I.J. Information Technology and Computer Science, **2014, 07, 32-39**
Published Online June 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2014.07.05

Improving the Performance of Semantic Web Services Discovery: Shortest Path based Approach

Maamar Khater
Computer Science department of Tahar Moulay University, Algeria
Email: khater.maamar@gmail.com

Mimoun Malki
Computer Science department of Djillali Liabes University, Algeria
Email: malki@univ-sba.dz

Abstract— Service discovery is the process of retrieving the service most similar to the query based on the description of functional and/or non-functional semantics. The original algorithm used in literature was proposed by Paolucci et al., 2002. Some research works, propose an extension or an improvement of this algorithm to correct the matchmaking used. In this paper we present an algorithm of matchmaking that resolves the problems of Paolucci algorithm by using the shortest path algorithm which determines the optimal matching between user query and provider service. This approach is validated within a framework proposed at the end of this paper and compared with the greedy approach and the bipartite graph based matching.

Index Terms— Web Services, Ontology of Service OWL-S, Discovery, Matchmaking, Graph, Shortest Path

I. INTRODUCTION

The semantic web services are services with semantic descriptions. This semantic description is provided by ontologies which are one of significant semantic web technologies where the main objective is to increase the degree of automation of standard tasks such as discovery, selection, composition, etc. In literature, there are two approaches to describe the semantic web services. The first approach presents a description based on annotations. In this category, the web service is in its syntactic form, and it is enriched with semantic annotations associated with ontology. In this approach, the description is independent of a particular ontology language. As implementation of this approach we find: Semantic Annotation for Web Service Description Language – SAWSDL-, Web Service Semantics -WSDL-S- [1], and Universal Service-Semantics Description Language – USDL-. Another approach for the semantic description of RESTful services is Semantic Annotation for REST services -SA-REST-. The second approach presents a description based on semantic language. In this category, we choose from the beginning a semantic language to describe the service. As implementation of this approach, we find: Ontology Web Language for Services -OWL-S-, Web Service Modeling Ontology –WSMO-[2].

In addition, there are other proposals aim to describe semantic web services, like easy-L, and pyramid-S [3]. In this paper, we focus our study on the ontology of services OWL-S which is defined as a semantic language for describing Web services in an unambiguous way; this ontology is based on OWL language. OWL-S [4] describes the service in three ways as depicted in figure 1.

The service profile tells "what the service does". It contains the name of the service and its textual description, the description of functional properties (Input Output Precondition Effect -IOPE-) and non-functional properties (Quality of Service –QoS-). Many approaches of service discovery are based on the elements of the profile as criteria (called black-box Service matching approaches).

The service model tells a client how to use the service. It describes the internal running of the service which is modelled as a process and a set of control flow. There are three types of processes:

Atomic process corresponds to a single operation (single interaction); composite Process corresponds to a combination of processes (atomic or not) using control constructs (Sequence, Split, If-Then-Else etc.); finally Simple Process is not executable (or invoked). It provides an abstraction mechanism to provide multiple views of the same process.

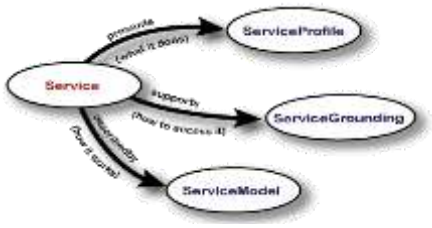

Fig. 1. Top level of the service ontology [4]

Service grounding specifies the details of how an agent can access a service. Typically grounding will specify a communication protocol, message formats, and other service-specific details such as port numbers used in contacting the service.

The rest of this paper is organised as follows. Section 2 presents an overview of some existing semantic web service matchmakers. Section 3 presents a review of some extensions or improvements of greedy algorithm. Section 4 describes our proposition for semantic discovery. Finally, the paper is concluded in section 5.

II. Semantic WEB Service Discovery

Service discovery is the process of retrieving the service most similar to the query based on the description of functional and/or non-functional semantics. In literature, some researchers deem service discovery system as matchmakers; and in other works it covers the entire spectrum of tasks from service request to service invocation which means the inclusion of the selection process. In general, any service discovery framework needs to define the matchmaker which evaluates the similarity metric between two services. We have two kinds of result returned by the discovery system:

- Exact discovery: in the case where exist a service which satisfies exactly the user requirements.
- Approximate discovery: is considered as realistic case, the discovery system returns a service (or set of services) which satisfies approximately the user requirements.

