Serving the Sky: Discovering and Selecting Semantic Web Services through Dynamic Skyline Queries

Dimitrios Skoutas[1,2] Dimitris Sacharidis[1] Alkis Simitsis[3] Timos Sellis[2,1]

[1]Nat'l Techn. Univ. of Athens
Athens, Greece
{dskoutas,dsachar}@dblab.ece.ntua.gr

[2]Inst. for the Mgmt of Inf. Syst.
Athens, Greece
timos@imis.athena-innovation.gr

[3]Stanford University
California, USA
alkis@db.stanford.edu

Abstract

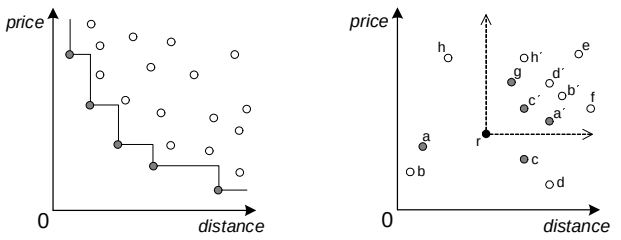
Semantic Web service descriptions are typically multi-parameter constructs. Discovering semantically relevant services, given a desirable service description, is typically addressed by performing a pairwise logic-based match between the requested and offered parameters. However, little or no attention is given to combining these partial results to compile the final list of candidate services. Instead, this is often done in an ad hoc manner, implying a priori assumptions regarding the user's preferences. In this paper, we focus on identifying the best candidate Semantic Web services given the description of a requested service. We model the problem as a skyline query, also known as the maximum vector problem, and we show how the service selection process can be performed efficiently. We consider different aspects, addressing both the requesters' and the providers' points of view. Experimental evaluation on real and synthetic data shows the effectiveness and efficiency of the proposed approach.

1 Introduction

Web services, as a key technology for realizing service-oriented architectures, promise to enable interoperability and integration between heterogeneous systems and applications. The discovery and selection of the appropriate services to fulfill a given request constitutes a fundamental task in such architectures. However, current industry standards for registering and locating Web services (WSDL, UDDI) aim at describing the structure of the service interface and of the exchanged messages, limiting the discovery process to essentially keyword-based search. Even though interoperability at the syntactic level is a necessary requirement, the identification and selection of appropriate services should be done in terms of the semantics of the requested and offered capabilities. To this direction, the Semantic Web, through the use of ontologies, provides the means to enrich the service descriptions with semantic information, allowing software agents to reason about the terms in these descriptions. This is a significant step for increasing the precision of the discovery process, as well as for minimizing the required human intervention. Several approaches have been proposed for adding semantics to Web service descriptions, including OWL-S, WSDL-S, and WSMO.

Semantic Web services matchmaking is basically addressed as a logic-based inference task [12, 11]. Service descriptions have multiple parameters, annotated with concepts from an associated ontology. A reasoner is used to match, in a pairwise manner, parameters from the requested and offered service. Then, the overall match is typically determined either as the worst match over all parameters or as a (weighted) average of the partial results. Clearly, both cases rely on an implicit, ad hoc assumption regarding the user's preferences, leading to biased results. This may have a significant impact on the perceived quality of the discovery process. In the case of a human user, a low precision on the top returned matches compromises the credibility of the discovery engine. Even worse, in a fully automated scenario, the software agent is expected to make a selection among the top returned services; choosing an inappropriate service breaks the whole workflow.

In this paper, we address the issue of selecting the best candidate services among those partially matching the given request. For this purpose, we use the notion of *skyline* [4], to define and compute the *potentially interesting* services for *any* type of user. A typical example used to illustrate this concept is searching for cheap hotels close to the beach. A sample set of hotels is depicted in Figure 1(a), characterized by two dimensions, *distance* and *price*. The drawn line indicates the skyline. A hotel belongs in the skyline if there is no other hotel that is better in both dimensions, i.e., both cheaper and closer to the beach. The distinguishing property of the skyline is that for any preference function f that is monotone on all attributes, if an object maximizes f, then



(a) Static Skyline (b) Dynamic Skyline

Figure 1. Static and dynamic skylines

this object is part of the skyline. Also, for every object in the skyline, there exists a monotone preference function that is maximized by this object. Intuitively, this means that (a) regardless of how a user weighs his/her preferences, his/her top preferred object will be one of the skyline objects, and (b) there is no skyline object which is nobody's top preference. The *dynamic skyline* is a variation, where the original objects are compared w.r.t. a given reference object r [13]. The reference object defines a new space, depicted as the inner coordinate system in Figure 1(b), and existing objects need to be transformed in this space. In this example, points a, c, b, d, h are projected to a', c', b', d', h' respectively (i.e., $p'_x = |p_x - r_x|, p'_y = |p_y - r_y|$). The dynamic skyline for the reference hotel r contains a', c' and g (i.e., a, c), but not b', d', e, f and h' (i.e., not b, d, h).

The analogy to our case is as follows: the space dimensions correspond to the service parameters being matched; the objects in the space correspond to the offered services; the reference object corresponds to the user's request.

