ISSN: 2229-6956(ONLINE)
DOI: 10.21917/ijsc.2012.0059

A SURVEY ON OPTIMIZATION APPROACHES TO SEMANTIC SERVICE DISCOVERY TOWARDS AN INTEGRATED SOLUTION

Chellammal Surianarayanan[1] and Gopinath Ganapathy[2]
Department of Computer Science and Engineering, Bharathidasan University, India
E-mail: [1]chelsrsd@rediffmail.com and [2]gganapathy@gmail.com

Abstract

The process of semantic service discovery using an ontology reasoner such as Pellet is time consuming. This restricts the usage of web services in real time applications having dynamic composition requirements. As performance of semantic service discovery is crucial in service composition, it should be optimized. Various optimization methods are being proposed to improve the performance of semantic discovery. In this work, we investigate the existing optimization methods and broadly classify optimization mechanisms into two categories, namely optimization by efficient reasoning and optimization by efficient matching. Optimization by efficient matching is further classified into subcategories such as optimization by clustering, optimization by inverted indexing, optimization by caching, optimization by hybrid methods, optimization by efficient data structures and optimization by efficient matching algorithms. With a detailed study of different methods, an integrated optimization infrastructure along with matching method has been proposed to improve the performance of semantic matching component. To achieve better optimization the proposed method integrates the effects of caching, clustering and indexing. Theoretical aspects of performance evaluation of the proposed method are discussed.

Keywords:

Optimization Infrastructure, Service Cache, Service Cluster, Service Index, Semantic Service Discovery

1. INTRODUCTION

Regardless of the existence of huge number of web services in the Web, in many scenarios such as Travel Plan, a single service is not sufficient to fulfill the required functionality and more than one atomic service from different organizations are combined as a composite service in a specific pattern to achieve the given goal. In this example, services from different heterogeneous domains, namely, Flight Service, Hotel Service and Cab Service are combined to produce the given business goal. Web service composition is a complex task due to existence of huge number of services, usage of different service description languages by different service providers, and constant update of service interfaces. With above difficulties, performing service composition task manually is impractical and automatic service composition becomes crucial to meet current business integration needs. In any automatic web service composition facility, a machine i.e. a program or software will analyze the given user's query, search the individual services required and compose them in a specific pattern to produce a specific functionality. This means that a service should be described in such a way so that a programme can interact with it automatically without any human intervention. In order for making web services as machine processable entities, they are being described using ontology languages such as Web Ontology Language for Services (OWL-S) [1] and Semantically Annotated Web Service Description Language (SAWSDL) [2]. Using semantic service description languages, the intended semantics of the services are expressed in ontologies (formal specification) which can be processed by software/machine.

With the advent of semantic service description and discovery, maximal automation and dynamism with sufficient accuracy have been brought into service discovery [3]. Despite the accuracy, as presently any semantic service discovery method employs a description logic based reasoning tool, the response time i.e. the time taken to find a set of matched services for a given query is significantly large. Such a large response time can even prohibit the usage of services in certain scenarios where the time involved in discovery and composition should be very short. A significant number of optimization approaches are being proposed to improve the performance of semantic service discovery. In this work, we investigate various approaches for performance improvement and classify the approaches. We propose an integrated optimization infrastructure along with an efficient matching method for semantic service discovery.

The paper is organized as follows. In section 1, we describe the need for optimization of semantic service discovery and composition. In section 2 we present a brief overview of semantic service discovery. In section 3 we review various optimization methods and classify them. In section 4, we propose an improved optimization infrastructure and matching method. In section 5, we discuss the theoretical evaluation of the proposed approach. Conclusion is given in section 6.

2. OVERVIEW OF SEMANTIC SERVICE DISCOVERY

It is known that automatic web service discovery becomes feasible only when web services are described and discovered semantically. The semantic languages used for web service description include OWL-S and Web Service Modeling Language (WSML). As these description languages are based on standard Web Ontology Language – Description Logic (OWL-DL), ontology reasoner (or Description Logic reasoner) becomes an indispensable tool for performing semantic reasoning during service matching. Prominent examples of description logic reasoners that are used for web service discovery include Pellet, RACER and Fact++.