We can classify the related work on service matching in two categories of criterion:

- Category 1: {logic-based matching, non-logic-based matching, hybrid matching}
- Category 2: {interface level, process level, hybrid level}

An overview of some existing semantic web service matchmakers is illustrated in table 1.

In this section, we define the principle of each element of both categories.

- Logic-based matching: the matchmakers use the semantic relations and logic inference to measure the similarity between two services. The degree of logic-based matching can be determined either (a) exclusively within the considered logic theory by means of logic reasoning, or (b) by a combination of logical inferences within the theory and algorithmic processing outside the theory [5]. Examples of logic-based matchmakers are illustrated in table 1. The original algorithm used in literature was proposed by Paolucci et al., 2002 [6] which determines four degrees of match: exact, plugin, subsumes, and fail.
- Non-logic-based matching: the matchmaker performs out of any logic-reasoning to determine the similarity between services. They use other techniques from the search area like: graph isomorphism, information retrieval measurement … etc.
- Hybrid matching: the matchmaker uses a combination of logic and non-logic mechanisms to perform the matching process.
- Service profile matching level (so-called black-box service matching): in this case, the matchmaker is generally based on input/output matching. The algorithm of Paolucci performs the matching process at the interface level. Other matchmakers exploit other element of service profile like IOPE, PE, E, in the matching process.
- Process matching level (so-called glass-box service matching): in this case, the matchmaker is based on the behaviour matching in terms of control and data flow.
- Hybrid matching level: the matchmaker uses a combination of service profile matching level and process matching level mechanisms to perform the matching process.

In table 1, we summarize the categories of existing Semantic Web Service matchmakers. The description given by Klush et al., [5] is used with the integration of other categories (such as hybrid between profile and process) and other approaches.

In this paper, we focus our study on the improvement of Paolucci algorithm.

Table 1. Categories of some existing Semantic Web Service matchmakers

Hybrid (Profile and Process)	(Grigori et al., 2008)[14]	(Gunay et al., 2013)[15]	(Gater et al., 2010)[13], (BenMokhtar et al., 2005)[11] (Majithia et al., 2004)[18]
Process Model	(Nejati et al., 2007)[20] (Vander aalst et al 2006)[21] (Minor et al., 2007)[19]	OWSPM[5], (Ehrig M. Et al., 2007)[12]	IORPTM [5],
combined (QoS and functional)	iMacher1 [5], (Cassar G., et al 2013) [5]	GSD-MM [5]	Imatcher2[5], FC-MATCH [5] (Abhijit et al., 2004)[10]
IOPE	DSD-MM[5], XSSD, (Li Jing 2013) [17]	RFSD, GLUE, [5] (Kourtesis et al., 2008)[16]	WSMO-MX, LARKS, SA-WSDL-MX [5]
PE		PCEM [5]	
E		MAMAS, RACER [5]	
IO	MWSDI-LUMINA SE HotBlu [5]	OWLSM, SDS, OWLS-UDDI [5] [Paolucci et al., 2002][6]	OWLS-MX [5]
QoS	WSML-QoS SE [5]		ROWLS [5]
	Non-logic	**Logic**	**Hybrid**

III. Related Work

In this section, we review some extensions or improvements of greedy algorithm.

3.1 Greedy approach:

The algorithm proposed by Paolucci et al.,[6] is a greedy approach for matchmaking. It uses input/output concepts in the process of matching by defining four degrees of match as depicted in figure 2.

The algorithm of Paolucci [6] tries to find a max-match between each concept of the query (input/output) and concepts of the advertisement (input/output).

```
1 degreeOfMatch(outR,outA):
2    if outA=outR then return exact
3    if outR subclassOf outA then return exact
4    if outA subsumes outR then return plugIn
5    if outR subsumes outA then return subsumes
6    otherwise fail
```

Fig. 2. degree of match in greedy algorithm [6]

This algorithm presents an ambiguity where it doesn't describe whether a concept is removed once it has been matched.