Contributions. In particular, we make the following contributions:

(a) We study the problem of discovering and selecting the best candidate Semantic Web services w.r.t. the description of a desired service, formulating it as a skyline computation problem.

(b) Based on a state of the art skyline computation algorithm, we provide an effective and efficient way to handle the service selection process, dealing both with the requester's and the provider's perspectives.

(c) We experimentally evaluate the performance of the proposed approach using both real and synthetic data.

Outline. The rest of the paper is organized as follows. Section 2 defines the notion of static and dynamic skyline queries, and formulates the selection of Semantic Web services as a skyline computation problem, presenting also a suitable example. Section 3 shows how the best matches can be identified efficiently. Specific aspects of the service selection process, referring both to the service requester and provider, are addressed. Section 4 presents a detailed evaluation of the proposed approach, while Section 5 discusses related work. Section 6 concludes the paper.

2 Skyline Services

2.1 Background

Consider a set of points P in a d-dimensional space, with p^i denoting the value of point $p \in P$ in the i-th dimension.

Definition 1 ***(Dominance)*** *A point $p \in P$ dominates another point $q \in P$, denoted as $p \prec q$, iff p is as good or better than q in all dimensions and better in at least one dimension, i.e., $\forall i \in [1,d] : p^i \leq q^i$ and $\exists i \in [1,d] : p^i < q^i$.*

Definition 2 ***(Skyline)*** *The skyline of P, denoted by SL_P, comprises the set of points in P that are not dominated by any other point, i.e., $SL_P = \{p \in P | \nexists q \in P : q \prec p\}$.*

Definition 3 ***(Dynamic Dominance)*** *Given a reference point $r \in P$, a point $p \in P$ dominates another point $q \in P$ w.r.t. r, denoted as $p \prec^r q$, iff $\forall i \in [1,d] : |r^i - p^i| \leq |r^i - q^i|$ and $\exists i \in [1,d] : |r^i - p^i| < |r^i - q^i|$.*

Definition 4 ***(Dynamic Skyline)*** *Given a reference point $r \in P$, the dynamic skyline of P w.r.t. r, denoted by SL^r_P, comprises the set of points in P that are not dynamically dominated by any other point w.r.t. r, i.e., $SL^r_P = \{p \in P | \nexists q \in P : q \prec^r p\}$.*

2.2 Problem formulation

The functional part of a Semantic Web service can be described by a tuple $SWS = (I, O, P, E)$, where I, O, P, E are sets of inputs, outputs, preconditions, and effects, with each parameter semantically annotated by means of an associated ontology $\mathcal{O}$. We assume that the languages OWL and OWL-S are used to represent, respectively, the domain ontology and the requested and offered services. Matching a service request R with a service offer S is based on matching the individual parameters in the two descriptions. For this purpose, a semantic matching function f_m is used. For input and output parameters the degree of match is typically determined by checking for equivalence or subsumption relationship between the corresponding classes in the ontology $\mathcal{O}$. Similar to previously established approaches [12, 11], we consider the following degrees of match, in decreasing order: $DM = \{exact, direct\_subclass, subclass, direct\_superclass, superclass, sibling, fail\}$. Notice that the distinction between direct subclass (superclass) and subclass (superclass) refers to whether the considered subsumption relationship is explicitly stated in the ontology or inferred by the reasoner (e.g., by transitivity.) Different ordering or variations of these degrees may also be meaningful in different applications and contexts [11]. Our approach is generic and does not depend on this particular assumption. Preconditions and effects are represented by logical

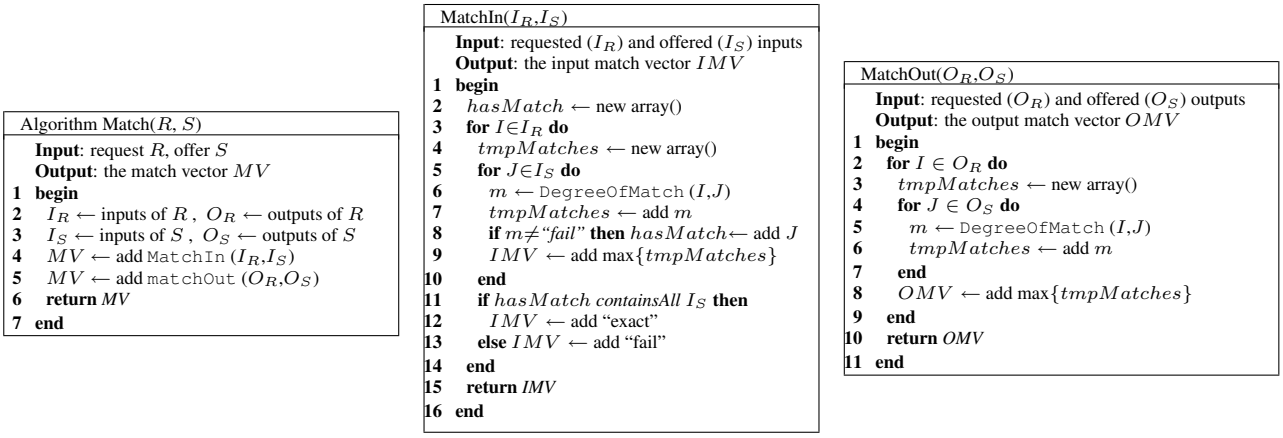


Figure 2. Matching service I/Os

formulae and are matched by checking for logical implication between them. The results of the match in this case is $exact$ or $fail$, depending on whether such an implication holds or not.