A typical semantic service discovery infrastructure is as shown in Fig.1.

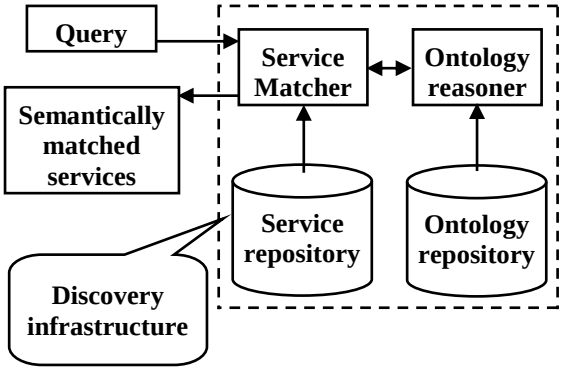

Fig.1. Semantic service discovery infrastructure

In Fig.1, the service matcher refers to a matching algorithm which finds matched services for a given query by matching the query with each published service in the service repository. While matching, the algorithm finds four types of standard semantic relationships namely, exact, plug-in, subsumes and fail as proposed in [4] that could exist between the concepts of query and that of published services with the help of ontology reasoner. Based on the semantic relationships between the query and the published service, the algorithm assigns a particular value for similarity score of the match and decides whether a service is a matched service or not. The ontology reasoner is the main component of the infrastructure which interacts with the ontologies and infers various subsumption relations, concept satisfiability, equivalence and disjointness [5]. The semantic service description consists of two files namely services files (such as .owls files) and ontology files (.owl files). The service file contains information about the service capabilities (mainly functional characteristics). The semantics of service capabilities are expressed as concepts in ontology files. The ontology files are stored in the ontology repository and service files are stored in service repository.

When a query is submitted to the service matcher, the matcher parses the query first and then performs the following tasks,

- Parses a service from the service repository
- Finds and loads the necessary ontologies to the semantic reasoner
- Finds the subsumption relationships between the query and service from repository with the help of ontology reasoner, by employing the logic filters and
- Discovers whether a service from repository is semantically similar or not to the query.

The algorithm repeats above tasks for each service from service repository and returns a set of semantically matched services.

From literature it is understood that with 'N' number of services in the repository, the time involved in semantic discovery can be classified into four categories as follows,

- Time taken to parse the query and parse *N* service descriptions
- Time taken to find and load the necessary ontologies of services into the ontology reasoner and produce the classified ontology
- Time taken by the reasoner to find the subsumption relations among various concepts from the classified ontology
- Time taken by the semantic matcher to find the semantic relations between query and all published services, 'N' in the repository.

It is understood from previous research works such as [6]–[8] that loading and classifying ontologies is major time consuming task which leads to very high response time of typically 4 to 5 seconds for performing a single semantic match between N = 1 number of services and query having typically ten concepts.

3. CLASSIFICATION OF OPTIMIZATION METHODS

A number of optimization methods are being proposed to reduce the response time of semantic service discovery. In the semantic web service discovery infrastructure, the optimization can be done either in semantic reasoner component or in semantic matcher component or in both. With a detailed survey, the optimization approaches to performance of semantic service discovery can be classified into two broad categories based on to which component the optimization is employed. They are optimization by efficient reasoning and optimization by efficient matching.

3.1 OPTIMIZATION BY EFFICIENT REASONING

The primary objective of this method is to optimize the semantic reasoner component so that the time involved in loading and classifying ontologies is reduced. Pre-classifying the ontologies for anticipated user queries and numerical encoding of concepts are the two prominent methods used to optimize the performance of a reasoner.

To implement optimization at reasoning component, one should work with the internal architecture of reasoner. As the method depends on the internal architecture of reasoner, the optimization methodology may not be arrived as common solution and it may vary from reasoner to reasoner. Because of this reason, optimization at matcher is preferred to optimization at reasoner.

