A GRAPH DECOMPOSITION APPROACH TO WEB SERVICE MATCHMAKING

S. Lagraa, H. Seba, R. Khennoufa and H. Kheddouci
Lab. GAMA, Université Claude Bernard Lyon1, Université de Lyon, IUT Lyon1, Villeurbanne Cedex, France

Keywords: Web services, Semantic web services, OWL-S, Graph matching, Graph decomposition, Graph edit distance, Graph kernel.

Abstract: Web service discovery is becoming a critical issue in several fields. The current approaches for service discovery and mainly semantic web services such as OWL-S are limited primarily to the matching of their inputs/outputs at the level of an ontology. Recent studies show that this approach does not suffice to discover web services and that the structure of web services has an important and necessary weight in the efficiency of the matching. The structure of a web service can be represented by a graph. The problem of matching semantic web services is then translated into a problem of graph matching. In this work, we propose a matching approach that first decomposes the web service graph into more simple sub-structures then, the matching process is achieved onto these substructures. The proposed algorithms achieve better time complexity than existing ones. We also propose a semantic similarity to enhance our structural similarity.

1 INTRODUCTION

With the rapid development of e-commerce over Internet, web services are becoming an essential technology for several enterprises. With the web service technology, an enterprise publishes its activities and services on the web and consequently has access to more clients. The challenge is then how to simplify service discovery to the potential clients? So, web service discovery is becoming a critical research area. The task of web service discovery implies comparing web service descriptions and select the one that corresponds to the demand. To address this problem many simple search engines that provide keyword search on web service descriptions have been proposed. This approach is insufficient mainly because keywords do not capture the underlying semantics of web services and do no suffice for accurately specifying users' information needs (Dong et al., 2004). To deal with these limitations, researchers follow two complementary directions:

1. Explore new matching methods that use other information about web service descriptions that can assist the user when selecting among similar web services.
2. Enhance the description of web services by adding more information that may be useful to the matchmaking process. In this context semantic descriptions of web services are of high importance. So web service description languages such as OWL-S (Web Ontology Language for Web Services) receive much interests.

This paper focuses on addressing the problem of web service matchmaking in the context of both approaches. The contribution of this work can be summarized as follows:

1. We propose two algorithms for web service matching. The key part of our algorithms is a decomposition approach that simplifies the complexity of the matching process while using both structural and semantic information in the matching process.
2. Based on graph kernels to extract similar entities, we propose a new semantic metric for the comparison of web services' inputs/ouputs.

We have implemented a prototype and conducted several experiments to evaluate the effectiveness and efficiency of our matching approach. Our results show that our solution outperforms existing ones.
The rest of the paper is organized as follows: in Section 2, we begin with a discussion of the current state of the art in web service matching. Following, in Section 3, we first present a general description of our scheme and its theoretical foundations. Then, we pro-

Lagraa S., Seba H., Khennoufa R. and Kheddouci H..
A GRAPH DECOMPOSITION APPROACH TO WEB SERVICE MATCHMAKING.
DOI: 10.5220/0003338200310040
In *Proceedings of the 7th International Conference on Web Information Systems and Technologies* (WEBIST-2011), pages 31-40
ISBN: 978-989-8425-51-5

vide the details of our similarity measures. Finally, we present the results of our experiments that evaluate our approach and compares it with existing solutions. Section 4 concludes this paper and presents our future work.

2 RELATED WORK

Web service matching is an active research area were several solutions are proposed. We focus here on semantic web service matchmaking where concepts of semantic has been introduced to overcome the limitations of the simple string matching provided by UDDI (Universal Description Discovery and Integration http://uddi.xml.org/) and WSDL (Web Services Description Language http://www.w3.org/TR/wsdl). This has been achieved by the means of description languages such as OWL-S (Web Ontology Language for Web Services, http://www.w3.org/Submission/OWL-S/). OWL-S (formerly, DAML-S) provides three facets to describe a web service: Service Profile, Service Model and Service Grounding that represent respectively Description, Functionality and Access Mechanism of web services (Bellur et al., 2008):

- Service Profile: describes inputs, outputs, preconditions and effects of the web service. Input and output terms of the service are expressed as concepts belonging to a set of ontologies. The use of an ontology allows to refer to a single concept from several syntactically different terms.
- Service Model: It describes a service as a process, either atomic or composite. Composite processes have a set of sub-processes associated with a control structure. The control structure will specify the order in which different sub-processes are executed. Different types of control structures are: Sequence, Split, Split+Join, Any-Order, Choice, If-Then-Else, Iterate, Repeat While and Repeat Until.
- Service Grounding: Grounding deals with the realization of services. It provides concrete details necessary to invoke the service such as message format, transfer protocol, etc.

The earlier matching approaches of OWL-S web services (Mandell and McIlraith, 2003) use only service profile in the matching process and mainly inputs and outputs parameters. In (Bellur and Vadodaria, 2008), Bellur et al. present a comprehensive review of current approaches in semantic web service matchmaking. One of the pioneers in this context is the work of Paloucci et al. (Paolucci et al., 2002) that defines four similarly degrees between concepts: Exact, Plug-in, Subsume and Fail. Let AdOp be one of the concepts of the outputs of an advertisement and QOp be one of the concepts of the outputs of a query. Exact means that QOp and AdOp are equivalent. Plug-in means that QOp is a superclass of AdOp in the ontology, so AdOp can be plugged in place of QOp. Subsumes means that AdOp is a superclass of QOp, so the advertised service may fulfill the requirements of the request. Fail means that no relation is found between the compared concepts.

