A Framework for Parameterized Semantic Matchmaking and Ranking of Web Services

Fatma Ezzahra Gmati[1], Nadia Yacoubi Ayadi[1], Afef Bahri[2], Salem Chakhar[3] and Alessio Ishizaka[3]
[1]*RIADI Research Laboratory, National School of Computer Sciences, University of Manouba, Manouba, Tunisia*
[2]*MIRACL Laboratory, High School of Computing and Multimedia, University of Sfax, Sfax, Tunisia*
[3]*Portsmouth Business School and Centre for Operational Research & Logistics, University of Portsmouth, Portsmouth, U.K.*

Keywords: Web service, Semantic Similarity, Matchmaking, Ranking, Performance.

Abstract: The Parameterized Semantic Matchmaking and Ranking (PMRF) is a highly configurable framework supporting a parameterized matching and ranking of Web services. The paper introduces the matching and ranking algorithms supported by the PMRF and presents its architecture. It also evaluates the performance of the PMRF and compares it to the iSeM-logic-based and SPARQLent frameworks using the OWLS-TC4 datasets. The comparison results show that the algorithms included in PMRF behave globally well in comparison to iSeM-logic-based and SPARQLent.

1 INTRODUCTION

The matchmaking is a crucial operation in Web service discovery and selection. The objective of the matchmaking is to discover and select the most appropriate Web service among the different available candidates. Different matchmaking frameworks are now available in the literature (Chakhar et al., 2014)(Chakhar et al., 2015)(Doshi et al., 2004) (Klusch and Kapahnke, 2012)(Samper Zapater et al., 2015)(Sbodio et al., 2010)(Sharma et al., 2015)(Sycara et al., 2003) but most of them present at least one of the following shortcomings: (1) use of strict syntactic matching, which generally leads to low recall and low precision rates; (2) use of capability-based matchmaking, which is proven (Doshi et al., 2004) to be inadequate in practice; (3) lack of customization support; and (4) lack of accurate ranking of matching Web services, especially within semantic-based matching.

The objective of this paper is to present the Parameterized Matching-Ranking Framework (PMRF), which uses semantic matchmaking, accepts capability and property attributes, supports different levels of customization and generates a ranked list of matching Web services. Hence, the proposed system can deal jointly with the previous shortcomings. The comparison of PMRF to iSeM-logic-based (Klusch and Kapahnke, 2012) and SPARQLent (Sbodio et al., 2010), using the OWLS-TC4 datasets, shows that the algorithms supported by PMRF behave globally well in comparison to iSeM-logic-based and SPARQLent.

The paper first reviews the matching and ranking algorithms (Section 2). Next, it presents the architecture of the PMRF (Section 3). Then, it studies the performance of the PMRF (Section 4). Lastly, the paper provides the comparative study (Section 5), comments on the user/provider acceptability (Section 6) and discusses some related work (Section 7). The last section concludes the paper.

2 MATCHING AND RANKING ALGORITHMS

In this section, we briefly present the matching and ranking algorithms supported by the PMRF.

2.1 Matching Algorithms

The PMRF supports three matching algorithms—basic, partially parameterized and fully parameterized—supporting different levels of customization (see Table 1). The basic matching algorithm supports no customization. The partially parameterized matching algorithm allows the user to specify the set of attributes to be used in the matching. Within the fully parameterized matching algorithm, three customizations are taken into account. A first customization consists in allowing the user to specify the list of attributes to consider. A second customization consists in allowing the user to specify the order in which the attributes are considered. A third customization is to allow the user to specify a desired similarity measure for each attribute. In the rest of this section, we present the third algorithm.

Table 1: Customization levels.

Matching algorithm	List of attributes	Order of attributes	Desired similarity
Basic			
Partially parameterized	✓		
Fully parameterized	✓	✓	✓

In order to support all the above-cited customizations of the fully parameterized matching, we used the concept of Criteria Table, introduced by (Doshi et al., 2004), that serves as a parameter to the matching process. A Criteria Table, C, is a relation consisting of two attributes, $C.A$ and $C.M$. The $C.A$ describes the service attribute to be compared, and $C.M$ gives the *least preferred similarity measure* for that attribute. Let $C.A_i$ and $C.M_i$ denote the service attribute value and the desired measure in the ith tuple of the relation. The $C.N$ denotes the number of tuples in C.

Let S^R be the service that is requested, S^A be the service that is advertised and C a criteria table. A sufficient match exists between S^R and S^A if for *every* attribute in $C.A$ there exists an identical attribute of S^R and S^A and the values of the attributes satisfy the desired similarity measure specified in $C.M$. Formally,

$$\forall_i \exists_{j,k}(C.A_i = S^R.A_j = S^A.A_k) \wedge \mu(S^R.A_j, S^A.A_k) \succeq C.M_i \Rightarrow \text{SuffMatch}(S^R, S^A) \quad 1 \leq i \leq C.N. \tag{1}$$

The computing of the similarity degrees $\mu(\cdot,\cdot)$ is addressed in Section 2.2. The fully parameterized matching process is formalized in Algorithm 1, which follows directly from Sentence (1). Algorithm 1 proceeds as follows. First, it loops over the attributes in the Criteria Table C and for each attribute it identifies the corresponding attribute in the requested service S^R and the potentially advisable service under consideration S^A. The corresponding attributes are appended into two different lists rAttrSet (requested Web service) and aAttrSet (advisable Web service). This operation is implemented by sentences 1 to 10 in Algorithm 1. Second, it loops over the Criteria Table and for each attribute it computes the similarity degree between the corresponding attributes in rAttrSet and aAttrSet. This operation is implemented by sentences 11 to 14 in Algorithm 1. The output of Algorithm 1 is either success (if for every attribute in C there is a similar attribute in the advertised service S^A with a sufficient similarity degree) or fail (otherwise).

The Criteria Table C used as parameter to Algorithm 1 permits the user to control the matched attributes, the order in which attributes are compared, as well as the minimal desired similarity for each attribute. The structure of partially matching algorithm is similar to Algorithm 1 but it takes as input an unordered collection of attributes with no desired similarities. The basic matching algorithm do no support any customization and the only possible inputs are the specification of the requested S^R and advertised S^A services. A further discussion about the different customization levels is given in Section 3.4.