3.2 OPTIMIZATION BY EFFICIENT MATCHING

The primary objective of this method to optimize the service matcher by filtering out the services which are obviously irrelevant to a given query so that the semantic service discovery will be employed only to those services which really need complete semantic matching of service capabilities. With this kind of optimization, the number of services that undergo ontology reasoning is significantly reduced.

Optimization by efficient matching is further sub categorized as follows,

- Optimization by clustering
- Optimization by inverted indexing
- Optimization by caching
- Optimization by hybrid approaches
- Optimization by efficient data structures
- Optimization by efficient matching algorithms

3.2.1 Optimization by Clustering:

This approach initially organizes all the available published services into groups of similar services using a clustering algorithm. Each cluster is associated with a cluster centre which is the representative of all the services in that cluster. Once the clusters are formed, when a query is submitted, during service matching, firstly, the cluster which is relevant to the query is found out by computing the distance between the query and each cluster centre. Now it is sufficient to perform complete semantic matching of service capabilities only to those services which belong to the relevant cluster. Hence, for a given query, the search space is reduced from an entire pool of services to a specific cluster. With clustering approach, once the services are clustered, then the time taken to find matched services for a given query will be equal to the time involved in finding the relevant cluster to which the query belongs to and time involved in matching the services present in that relevant cluster. This improves the performance of semantic matcher greatly. Significant number of research works is being carried out in this direction.

A Single-Linkage based clustering method proposed in [9] clusters the services based on a dissimilarity measure computed from service textual description, OWL-S profile elements and Web Service Description Language (WSDL) elements. A prototype semantic service retrieval engine along with an efficient clustering algorithm for organizing the services returned by the engine has been proposed in [10]–[11]. An output parameter based clustering algorithm has been proposed in [12] to assist semantic discovery. A neural network based learning and clustering algorithm has been proposed for automatic classification of web services in [13] to improve the efficiency of service discovery. A number of research efforts such as [14]–[16] exploit the method of clustering in filtering out irrelevant services during semantic service discovery.

3.2.2 Optimization by Inverted Indexing:

This method basically uses an index (a sorted list) of output parameters having links to services containing that output parameter. The output parameter acts as the key of the inverted index. The matcher matches the output parameter of a given query with each output parameter in the inverted index to find any semantic relations exist between them. When it finds a semantic match, it means that all the services which are linked by that output parameter are semantically related to the query. In this way, the indexing helps to filter out services which are irrelevant to the output of query and makes the semantic discovery efficient.

Inverted indexing based approach proposed in [17] and an inverted indexing based Semantic Web Service Discovery search Model (SWSDSM) proposed in [18] produce better performance than sequential matching. Experiments with composition-oriented service discovery algorithm using indexing technique proposed in [19] prove that the response time using the indexing is always less than that of the sequential composition-oriented discovery. The performance of service discovery using indexing is a promising way to improve the efficiency of service discovery.

3.2.3 Optimization by Caching:

Caching technique is employed for performance optimization in several areas of computing. A Semantic Discovery Caching (SDC) which captures the knowledge on the functional usability of all available services and exploits this for matched services of queries has been proposed in [20].

3.2.4 Optimization by Hybrid Methods:

This approach uses hybrid techniques which combine non-logic methods such as syntactic, graph and Information Retrieval (IR) approaches with description logic based matching instead of purely logic based semantic matching techniques. The syntactic and IR methods are helpful to remove services which are irrelevant to a given query quickly without employing semantic reasoning for them.

WSMO-MX [21] is a hybrid matchmaker for WSML oriented services. It combines F-Logic reasoning with syntactic similarity where each of them alone would fail. The experimental results with WSMO-MX showed that the method outperforms pure logic based matching in terms of computation time. Another hybrid matcher OWLS-MX [22] performs semantic service matching with five different filters, exact, plug-in, subsumes, subsumed-by and nearest-neighbor of which the last two are hybrid methods. The last two methods employ IR similarity along with a user defined threshold. Experimental results with OWLS-MX show that under certain constraints, the matcher outperforms the purely logic based matching in terms of performance and scalability. SAWSDL-MX [23] is another hybrid matchmaker for SAWSDL oriented services, which combines pure logic based matching with text retrieval strategies. The experimental results with SAWSDL-MX show that the hybrid method using cosine similarity measure performs the best.