The inputs and preconditions of the request should match those of the service, while the service outputs and effects should match those of the request. Thus, applying the function f_m to pairs of corresponding parameters from the requested and offered service, results in a match vector $MV \in DM^k$, $k = |MV| = |S_I| + |S_P| + |R_O| + |R_E|$. However, the number, as well as the order, of the parameters may vary among the set of available services, rendering the match vectors not comparable. To deal with this problem, i.e., to fix the number and the order of the dimensions, we use as reference dimensions the ones specified by the user's request. Still, two issues need to be resolved in this case. First, the same request input/precondition may provide a match for more than one service inputs/preconditions. Then, the best degree of match is considered for the corresponding position in MV. Second, it is possible that not all service inputs/preconditions are matched. To capture this, we introduce two additional fields in MV, corresponding respectively to inputs and preconditions, with the values *exact* or *fail*, indicating accordingly whether there exists a parameter that has not been matched (alternatively, the number of parameters that failed to match can be used). Thus, the size of MV becomes $|MV| = |R_I| + |R_O| + |R_P| + |R_E| + 2$ (i.e., fixed for a given R). The matching algorithm, $Match(R, S)$ is presented in detail in Figure 2. The function $DegreeOfMatch(I, J)$ uses a reasoner to determine the degree of match between the ontology concepts I,J. For brevity, we only consider inputs and outputs; preconditions and effects can be matched accordingly.

We can now define the notions of (dynamic) dominance and (dynamic) skyline for Semantic Web services selection.

Definition 5 ***(Service Dominance)*** *Given a set of Semantic Web services $\mathcal{S}$ and a request R, a service $S_1 \in \mathcal{S}$ dominates another service $S_2 \in \mathcal{S}$ w.r.t. R, denoted as $S_1 \prec^R S_2$, iff $\forall i \in [1, |MV_{R,S_1}|] : MV^i_{R,S_1} \leq MV^i_{R,S_2}$ and $\exists i \in [1, |MV_{R,S_1}|] : MV^i_{R,S_1} < MV^i_{R,S_2}$.*

Definition 6 ***(Skyline Services)*** *Given a set of Semantic Web services $\mathcal{S}$ and a request R, the skyline services of $\mathcal{S}$ w.r.t. R, denoted by $SL^R_{\mathcal{S}}$, are those not dominated by another service w.r.t. R: $SL^R_{\mathcal{S}} = \{S \in \mathcal{S} | \nexists S' \in \mathcal{S} : S' \prec^R S\}$.*

2.3 Illustrative Example

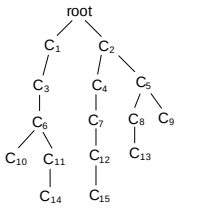
Assume a sample service request and six available services, as shown in Figure 3(a). For simplicity, we consider only input and output parameters, which are classes from the hierarchy depicted in Figure 3(b). The derived match vectors are presented in Figure 3(c), with IN_X indicating whether all service inputs are matched or not. For instance, in the case of S_4, the provided input C_6 provides a $direct\_superclass$ match with C_{10} and a $superclass$ match with C_{14}. Thus, $MV^{IN_1}_{R,S_4}$ = $direct\_superclass$ and $MV^{IN_X}_{R,S_4}$ = $exact$. For the service S_2, $MV^{IN_1}_{R,S_2} = fail$ and $MV^{IN_X}_{R,S_2}$ = $fail$, since the request input C_6 does not provide a match for the service input C_9. The rest of the results can be verified similarly.

Given the match vectors shown in Figure 3(c), the problem is to identify the best matches. Even for such a small number of services this is no trivial task. For this purpose, we consider as best matches those services that belong in the skyline for the given request. Based on the definitions in Section 2.2, we can conclude that (a) the services S_2, S_4 and S_5 are dominated by both S_1 and S_3, (b) S_6 is dominated by S_3, and (c) S_1 and S_3 are not dominated by any service. Therefore, S_1 and S_3 constitute the skyline, i.e., the best matches, for the request R.

One might argue that S_1 constitutes an "overall" better match than S_3, given that $direct\_subclass$ indicates a

	INPUTS	OUTPUTS
R	C_6	C_7, C_8
$\mathbf{S}_1$	C_3	C_5, C_7
$\mathbf{S}_2$	C_9	C_2, C_9
$\mathbf{S}_3$	C_3	C_2, C_8
$\mathbf{S}_4$	C_{10}, C_{14}	C_{15}
$\mathbf{S}_5$	C_1	C_6
$\mathbf{S}_6$	-	C_6, C_8, C_9