Algorithm 1: Fully Parameterized Matching.

```
   Input  : $S^R$, // Requested service.
            $S^A$, // Advertised service.
            $C$, // Criteria Table.
   Output: Boolean, // fail/success.
1  while $(i \leq C.N)$ do
2      while $(j \leq S^R.N)$ do
3          if $(S^R.A_j = C.A_i)$ then
4              Append $S^R.A_j$ to rAttrSet;
5          $j \longleftarrow j+1$;
6      while $(k \leq S^A.N)$ do
7          if $(S^A.A_k = C.A_i)$ then
8              Append $S^A.A_k$ to aAttrSet;
9          $k \longleftarrow k+1$;
10     $i \longleftarrow i+1$;
11 while $(t \leq C.N)$ do
12     if $(\mu(\text{rAttrSet}[t], \text{aAttrSet}[t]) \prec C.M_t)$ then
13         return fail;
14     $t \longleftarrow t+1$;
15 return success;
```

Let us now focus on the complexity of Algorithm 1. Generally, we have $S^A.N \gg S^R.N$, hence the complexity of the first outer *while* loop is $O(C.N \times S^A.N)$. Then, the worst case complexity of Algorithm 1 is $O(C.N \times S^A.N) + \alpha$ where α is the complexity of computing $\mu(\cdot,\cdot)$. The value of α depends on the approach used to infer $\mu(\cdot,\cdot)$. As underlined in (Doshi et al., 2004), inferring $\mu(\cdot,\cdot)$ by ontological parse of pieces of information into facts and then utilizing commercial rule-based engines, which use the fast Rete pattern-matching algorithm leads to $\alpha = O(|R||F||P|)$ where $|R|$ is the number of rules, $|F|$ is the number of facts, and $|P|$ is the average number of patterns in each rule. Furthermore, we observe, as in (Doshi et al., 2004), that the process of computing $\mu(\cdot,\cdot)$ is the most 'expensive' step of Algorithm 1. Hence, the complexity of the matching algorithm will be $O(C.N \times S^A.N) + O(|R||F||P|) \asymp O(|R||F||P|)$.

Different versions and extensions of this algorithm are available in (Chakhar, 2013)(Chakhar et al., 2014)(Chakhar et al., 2015)(Gmati et al., 2014). Finally, we remark that Algorithm 1 permits to compute the similarly between a requested Web service S^R and an advertised Web service S^A. In practice, however, matching process should consider all the Web services available in the registry. An extended version of Algorithm 1 that takes into account this fact is given in (Gmati et al., 2014).

2.2 Computing Similarity Degrees

To compute the similarity degree $\mu(\cdot,\cdot)$, we extended the solution of (Bellur and Kulkarni, 2007) where the authors define four degrees of match, namely `Exact`, `Plug-in`, `Subsumes` and `Fail` as default. During the matching process, the inputs and outputs of the requested Web service are matched with the inputs and outputs of the advertised Web service by constructing a bipartite graph where the vertices correspond to concepts associated with the attributes. The bipartite graph is defined such that: (i) the vertices in the left side of the bipartite graph correspond to advertised services, (ii) the vertices in the right side correspond to the requested service; and (iii) the edges correspond to the semantic relationships between the concepts in left and right sides of the graph. Then, they assign weights to each edge as follows: `Exact`: w_1, `Plug-in`: w_2, `Subsumes`: w_3, `Fail`: w_4; with $w_4 \succ w_3 \succ w_2 \succ w_1$. Finally, they apply the Hungarian algorithm to identify the complete matching that minimizes the maximum weight in the graph. The final returned degree is the one corresponding to the maximum weight in the graph. Then, the selected assignment is the one representing a strict injective mapping, such that the maximal weight is minimized.

The optimality criterion used in (Bellur and Kulkarni, 2007) is designed to minimize the false positives and the false negatives. In fact, minimizing the maximal weight would minimize the edges labeled `Fail`. However, the choice of $\max(w_i)$ as a final return value is restrictive and the risk of false negatives in the final result is higher. To avoid this problem, we propose to consider both $\max(w_i)$ and $\min(w_i)$ as pertinent values in the matching.

A further discussion of similarity degree computing is available in (Gmati et al., 2015).

2.3 Ranking Algorithms

The PMRF supports three ranking algorithms: score-based, rule-based and tree-based. The first algorithm relies on the scores only. The second algorithm defines and uses a series of rules to rank Web services. It permits to solve the ties problem encountered by the score-based ranking algorithm. The tree-based algorithm, which is based on the use of a tree data structure, permits to solve the problem of ties of the first algorithm. In addition, it is computationally better than the rule-based ranking algorithm. The score-based ranking is given in Algorithm 2. The rule-based and tree-based ranking algorithms are available in (Gmati et al., 2014) and (Gmati et al., 2015), respectively.

Algorithm 2: Score-Based Ranking.

```
Input  : mServices,// List of matching Web services.
         $N$,// Number of attributes.
Output: mServices,// Ranked list of Web services.
1  mServices ← ComputeNormScores(mServices,$N$);
2  $r$ ← length(mServices);
3  for ($i = 1$ to $r - 1$) do
4      $j \leftarrow i$;
5      while
       ($j \geq 0 \wedge$ mServices[$j-1, N+2$] > mServices[$j, N+2$]) do
6          swap mServices[$j, N+2$] and mServices[$j-1, N+2$];
7          $j \leftarrow j - 1$;
8  return mServices;
```

The main input of this algorithm is a list mServices of matching Web services. The function ComputeNormScores in Algorithm 2 permits to calculate the normalized scores of Web services. It implements the idea we proposed in (Gmati et al., 2014). The score-based ranking algorithm uses then an *insertion sort* procedure (implemented by lines 3-7 in Algorithm 2) to rank the Web services based on their normalized scores.

The list mServices used as input to Algorithm 2 has the following generic definition:

$$(S_i^A, \mu(S_i^A.A_1, S^R.A_1), \cdots, \mu(S_i^A.A_N, S^R.A_N)),$$

where: S_i^A is an advertised service, S^R is the requested service, N the total number of attributes and for $j \in \{1, \cdots, N\}$, $\mu(S_i^A.A_j, S^R.A_j)$ is the similarity measure between the requested Web service and the advertised Web service on the jth attribute A_j.

The list mServices will be first updated by function ComputeNormScores and it will have the following new generic definition:

$$(S_i^A, \mu(S_i^A.A_1, S^R.A_1), \cdots, \mu(S_i^A.A_N, S^R.A_N), \rho'(S_i^A)),$$

where: S_i^A, S^R, N and $\mu(S_i^A.A_j, S^R.A_j)$ $(j = 1, \cdots, N)$ are as above; and $\rho'(S_i^A)$ is the normalized score of advertised Web service S_i^A. The normalized score $\rho'(S_i^A)$ is computed by ComputeNormScores.