In (Bellur and Kulkarni, 2007), the authors extends the algorithm of (Paolucci et al., 2002) by computing a matching in a bipartite graph composed of the concepts of the published services and those of the query. In (Bellur et al., 2008), the authors consider pre-condition matching in addition to inputs and outputs. In (Guo and Chen, 2005), the authors present a similarity measure that allows advertisements and requests to be expressed in different ontologies. In (Beck and Freitag, 2006), the authors present a method for semantic matchmaking which takes into account the preference of concepts provided by the user. (Vu et al., 2006) consider the QoS information through the matching process. For this, the author suppose the existence of an interface where users can submit their feedback on the perceived QoS of consumed services. Wand and Stroulia (Wang and Stroulia, 2003) were the first researchers that introduced the structure of web service operations during web service discovery. Their matching method is based on the hierarchical structure of the XML syntax of WSDL specifications. For this, they adapted a tree edit distance algorithm (Garofalakis and Kumar, 2003) to the matching of WSDL specifications. In (Dong et al., 2004), Dong et al. built Woogle, a web service search engine that uses a clustering algorithm to group names of parameters of web-service operations into semantically meaningful concepts. Then these concepts are used to measure similarity of web-service operations. In (Shen and Su, 2005), Shen and Su formally define a behavior model for web service by automata and logic formalisms. In (Dijkman et al., 2009), the authors use graphs to represent web service operations. Then, they use graph matching algorithms proposed in (Messmer and Bunke, 1999) to compare graphs. However, these algorithms are space search based and consequently have an exponentiel time complexity. In (Corrales et al., 2008), authors also use graphs to match semantic web services. To match web service graphs, they use an A* based algorithm proposed in (Messmer, 1995). A* is an algorithm that uses a depth first search approach to traverse a tree representing all the possible matching

situations of the compared graphs. Hao et al. (Hao and Zhang, 2007) discover similar web services by matching the XML trees corresponding to data types in WSDL. In (Mbareck and Tata, 2006), authors use Petri nets to represent and compare web service operations. In (Nejati et al., 2007) and (Wombacher, 2006), authors use finite state machine to represent web service operations.

3 A DECOMPOSITION APPROACH TO WEB SERVICE MATCHMAKING

3.1 Overview

To address the challenges involved in searching for web services more information must be considered. Among these information, the structure of web service operations is perhaps the most important. With OWL-S, this structure can be represented by a graph. A graph of an OWL-S process is a labeled directed graph $G = (V_A, V_C, E)$ where:

- V_A is the set of vertices that represent the atomic activities of the service.
- V_C est the set of vertices that correspond to control structures such as choice, sequence, if-then-else, etc. These structures can be represented by logic connectors namely XOR et AND. For instance, a choice is represented by an XOR.
- E is the set of edges that bind the different vertices.

Furthermore, a process graph can have a vertex "start" and one or several vertices "end" (Mendling et al., 2006). Figure 1 gives un example of an OWL-S process and its corresponding graph.

In the case of non cyclic operations, an OWL-S process can also be represented by a tree where the root and the internal nodes are the different control structures (choice, sequence, if-then-else,..., etc.) and the leafs are the atomic services.

When web service operations are represented by graphs, the problem of their matching is a problem of comparing the corresponding graphs. The problem of graph comparison received a lot of interest (Bunke, 2000). The main traditional algorithmic approaches to graph comparison are graph isomorphism and inexact matching (Borgwardt and Kriegel, 2005). Graph isomorphism checks for topological identity. Inexact matching or error-tolerant matching does not enforce strict matching of graphs. Inexact matching algorithms generally computes a distance between the compared graphs. This distance measures how much these graphs are similar and is associated to a cost function. Graph edit distance is the most used inexact matching measure. Similarity or edit distance between two graphs is the minimum number of edit operations needed to transform one graph into the other (Bunke and Allermann, 1983; Sanfeliu and Fu, 1983). An edit operation is an insertion, a suppression or a relabeling of a vertex or an edge in a graph. To each edit operation e_i is associated a cost $c(e_i)$ and the edit distance between two graphs G_1 and G_2 is then the minimum cost related to the set of operations that transform G_1 into G_2.

$$d(G_1, G_2) = \min_{(e_1,\ldots,e_k)\in\Upsilon(G_1,G_2)} \sum_{i=1}^{k} c(e_i) \qquad (1)$$

where $\Upsilon(G_1, G_2)$ is the set of edit operations that transform G_1 into G_2. However, finding the minimal edit distance is NP-hard (Bunke, 1999) and it is often difficult to find the appropriate costs for individual edit operations (Borgwardt and Kriegel, 2005). So, searching for other graph matching approaches is still an open question. Graph kernels is one of the most recent approaches to graph comparison that is presented as an alternative to existing solutions. Graph kernels have the particularity of integrating concepts from all former approaches to graph comparison. Graph kernels belong to a class of kernels on structured data called *R*-convolution kernels or decomposition kernels (Haussler, 1999). Roughly speaking, the idea of graph kernels is constructing similarity of graphs based on the similarity of small parts, i.e. sub-graphs, of the compared graphs. The concept of decomposing a discrete compound object x into parts $(x_1, x_2, ..., x_d)$ can be formally modeled by a predicate R that returns true if $(x_1, \ldots, x_d)$ is a valid decomposition of x. The set of all valid decompositions of x is noted $R^{-1}(x) = \{(x_1, \ldots, x_d) : R(x_1, \ldots, x_d)\}$. The kernel function k between two objects x and x' is given by (Haussler, 1999):