3.1.1 Complexity

Let n_1 and n_2 the number of input concepts of query and advertise. Let m_1 and m_2 the number of output concepts of query and advertise. The complexity of matching is given by $((n_1*n_2)+(m_1*m_2))$

The matching algorithm iterates over t advertisement services in repository, then the total complexity is given by $(t*((n_1*n_2)+(m_1*m_2)))$. The complexity of this algorithm is polynomial, and at worst case where $t=n_1=n_2=m_1=m_2=N$, the complexity is bounded by $O(N^3)$.

3.1.2 Performance of matchmaking

The matchmaking results given by the greedy algorithm are not reliable. We consider the following scenarios. The concepts of advertise 'A' and query 'Q' are defined in Books ontology illustrated in figure 3. We denote 'dom' as degree of matching, and 'Gdom' as global degree of match.

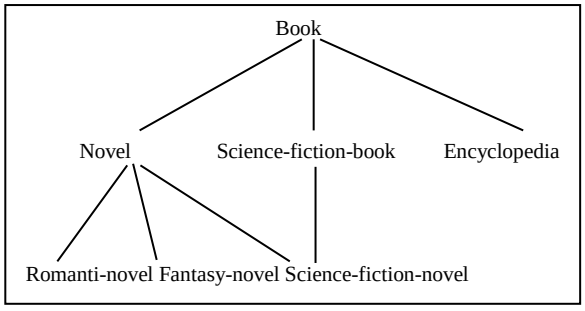

Fig. 3. Part of book ontology [4]

a. The first scenario: without removal of advertise concepts.

Advertise 'A'

Input	publisher
Output	novel, price

Query 'Q'

Input	publisher
Output	romantic novel, science-fiction novel

dom(romantic novel, novel)=exact=1.
dom(romantic novel, price)=fail=0.
dom(science-fiction novel, novel)=exact=1.
dom(science-fiction novel, price)=fail=0.
Gdom=exact.

The matchmaker returns 'A' as correct response to 'Q', where 'A' presents a false positive (two or more concepts from 'Q' match a single concept in 'A').

b. The second scenario: with removal of advertise concepts

Advertise 'A'

Input	publisher
Output	novel, science-fiction book

Query 'Q'

Input	publisher
Output	science-fiction novel, romantic novel

dom(science-fiction novel, novel)=exact=1.
dom(science-fiction novel, science-fiction book)=1= exact.
Science-fiction novel match novel, and novel is removed from advertise concepts.
dom(romantic novel, science-fiction book)=fail=0.

The matchmaker returns as a response 'A' don't match 'Q', where 'A' presents a false negative (the order of query concepts influences in the matching process, and changes the global degree of match).

Some research works, propose an extension or an improvement of Paolucci algorithm. Bellure U., et al [7] uses the Hungarian algorithm to determine the matching of bipartite graphs. Phatak J. et al., [8] adds ontology mappings and QoS constraints. Michael C. Et al., [9] proposes a matching based on properties matching and service profile hierarchy. We compare our approach with the bipartite graph based matching.

3.2 Bipartite graph based matching:

To solve the problems of greedy algorithm, Bellure U., et al [7] proposes a semantic matchmaking based on bipartite matching and the Hungarian algorithm to achieve an optimal solution of concepts matching. As depicted in table 2, the algorithm assigns different numerical weights to degrees of match.

Table 2. weight of degree of match

Degree of match	Weight
Exact	$W_1=1$
Plugin	$W_2=(w_1*\|v_0\|)+1$
Subsume	$W_3=(w_2*\|v_0\|)+1$

V_0 is the vertices cardinality of bipartite graph.

The Hungarian algorithm computes a complete matching for weighted bipartite graph, and the optimal matching is given by a complete matching with a minimum of all maximum weighted edges in the matching.

3.2.1 Complexity

We use the same variables to compute the complexity of this algorithm. The time complexity of Hungarian algorithm is bounded by $O(n^3)$ where "n" is the cardinality of concepts (input/output). The complexity of the matching is given by $((n_1*n_2)+n_2^3+(m_1*m_2)+m_1^3)$ The repository cardinality is t, then the global complexity is given by $(t*((n_1*n_2)+n_2^3+(m_1*m_2)+m_1^3))$. The complexity of the matchmaking algorithm is polynomial. At worst case, where $t=n_1=n_2=m_1=m_2=N$, the complexity of bipartite graph based matching is bounded by $O(N^4)$.