Based on the discussion in Section 2.2, we designed two versions for computing similarity degrees. Accordingly, two versions can be distinguished for the definition of the list mServices at the input level, along with the way the similarity degrees are computed. The first version is as follows:

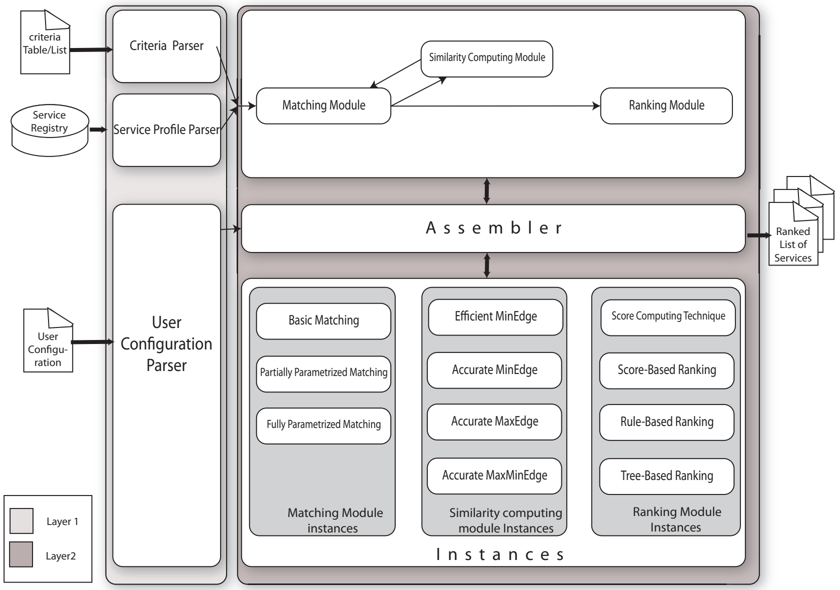


Figure 1: Conceptual architecture of the PMRF.

$(S_i^A, \mu_{\max}(S_i^A.A_1, S^R.A_1), \cdots, \mu_{\max}(S_i^A.A_N, S^R.A_N))$, where: S_i^A, S^R and N are as above; and $\mu_{\max}(S_i^A.A_j, S^R.A_j)$ $(j = 1, \cdots, N)$ is the similarity measure between the requested Web service and the advertised Web service on the jth attribute A_j computed by selecting the edge with the **maximum weight** in the matching graph.

The second version of mServices is as follows: $(S_i^A, \mu_{\min}(S_i^A.A_1, S^R.A_1), \cdots, \mu_{\min}(S_i^A.A_N, S^R.A_N))$, where S_i^A, S^R and N are as above; and $\mu_{\min}(S_i^A.A_j, S^R.A_j)$ $(j = 1, \cdots, N)$ is the similarity measure between the requested Web service and the advertised Web service on the jth attribute A_j computed by selecting the edge with the **minimum weight** in the matching graph.

To obtain the final rank, we need to use these two versions separately and then combine the obtained rankings. However, a problem of ties may occur since several Web services may have the same scores with both versions. This will deteriorate the precision rate. The tree-based ranking algorithm (Gmati et al., 2015) permits to solve this new ties problem.

The function ComputeNormScores in Algorithm 2 has a complexity of $O(rN^2)$ where r is the number of Web services and N is the number of attributes. The length in line 2 is assumed to be a built-in function and its complexity is not considered here. The sentences in lines 3-7 in Algorithm 2 implement an insertion sort procedure, which at best has a time complexity of $O(r)$ and in worst case, it makes $O(r^2)$. Hence, the overall complexity of Algorithm 2 is $O(rN^2) + O(r)$ in best case and $O(rN^2) + O(r^2)$ in worst case.

3 SYSTEM ARCHITECTURE AND IMPLEMENTATION

In this section, we first introduce conceptual and functional architecture of the PMRF. Then, we briefly present the implementation environment and choices. Finally, we show how the system can be customized.

3.1 Conceptual Architecture

Figure 1 provides the conceptual architecture of the PMRF. The inputs of the system are: the Criteria Table, the Attributes List, the published Web services repository, the user request and its corresponding Ontologies. The Attributes List is a criteria table without similarity measures. It is used during the partially parameterized matching. The weights of similarity degrees and order functions are computed by the PMRF. The weights are used to compute the scores of Web services and the order functions are employed in the rule-based ranking (see (Gmati et al., 2014)). The output of the PMRF is a ranked list of Web services.

The PMRF is composed of two layers. The role of the first layer is to parse the input data and parameters and then transfer it to the second layer, which represents the matching and ranking engine. The Matching Module filters Web service offers that match with the Criteria Table/List. The result is then passed to the Ranking Module. This module produces a ranked list of Web services. The assembler guarantees a coherent interaction between the different modules in the second layer.

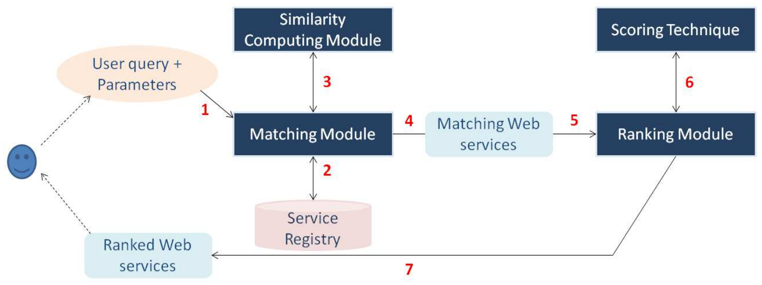


Figure 2: Functional architecture of the PMRF.

The three main components of the second layer are:

- **Matching Module**: This component contains the different matching algorithms: basic, partially parameterized and fully parameterized matching algorithms.
- **Similarity Computing Module**: This component supports the different similarity measure computing approaches: Efficient similarity with MinEdge, Accurate similarity with MinEdge, Accurate similarity with MaxEdge and Accurate similarity with MaxMinEdge.
- **Ranking Module**: This component is the repository of the score computing technique and the different ranking algorithms, namely score-based, rule-based and tree-based ranking algorithms.

3.2 Functional Architecture

The functional architecture of the PMRF is given in Figure 2. It shows graphically the different steps from receiving the user query (specifications of the requested Web service and the different parameters) until the delivery of the final results (ranked list of matching Web services) to the user.

We can distinguish the following main operations:

- The PMRF receives (1) the user query including the specifications of the desired Web service and the required parameters;
- The Matching Module scans (2) the Registry in order to identify the Web services matching the user query;
- During the matching process, the Matching Module uses (3) the Similarity Computing Module to calculate the similarity degrees;
- The Matching Module delivers (4) the Web services matching the user query to the Ranking Module;
- The Ranking Module receives (5) the matching Web services and processes them for ranking;
- During the ranking operation, the Ranking Module uses (6) the Scoring Technique to compute the scores of the Web services;
- The Ranking Module generates a ranked list of Web services, which is then delivered (7) by the PMRF to the user.