$$k(x, x') = \sum_{\substack{(x_1,\ldots,x_d)\in R^{-1}(x) \\ (x'_1,\ldots,x'_d)\in R^{-1}(x')}} \prod_{i=1}^{d} k_i(x_i, x'_i) \qquad (2)$$

where k_i is a kernel that gives the similarity between two parts.

For graphs, the natural and most general *R*-convolution kernel would decompose each of the two compared graphs G and G' into all of their subgraphs and compare them pairwise. This all-subgraphs kernel is defined as follows (Borgwardt and Kriegel, 2005):

$$k(G, G') = \sum_{\substack{S\subseteq G \\ S'\subseteq G'}} k_i(S, S') \qquad (3)$$

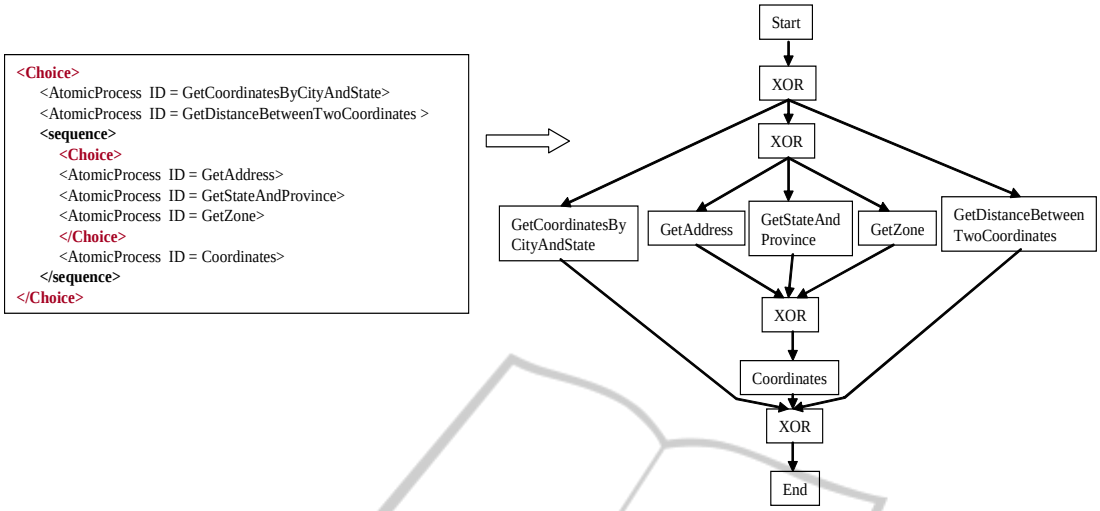

Figure 1: A graph representation of an OWL-S process.

In (Gartner et al., 2003) the authors show that the problem of computing this all-subgraphs kernel is NP-hard. So, several other alternatives have been defined in the literature. These alternatives focus on proposing simpler decompositions such as sequences of vertices (Neuhaus and Bunke, 2006), paths (Suard and Rakotomamonjy, 2007), shortest paths (Borgwardt and Kriegel, 2005), trees (Ramon and Gartner, 2003),etc. The idea of decomposing the graphs to be matched prior to the matching process has also been used with graph edit distance and the results are promising. In (Jouili and Tabbone, 2009; Riesen and Bunke, 2009) the authors use a star decomposition of the graphs and then define graph edit distance onto the obtained sub-structures.

3.2 Our Approach

It follows from the above description of graph matching approaches that graph decomposition allows to have more efficient solutions either via graph kernels or graph edit distance. The question here is what is the decomposition to use when matching web services? To attempt to give a satisfactory answer, we propose here two kinds of decomposition of web service graphs through two matching algorithms: Algorithm 1 and Algorithm 2. Both of them focus on the main characteristic of the graphs of web service business processes: directed edges. Algorithm 1 uses graph kernels and Algorithm 2 uses graph edit distance. We also attempt to reduce the matching delay by avoiding redundancy in the obtained sub-structures.

3.3 Algorithm 1

In this algorithm, we extend an existing graph decomposition, the star decomposition, to take into account edge-direction. We call this star, where all edges are out-coming edges, a *FaS* for Father and Sons substructure. A *FaS* substructure is composed of a node connected to its sons by out-coming edges. This decomposition is adapted for web services which graphs do not contain cycles. So we apply it for trees of web service operations. Figure 2 shows an example of a decomposition of an OWL-S process tree into *FaS* substructures.

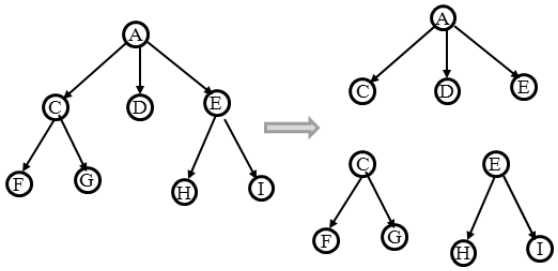

Figure 2: Decomposition of a service tree into *FaS*s.