(a) Service request and offers



(b) A sample class hierarchy

	$\mathbf{IN}_1$	$\mathbf{IN}_X$	$\mathbf{OUT}_1$	$\mathbf{OUT}_2$
$\mathbf{S}_1$	dir_subcls	$exact$	$exact$	dir_subcls
$\mathbf{S}_2$	$fail$	$fail$	$subcls$	$sibling$
$\mathbf{S}_3$	dir_subcls	$exact$	$subcls$	$exact$
$\mathbf{S}_4$	$dir\_supercls$	$exact$	$supercls$	$fail$
$\mathbf{S}_5$	$subcls$	$exact$	$fail$	$fail$
$\mathbf{S}_6$	$fail$	$exact$	$fail$	$exact$

(c) The resulting match vectors

Figure 3. Illustrative example

closer match than $subclass$. However, this would only be true for users with an equal preference on both output parameters or a higher preference on OUT_1. Instead, a user concerned about parameter OUT_2 would probably be more interested in the service S_3. Selecting the skyline services guarantees the retrieval of the best matches regardless of user preferences.

3 Selecting the Best Candidates

3.1 Main algorithm

We leverage work existing in the database literature, and in particular the Bitmap algorithm, introduced in [15]. Since the semantic match between requested and offered services is typically expressed by a small set of discrete degrees of match, as discussed in the previous section, the choice of the Bitmap algorithm is natural, as it is especially designed for discrete, low cardinality domains. Specifically, it employs a bitmap representation to encode the data points, and uses bit-wise operations to determine the skyline. The efficiency of the algorithm relies on the high speed of bit-wise operations. Note that, even though more efficient skyline algorithms have been proposed ([13]), they rely on the assumption that the data set is indexed.

The skyline service selection algorithm works as follows. First, the match vectors are translated to an appropriate bitmap representation. In fact, to avoid any extra overhead, this step can be integrated with the matching phase, i.e., the result of the matcher can be directly encoded in this representation. Then, each match vector is checked for dominance against all other match vectors. The latter step is efficiently performed by fast bit-wise AND/OR operations on the bitmap representations obtained in the former step.

Obtaining the bitmap representation. We assume the dominance relationship described in Section 2.2 and assign the values $\{1, 2, \ldots, 7\}$ to the 7 possible degrees of match,

	$\mathbf{IN}_1$	$\mathbf{IN}_X$	$\mathbf{OUT}_1$	$\mathbf{OUT}_2$
$\mathbf{S}_1$	01***1***1111	**1**111111	111***1***111	0111***1***1
$\mathbf{S}_2$	00***0***0001	**0**000001	001***1***111	00000***1***1
$\mathbf{S}_3$	01***1***1111	**1**111111	001***1***111	1111***1***1
$\mathbf{S}_4$	00***0***1111	**1**111111	000***0***111	00000***0***1
$\mathbf{S}_5$	00***1***1111	**1**111111	000***0***001	00000***0***1
$\mathbf{S}_6$	00***0***0001	**1**111111	000***0***001	1111***1***1

Figure 4. Bitmap representation

$$A^1 = 101110 \quad B^1 = 101010$$
$$A^2 = 101111 \quad B^2 = 000000$$
$$A^3 = 111100 \quad B^3 = 111000$$
$$A^4 = 111111 \quad B^4 = 111001$$
$$A = 101100 \quad B = 111011$$

$$A\&B = 101000$$

Figure 5. Dominance check for S_4

with 1 corresponding to $exact$ and 7 to $fail$ [1]. We represent these values in a bitmap of size 7, as follows: if $q \in \{1, 2, \ldots, 7\}$ is the degree of match, then its bitmap representation has value 0 for the bits 1 to $q-1$, and 1 for the bits q to 7. For example, an $exact$ degree of match, i.e., value 1, is represented as 1111111, whereas a $sibling$ degree of match, i.e., value 6, is represent as 0000011. Returning to our running example, the corresponding bitmap representations are depicted in Figure 4 (the function of the bold and italicized bits will be discussed in the following).

Checking for dominance. Determining whether a service belongs to the skyline involves extracting vertical *bitslices* and performing bitwise AND/OR operations. This process is best illustrated through our running example. Assume we wish to discover whether service S_4 with match vector $MV_{S_4} = (4, 1, 5, 7)$ is part of the skyline. For each field $i \in [1, |MV_{S_4}|]$ of MV_{S_4}, two vertical bitslices, A^i and B^i, are extracted. In particular, letting $q = MV^i_{S_4}$, we obtain the bitslice A^i (resp., B^i) by juxtaposing the q-th (resp., the preceding $(q-1)$-th) bit of the i-th field for all services. Note that when $q - 1 < 1$, B^i is explicitly set to all zeros. Since $MV^1_{S_4} = 4$ the bitslice $A^1 = 101110$ is obtained by juxtaposing the 4th bits of the first field for all services. Similarly, $B^1 = 101010$ is obtained by juxtaposing the 3rd bits. Figure 4 shows the A^i bitslices in bold typeface; the B^i bitslices are shown italicized ($B^2 = 000000$ is omitted).

Assume a service request R and an offer S. Observe that the bitslice A^i of S encodes which services (i.e., those whose bit is set) are equally as good or better matches than S w.r.t. the i-th field of the match vector. On the other hand, the bitslice B^i of S encodes the services that are strictly better matches for the i-th field. Let $A = A^1 \& A^2 \& \ldots \& A^{|MV_S|}$, where $\&$ represents the bitwise AND operation. Then, A indicates the services that

[1] The adaptation to different degrees of match and dominance relationships is straightforward.