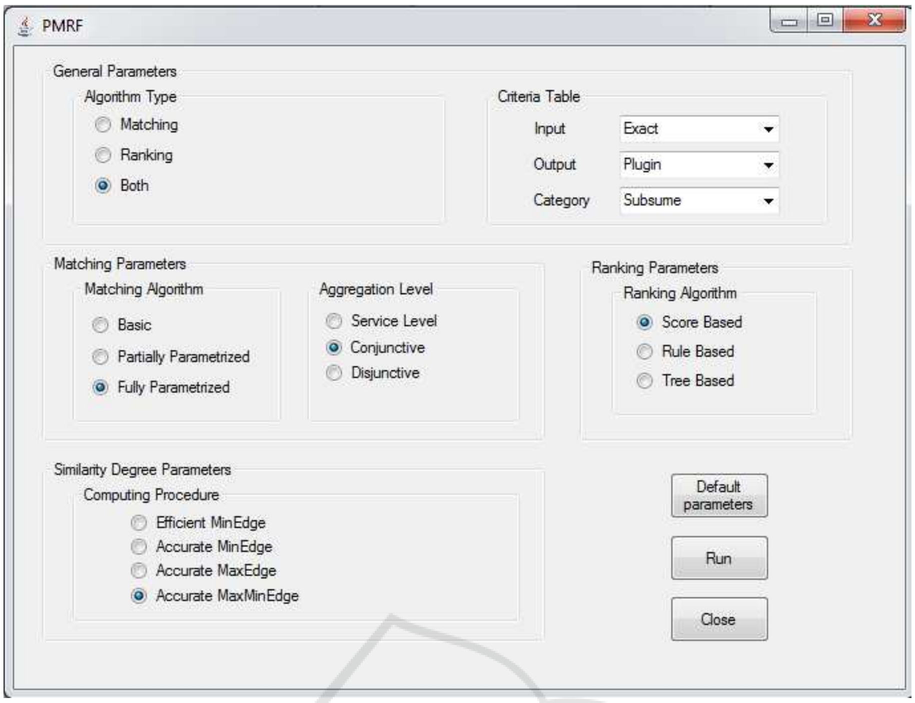
3.3 Implementation

To develop the PMRF, we have used the following tools: (i) Eclipse IDE as the developing platform, (ii) OWLS-API to parse the OWLS service descriptions, and (iii) OWL-API and the Pellet-reasoner to perform the inference for computing the similarity degrees. In order to minimize resources consumption (especially memory), we used the following procedure for implementing the inference operation: (1) A local Ontology is created at the start of the matchmaking process. The incremental classifier class, taken from the Pellet reasoner library, is associated to this Ontology. (2) The service parser based on the OWLs-API retrieves the Uniform Resource Identifier (URI) of the attributes values of each service. The concepts related to these URIs are added incrementally to the local Ontology and the classifier is updated accordingly. (3) In order to infer the semantic relations between concepts, the similarity measure module uses the knowledge base constructed by the incremental classifier.

3.4 System Customization

The parameterizing interface of the PMRF is given in Figure 3. The PMRF permits the user to choose the type of algorithm to use and to specify the criteria table to consider during the matching. The PMRF offers three matching algorithms (basic, partially parameterized and fully parameterized) and three ranking algorithms (score-based, rule-base and tree-based).

In addition, the PMRF supports different aggregation levels: conjunctive-attribute level, disjunctive-attribute level and service level. The attribute-level matching involves capability and property attributes



PMRF

General Parameters

Algorithm Type: Matching, Ranking, [✓ Both]

Criteria Table

Input	Exact
Output	Plugin
Category	Subsume

Matching Parameters

Matching Algorithm: Basic, Partially Parametrized, [✓ Fully Parametrized]

Aggregation Level: Service Level, [✓ Conjunctive], Disjunctive

Ranking Parameters

Ranking Algorithm: [✓ Score Based], Rule Based, Tree Based

Similarity Degree Parameters

Computing Procedure: Efficient MinEdge, Accurate MinEdge, Accurate MaxEdge, [✓ Accurate MaxMinEdge]

Figure 3: Parameterizing Interface.

and consider each matching attribute independently of the others. In this type of matching, the PMRF offers two types of aggregation, namely conjunctive and disjunctive, where the individual (for each attribute) similarity degrees are combined using either AND or OR logical operators. The service-level matching considers capability and property attributes but the matching operation implies attributes both independently and jointly.

The PMRF also allows the user to select the procedure to use for computing the similarity degrees. Four procedures are supported by the system: Efficient similarity with MinEdge, Accurate similarity with MinEdge, Accurate similarity with MaxEdge and Accurate similarity with MaxMinEdge.

4 PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed matching and ranking framework.

4.1 Evaluation Framework

To evaluate the performance of the PMRF, we used the SME2 (Klusch et al., 2010), which is an open source tool for testing different semantic matchmakers in a consistent way. The SME2 uses OWLS-TC collections to provide the matchmakers with Web service descriptions, and to compare their answers to the relevance sets of the various queries. The SME2 provides several metrics to evaluate the performance and effectiveness of a Web service matchmaker. The metrics that have been considered in this paper are: precision and recall, average precision, query response time and memory consumption. The definition of these metrics are given in (Klusch et al., 2010).

Experimentations have been conducted on a Dell Inspiron 15 3735 Laptop with an Intel Core i5 processor (1.6 GHz) and 2 GB of memory. The test collection used is OWLS-TC4, which consists of 1083 Web service offers described in OWL-S 1.1 and 42 queries.

4.2 Performance Evaluation Analysis

To study the performance of the different modules supported by the PMRF, we implemented seven plugins (see Table 2) to be used with the SME2 tool. Each of these plugins represents a different combination of the matching, similarity computing and ranking algorithms.

4.2.1 Comparison of Configurations 1 and 2

The difference between configurations 1 and 2 is the similarity measure module instance: configuration 1 employs the **Accurate MinEdge** instance while the second employs the **Efficient MinEdge** instance.

Table 2: Evaluated Configurations.

Config.	Similarity Measure	Matching	Ranking
1	Accurate MinEdge	Basic	Basic
2	Efficient MinEdge	Basic	Basic
3	Accurate MaxEdge	Basic	Basic
4	Accurate MinEdge	Fully Parameterized	Basic
5	Accurate MaxMinEdge	Basic	RankMinMax
6	Accurate MinEdge	Basic	Rule Based
7	Efficient MinEdge	Basic	Rule Based

Figure 4 shows the Average Precision and Figure 5 illustrates the Recall/Precision plot of configurations 1 and 2. We can see that configuration 1 outperform configuration 2 for these two metrics. This is due to the use of logical inference, that obviously enhances the precision of the first configuration.

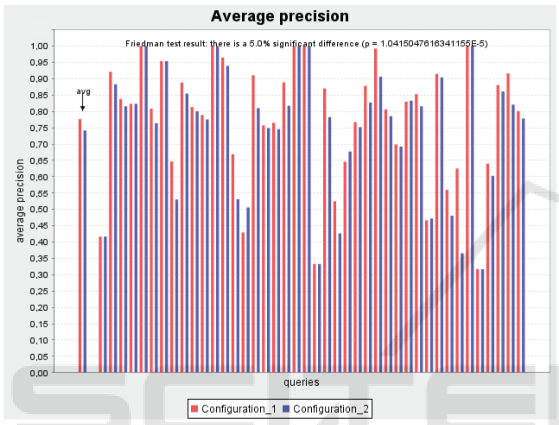


Figure 4: Config. 1 *vs* Config. 2: Average Precision.

