate the effects of one parameter, while we set the remaining ones to their default values, shown in bold in Table I. Due to lack of space, the effects of the parameters ε and λ are left for future work. For this purpose, we used $\varepsilon = 0.05$ and $\lambda = 0.2$.

Table I
Parameters and examined values

Parameter	Symbol	Values
Number of services	n	1K, **10K**, 100K, 1M
QoS dimensions	d	$[2, 5]$d, **3d**
Dominance degree	α	$[0.2, 0.8]$, **0.5**
Parameter correlation	$corr$	**ind**, cor, ant

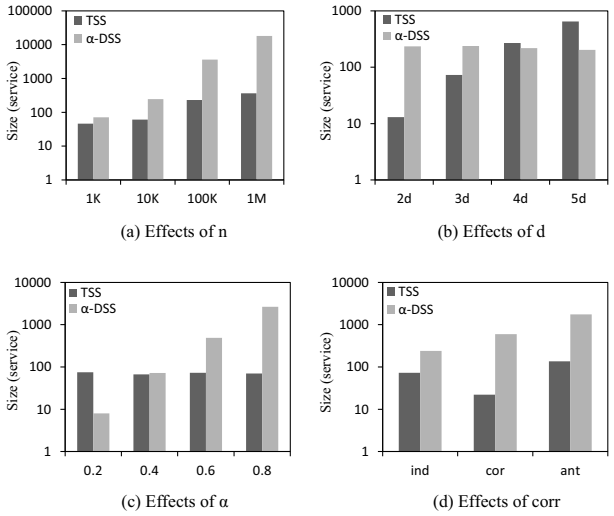

Figure 5. The effects of parameters on the size of the traditional service skyline and the α-*dominant* service skyline

B. Size of the α-dominant service skyline

1) The effects of n: Figure 5 (plot-a) shows that the size of the α-*dominant* service skyline follows a similar trends as the traditional skyline with the increase of n. With α set to 0.5, the size of the α-*dominant* service skyline is larger than that of the traditional service skyline. In addition, the difference between the two sizes is proportional to n, since the number of services with a good compromise increases as n increases.

2) The effects of d: In contrast to the traditional service skyline whose size increases significantly as d increases, d has no obvious effect on the size of the α-*dominant* service

skyline as shown in Figure 5 (plot-b) (the sizes can be regarded as in the same scale varying d). Since the number of services with a good compromise is approximately the same when d varies.

3) The effects of α: The dominance degree α has a significant effect on the size of the α-*dominant* service skyline as shown in Figure 5 (plot-c) (the size of the traditional service skyline does not change as it is not related to α). This is because the increase (resp. decrease) of α leads to the inclusion (resp. exclusion) of services with a bad compromise.

4) The effects of corr: Figure 5 (plot-d) shows that the α-*dominant* service skyline and the traditional service skyline exhibit different behaviors w.r.t. corr parameter. Furthermore, the size of the α-*dominant* service skyline is larger than that of the traditional service skyline especially for the correlated and anti-correlated datasets. On the correlated datasets many services with a good compromise occur. They are thus included in the α-*dominant* service skyline. On the anti-correlated datasets, services with a good compromise less likely to occur, thus there are not enough services which α-dominate others.