3.2.5 Optimization by Efficient Data Structures:

This method improves the efficiency of searching by storing web services in efficient data structures.

A discovery method proposed in [24] uses tree-form data structure for storing service information in which upper nodes are given more weight than lower nodes. The method stores properties specific to a service called as, Special Properties (SP) in the upper nodes and Common Properties (CP) such as service ID in the lower nodes. With this method, if the upper nodes have been matched in searching, the lower nodes could be omitted and leads to improved efficiency. A novel data structure, *Double Parameter Inverted File* (*DuoParaInvertedFile*) has been proposed in [25] to facilitate the mutual search operations among service operations, input and outputs.

3.2.6 Optimization by efficient matching algorithms:

The primary objective of this method is to improve the performance of match making algorithm in the semantic matching component. In contrast to conventional semantic matching algorithm proposed by Paolucci, et al., [4], various other approaches are being proposed to improve both the accuracy of matching and execution time of matching.

An improved matchmaking algorithm based on bipartite graph matching proposed in [26] offers a much better performance as compared to that of Brute-Force technique. A two-phase semantic service matching method proposed in [27]

improves the performance of semantic service discovery by first finding the functionally matched services in the first phase. In the second phase, the method employs a qualitative filtering to all functionally matched services found in the first phase.

4. PROPOSED APPROACH

An approach consists of an optimization infrastructure along with a corresponding matching method is proposed to improve the performance of semantic matcher. This approach is distinct from existing mechanisms as follows. Existing methods generally employ caching, clustering and indexing as individual mechanisms to improve the performance. Whereas the proposed approach combines the three mechanisms and introduces optimization in three stages using three core elements namely, service cache, service clusters and service indices of the infrastructure.