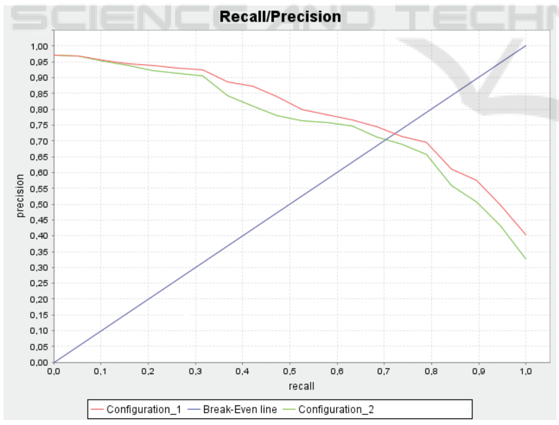


Figure 5: Config. 1 *vs* Config. 2: Recall/Precision.

In Figure 6, however, configuration 2 is shown to be remarkably faster than configuration 1. This is due to the inference process used in configuration 1 that consumes considerable resources.

4.2.2 Comparison of Configurations 1 and 4

The configurations 1 and 4 use different matching module instances. The first configuration is based on

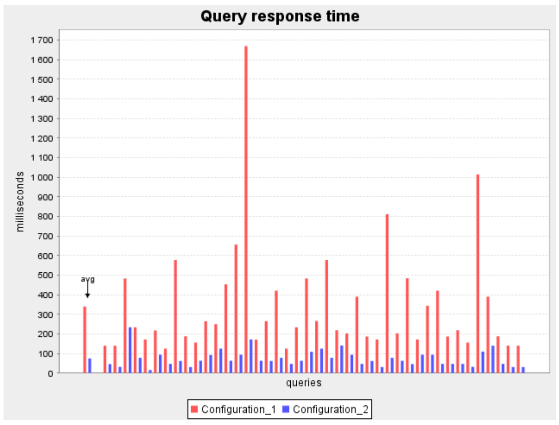


Figure 6: Config. 1 *vs* Config. 2: Query Response Time.

the basic matching algorithm while the second uses the fully parameterized matching. Figure 7 shows the Average Precision metric results. It is easy to see that configuration 4 outperforms configuration 1. This is due to the fact that the Criteria Table restricts the results to the most relevant Web services, which will have the best ranking leading to a higher Average Precision. Figure 8 illustrates the Recall/Precision plot. It shows that configuration 4 has a low recall rate. The overly restrictive Criteria Table explains these results, since it fails to return some relevant services.

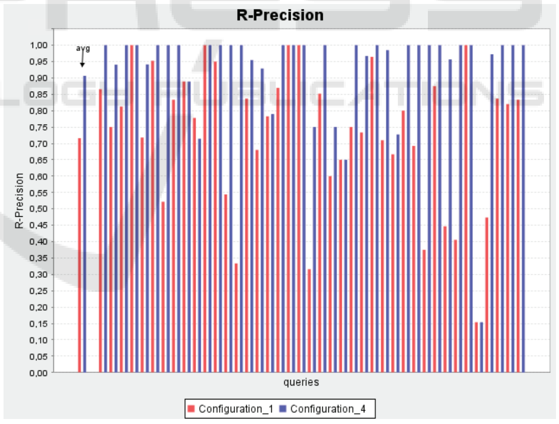


Figure 7: Config. 1 *vs* Config. 4: Average Precision.

4.2.3 Comparison of Configurations 5 and 6

The difference between configurations 5 and 6 is the ranking module instance and the similarity measure computing procedure. The first uses the tree-based ranking algorithm while the second employs the rule-based ranking algorithm. Figure 9 shows that configuration 5 has a slightly better Average Precision than configuration 6 while Figure 10 shows that configuration 6 is obviously faster than configuration 5.

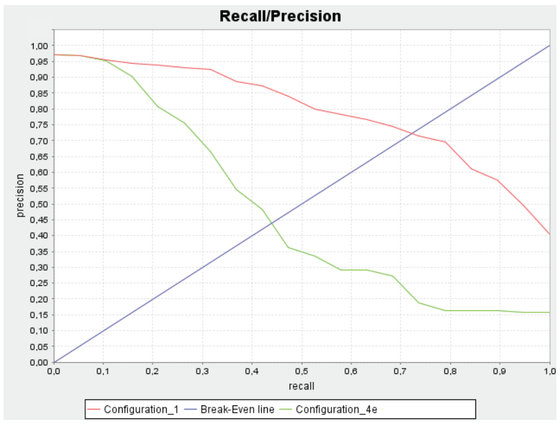


Figure 8: Config. 1 *vs* Config. 4: Recall/Precision.

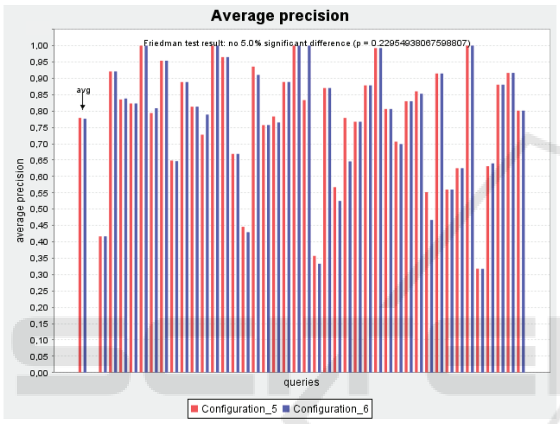


Figure 9: Config. 5 *vs* Config. 6: Average Precision.

5 COMPARATIVE STUDY

We compared the results of the PMRF matchmaker with SPARQLent (Sbodio et al., 2010) and iSeM (Klusch and Kapahnke, 2012) frameworks. Configuration 7 was chosen to perform this comparison. The SPARQLent is a logic-based matchmaker based on the OWL-DL reasoner Pellet to provide exact and relaxed Web services matchmaking. The iSeM is an hybrid matchmaker offering different filter matchings: logic-based, approximate reasoning based on logical concept abduction for matching Inputs and Outputs. We considered only the I-O logic-based in this comparative study. We note that SPARQLent and iSeM consider preconditions and effects of Web services, which are not considered in our work.

5.1 Average Precision

The Average Precision is shown in Figure 11. This figure shows that the PMRF has a more accurate Average Precision than iSeM logic-based and SPARQLent, leading to a better ranking precision than the two other frameworks. In addition, the generated ranking is more fine-grained than SPARQLent and iSeM. This is due to the score-based ranking that gives a more coarse evaluation than a degree aggregation. Indeed, SPARQLent and iSeM approaches adopt a subsumption-based ranking strategy as described in (Paolucci et al., 2002), which gives equal weights to all similarity degrees.