3.2.2 Performance of matchmaking

Bipartite graph based matching regulates false positives and false negatives as discussed in the previous scenarios. The performance of the matchmaker is better than the greedy algorithm.

IV. Proposition

4.1 Principles:

Our proposal consists to resolve the problem of the concepts order in the algorithm of Paolucci, we propose to represent the matching process as a matrix $M_{n,m}$ where we consider the following weights:

Table 3. degree of match of OWLS-SP

degree of match	Weight
Exact	W1=1
Plugin	W2=(w1*|M|)+1
Subsumes	W3=(w2*|M|)+1
Fail	W4=(w3*|M|)+1

Where |M| is the dimension of the matrix M.

Definition "matching matrix": the matching is represented as matrix $M_{n,m}$ where :

m_{ij}= dom(C_i,C_j) such C_i and C_j are ontological concepts.

For example, let M a matching matrix for output concepts between query and advertise services:

$$M = \begin{pmatrix} 1 & 4 & 40 \\ 4 & 40 & 1 \\ 40 & 13 & 4 \end{pmatrix}$$

Before determining the global degree of matching (for the output or input concepts), we define the following rules:

- Transform the matching matrix $M_{n,m}$ to graph G where: The vertices are m_{ij}, the arcs are organized by column (arc (ci, ci +1)), and it is not allowed to create an arc between the first and the 3rd column. It is not allowed to connect two vertices of the same line or column. The source vertex of G is connected to all the vertices of the first column. Each vertex (element) in a column is connected to all other vertices (elements) of the next column, if exist. The terminus vertex of G is connected to all the vertices of the last column.
- Each arc Ai that connects two vertices ni and nj is weighted by the ni value where ni presents the source of this arc. P_{ij} is the weight of the transition between nodes ni and nj, where P_{ij} = ni.
- Outgoing arcs of the source vertex are weighted by zero "0".
- We search the shortest path in G; Dijkstra's algorithm is applied according to the specification of our graph G (a path is valid if it is made up of independent nodes, its nodes do not share the same line).
- The optimal solution of the matching matrix M is done by the vertices of shortest path, denoted π.

Lemma: a matching in which ∑wi is minimized, is equivalent to a matching with a shortest path.

We use a proof by contradiction to prove this lemma.

Let $\pi=V_1V_2V_3\ldots V_n$ denotes the path with minimal sum of weights in G: $|\pi|=\sum Vi$ where Vi are the vertices of π.

Let $\pi'=N_1N_2N_3\ldots N_m$ denotes the shortest path in G: $|\pi'|=\sum N_i$ where N_i are the vertices of π'.

Assume that the lemma is untrue, that means $|\pi|=\sum V_i < |\pi'|=\sum N_i$ ….(I)

By definition, if π' is the shortest path in G then there is no path with minimum length (minimal sum of weights) than π' in G, which means that (I) is untrue.

We can hence infer that $|\pi|=\sum V_i = |\pi'|=\sum N_i$ and both π and π' are a shortest path in G.

Definition "global degree of matching Gdom": the Gdom in both output matching and input matching is given by the following rule: Gdom= |π| where π represents the shortest path of G.

For the previous example, the optimal matching is given by π where: $\pi=m_{21}m_{12}m_{33}$

π is a valid path, and Gdom=|π|=12

We use the Gdom for ranking the results returned by the matchmaker.