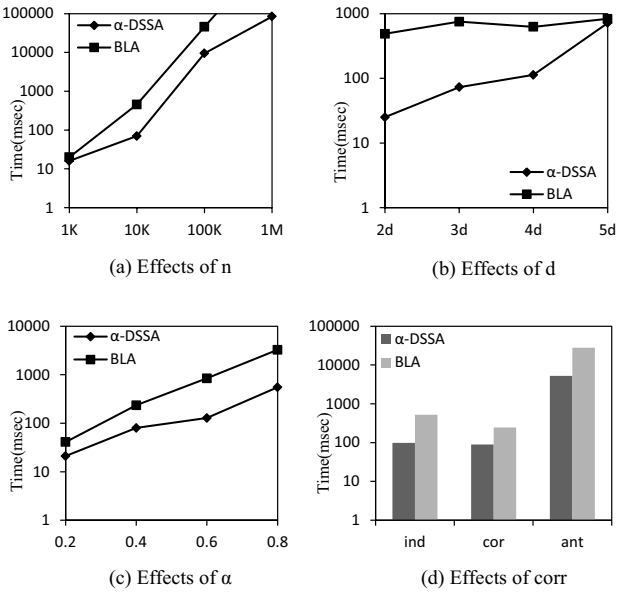

Figure 6. The effects of parameters on the time of $\mathcal{BLA}$ and α-$\mathcal{DSSA}$

C. Performance and Scalability

1) The effects of n: Figure 6 (plot-a) shows that α-$\mathcal{DSSA}$ is consistently more efficient than $\mathcal{BLA}$ and $\mathcal{BLA}$ does not terminate successfully for $n > 100K$. The difference between α-$\mathcal{DSSA}$ and $\mathcal{BLA}$ increases significantly as n increases. This is due to the high number of pruned services by α-$\mathcal{DSSA}$.

2) The effects of d: α-$\mathcal{DSSA}$ is consistently more efficient than $\mathcal{BLA}$ as shown in Figure 6 (plot-b). The difference between α-$\mathcal{DSSA}$ and $\mathcal{BLA}$ decreases as d increases. This is because the size of the skyline increases significantly as d increases, thus the number of pruned services by α-$\mathcal{DSSA}$ decreases. In other words, the number of services selected to the second step of α-$\mathcal{DSSA}$ increases.

3) The effects of α: Figure 6 (plot-c) shows that α-$\mathcal{DSSA}$ is faster than $\mathcal{BLA}$ in a consistent manner and its performance advantage over $\mathcal{BLA}$ becomes more obvious with increasing α. Since the size of the α-*dominant* service skyline increases significantly as α increases. Then, the number of comparison involved in $\mathcal{BLA}$ is significant.

4) The effects of corr: α-$\mathcal{DSSA}$ is more efficient than $\mathcal{BLA}$ on all distributions as shown in Figure 6 (plot-d). In addition, α-$\mathcal{DSSA}$ is lower on anti-correlated datasets than correlated and independent datasets. This is because the size of the skyline is quite large on on anti-correlated datasets, thus the number of pruned services decreases. Then, the number of services selected to the second step of α-$\mathcal{DSSA}$ increases.

V. Related work

During the last years, the problem of QoS-based web service selection has received a lot of attention in the service computing community. In [8], the authors propose an extensible and dynamic QoS computation model that takes into account the feedback from users as well as other business related criteria. Zeng et al. [9][10] propose a general and extensible model to evaluate QoS of both elementary and composite services. The authors use linear programming techniques to find the optimal selection of component services. In [11] the authors studied the problem of service selection with multiple QoS constraints. The authors propose two models for the QoS-based service composition problem: (i) a combinatorial model and (ii) a graph model. A heuristic algorithm is introduced for each model. Wang et al. [12] introduces QoS-based selection of semantic web services, the authors present a QoS ontology and selection algorithm to evaluate multiple qualities. However, these approaches require the users to assign weights on QoS attributes. They thus suffer from lack of flexibility.

Skyline analysis was introduced in the database domain by Boïzsoñyi et al. [13], which develop basic algorithms based on block nested loops (BNL), divide-and-conquer and index scanning (B-tree). Tan et al. [14] introduce techniques, which can output the skyline without having to scan the entire dataset. The BNL algorithm was further improved with a pre-sorting scheme in [15][16]. NN (Nearest Neighbor) and BBS (Branch and Bound) are among the pioneer algorithms that can progressively process the skyline based on a R-tree indexing structure. Kossmann et al. [17] present an improved algorithm, called NN due to its reliance on nearest neighbor search, which applies the divide-and-conquer framework on

datasets indexed by R-trees. In the work [18], which also uses R-trees, the authors propose an optimal and progressive algorithm for skyline computation based on the Branch and Bound paradigm.