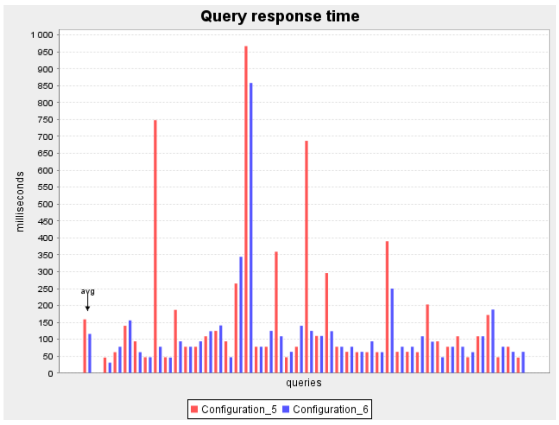


Figure 10: Config. 5 *vs* Config. 6: Query Response Time.

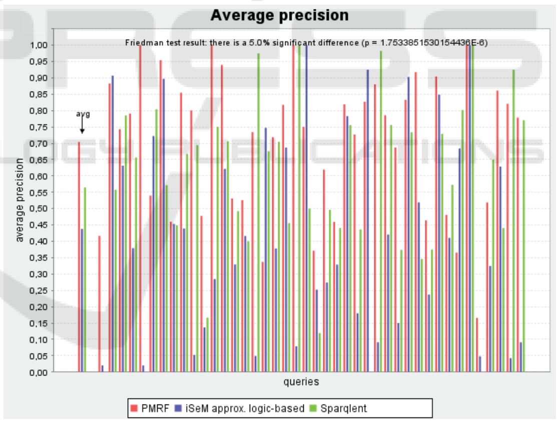


Figure 11: Comparative study: Average Precision.

5.2 Recall/Precision

Figure 12 presents the Recall/Precision of the PMRF, iSeM logic-based and SPARQLent. This figure shows that PMRF recall is significantly better than both iSeM logic-based and SPARQLent. This means that our approach is able to reduce the amount of false positives (see (Bellur and Kulkarni, 2007) for a discussion on the false positives problem).

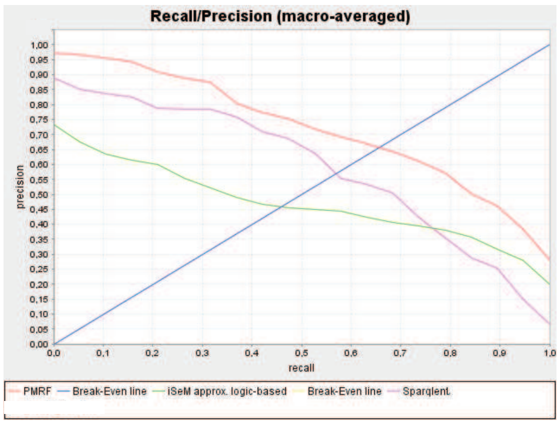


Figure 12: Comparative study: Recall/Precision.

5.3 Query Response Time

The comparison of the Query Response Time of the PMRF, logic-based iSeM and SPARQLent is shown in Figure 13. The first column (Avg) gives the average response time for the three matchmakers. The experimental results show that the PMRF is faster than SPARQLent (760ms for SPARQLent versus 128ms for PMRF) and slightly less faster than logic-based iSeM (65ms for iSeM). We note that SPARQLent has especially high query response time if the query include preconditions/effects. The SPARQLent is also based on an OWL DL reasoner, which is an expensive processing. PMRF and iSeM have close query response time because both consider direct parent/child relations in a subsumption graph, which reduces significantly the query processing. The PMRF highest query response time limit is 248ms.

5.4 Memory Usage

Figure 14 shows the Memory Usage for PMRF, iSeM logic-based and SPARQLent. It is easy to see that

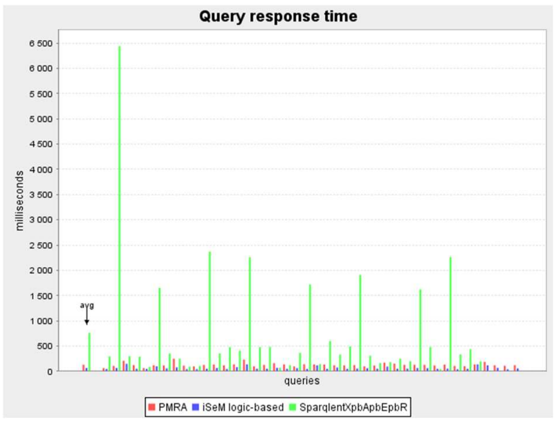


Figure 13: Comparative study: Query Response Time.

PMRF consumes less memory than iSeM logic-based and SPARQLent. This can be explained by the fact that the PMRF does not require a reasoner (in the case of Configuration 7) neither a SPARQL queries in order to compute similarities between concepts. We note, however, that the memory usage of the PMRF increases monotonically in contrast to SPARQLent. In the long-term operation of the proposed system, its memory consumption might be equivalent to or more than SPARQLent.

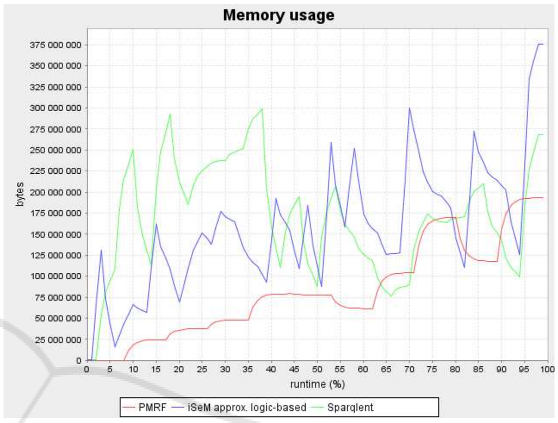


Figure 14: Comparative study: Memory Usage.

6 DISCUSSION

In this section, we discuss the user/provider acceptability of the proposed customizations. Indeed, one important characteristic of the proposed framework is its configurability by allowing the user to specify a set of parameters and apply different algorithms supporting different levels of customization. This, however, leads to the problem of user/provider acceptability and ability to specify the required parameters, especially the criteria Table. Indeed, the specification of these parameters may require some cognitive effort from the user/provider.

A possible solution to reduce this effort is to use a predefined Criteria Table. This solution can be further enhanced by including in the framework some appropriate Artificial Intelligence techniques to learn from the previous choices of the user.

Another possible solution to reduce the cognitive effort consists in exploiting the context of the user queries. First, the description of elementary services can be textually analysed and based on the query domain, the system uses either the efficient or the accurate configurations. Second, a global time limit to the matchmaking process can be used to orient the system towards the use of the accurate version or efficient version of the similarity measure computing algorithm. Third, the context of the query in the workflow can be used to determine the level of customization needed and also in the generation of a suitable Criteria Table or Attributes List.

A more advanced solution consists in combining all the idea cited above.

7 RELATED WORK

Several matchmaking frameworks have been proposed in the literature. However, most of these frameworks present at least one of the four shortcomings cited in the introduction: use of strict syntactic matching, use of capability-based matchmaking, lack of customisation support and lack of accurate ranking of matching Web services. In what follows, we discuss each of these shortcomings.