Examples: let M1, M2, M3, M4 four output matching matrixes, and we aim to rank them:

$$M1 = \begin{pmatrix} 1 & 4 & 40 \\ 40 & 1 & 13 \\ 4 & 1 & 40 \end{pmatrix}$$ π=$m_{11}m_{32}m_{23}$, Gdom_out=15

$$M2 = \begin{pmatrix} 4 & 1 & 13 \\ 1 & 13 & 4 \\ 13 & 4 & 1 \end{pmatrix}$$ π=$m_{21}m_{12}m_{33}$, Gdom_out=3

$$M3 = \begin{pmatrix} 1 & 13 & 40 \\ 4 & 1 & 4 \\ 40 & 1 & 13 \end{pmatrix}$$ π=$m_{11}m_{32}m_{23}$, Gdom_out=6

$$M4 = \begin{pmatrix} 1 & 4 & 40 \\ 40 & 4 & 13 \\ 4 & 13 & 40 \end{pmatrix}$$ π=$m_{31}m_{12}m_{23}$, Gdom_out=21

The results ranked in descending order of Gdom are: M2, M3, M1, and M4.

4.2 Algorithm:

In this section, we present the procedures used in our matchmaking algorithm.

Matchmaking Algorithm

Input: Query Q
Output: set of services ranked in descending order, called Result
Result=empty
For each service A_i in repository do
If card(Q_{out})>card(A_{out}) then Gdom_out=0
Else Gdom_out=Matching output concepts (Q_{out}, A_{out})
If Gdom_out ≠ 0 then If card(A_{in})>card(Q_{in}) then Gdom_in=0
Else Gdom_in= Matching input concepts (A_{in}, Q_{in})
score[A_i]=(Gdom_out + Gdom_in)/2
Append.Result(A_i, score[A_i])
End_for
Ranked Result in ascending order of score, the best score is which have the lowest value (different to zero).
Return Result.

Degree of match_out
Input: two concepts: Q_{out}, A_{out}, Max(n,m)
Output: dom, where dom={w1, w2 , w3 , w4}
If Q_{out} = A_{out} then return w1 //exact
If Q_{out} subclassOf A_{out} then return w1 //exact
If Q_{out} subsumed by A_{out} then return w2 //plugin
If Q_{out} subsumes A_{out} then return w3 //subsumes
Otherwise return w4 //fail

Degree of match_in
Input: two concepts: A_{in}, Q_{in}, Max(n,m)
Output: dom, where dom={ w1, w2 , w3 , w4}
If A_{in}=Q_{in} then return w1 //exact
If A_{in} subclassOf Q_{in} then return w1 //exact
If A_{in} subsumed by Q_{in} then return w2 //plugin
If A_{in} subsumes Q_{in} then return w3 //subsumes
Otherwise return w4 //fail

Output Matching Matrix
Input: two set of output concepts: Q_{out}, A_{out} //vectors
Output: $M\_out_{n,m}$
For i= 1 to n do
For j= 1 to m do M[i,j]=degree of match_out(Q_{out}[i], A_{out}[j])
Return $M\_out_{n,m}$

Input Matching Matrix
Input: two set of input concepts: A_{in}, Q_{in} // two vectors
Output: $M\_in_{n,m}$
For i= 1 to n do
For j= 1 to m do M[i,j]=degree of match_in(A_{in}[i], Q_{in}[j])
Return $M\_in_{n,m}$

Matching output concepts: // each Q_{out} needs to be // matched with A_{out}
Input: n concepts of Q_{out}, m concepts of A_{out}
Output: Gdom_out // the global degree of matching for // output concepts
1. Call the procedure: Output Matching Matrix // return $M\_out_{n,m}$
2. If all elements of a line in the matrix $M\_out_{n,m}$ equal to max(n,m) then Gdom_out=0, goto (5)
3. Call the procedure: Dijkstra algorithm for shortest path($M\_out_{n,m}$) //search an optimal matching (π)
4. Gdom_out=$|\pi|=\sum Vi$
5. Return Gdom_out, END.

Matching input concepts: // each A_{in} needs to be // matched with Q_{in}
Input: n concepts of A_{in}, m concepts of Q_{in}
Output: Gdom_in //the global degree of matching for // input concepts
1. Call the procedure: input Matching Matrix // return $M\_in_{n,m}$
2. If all elements of a line in the matrix $M\_in_{n,m}$ equal to max(n,m) then Gdom_in=0, goto (5)
3. Call the procedure: Dijkstra algorithm for shortest path ($M\_in_{n,m}$) //search an optimal matching (π)
4. Gdom_in=$|\pi|=\sum Vi$
5. Return Gdom_in, END.