3.3.1 Principle

To compare two OWL-S processes, we first decompose their corresponding trees into *FaS*s. Then, each *FaS* of the first process is compared to every *FaS* of the second process (see Algorithm 1). Comparing a pair of *FaS*s FaS_i and FaS_j consists to compute the kernel function

$$k(FaS_i, FaS_j) = exp^{-d(FaS_i, FaS_j)} \quad (4)$$

which allows us to find for a given FaS_i of the first tree the most similar FaS_j of the second tree given by the best value of $k(FaS_i, FaS_j)$ noted $Sim(i, j)$. Then, we compute the kernel of the two compared trees by summing all these best values :

$$K(A_1, A_2) = \sum_{i=1,j=1}^{nA_1,nA_2} Sim(i, j) \quad (5)$$

Algorithm 1 : Matching algorithm using a decomposition into *FaS*s and tree kernels.

Input:Two OWL-S process trees A_1 and A_2
Output: similarity degree between A_1 and A_2 computed as a tree kernel
Begin
Decompose A_1 and A_2 into *FaS*s.
For $i = 1, nA_1$ do For $j = 1, nA_2$ do $Sim(i, j) = 0$; EndFor EndFor;
For $i = 1, nA_1$ do // nA_1 is the number of *FaS*s in A_1
For $j = 1, nA_2$ // $nA_2$is the is the number of *FaS*s in A_2.
do /* compare FaS_i and FaS_j */

$$d_{father}(FaS_i, FaS_j) = \begin{cases} 0 & if\ father_{FaS_i} = father_{FaS_j} \\ 1 & otherwise \end{cases}$$

$d_{son}(FaS_i, FaS_j)) =$
$max(\|FaS_i\|, \|FaS_j\|) - \|FaS_i \cap FaS_j\|$
$d_{edge}(FaS_i, FaS_j) = |\,\|edge(FaS_i)\| - \|edge(FaS_j)\|\,|$
$d(FaS_i, FaS_j) = d_{father}(FaS_i, FaS_j) + d_{son}(FaS_i, FaS_j)) + d_{edge}(FaS_i, FaS_j)$ $k(FaS_i, FaS_j) = exp^{-d(FaS_i, FaS_j)}$
If $k(FaS_i, FaS_j) > Sim(i, j)$ Then
$Sim(i, j) = k(FaS_i, FaS_j)$; EndIF
EndFor EndFor
$K(A_1, A_2) = \sum_{i=1,j=1}^{nA_1,nA_2} Sim(i, j)$ //tree kernel of A_1 and A_2.
END

3.3.2 Example

Figure 3 illustrates a matching example of two web service trees A_1 and A_2 with Algorithm 1. The decomposition gives 2 *FaS* substructures for each tree. The best value of the kernel function for $FaS_1(A_1)$ is the one that matches it with $FaS_1(A_2)$ and it is equal to e^{-2}. The best value of the kernel function for $FaS_2(A_1)$ is the one that matches it with $FaS_2(A_2)$ and it is also equal to e^{-2}. So, the final value of the similarity between the two trees is $e^{-2} + e^{-2} = 2e^{-2}$.

3.3.3 Time Complexity

The algorithm has a complexity of $O(\triangle n^3)$ time steps where n is the number of vertices in the largest tree and $\triangle$ is the maximum degree in the tree. In fact the complexity of the decomposition into *FaS* substructures is polynomial. In the worst case it is $O(\triangle n)$. To match the obtained sets of *FaS*s, we need $O(\triangle n^3)$ time steps. This gives a global complexity of $O(\triangle n^3 + \triangle n = \triangle n^3)$ time steps. As in the worst case $\triangle = n$, we have a complexity of $O(n^4)$.

3.4 Algorithm 2

In this algorithm, we propose a new graph decomposition that emphasizes edges and their direction. This decomposition, called EaS for (***E***dge and its endpoint ***S***tars) is a graph sub-structure composed of a directed edge, its endpoint vertices and all the outcoming edges from the two endpoints. Figure 4 shows an example of decomposition of a graph into EaSs.

3.4.1 Principle

To compare two OWL-S processes, we first decompose their corresponding graphs into EaSs. Then each EaS of the first process is compared to every EaS of the second process (see Algorithm 2). Comparing a pair of EaSs: EaS_i and EaS_j consists to compute the edit distance between two sub-graphs $Sim(i, j) = d(EaS_i, EaS_j)$. Sim is a $nG_1 \times nG_2$ matrix, where nG_1(resp. nG_2) is the number of EaSs in G_1(resp. G_2), that saves the computed distances (see Figure 5). Then, the distance between the two graphs is the sum of the minimal distances between EaSs.

Algorithm 2 : Matching algorithm based on EaS. and Edit distance.