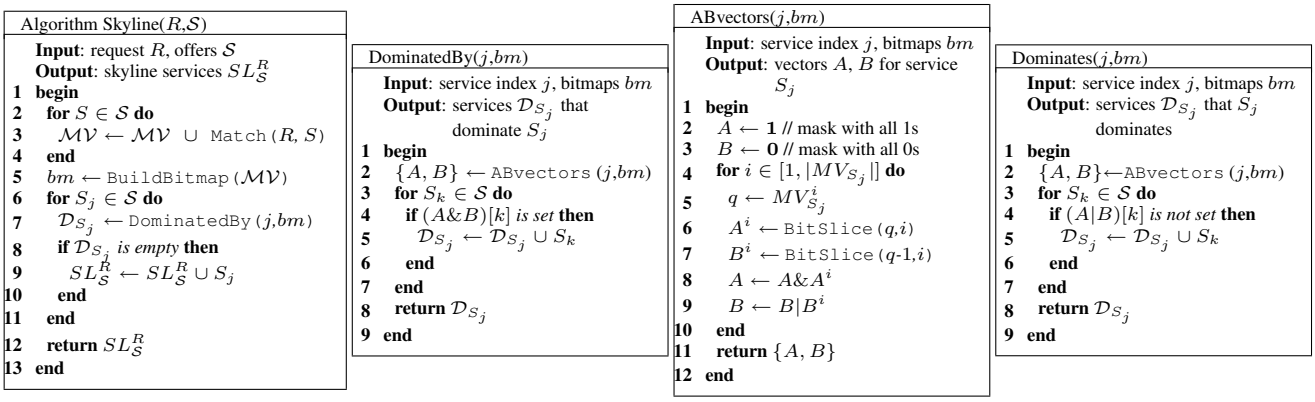


Figure 6. Determining the skyline services

are equally as good or better in *all* fields of the match vector. Similarly, let $B = B^1|B^2|\ldots|B^{|MV_S|}$, where $|$ represents the bitwise OR operation. Then, B indicates the services that are strictly better in *at least one* field of the match vector. According to Definition 5, if a service has its bit set both in A and B, then it dominates S, and, hence, the latter is not in the skyline. On the other hand, if $A\&B$ has no bit set, then S is not dominated by any other service, and thus belongs to the skyline. Figure 5 illustrates the dominance check for S_4, which is dominated by S_1 and S_3. The algorithm is presented in detail in Figure 6.

Next, we extend the algorithm to provide key aspects of functionality desirable by service requesters and providers.

3.2 Requester's perspective

Three key elements of functionality for service requesters are (a) ranking, (b) redefinition, and (c) relaxation. We discuss each aspect in detail.

Ranking. The selected skyline services are determined regardless of specific user preferences; hence, are not ranked. However, in many cases, e.g., when the number of returned results is large, ranking is required. To this end, we present a ranking function that is user preference agnostic and is well aligned with the dominance notion. Intuitively, services that dominate a large number of other services are potentially more interesting and should be examined first.

Definition 7 ***(Dominance Set and Score)*** *Given a set of Semantic Web services $\mathcal{S}$, a request R, and a service $S \in \mathcal{S}$, the dominance set of S comprises those services dominated by S, i.e., $\mathcal{D}_S = \{S_i \in \mathcal{S} | S \prec^R S_i\}$. The dominance score of S is the cardinality of $\mathcal{D}_S$, i.e., $ds_S = |\mathcal{D}_S|$.*

The skyline services are ranked based on their dominance score. To calculate this score we utilize the A and B bitmaps for service S. Observe that $\neg A$, where $\neg$ denotes negation, indicates the services that are strictly worse than S in *at least one* field of the match vector. Similarly, $\neg B$ indicates those services being worse or equal to S in *all* fields of the match vector. It is easy to show that if a service has its bit set both in $\neg A$ and $\neg B$, then it is dominated by S. Hence, calculating the dominance score of S resolves to counting the bits set in $(\neg A)\&(\neg B)$.

Redefinition. Suppose that the user would like to redefine his/her request in terms of removing or adding request parameters, either because he/she is not satisfied by the matchmaking, or due to exploratory behaviour. The proposed methodology handles such a scenario efficiently, requiring minimum invocation of the matcher and the fewest changes to the bitmap representation of the match vectors. We distinguish 4 cases and examine the necessary changes to the services selection process.

Adding input parameter. We need to run the match algorithm for the new parameter. Note also that the IN_X field of the match vector might be affected by the matching, and thus, it needs to be re-computed (if the previous value was $fail$). Then, we need to build the bitmap representation for the field corresponding to the new parameter, to update the representation for the IN_X field, if changed, and to execute the bitmap algorithm.

Deleting input parameter. Only the IN_X field of the match vector may be affected by the matching (if it was previously set to $exact$). Therefore, we need to rebuild its bitmap representation. Since the deleted parameter might be needed in a future request, we do not delete the representation corresponding to its match vector field; rather, we modify the bitmap algorithm to skip that field in the calculation of the A, B bitmaps.