Recently, skyline computation is adopted in Web services selection. The work in [3] focuses on the selection of skyline services for QoS based Web service composition. A method for determining which QoS levels of a service should be improved so that it is not dominated by other services is also discussed. In [4], the authors propose a skyline computation approach for service selection. The resulting skyline, called multi-service skyline, enables services users to optimally and efficiently access sets of service as an integrated service package. In the robust work [5], Yu and Bouguettaya address the problem of uncertainty inherent in QoS and compute the skylines from service providers (referred to as service skylines). A service skyline can be regarded as a set of service providers that are not dominated by others in terms of QoS aspects that interest all users. To this end, a concept called p-dominant service skyline is defined. A provider S belongs to the p-dominant skyline if the probability that S is dominated by any other provider is less than p. The authors provide also a discussion about the interest of p-dominant skyline w.r.t. the notion of p-skyline proposed in [19].

All the previous studies on skyline computation rely on Pareto dominance. The α-*dominant* service skyline (proposed in this work) leverages fuzzy dominance. A formal comparison between Pareto dominance and fuzzy dominance, then between the service skyline and the α-*dominant* service skyline is given in section 2. In addition, our experimental study also demonstrates the effectiveness of the α-*dominant* service skyline.

VI. Conclusion

In this paper, we have addressed the problem of QoS-based Web service selection. We have introduced a new concept, called α-*dominant* service skyline, to overcome the major issues of the current approaches: (i) requiring users to assign weights to QoS attributes, (ii) privileging the services with a bad compromise between different QoS attributes and (iii) not allowing users to control the size of the returned set of services. Further, we have developed a suitable algorithm for computing the α-*dominant* service skyline using pruning techniques. An interesting future direction is to extend this work to QoS-based Web service composition.

References

[1] M. P. Papazoglou and D. Georgakopoulos, "Service-oriented computing: Introduction," *Commun. ACM*, vol. 46, no. 10, pp. 24–28, 2003.

[2] F. Curbera, M. J. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the web services web: An introduction to soap, wsdl, and uddi," *IEEE Internet Computing*, vol. 6, no. 2, pp. 86–93, 2002.

[3] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for qos-based web service composition," in *WWW*, 2010, pp. 11–20.

[4] Q. Yu and A. Bouguettaya, "Computing service skylines over sets of services," in *ICWS*, 2010, pp. 481–488.

[5] ——, "Computing service skyline from uncertain qows," *IEEE T. Services Computing*, vol. 3, no. 1, pp. 16–29, 2010.

[6] K. Benouaret, D. Benslimane, and A. Hadjali, "Top-k service compositions: A fuzzy set-based approach," in *SAC*, 2011, pp. (1033–1038).

[7] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *SIGMOD Conference*, 1984, pp. 47–57.

[8] Y. Liu, A. H. H. Ngu, and L. Zeng, "Qos computation and policing in dynamic web service selection," in *WWW (Alternate Track Papers & Posters)*, 2004, pp. 66–73.

[9] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality driven web services composition," in *WWW*, 2003, pp. 411–421.

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

[11] T. Yu, Y. Z. 0001, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *TWEB*, vol. 1, no. 1, 2007.

[12] X. Wang, T. Vitvar, M. Kerrigan, and I. Toma, "A qos-aware selection model for semantic web services," in *ICSOC*, 2006, pp. 390–401.

[13] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *ICDE*, 2001, pp. 421–430.

[14] K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient progressive skyline computation," in *VLDB*, 2001, pp. 301–310.

[15] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *ICDE*, 2003, pp. 717–816.

[16] P. Godfrey, R. Shipley, and J. Gryz, "Maximal vector computation in large data sets," in *VLDB*, 2005, pp. 229–240.

[17] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in *VLDB*, 2002, pp. 275–286.

[18] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *SIGMOD Conference*, 2003, pp. 467–478.

[19] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *VLDB*, 2007, pp. 15–26.