Strict Syntactic Matching. The first and traditional matchmaking frameworks, such as Jini (Arnold et al., 1999), Konark (Lee et al., 2003) and Salutation (Miller and Pascoe, 2000), are based on strict syntactic matching. Such syntactic matching approaches only perform service discovery and service matching based on particular interface or keyword queries from the user, which generally leads to low recall and low precision of the retrieved services (Lv et al., 2009).

In order to overcome the limitation of strict syntactic matching, some advanced techniques and algorithms (e.g., genetic algorithmic as in (Ludwig, 2011), utility function as in (Wang et al., 2009)(Yu and Lin, 2004)) have been used. Additionally, many authors propose to include the concept of semantics as in (Bellur and Kulkarni, 2007)(Ben Mokhtar et al., 2006)(Fu et al., 2009)(Goncalves et al., 2010)(Guo et al., 2005)(Klusch and Kapahnke, 2012)(Li and Horrocks, 2003)(Paolucci et al., 2002)(Sbodio et al., 2010)(Sycara et al., 2003) to deal with the limitation of strict syntactic matching. The use of ontology eliminates the limitations caused by syntactic difference between terms since matching is now possible on the basis of concepts of ontologies used to describe input and output terms (Bellur et al., 2008).

Capability-based Matchmaking. Most of existing matchmaking frameworks such as (Ben Mokhtar et al., 2006)(Goncalves et al., 2010)(Guo et al., 2005)(Li and Horrocks, 2003)(Paolucci et al., 2002)(Sycara et al., 2003) utilize a strict capability-based matchmaking, which is proven (Doshi et al., 2004) to be inadequate in practice. Some recent proposals including (Ben Mokhtar et al., 2006)(Guo et al., 2005) propose to use semantics to enhance the matchmaking process but most of them still consider capability attributes only.

The author in (Chakhar, 2013) distinguishes three types of service attributes (i) capability attributes that directly relate to working of the service, (ii) quality attributes related to the service quality and property attributes including all attributes other than those included in service capability or service quality. The authors in (Chakhar et al., 2014) extend the works of (Doshi et al., 2004) and (Chakhar, 2013) and propose different matchmaking algorithms devoted to different types of attributes (capability, property and service quality).

Lack of Customisation Support. An important shortcoming of most of existing Web service matchmaking frameworks is the lack of customisation support. To deal with this shortcoming, some authors allow the user to specify some parameters. For instance, the authors in (Doshi et al., 2004) present a parameterized semantic matchmaking framework that exhibits a customizable matchmaking behavior. One important shortcoming of (Doshi et al., 2004) is that the sufficiency condition defined by the authors is very strict since it requires that all the specified conditions hold at the same time. This seems to be very restrictive in practice, especially for attributes related to the service quality.

Recently, (Chakhar et al., 2014) extend the work of (Chakhar, 2013) and propose a series of algorithms for the different types of matching. These algorithms are designed to support a customizable matching process that permits the user to control the matched attributes, the order in which attributes are compared, as well as the way the sufficiency is computed for all matching types.

Lack of Accurate Ranking of Matching Web Services. Although the semantic matchmaking (Paolucci et al., 2002)(Doshi et al., 2004)(Bellur and Kulkarni, 2007)(Fu et al., 2009)(Chakhar, 2013)(Chakhar et al., 2014)(Chakhar et al., 2015) permits to avoid the problem of simple syntactic and strict capability-based matchmaking, it is not very suitable for efficient Web service selection. This is because it is difficult to distinguish between a pool of similar Web services (Rong et al., 2009). Indeed, since we have a limited number of similarity degrees, semantic matchmaking frameworks will most often face the problem of ties when several Web services have the same similarity degree.

A possible solution to this issue is to use some appropriate techniques and some additional informa-

Table 3: Comparison of Matchmaking Frameworks.

Matchmaker	Matching	Attributes	Customization Type	Ranking	Description Language
Jini(Arnold et al., 1999)	Syntactic	Capability	No	No	No
Konark(Lee et al., 2003)	Syntactic	Capability	No	No	XML
Salutation(Miller and Pascoe, 2000)	Logical	Capability	No	Yes	OWL-S
MatchMaker (Sycara et al., 2003)	Syntactic	Capability	No	No	DAMS, UDDI
RACER(Li and Horrocks, 2003)	Syntactic	Capability	No	No	DAML-S
PSMF(Doshi et al., 2004)	Logical	Capability	Yes	No	DAML-S, WSDL, UDDI
SPARQLent(Sbodio et al., 2010)	Logical	Capability	No	Yes	OWL-S
iSeM-logi-based(Klusch and Kapahnke, 2012)	Logical	Capability	No	Yes	OWL-S, SAWSDL
QoSeBroker(Chakhar et al., 2014)(Chakhar et al., 2015)	Logical	Capability, Property, Service quality	Yes	No	OWL-S
PMRF	Logical	Capability, Property	Yes	Yes	OWL-S

tion to rank the Web services delivered by the semantic matching algorithm and then provide a manageable set of ‘best’ Web services to the user from which s/he can select one Web service to deploy. Several approaches have been proposed to implement this idea (Manikrao and Prabhakar, 2005)(Maamar et al., 2005)(Kuck and Gnasa, 2007).

Table 3 summarizes the main characteristics of the above cited frameworks. As shown in this table, the discussed frameworks fail to jointly take into account the shortcomings of Web services matchmaking enumerated in the introduction. The proposed system PMRF uses semantic matchmaking, accepts capability and property attributes, support different levels of customization and generates a ranked list of matching Web services. It can be easily extended, based on our previous work (Chakhar et al., 2014)(Chakhar et al., 2015), to support attributes related to the quality of service.

8 CONCLUSION

In this paper, we presented a highly customizable framework, called PMRF, for matching and ranking Web services. We briefly reviewed the matching and ranking algorithms supported by the PMRF, provided its architecture and discussed some implementation issues. We also presented the results of the performance evaluation of the PMRF using the OWLS-TC4 datasets and compared it to iSeM-logic-based and SPARQLent. The results show that the algorithms supported by PMRF behave globally well in comparison to iSeM-logic-based and SPARQLent frameworks.

In the future, we intend to enhance PMRF by (i) including other matching techniques, namely textual matching and Ontology distance calculation; (ii) adapt it to a dynamic Web service environment, (iii) use AI techniques to reduce the cognitive effort required from user/provider; and (iv) make the PMRF useable over the cloud technology.

REFERENCES

Arnold, K., O’Sullivan, B., Scheifler, R., Waldo, J., and Woolrath, A. (1999). *The Jini Specification*. Addison-Wesley, Reading, MA.

Bellur, U. and Kulkarni, R. (2007). Improved matchmaking algorithm for semantic Web services based on bipartite graph matching. In *IEEE Inter. Conf. on Web Services*, pages 86–93, Salt Lake City, Utah, USA.

Bellur, U., Vadodaria, H., and Gupta, A. (2008). Semantic matchmaking algorithms. In Bednorz, W., editor, *Advances in Greedy Algorithms*, pages 481–502. SIn-Tech, Vienna, Austria.