Adding output parameter. We need to run the match algorithm for the added output parameter. Then, the bitmap representation for the new parameter must be built and the bitmap algorithm must be executed.

Deleting output parameter. In this case, the match algorithm need not run. We choose to preserve the bitmap representation for the corresponding field and modify the bitmap algorithm to skip that field in the A, B calculation.

Relaxation. Consider the case that the user would like to relax the dominance requirement and retrieve additional relevant services besides those included in the skyline. Such a functionality would prove useful when there are a few very dominant services that *hide* some other potentially interesting offers. For this purpose, we provide the user with the option to examine the next most dominant services, i.e., the next *skylayer*.

Definition 8 *(**l-Skylayer Services**) Given a set of Semantic Web services $\mathcal{S}$ and a request R, the l-skylayer services of $\mathcal{S}$ w.r.t. R, denoted by $SL^R_{\mathcal{S}}(l)$, is defined recursively as follows: $SL^R_{\mathcal{S}}[1,l] = \bigcup_{0<k\leq l} SL^R_{\mathcal{S}}(k)$, where $SL^R_{\mathcal{S}}(1)$ is the skyline services $SL^R_{\mathcal{S}}$ and $SL^R_{\mathcal{S}}(l) = SL^R_{\mathcal{S}\setminus SL^R_{\mathcal{S}}[1,l]}$.*

Finding the l-skylayer services can be performed by some tweaking of the bitmap algorithm, without invoking the matcher. Assume that the (l-1)-skylayer has been found. We maintain a bitmap mask C that indicates which services belong to one of the previous skylayers, i.e., in $SL^R_{\mathcal{S}}[1,l]$. In the calculation of the A bitmaps for the l-skylayer we need to mask it (i.e., perform bitwise AND operation) with the negation of C, so as to suppress services previously found. Finally, the bitmap mask C is updated by setting the bits of the l-skylayer services.

3.3 Provider's perspective

Existing works on service discovery focus on locating one or more services that are appropriate for fulfilling the client's request. In the remaining of this section, we turn our attention towards the provider's view of the service selection process. From this perspective, a provider would be interested in analyzing the position of his/her services in the market and their potential to attract clients. We consider two scenarios that might be of interest for a service provider.

Service competitiveness. In this scenario, the provider is interested in evaluating how competitive his/her provided service S is w.r.t. a request R and a set of other available services $\mathcal{S}$. This can be accomplished by means of two measures: (a) the number of services dominated by S w.r.t. R; (b) the number of services dominating S w.r.t. R. The first is the dominance score of S (see Definition 7) and is calculated by the function *Dominates*, shown in Figure 6. The second is provided similarly, through the function *DominatedBy*, also shown in Figure 6.

Service adaptation. In this scenario, the provider would like to appropriately modify the offered service S in order to target specific user requests, i.e., so that the service would be in the skyline for a considered request R. To keep the required modifications to a minimum, we consider the case where only one parameter is subject to change, and our goal is to determine the parameter for which the required change is minimized. For this purpose, we calculate the services that dominate S and are part of the skyline for the request R. Then, we compare the values in all the dimensions of the selected match vectors, to find the maximum differences in each dimension. The dimension having the minimum among these differences is selected. The intuition lies in the fact that a service is included in the skyline, when it becomes better than all its competitors in at least one dimension. This process is formally described by the algorithm in Figure 7. As an example, consider the service shown in black in the same figure. The shaded area contains the services that dominate it, including two in the skyline. The arrows represent the maximum differences for each dimension; clearly, the dimension of the shortest arrow corresponds to the minimal change required.

Algorithm ModDim(R,S_j,$\mathcal{S}$)
Input: request R, service S_j, competing services $\mathcal{S}$
Output: the dimension d to modify
begin
  $SL^R_{\mathcal{S}} \leftarrow$ `Skyline` (R,$\mathcal{S}$)
  $\mathcal{D}_{S_j} \leftarrow$ `DominatedBy` (j,bm)
  $SLD \leftarrow SL^R_{\mathcal{S}} \cap \mathcal{D}_{S_j}$
  $d \leftarrow i \in [1, |MV_{S_j}|]$ s.t. $\max_{S \in SLD} |MV^i_{S_j} - MV^i_S|$ is minimized
  return d
end