Input: Two OWL-S process graphs G_1 and G_2
Output: similarity degree between G_1 and G_2 computed as a graph edit distance
Begin
Decompose G_1 and G_2 into EaSs.
For $i = 1, nG_1$ do For $j = 1, nG_2$ do $Sim(i, j) = 0$; EndFor EndFor;
For $i = 1, nG_1$ do
For $j = 1, nG_2$ do
do /* compare EaS_i and EaS_j */
$d(EaS_i, EaS_j) = c(father_i, father_j) + c(son_i, son_j) +$
$|\,\|E(EaS_i)\| - \|E(EaS_j)\|\,|$
where $c(x, y)$ is the cost of the substituting x by y.
$Sim(i, j) = d(EaS_i, EaS_j)$
EndFor
EndFor

$$d(G_1, G_2) = \frac{\sum_{i=1,j=1}^{NG_1,NG_2} min \quad Sim(i, j)}{|nG_1 - nG_2|} + |\,\|G_1\| - \|G_2\|\,|$$

where nG_1(resp. nG_2) is the number of EaSs in G_1(resp. G_2)
End

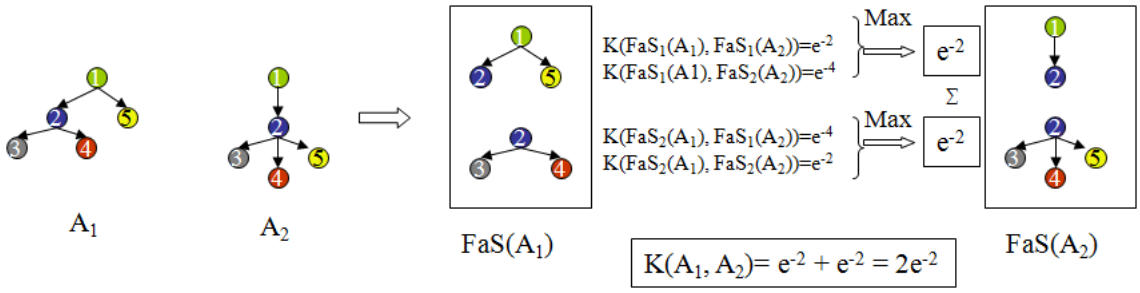

Figure 3: A matching example with Algorithm 1.

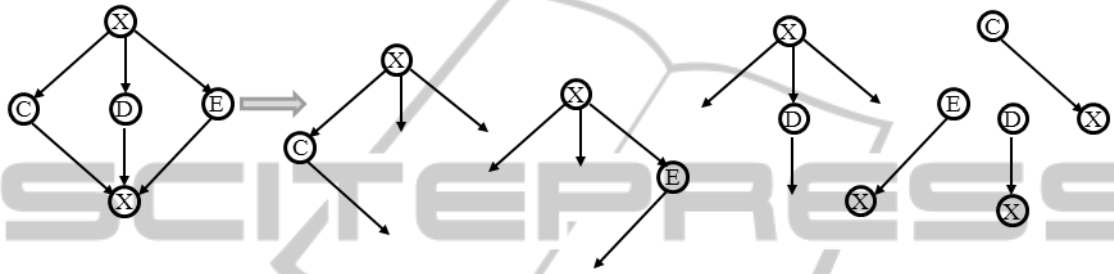

Figure 4: Example of decomposing a graph into *EaS*s.

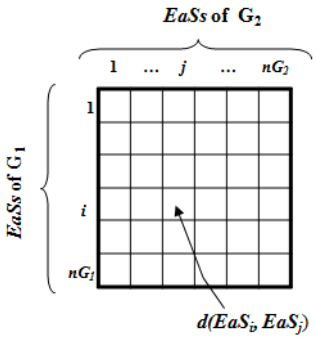

Figure 5: Similarity matrix between *EaS*s.

3.4.2 Example

Figure 6 illustrates a matching example of two graphs G_1 and G_2 with Algorithm 2. The decomposition gives 5 *EaS* substructures for G_1 and 3 *EaS* substructures for G_2. The matrix of similarity *Sim* between *EaS*s is 5×3. In Figure 6, each *EaS* is represented by the directed edge that determines it i.e. a pair of nodes $(father, son)$. Each cell of the matrix contains the edit distance between two *EaS*s. For example $Sim(1,1) = 2$ because if we consider that all edit operations have the same cost 1, then the distance between $EaS_1(G_1)$ and $EaS_1(G_2)$ is equal to the cost of two re-labelling operations: 2 to 1 and 3 to 2. By summing the minimal distance between all pairs of *EaS*s, we obtain a distance between the two graphs equal to 7.

3.4.3 Complexity

The complexity of Algorithm 2 is $O(n^4)$ where n is the number of vertices in the largest graph. In fact, to decompose a graph of n nodes into *EaSs* we need $O(n^2)$ time steps. To compare two *EaS*s, we need $2\triangle^2$ time steps where $\triangle$ is the maximum degree in the graph. So, if we consider that, in the worst case a graph of n nodes has $2n$ *EaS*s then to construct the matrix *Sim* of all comparisons between the *EaSs*, of two graphs, we need $O(4n^2 * 2\triangle^2) = O(n^2\triangle^2)$ time steps.

To compute the minimum distance between *EaS*s, we need $O((2n)^2) = O(4n^2)$ time steps.

So, the complexity of the algorithm is $O(n^2\triangle^2)$. As in the worst case $\triangle = n$, we have a complexity of $O(n^4)$.