Dijkstra algorithm for shortest path:
Input: matching matrix $M_{n,m}$
Output: shortest path denoted π,
1. $T \leftarrow \emptyset$
2. For $v \in V$ do $d(v) \leftarrow \infty$
3. $d(s) \leftarrow 0$
4. while ($T \neq V$)
$v \leftarrow rgmin\{d(u): u \notin T\}$
$T \leftarrow T \cup \{v\}$
For $u \in$ voisins(v) do
$d(u) \leftarrow min\{d(u), d(v) + w_{vu}\}$
5. Return optimal matching expressed by π, END.

The complexity of our matchmaker algorithm is computed as follows. We denote: Card(Adv), the number of advertise services, Card(Q_{out}), the number of query output concepts, Card(Q_{in}), the number of query input concepts, Card(A_{out}), the number of advertise output concepts, Card(A_{in}), the number of advertise input concepts.

The complexity formula can be expressed as:

Time complexity of *Matchmaking Algorithm* is bounded by $O(n^3)$

Time complexity of *Dijkstra algorithm for shortest path ($M_{n,m}$)* is bounded by $O(n^2)$

Time complexity of *Matching output concepts(vector Q_{out}, vector A_{out})* is bounded by $O(n^2)$

Time complexity of *Output Matching Matrix(vector Q_{out}, vector A_{out})* is bounded by $O(n^2)$

Time complexity of Degree of match_out(Q_{out}, A_{out}) is bounded by O(1)

The proposed algorithm has cubic time complexity.

4.3 Experimentation:

In this section, we implement our algorithm of matchmaking, and we use some tools like: Owl-s API [22] (to parse queries, services, and ontologies), Owls-tc as

benchmark [23]. The architecture of our application is illustrated in figure 4. For measuring the accuracy of our algorithm we use a collection of Web services (OWLS-TC). This collection has more than 500 services covering several application domains.

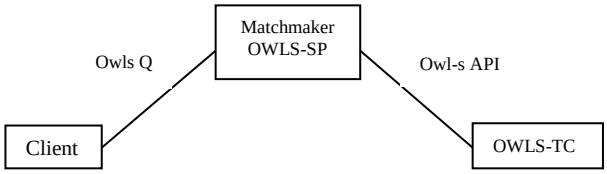

Fig. 4. OWLS-SP matchmaker architecture

In order to analyze the improvement of accuracy obtained by our algorithm, we use the individual precision-recall chart technique. Precision and Recall are two indicators of effectiveness [24].

Precision is the ratio of the number of relevant records retrieved (TP -true positive-) to the total returned (TP and FP -false positive-). It is expressed as: precision= $\frac{TP}{TP+FP}$

Recall is the ratio of number of relevant records retrieved to the total number of relevant records (TP and FN -false negative-). It is expressed as: recall= $\frac{TP}{TP+FN}$

Numerator is same for precision and recall: number of correct returned; denominator for precision is all that is returned; denominator for recall is all that is relevant.

In this paper both indicators were used to measure the effectiveness of our algorithm and compared to greedy algorithm and bipartite graph based matching, for this purpose we made a test set (queries and services) for the evaluation of the performance of the three discovery matchmakers.

Query:
Input: BOOK, NOVEL, PUBLICATION
Output: RECOMMENDEDPRICE, PRICE, TAXEDPRICE
Ontology:
C:\Program Files\Apache Software Foundation\Tomcat 6.0\webapps\ ontology\books.owl
C:\Program Files\Apache Software Foundation\Tomcat 6.0\webapps\ ontology\concept.owl

We use five services for parsing our algorithm against greedy algorithm and bipartite graph based matching.