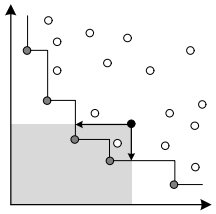


Figure 7. Modifying service parameter

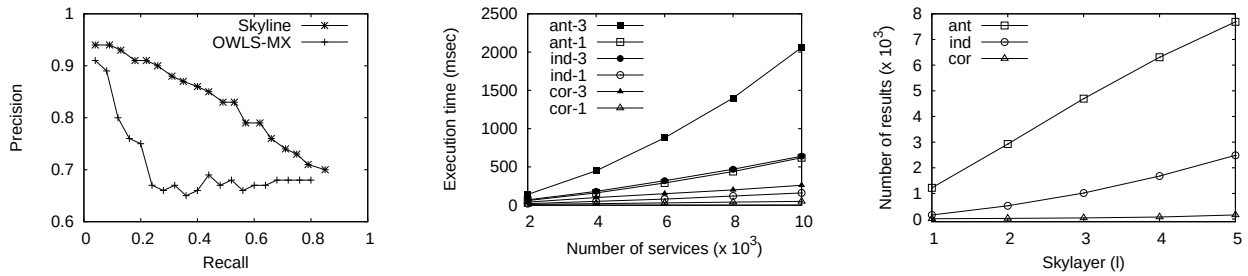
4 Experimental Evaluation

In this section, we present a comprehensive study evaluating the effectiveness and the efficiency of our skyline-based approach, termed `Skyline`, for selecting the best Semantic Web services w.r.t. a desirable service description, using both real and synthetic data.

Retrieval Effectiveness. To simulate a real-world scenario, we use the OWL-S service retrieval test collection OWLS-TC v2[2]. This collection contains services retrieved mainly from public IBM UDDI registries, and semi-automatically transformed from WSDL to OWL-S. More specifically, it comprises: (a) a set of ontologies, derived from 7 different domains (education, medical care, food, travel, communication, economy and weapons), used to semantically annotate the service parameters, (b) a set of 576 OWL-S services, (c) a set of 28 sample requests, and (d) the relevance set for each request (manually identified).

To better gauge the performance of our approach we consider the matchmaking algorithm of [9], termed `OWLS-MX`. `OWLS-MX` is a hybrid matchmaker, which, apart from logic-based match, supports different IR similarity metrics for

[2] http://projects.semwebcentral.org/projects/owls-tc/



(a) Recall-precision curve (b) Execution time (c) Number of retrieved services

Figure 8. Experimental evaluation on real and synthetic data

content-based retrieval (e.g., cosine similarity or extended Jacquard similarity). Since the focus of this paper is on semantic matching, we only consider the logic-based filters while running OWLS-MX. It is important to note, however, that our approach is generic in that it can straightforwardly consider other types of filters, simply by terms of increasing the size of the match vectors to accommodate for the additional parameters; hence, the process of selecting the best matches does not change.

For each request, we calculate the match vectors and apply `Skyline` to select the best candidates. Since the number of services in the skyline may vary, successive skylayers are computed until an adequate number of services has been retrieved (see the *relaxation* case in Section 3.2). Thus, to each obtained service S that belongs to the l-skylayer and has dominance score ds_S, we assign the tuple $\langle l, ds_S \rangle$. As discussed in Section 3, we consider better matches the services that belong to the lowest l skylayer, and among those that belong to the same skylayer we consider better matches the ones with higher dominance score ds_S, i.e., we rank the obtained services by l, solving ties using ds_S.

Similarly, for each request the logic-based filters of `OWLS-MX` are applied to all services. `OWLS-MX` assigns to each service a score based on the worst degree of match among all parameters. Finally, the services are ranked according to their score, i.e., the best match is a service that has $exact$ match on all parameters.

We apply well-established IR metrics to measure the performance of the two methods, w.r.t. the corresponding relevance sets [1]. In particular, Figure 8(a) depicts the micro-averaged recall-precision curves for all the queries in the test collection. It is clear that `Skyline` outperforms `OWLS-MX` in terms of precision at all recall levels, as well as achieving a higher final recall. The results for the measures (a) Mean Average Precision (MAP), where average precision refers to the average of the precision after each relevant service retrieved, and (b) precision at N are detailed below:

Method	MAP	P@1	P@2	P@3	P@5	P@10
Skyline	0.83	0.94	0.93	0.91	0.87	0.76
OWLS-MX	0.71	0.91	0.79	0.75	0.67	0.67

These measures emphasize on returning relevant results earlier, which is important as users often tend to examine only the first few results retrieved. In particular, P@1 is especially important, as it determines the success in fully automated service discovery scenarios, where no human user is involved in the process, and thus the top-1 result is selected. Again, `Skyline` outperforms `OWLS-MX` in all cases.

Synthetic data. We measure the performance overhead associated with our approach for computing the skyline services. The algorithm was implemented in Java and the experiments were conducted on a Pentium D 2.4GHz with 2GB of RAM, running Linux. The reported measurements refer only to the process of computing the skyline, and do not include the time to perform the logic-based match for each parameter. The later depends on factors which are outside the scope of this paper, e.g., the size and type of ontologies used or the performance of the employed reasoner.

We construct match vectors of 6 parameters and assign to each degrees of match under three distributions: in `independent` (`ind`), degrees of match are assigned independently to each parameter; in `correlated` (`cor`), the values in the match vector are positively correlated, i.e., a good match in some service parameters increases the possibility of a good match in the others; in `anti-correlated` (`ant`) the values are negatively correlated, i.e., good matches (or bad matches) in *all* parameters are less likely to occur.

Figure 8(b) illustrates the running time, in milliseconds, for determining the services that belong to the l-skylayer, for $l = 1$ (i.e., the skyline), and $l = 3$, for the three types of distributions, while varying the number of services from 2K up to 10K. Observe that the time required is higher (lower) for anti-correlated (correlated) data, as the number of skyline services in this case is also higher (lower). Still, it does not exceed roughly 0.5 seconds for all cases, except that of 3 skylayers of anti-correlated data, where it takes roughly 2 seconds. Notice, however, that since for the anti-correlated case the number of services contained in the first skylayer is already quite large, computing additional layers is normally not required.