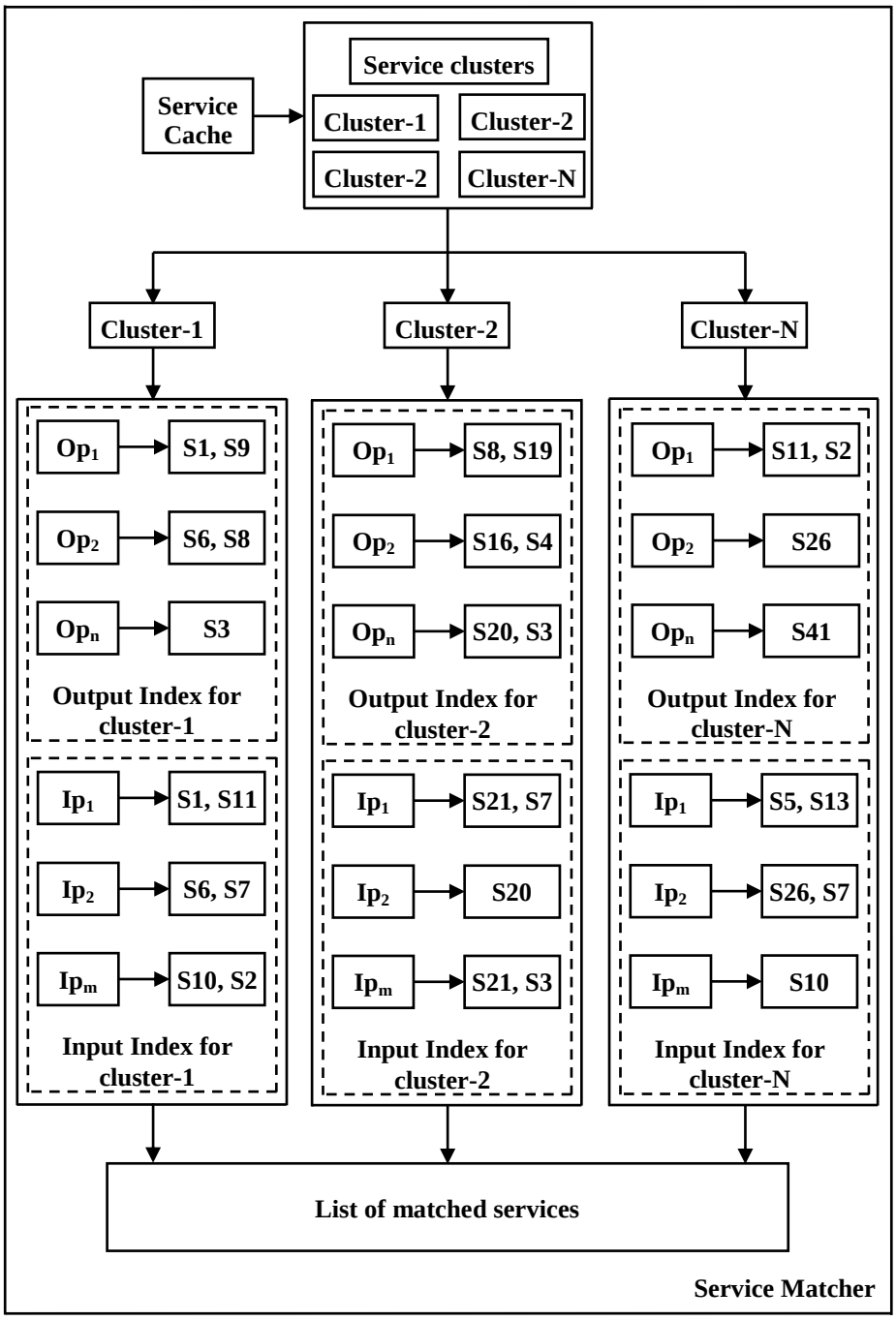

Fig.2. Optimization infrastructure

4.1 OPTIMIZATION INFRASTRUCTURE

The proposed optimization infrastructure is shown in Fig.2.

It consists of three major elements namely, service cache, service clusters and service indices.

Service cache represents a cache which caches all previously occurred queries along with their results.

All the available published services are grouped into clusters of similar services called 'service clusters' by using a standard clustering algorithm based on service similarity measure. Each service cluster has its own characteristic representation which is representative of all the services in that cluster. The service similarity two services s_1 and s_2 can be computed using formula,

$$Sim(s_1,s_2)=w_1 x DeSim(s_1,s_2)+w_2 x(OutSim(s_1,s_2)+InSim(s_1,s_2)) \quad (1)$$

In Eq.(1),$Sim(s_1,s_2)$ represents the semantic similarity between the services, s_1 and s_2, $Desim(s_1, s_2)$ represents the description similarity between the textual description of s_1 and s_2, $OutSim(s_1,s_{2)}$ represents semantic similarity between output parameters of s_1 and s_2 and $InSim(s_1,s_{2)}$ represents semantic similarity between input parameters of s_1 and s_2 and w_1 and w_2 are weight parameters affecting the search performance. The values are set as w_1 = 0.382 and w_2 = 0.618. The weight parameters are taken from [28]. The semantic similarity between services is computed used standard algorithms such as [4]. The semantic similarity among all available published services is computed in a pair-wise manner and a similarity matrix is constructed. Now this similarity matrix will be used as input for clustering algorithm. Hierarchical agglomerative clustering is used to cluster the services. In each agglomerative step a pair of most similar clusters is merged using group-average scheme. Depending on the application, a similarity threshold is specified as stopping condition during clustering. After services are clustered, each cluster is represented by a characteristic representation using the most frequent set of parameters.

After service clusters are constructed, for each service cluster, two indices are created, namely output index and input index. Output parameters are used as keys of output index and input parameters are used as keys of input index.

Each key in the output index is linked to a set of all services which contain that key as one of its output parameters. Similarity, each key in the input index is linked to a set of all services which contain that key as one of its input parameters. These indices are called as service indices. When a query is submitted, the matching component efficiently finds the list of matched services for a given query as explained in the subsequent section.

4.2 MATCHING METHOD

A matching method corresponding to the proposed infrastructure of semantic matcher is described in this sub section. When a query is submitted, the matcher performs the tasks as given in Fig.3.