3.5 Semantic Similarity

To obtain an accurate similarity measure, we also consider semantic similarity of inputs and outputs. To do so, we extended the semantic similarity of (Paolucci et al., 2002) with Kernels. In fact, we propose a kernel function to evaluate the four degrees of similarity *Exact*, *PlugIn*, *Subsume* and *Fail* introduced in (Paolucci et al., 2002). We affect a weight w to each of the four degrees. This weight is either given by the user or determined according to the application. So, we compare two concepts cx_1 and cx_2 that correspond to the inputs or outputs x_1 and x_2 of the compared ser-

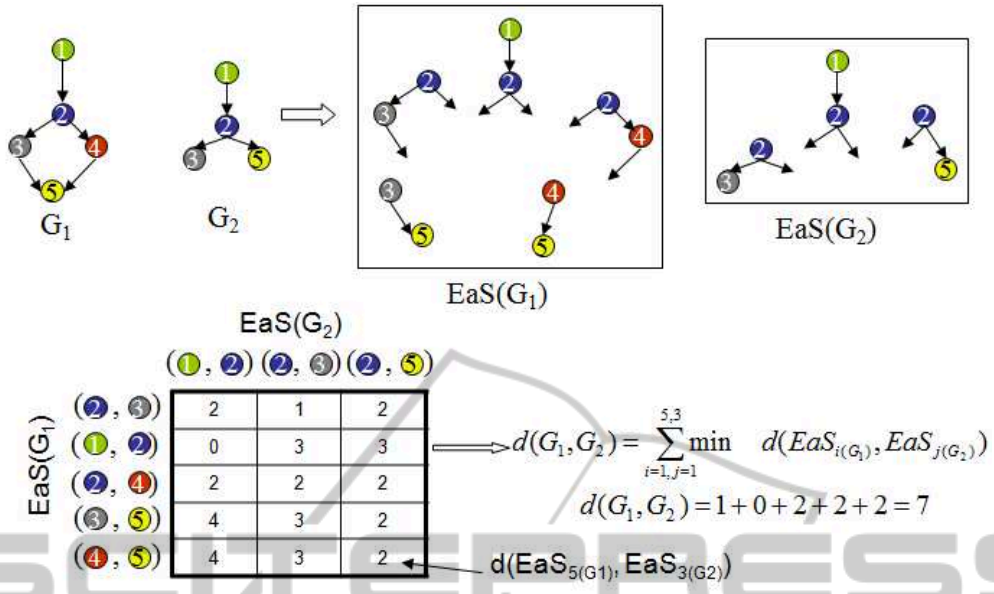

Figure 6: A matching example with Algorithm 2.

vices by the following function:

$$k_{Sem}(x_1,x_2) = \begin{cases} w_{Exact} & if\ Exact(cx_1,cx_2) \\ w_{PlugIn} * \frac{1}{d(cx_1,cx_2)} & \text{if } PlugIn(cx_1,cx_2) \\ w_{Subsumes} * \frac{1}{d(cx_1,cx_2)} & \text{if } Subsumes(cx_1,cx_2) \\ w_{Fail} & \text{otherwise} \end{cases} \quad (6)$$

where $d(cx_1,cx_2)$ is the number of edges that separate the two concepts cx_1 and cx_2 in the ontology.

Thus, our semantic similarity of two services S_1 and S_2 measures the sum of the most similar inputs and outputs of the two services as follows:

$$k(S_1,S_2) = \sum_{x_1 \in S_1, x_2 \in S_2}^{nbS_1, nbS_2} max\ k_{Sem}(x_1,x_2) \quad (7)$$

where nbS_1 (resp. nbS_2) is the number of inputs/outputs of S_1 (resp. S_2) and $k_{Sem}(x_1,x_2)$ is the similarity between two concepts.

3.5.1 Example

Let S be a published service with two input parameters: "vehicle" and "parts" and one output "price". Let Q a query with input parameters "Car" and "parts" and one output "price". We suppose that the ontology is the one presented on Figure 7 and that $w_{Exact} = 1$, $w_{Plug-In} = 0.8$, $w_{Subsumes} = 0.5$, $w_{Fail} = 0$.

The similarity between the inputs of the published service and the query is :

$$\begin{cases} k_{Sem}(car,vehicle) = w_{PlugIn} * \frac{1}{1} = 0.8 \\ k_{Sem}(car,parts) = w_{Fail} = 0 \end{cases} \overset{Max}{\Rightarrow} 0.8 \quad (8)$$

$$\begin{cases} k_{Sem}(parts,vehicle) = w_{Fail} = 0 \\ k_{Sem}(parts,parts) = w_{Exact} = 1 \end{cases} \overset{Max}{\Rightarrow} 1 \quad (9)$$

The similarity between the outputs of the two services is:

$$k_{Sem}(price,price) = w_{Exact} = 1 \overset{Max}{\Rightarrow} 1 \quad (10)$$

This gives the following similarity between the two services:

$$k(S_1,S_2) = 0.8 + 1 + 1 = 2.8 \quad (11)$$

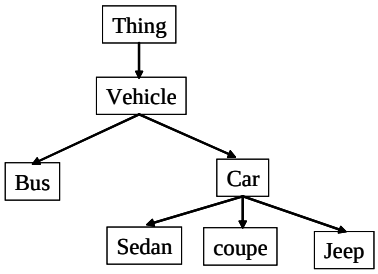

Figure 7: Vehicle ontology.