Figure 8(c) illustrates, for each distribution, the number of services retrieved by the first l-skylayers, for $l = 1$ to 5, and for an initial set of 10K services. As shown, the correlation of the degrees of match directly affects the number of selected services for each layer. For instance, the skyline comprises 16, 162, and 1221 services for the correlated, independant, and anti-correlated case respectively. These results prove the necessity of the extensions proposed in Section 3.2 for *ranking* and *relaxation*.

5 Related Work

Service discovery is a fundamental task in service-oriented architectures, and the use of semantics is critical in pursuing a higher degree of interoperability and automation. Several works have dealt with the problem of matchmaking for Semantic Web services. The basic idea, as presented in [12, 11], is to determine the degree of match as a result of logic inference between concepts contained in the service request and offer. Matching of OWL-S services is performed in [6], by a similarity measure for OWL objects, based on the ratio of common RDF triples in their descriptions. The algorithm in [5] assesses the similarity between requested and offered I/Os, by means of the proportion of shared properties between the corresponding concepts in the ontology. In [14] the ranking of Semantic Web services is determined by a semantic similarity measure defined on the associated domain ontology. The matchmaking algorithm in [3] matches sets of requested and offered parameters, based on the concept of matching bipartite graphs. Existing implemented systems include OWLS-MX [9], a hybrid matchmaker for OWL-S services, and WSMO-MX [7], a similar matchmaker for WSMO-oriented service descriptions. These works focus on matching pairs of parameters from the requested and offered services, while the overall match is typically calculated as a weighted average, assuming the existence of an appropriate weighting scheme.

The application of user preferences, expressed as soft constraints, in Web services selection has also been studied: (a) as a query language for expressing preferences, and an algebraic optimization of preference queries [8]; (b) as expansion of service requests with user-specific preferences, either stated explicitly or acquired implicitly (e.g., previous interactions, user profile) [2]. The approach in [10] uses utility function policies to model service configurations and associated prices and preferences, and develops an algorithm for optimal service selection. These approaches are complementary to our work, as determining the skyline services identifies the best matches w.r.t. *any* user preferences. Then, in the presence of user-specific preferences, the optimal services can be selected from this subset, instead of considering all the available services, thus reducing the search time.

6 Conclusions

In this paper, we have formulated the problem of Semantic Web services selection, given a desirable service description, using the notion of skyline query. We have shown how the best matches can be identified efficiently by a skyline computation algorithm, and we have addressed common tasks involved in the service selection process, referring both to the requesters' and the providers' perspectives. Experimental evaluation on real and synthetic data shows that the best matches can be identified very efficiently, with a significant increase in recall and precision.

On-going work is focused on the improvement of our prototype w.r.t. the service selection process, by facilitating the user in expressing and refining his/her queries and providing faceted browsing capabilities.

References

[1] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[2] W.-T. Balke and M. Wagner. Towards personalized selection of web services. In *WWW (Alternate Paper Tracks)*, 2003.

[3] U. Bellur and R. Kulkarni. Improved matchmaking algorithm for semantic web services based on bipartite graph matching. In *ICWS*, pages 86–93, 2007.

[4] S. Börzsönyi, D. Kossmann, and K. Stocker. The Skyline Operator. In *ICDE*, pages 421–430, 2001.

[5] J. Cardoso. Discovering Semantic Web Services with and without a Common Ontology Commitment. In *Services Computing Workshops*, pages 183–190, 2006.

[6] J. Hau, W. Lee, and J. Darlington. A Semantic Similarity Measure for Semantic Web Services. In *Web Service Semantics Workshop at WWW*, 2005.

[7] F. Kaufer and M. Klusch. WSMO-MX: A logic programming based hybrid service matchmaker. In *ECOWS*, pages 161–170, 2006.

[8] W. Kießling and B. Hafenrichter. Optimizing preference queries for personalized web services. In *Communications, Internet, and Information Technology*, pages 461–466, 2002.

[9] M. Klusch, B. Fries, and K. P. Sycara. Automated semantic web service discovery with OWLS-MX. In *AAMAS*, pages 915–922, 2006.

[10] S. Lamparter, A. Ankolekar, R. Studer, and S. Grimm. Preference-based selection of highly configurable web services. In *WWW*, pages 1013–1022, 2007.

[11] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *WWW*, pages 331–339, 2003.

[12] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. In *ISWC*, pages 333–347, 2002.

[13] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An Optimal and Progressive Algorithm for Skyline Queries. In *SIGMOD*, pages 467–478, 2003.

[14] D. Skoutas, A. Simitsis, and T. K. Sellis. A ranking mechanism for semantic web service discovery. In *Services Computing Workshops*, pages 41–48, 2007.

[15] K.-L. Tan, P.-K. Eng, and B. C. Ooi. Efficient Progressive Skyline Computation. In *VLDB*, pages 301–310, 2001.