Step 1: The matcher checks that whether the query has occurred already by checking the service cache, in which case, the matched services will be available in the service cache. It simply fetches the matched services from cache and returns them as semantically matched services. If the query is not available in the service cache, the matcher performs from Step 2 to Step 6.

Step 2: The matcher finds the relevant cluster of the query by finding the service distance between each cluster centre and query. The cluster which has minimum service distance with the query is chosen as the relevant cluster of the query. Let the relevant cluster of the query be Cluster-R

Step 3: It finds the output index of Cluster-R.

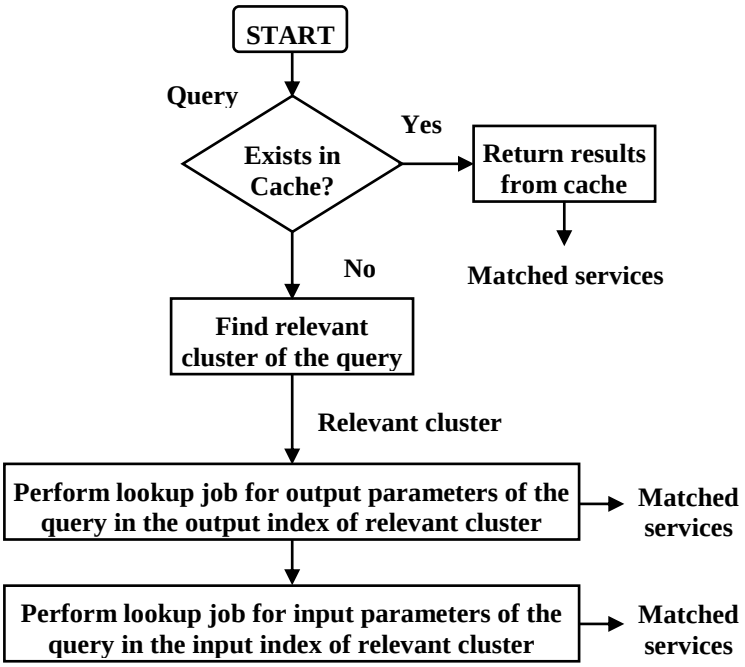

Fig.3. Tasks of matching method

Step 4: It performs semantic matching for a particular output parameter of the query with each key of the output index of Cluster-R. If it finds a match between the output parameter of query and any key of the output

index of Cluster-R, then it retrieves all the services linked by that key and returns them as semantically matched services for the output parameter of the query under consideration.

Step 5: It repeats the step 4 for every output parameter in the query.

Step 6: It performs semantic matching for each input parameter of the query with each key of the input index of Cluster-R. If it finds a match between the input parameter of the query and any key of input index of Cluster-R, then it retrieves all the services linked by that key and returns them as semantically matched services for the input parameter of the query under consideration.

Step 7: It repeats the step 6 for every input parameter in the query.

Step 8: Using suitable intersection operations on different sets of services returned in Step 4 and Step 6, matched services of the given query is computed.

5. DISCUSSION

The proposed method is analyzed for its performance theoretically as follows. When a query is submitted to a matching component, as a first step it checks whether the query has occurred already by checking the service cache, in such case, the matching component simply returns the stored results from service cache as semantically matched services and the time involved in semantic matching process is completely avoided. This means that for already occurred queries, there is no need for repetitive semantic matching for every time the query is raised. Thus by avoiding repetitive matching, the performance is improved.

Secondly, if the query does not exist in the cache, the matching component tries to find a relevant cluster from service clusters to which the query belongs to. Once it identifies a relevant cluster for the given query, the search space of the query is reduced from all available services to a relevant cluster of services. Suppose, the total number of available services is 1000 and it is grouped into say 10 clusters, each cluster with 100 services, then using clustering, the search space is reduced from 1000 services to 100 services. Hence with clustering, the search space is reduced from all available services to a single cluster of services.