Ben Mokhtar, S., Kaul, A., Georgantas, N., and Issarny, V. (2006). Efficient semantic service discovery in pervasive computing environments. In *ACM/IFIP/USENIX 2006 Inter. Conf. on Middleware*, pages 240–259, Melbourne, Australia.

Chakhar, S. (2013). Parameterized attribute and service levels semantic matchmaking framework for service composition. In *Fifth International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA 2013)*, pages 159–165, Seville, Spain.

Chakhar, S., Ishizaka, A., and Labib, A. (2014). QoS-aware parameterized semantic matchmaking framework for web service composition. In Monfort, V. and Krempels, K.-H., editors, *Proceedings of the 10th international conference on web information systems and technologies (WEBIST 2014), volume 1, Barcelona, Spain, April 3-5, 2014:*, volume 1, pages 50–61. SciTePress.

Chakhar, S., Ishizaka, A., and Labib, A. (2015). Semantic matching-based selection and QoS-aware classification of web services. In Monfort, V. and Krempels, K.-H., editors, *Proceedings of the 10th international conference, WEBIST 2014, Barcelona, Spain, 3-5 April, 2014, Revised Selected Papers:*, Lecture Notes in Business Information Processing, pages 96–112. Springer, Switzerland.

Doshi, P., Goodwin, R., Akkiraju, R., and Roeder, S. (2004). Parameterized semantic matchmaking for workflow composition. IBM Research Report RC23133, IBM Research Division.

Fu, P., Liu, S., Yang, H., and Gu, L. (2009). Matching algorithm of Web services based on semantic distance. In *International Workshop on Information Security and Application (IWISA 2009)*, pages 465–468, Qingdao, China.

Gmati, F.-E., Yacoubi Ayadi, N., Bahri, A., Chakhar, S., and Ishizaka, A. (2015). A tree-based algorithm for ranking web services. In Monfort, V. and Krempels, K.-H., editors, *The 11th International Conference on Web Information Systems and Technologies (WEBIST 2015), Lisbon, Portugal, May 20-22:*, pages 170–178. SciTePress.

Gmati, F.-E., Yacoubi Ayadi, N., and Chakhar, S. (2014). Parameterized algorithms for matching and ranking web services. In Meersman, R., Panetto, H., Dillon, T., Missikoff, M., Liu, L., Pastor, O., Cuzzocrea, A., and Sellis, T., editors, *On the move to meaningful internet systems, OTM 2014 conferences:confederated international conferences: CoopIS, and ODBASE 2014, Amantea, Italy, October 27-31, 2014, proceedings*, Lecture Notes in Computer Science, pages 784–791. Springer, Berlin Heidelberg.

Goncalves, M., Vidal, M.-E., Regalado, A., and Yacoubi-Ayadi, N. (2010). Efficiently selecting the best web services. In *The Second International Workshop on Resource Discovery RED 2009, Lyon, France, August 28, 2009. Revised Papers*, volume 6162 of *Lecture Notes in Computer Science*, pages 120–139. Springer.

Guo, R., Le, J., and Xiao, X. (2005). Capability matching of Web services based on OWL-S. In *Sixteenth Inter. Workshop on Database and Expert Systems Applications*, pages 653–657.

Klusch, M., Dudev, M., Misutka, J., Kapahnke, P., and Vasileski, M. (2010). *SME^2 Version 2.2. User Manual*. The German Research Center for Artificial Intelligence (DFKI), Germany.

Klusch, M. and Kapahnke, P. (2012). The iSeM matchmaker: A flexible approach for adaptive hybrid semantic service selection. *Web Semantics: Science, Services and Agents on the World Wide Web*, 15:1–14.

Kuck, J. and Gnasa, M. (2007). Context-sensitive service discovery meets information retrieval. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom'07)*, pages 601–605.

Lee, C., Helal, A., Desai, N., Verma, V., and Arslan, B. (2003). Konark: A system and protocols for device independent, peer-to-peer discovery and delivery of mobile services. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 33(6):682–696.

Li, L. and Horrocks, I. (2003). A software framework for matchmaking based on semantic web technology. In *The 12th Inter. Conf. on World Wide Web*, WWW '03, pages 331–339, New York, NY, USA. ACM.

Ludwig, S. (2011). Memetic algorithm for Web service selection. In *The 3rd Workshop on Biologically Inspired Algorithms for Distributed Systems*, BADS '11, pages 1–8, New York, NY, USA. ACM.

Lv, Q., Zhou, J., and Cao, Q. (2009). Service matching mechanisms in pervasive computing environments. In *Intelligent Systems and Applications, 2009. ISA 2009. Inter. Workshop on*, pages 1–4.

Maamar, Z., Mostefaoui, S., and Mahmoud, Q. (2005). Context for personalized Web services. In *In Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, pages 166b–166b.

Manikrao, U. and Prabhakar, T. (2005). Dynamic selection of Web services with recommendation system. In *Proceedings of the International Conference on Next Generation Web Services Practices (NWeSP 2005)*, pages 117–121.

Miller, B. and Pascoe, R. (2000). Salutation service discovery in pervasive computing environments,. White paper, IBM Pervasive Computing.

Paolucci, M., Kawamura, T., Payne, T., and Sycara, K. (2002). Semantic matching of web services capabilities. In *The First Inter. Semantic Web Conf. on The Semantic Web*, ISWC '02, pages 333–347, London, UK. Springer-Verlag.

Rong, W., Liu, K., and Liang, L. (2009). Personalized Web service ranking via user group combining association rule. In *IEEE International Conference on Web Services (ICWS 20090*, pages 445–452.

Samper Zapater, J., Llido Escriva, D., Soriano Garcia, F., and Martinez Dura, J. (2015). Semantic Web service discovery system for road traffic information services. *Expert Systems with Applications*, 42(8):3833–3842.

Sbodio, M., Martin, D., and Moulin, C. (2010). Discovering semantic Web services using SPARQL and intelligent agents. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4):310–328.

Sharma, S., Lather, J., and Dave, M. (2015). Google based hybrid approach for discovering services. In *IEEE Inter. Conf. on Semantic Computing (ICSC 2015)*, pages 498–502, Anaheim, CA. IEEE.

Sycara, K., Paolucci, M., van Velsen, M., and Giampapa, J. (2003). The retsina mas infrastructure. *Autonomous Agents and Multi-Agent Systems*, 7(1-2):29–48.

Wang, R., Chi, C.-H., and Deng, J. (2009). A fast heuristic algorithm for the composite Web service selection. In *The Joint Inter. Conf. on Advances in Data and Web Management*, APWeb/WAIM '09, pages 506–518, Berlin, Heidelberg. Springer-Verlag.

Yu, T. and Lin, K.-J. (2004). Service selection algorithms for Web services with end-to-end QoS constraints. In *The IEEE Inter. Conf. on e-Commerce Technology (CEC 2004)*, pages 129–136.