3.6 Evaluation

We implemented our matching scheme to evaluate its performance and compare it with the performance of exiting matching approaches of web services. In this section, we provide an overview of our experiments and some of the results we obtained. The experiments were conducted on a Windows XP Laptop with a 1.73 GHz Pentium IV CPU and 1Gb main memory. The data set used in our tests is a web service repository

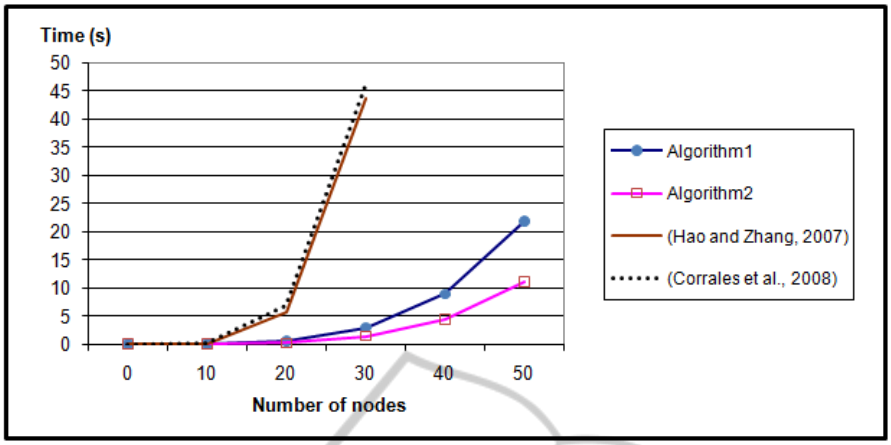

Figure 8: Execution time vs. the size of the graph.

collected by (Gater et al., 2010). We evaluated our scheme by comparing it with two other methods that embed structure matching within web service matchmaking: (Corrales et al., 2008) and (Hao and Zhang, 2007) described in Section 2.

We first evaluated the execution time performance of the four algorithms. The time performance is tested with the increase of the number of nodes of the compared graphs. Figure 8 shows the variation of execution times according to the size of target graphs. The results reported in this graphic represent the average execution times found for each given target graph size. It can be seen that the execution times of Algorithm 1 and Algorithm 2 are less than those of (Corrales et al., 2008) and (Hao and Zhang, 2007). The execution time is polynomial w.r.t the number of nodes in the target graph. Moreover, the slope of Algorithm 1 id higher than for Algorithm 2.

We then evaluated the effectiveness of our approach by computing the recall and precision ratios (Dong et al., 2004). The precision p and recall r are defined as follows:

$$p = \frac{A+B}{B} \tag{12}$$

$$r = \frac{A}{A+C} \tag{13}$$

where A stands for the number of returned relevant services, B stands for the number of returned irrelevant services, C stands for the number of missing relevant services, $A+C$ stands for the total number of relevant services, and $A+B$ stands for the total number of returned services. As shown in Figure 9, the precisions of our schemes is 93% for Algorithm 1 and 92% for Algorithm 2 outperforming the two other methods. As it can be seen in Figure 10, the recall of our schemes is 96% for Algorithm 1 and 95% for Algorithm 2 also outperforming the two other methods.

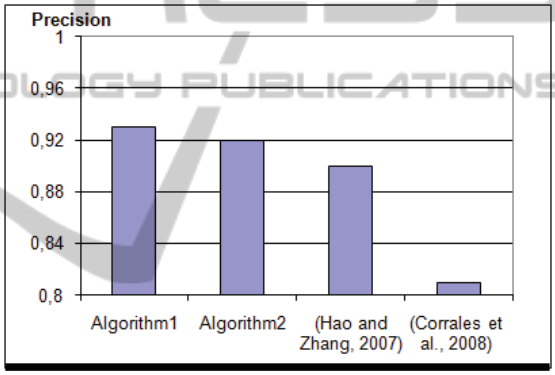

Figure 9: Precision.

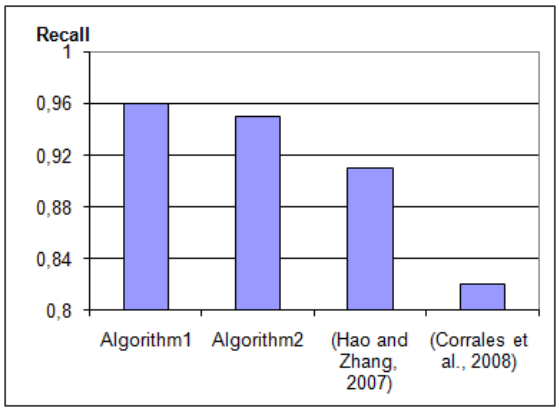

Figure 10: Recall.

4 CONCLUSIONS

In this paper, we presented two decomposition-based web service matchmaking methods. The main advantage of decomposing the web service graph into smaller sub-structures is to reduce the time complexity of the matching. The proposed algorithms take into account the main characteristic of web service graphs: directed edges and use the most efficient graph matching tools: graph kernels and graph edit distance. We also augmented our structural matching by a semantic similarity measure that enhance the matching precision. Experimental results show that the proposed algorithms are efficient and outperform existing ones. An interesting future extension consists in using this approach with large web service databases.

ACKNOWLEDGEMENTS

This work was funded by the ANR AOC and the CCI de l'Ain.