Thirdly, after finding the relevant cluster, the service matcher finds the output and input indices of the relevant cluster. The matched services of the given query are retrieved from the indices. Let us consider a typical query having one output parameter, say 'temperature'. In this example, after finding the relevant cluster of the query, the matcher matches the output parameter of the query, i.e. 'temperature' with each key of the output index of the relevant cluster. If it finds a match, then it fetches all the services linked by that key and return them as semantically matched services of the query. Let us assume the number of web services linked by a matched output parameter is five. In this case, individual matching of the five services with the query is prevented. This is because if any key is found be matched with output parameter of query, it means that all the services linked by that key are similar to the query. In the above example, the matching five services individually with the query are avoided and thereby the performance is improved.

Further, in existing indexing based methods where all the available services are indexed based on output parameters and there exists a single index file for the entire services. This results in a set of candidate services. After finding candidate services, semantic matching of capabilities is performed to candidate services. Here it invokes semantic reasoning during querying. But in the proposed approach, two indices are used to retrieve matched services. This feature fully eliminates the invoking of semantic reasoning during querying.

6. CONCLUSION

An investigation has been done on various optimization approaches to semantic service discovery. Existing approaches are classified according to the component where the optimization is employed. Based on the analysis, an integrated optimization infrastructure along with an efficient matching method has been proposed to optimize the matching component. The infrastructure consists of three core elements, service cache, service clusters and service indices and optimization is achieved through above elements. Theoretical analysis of the proposed approach shows that the proposed approach is an effective approach to improve the performance of the semantic service discovery as it combines optimization at three levels, initially at service cache, secondly at service clusters and thirdly at service index. Further, the usage of two service indices for each cluster fully avoids the invoking of semantic reasoner during query. This feature makes the approach more applicable to real time service composition. Efforts are towards the implementation of the proposed approach.

REFERENCES

[1] Ruiqiang Guo, Jiajin Le and XiaoLing Xia, "Capability matching of Web services based on OWL-S", *Proceedings of 16th International Workshop on Database and Expert Systems Applications*, pp. 653-657, 2005.

[2] Jacek Kopecky, Tomas Vitvar, Carine Bournez and Joel Farrell, "SAWSDL: Semantic Annotations for WSDL and XML Schema", *IEEE Internet Computing*, Vol. 11, No. 6, pp. 60-67, 2007.

[3] Katia Sycara, Massimo Paolucci, Anupriya Ankolekar and Naveen Srinivasan, "Automated discovery, interaction and composition of Semantic Web Services", *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 1, No. 1, pp. 27-46, 2003.

[4] Massimo Paolucci, Takahiro Kawamura, Terry R Payne and Katia Sycara, "Semantic Matching of Web Service Capabilities", *Proceedings of the First International Semantic Web Conference on the Semantic Web*, Vol. 2342, pp. 333-347, 2002.

[5] Umesh Bellur, Harin Vadodaria and Amit Gupta, "Semantic Matchmaking Algorithms", *Advances in Greedy Algorithms by: Witold Bednorz*, pp. 481-502, 2008.

[6] Sonia Ben Mokhtar, Anupam Kaul Nikolaos Georgantas and Valerie Issarny, "Towards Efficient Matching of

Semantic Web Service Capabilities", *International Workshop on Web Services Modeling and Testing*, pp. 137-152, 2006.

[7] Sonia Ben Mokhtar, Anupam Kaul Nikolaos Georgantas and Valerie Issarny, "Efficient Semantic Service Discovery in Pervasive Computing Environments", *M. Van Steen and M. Henning (Eds): Middleware 2006, LNCS*, Vol. 4290, pp. 240-259, 2006.

[8] Sonia Ben Mokhtar, Davy Preuveneers, Nikolaos Georgantas, Valerie Issarny and Yolande Berbers, "EASY: Efficient semantic Service discovery in pervasive computing environments with QoS and context support", *The Journal of Systems and Software*, Vol. 81, pp. 785-808, 2008.

[9] Jingyu Zhang, Xueli Yu, Peng Liu and Zhen Wang, "Research on improving Performance of Semantic Search in UDDI", *Proceedings of the WRI Global Congress on Intelligent Systems*, Vol. 4, pp. 572-576, 2009.

[10] Dong Shou and Chi-Hung Chi, "A Clustering-based Approach for Assisting Semantic Web Service Retrieval", *IEEE International Conference on Web Services*, pp. 838-839, 2008.