Fig. 5. OWLS-SP interface

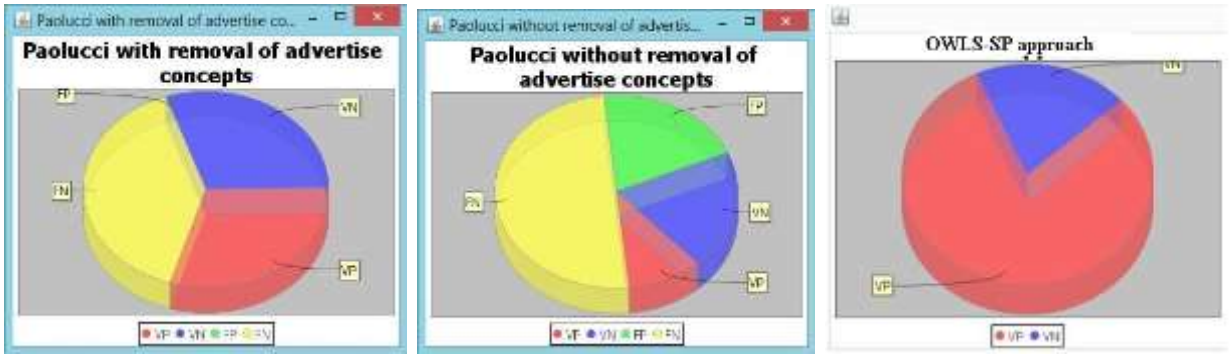

Fig. 6. performance evaluation.

Greedy algorithm:
Paolucci with removal concepts: VP=3, VN=3, FP=0, FN=4
Paolucci without removal concepts: VP=1, VN=2 FP=2, FN=5
Bipartite graph based matching: VP=7, VN=3, FP=0, FN=0
Our approach: OWLS-SP: VP=7, VN=3, FP=0, FN=0

We conclude that our approach OWLS-SP which uses the shortest path algorithm presents the same results as the bipartite graph based matching which uses the Hungarian algorithm. Both approaches present results better than greedy algorithm. The complexity of our algorithm is bounded by $O(N^3)$. Indeed, this result presents an advantage compared to bipartite graph based matching algorithm.

Table 4. comparison results

	Paolucci approach	Bellur approach	Our approach
Dom	exact, plugin, subsume, fail	exact=w1=1 plugin=w2 =(w1*|v|)+1 subsumes=w3 =(w2*|v|)+1 fail=w4 =(w3*|v|)+1	exact=w1=1 plugin=w2 =(w1*|M|)+1 subsumes=w3 =(w2*|M|)+1 fail=w4 =(w3*|M|)+1
Matching algorithm	Greedy algorithm	Hungarian algorithm	Shortest path algorithm
ranking	Descending order of Gdom_out	Gdom=max-weight edge in Graph	Gdom=Length of shortest path in Graph
complexity	$O(N^3)$	$O(N^4)$	$O(N^3)$

V. CONCLUSION

The quality of results of the discovery process is based mainly on the correctness of the matchmaker used. In this paper we have presented an algorithm of matchmaking that resolves the problems of Paolucci algorithm by using the shortest path algorithm which determines the optimal matching between user query and provider service. The complexity of the proposed matchmaker is polynomial and is bounded by $O(N^3)$, this result is better than the complexity of the bipartite graph based matching which is bounded by $O(N^4)$. We performed some experiments to validate our approach and to analyze the improvement of accuracy obtained by our algorithm based on two indicators of effectiveness: Precision and Recall. We concluded that our approach had better results than the greedy approach and presents the same performance as the bipartite graph based matching. Finally we developed a tool called OWLS-SP Discovery to disseminate our algorithm. Our future work is focused on analysis our algorithm in the case of composite services.