REFERENCES

Beck, M. and Freitag, B. (2006). Semantic matchmaking using ranked instance retrieval. In *SMR '06: 1st International Workshop on Semantic Matchmaking and Resource Retrieval, Co-located with VLDB*.

Bellur, U. and Kulkarni, R. (2007). Improved matchmaking algorithm for semantic web services based on bipartite graph matching. In *ICWS'07, IEEE International Conference on Web Services*.

Bellur, U. and Vadodaria, H. (2008). On extending semantic matchmaking to include precondition and effect matching. In *International Conference on Web Services, 2008, Beijing, China*.

Bellur, U., Vadodaria, H., and Gupta, A. (2008). *Semantic Matchmaking Algorithms*, chapter Greedy Algorithms. Witold Bednorz, InTech, Croatia.

Borgwardt, K. and Kriegel, H.-P. (2005). Shortest-path kernels on graphs. In *5th Int. Conference on Data Mining*, page 74–81.

Bunke, H. (1999). Error correcting graph matching: On the influence of the underlying cost function. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(9):917–922.

Bunke, H. (2000). Recent developments in graph matching. In *ICPR*, pages 2117–2124.

Bunke, H. and Allermann, G. (1983). Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1:245–253.

Corrales, J. C., Grigori, D., and Bouzeghoub, M. (2008). Behavioral matchmaking for service retrieval: Application to conversation protocols. *Inf. Syst.*, 33(7-8):681–698.

Dijkman, R., Dumas, M., and Garcia-Banuelos, L. (2009). *Business Process Management, LNCS 570*, page 48–63. Springer.

Dong, X., Halevy, A., Madhavan, J., Nemes, E., and Zhang, J. (2004). Simlarity search for web services. In *VLDB2004*, pages 372–383.

Garofalakis, M. and Kumar, A. (2003). Correlating xml data streams using tree-edit distance embeddings. In *ACM PODS'2003. San Diego, California, June 2003*, pages 143–154. ACM Press.

Gartner, T., Flach, P., and Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. In Springer, editor, *Annual Conf. Computational Learning Theory*, page 129–143.

Gater, A., Grigori, D., and Bouzeghoub, M. (2010). Owl-s process model matchmaking. In *IEEE International Conference on Web Services, July 5-10, Miami, Florida, USA*.

Guo, J. L. R. and Chen, D. (2005). Matching semantic web services across heterogenous ontologies. In *CIT 05, the Fifth international conference on computer and information technology*.

Hao, Y. and Zhang, Y. (2007). Web services discovery based on schema matching. In *the thirtieth Australasian conference on Computer science - Volume 62*.

Haussler, D. (1999). Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California, Santa Cruz.

Jouili, S. and Tabbone, S. (2009). Attributed graph matching using local descriptions. In *ACIVS 2009, LNCS 5807*, page 89–99.

Mandell, D. and McIlraith, S. (2003). A bottom-up approach to automating web service discovery, customization, and semantic translation. In *Proceedings of the Twelfth International World Wide Web Conference Workshop on E-Services and the Semantic Web (ESSW),Budapest*.

Mbareck, N. O. A. and Tata, S. (2006). Bpel behavioral abstraction and matching. *Business Process Management Workshops*, pages 495–506.

Mendling, J., Lassen, K., and Zdun, U. (2006). Transformation strategies between block-oriented and graph-oriented process modelling languages. *F. Lehner, H. Nsekabel, P. Kleinschmidt, eds. Multikonferenz Wirtschaftsinformatik*, pages 297–312.

Messmer, B. (1995). *Efficient Graph Matching Algorithms for Preprocessed Model Graphs*. PhD thesis, University of Bern, Switzerland.

Messmer, B. T. and Bunke, H. (1999). A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition*, 32:1979–1998.

Nejati, S., Sabetzadeh, M., Chechik, M., Easterbrook, S., and Zave, P. (2007). Matching and merging of statecharts specifications. In *ICSE 2007*, page 54–63.

Neuhaus, M. and Bunke, H. (2006). A convolution edit kernel for errortolerant graph matching. In *IEEE international conference on pattern recognition, Hong Kong*, page 220–223.

Paolucci, T., Kawmura, T., and Sycara, K. (2002). Semantic matching of web service capabilities. In *Springer Verlag, LNCS, Proceedings of the International Semantic Web Conference*.

Ramon, J. and Gartner, T. (2003). Expressivity versus efficiency of graph kernels. In *First International Workshop on Mining Graphs, Trees and Sequences*.

Riesen, K. and Bunke, H. (2009). Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27:950–959.

Sanfeliu, A. and Fu, K. (1983). A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics (Part B)*, 13(3):353–363.

Shen, Z. and Su, J. (2005). Web service discovery based on behavior signatures. *SCC*, 1:279–286.

Suard, F. and Rakotomamonjy, A. (2007). Mesure de similarité de graphes par noyau de sacs de chemins. In *21e colloque GRETSI sur le traitement du signal et des images, Troyes*.

Vu, F. P. L.-H., Hauswirth, M., and Aberer, K. (2006). A search engine for qosenabled discovery of semantic web services. *International Journal of Business Process Integration and Management*, 1(4):244–255.

Wang, Y. and Stroulia, E. (2003). Flexible interface matching for web-service discovery. In *WISE'2003*.

Wombacher, A. (2006). Evaluation of technical measures for workflow similarity based on a pilot study. *Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS*, 4275:255–272.