[11] Dong Shou and Chi-Hung Chi, "Effective Web Service Retrieval Based on Clustering", *Fourth International Conference on Semantics, Knowledge and Grid*, pp. 469-472, 2008.

[12] Li Ying, "Algorithm for Semantic Web Services Clustering and Discovery", *International Conference on Communications and Mobile Computing*, 2010.

[13] Huiying Goa, Wolffried Stucky and Lei Liu, "Web Services Classification Based on Intelligent Clustering Techniques", *International Forum on Information Technology and Applications*, Vol. 3, pp. 242-245, 2009.

[14] Shun Hon, Haiyang Wang and Lizhen Cui, "A User Experience-Oriented Service Discovery Method with Clustering Technology", *Second International Symposium on Computational Intelligence and Design*, Vol. 2, pp. 64-67, 2009.

[15] Peng Liu, Jingyu Zhang and Xueli Yu, "Clustering-Based Semantic Web Service Matchmaking with Automated Knowledge Acquisition", *International Conference on Web Information Systems and Mining*, Vol. 5854, pp. 261-270, 2009.

[16] Richi Nayak and Bryan Lee, "Web Service Discovery with additional Semantics and Clustering", *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 555-558, 2007.

[17] Gao Ting, Wang Hiayang, Zheng Naihui and Li Fei, "An Improved Way to Facilitate Composition Oriented Semantic Service Discovery", *International Conference on Computer Engineering and Technology*, pp. 156-160, 2009

[18] Bo Zhou, Tinglei Huang, Jie Liu and MeizhouShen, "Using Inverted Indexing to Semantic WEB Service Discovery Search Model", *Proceedings of 5th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 4872-4875, 2009.

[19] Li Kuang, Ying Li, Jian Wu, Shuiguang Deng and Zhaohui Wu, "Inverted Indexing for Composition-Oriented Service Discovery", *IEEE International Conference on Web Services*, pp. 257-264, 2007

[20] Michael Stollberg, Martin Hepp and Jorg Hoffmann, "A Caching Mechanism for Semantic Web Service Discovery", *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, Vol. 4825, pp. 480-493, 2007.

[21] Matthias Klusch and Frank Kaufer, "WSMO-MX: A Hybrid Semantic Web Service Matchmaker", *Web Intelligence and Agent Systems*, Vol. 5, No. 1, pp. 23-42, 2009.

[22] Matthias Klusch, Benedikt Fries and Katia Sycara, "Automated Semantic Web Service Discovery with OWLS-MX", *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems*, pp. 915-922, 2006.

[23] Matthias Klusch, Patrick Kapahnke and Ingo Zinnikus, "Hybrid Adaptive Web Service Selection with SAWSDL-MX and WSDL-Analyzer", *Proceedings of the 6th European Semantic Web Conference on the Semantic Web: Research and Applications*, Vol. 5554, pp. 550-564, 2009.

[24] Wuling Ren and Zhujun Xu, "A New Web Service Discovery Method Based on Semantic", *Workshop on Power Electronics and Intelligent Transportation System*, pp. 223-226, 2008

[25] Zhumin Chen, Jun Ma, Ling Song and Li Lian, "An Efficient Approach to Web Service Discovery and Composition when Large Scale Services are Available", *Proceedings of IEEE Asia-Pacific Conference on Services Computing*, pp. 34-41, 2006.

[26] Umesh Bellur and Roshan Kulkarni, "Improved Matchmaking algorithm for Semantic Web Services Based on Bipartite Graph Matching", *IEEE International Conference on Web Services*, pp. 86-93, 2007.

[27] Alireza Zohali and Kamran Zamanifar, "Matching Model for Semantic Web Services Discovery", *Journal of Theoretical and Applied Information Technology*, Vol. 7, No. 2, pp. 139-144, 2000.

[28] Dong Shou and Chi-Hung Chi, "A Clustering-based Approach for Assisting Semantic Web Service Retrieval", *IEEE International Conference on Web Services*, pp. 838-839, 2008.