REFERENCE

[1] WSDL-S. http://www.w3.org/Submission/WSDL-S/ visited 15/09/2013
[2] WSMO. http://www.w3.org/Submission/WSMO/ visited 15/09/2013
[3] Ngan L.D., Kanagasabai R. Semantic Web service discovery: state-of-the-art and research challenges. Personal and Ubiquitous Computing journal, september 2012. Springer-Verlag. Online ISSN 1617-4917
[4] OWL-S. http://www.w3.org/Submission/OWL-S/ visited 15/09/2013
[5] Klusch M, "Semantic Web Service Coordination", CASCOM - Intelligent Service Coordination in the Semantic Web, Chapter 4, Springer, pp.69-pp.114, 2008.
[6] Paolucci, M., Kawamura, T., Payne, T. R., & Sycara, K. (2002). Semantic matching of Web services capabilities. In Proc. of 1st International Semantic Web Conference (ISWC). LNCS Volume 2342, 2002, pp 333-347
[7] Bellur Umesh, Kulkarni Roshan, Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching. Conference: International Conference on Web Services - ICWS , pp. 86-93, 2007
[8] Phatak J. et al. A Framework for Semantic Web Services Discovery. WIDM, 2005.
[9] Michael C. Jaeger, Stefan Tang, Ranked Matching for Service Descriptions using DAML-S Conference on Advanced Information Systems Engineering - CAiSE , pp. 217-228, 2004
[10] Abhijit A. Patil, Swapna A. Oundhakar, Amit P. Sheth, Kunal Verma. Meteor-s web service annotation framework. World Wide Web Conference Series - WWW , pp. 553-562, 2004
[11] BenMokhtar S, Nikolaos Georgantas, Valérie Issarny: Ad Hoc Composition of User Tasks in Pervasive Computing Environments. Software Composition 2005: 31-46
[12] Ehrig M, A. Koschmider, and A. Oberweis. Measuring similarity between semantic business process models. In Proc. of APCCM 2007, pages 71-80, 2007.
[13] Gater A, Daniela Grigori, Mokrane Bouzeghoub: A Graph-Based Approach for Semantic Process Model Discovery. Graph Data Management 2011: 438-462
[14] Grigori D, J.C. Corrales, and M. Bouzeghoub. Behavioral matchmaking for service retrieval: Application to conversation protocols. Inf. Syst., 33(7-8):681-698, 2008.
[15] Günay A, Pinar Yolum: Service matchmaking revisited: An approach based on model checking. J. Web Sem. 8(4): 292-309 (2010)
[16] Kourtesis Dimitrios, Iraklis Paraskakis: Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery. ESWC 2008: 614-628
[17] Li Jing. A Fast Semantic Web Services Matchmaker for OWL-S Services. JOURNAL OF NETWORKS, VOL. 8, NO. 5, MAY 2013.
[18] Majithia S., David W. Walker and W. A. Gray. A framework for automated service composition in service-

oriented architecture. In 1st European Semantic Web Symposium, 2004.
[19] Minor M, A. Tartakovski, and R. Bergmann. Representation and structure-based similarity assessment for agile workows. In Proc. of the Intl. Conf. on Case-Based Reasoning, volume 4626 of LNAI, pages 224-238. Springer, 2007.
[20] Nejati S, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave. Matching and merging of statecharts speci_cations. In Proc. of ICSE 2007, pages 54-63, 2007.
[21] van der Aalst W, A.K. Alves de Medeiros, and A. Weijters. Process Equivalence: Comparing two process models based on observed behavior. In Proc. of BPM 2006, volume 4102 of LNCS, pages 129-144. Springer, 2006.
[22] OWL-S API. MINDSWAP: Maryland Information and Network Dynamics Lab Semantic Web Agents Project. http://www.mindswap.org/2004/owl-s/api/
[23] OWL-S Service Retrieval Test Collection. http://projects.semwebcentral.org/projects/owls-tc/
[24] Rijsbergen C. J. V., "Information Retrieval," Buttersworth, London, 1976.

Authors' Profiles

Maamar Khater is a PhD student in Computer Science at the Department of Computer Science in the University of Djillali Liabes (Sidi Bel-Abbes, Algeria). His academic interests include semantic web services, workflow management and multi-agent systems.
http://fsi.univ-tlemcen.dz/depart-inf/fiche-KHATER.html

Mimoun Malki is graduated with Engineer Degree in Computer Science from National Institute of Computer Science, Algiers, in 1983. He received the Master of Science degree and the PhD degree in Computer Science from the University of Sidi Bel-Abbes, Algeria, in 1992 and 2002, respectively. He was a Senior Lecturer in the Department of Computer Science at the University of Sidi Bel-Abbes from 1986 to 2002. Currently, he is a Professor at University of Djillali Liabes, Sidi Bel-Abbes, Algeria. His research interests include databases, ontology engineering, web-based information systems, semantic web, web services and web reengineering.

How to cite this paper: Maamar Khater, Mimoun Malki,"Improving the Performance of Semantic Web Services Discovery: Shortest Path based Approach", International Journal of Information Technology and Computer Science(IJITCS), vol.6, no.7, pp.32-39, 2014. DOI: 10.5815/ijitcs.2014.07.05

