Soft Comput (2005)
DOI 10.1007/s00500-005-0029-3

ORIGINAL PAPER

Haibin Wang · Yan-Qing Zhang
Rajshekhar Sunderraman

Extensible soft semantic web services agent

Abstract Web services technology is critical for the success of business integration and other application fields such as bioinformatics. However, there are two challenges facing the practicality of Web services: (a) efficient location of the Web service registries that contain the requested Web services and (b) efficient retrieval of the requested services from these registries with high quality of service (QoS). The main reason for this problem is that current Web services technology is not semantic-oriented. Several proposals have been made to add semantics to Web services to facilitate discovery and composition of relevant Web services. Such proposals are being referred to as semantic Web services (SWS). However, most of these proposals do not address the second problem of retrieval of web services with high QoS. In this paper, we propose a framework called soft semantic Web services agent (soft SWS agent) for providing high QoS Semantic Web services using soft computing methodology. Since different application domains have different requirement for QoS, it is impractical to use classical mathematical modeling methods to evaluate the QoS of semantic Web services. We use fuzzy neural networks with Genetic Algorithms (GA) as our study case. Simulation results show that the soft computing methodology is practicable to handle fuzzy and uncertain QoS metrics effectively.

Keywords Quality of service · Soft computing · Semantic Web · Intelligent agents · Soft semantic web services

1 Introduction

Web services are playing an important role in e-business application integration and other application fields such as bioinformatics. So it is crucial for the success of both service providers as well as service consumers to provide and invoke the high quality of service (QoS) Web services. Unfortunately, current Web services technologies such as simple object access protocol (SOAP), web services description language (WSDL), universal description, discovery and Integration (UDDI), electronic business XML initiative (ebXML), XLANG, web services flow language (WSFL), business process execution language for web services (BPEL4WS), and bioinformatic sequence markup language (BSML) are all syntax-oriented with little or no semantics associated with them. Computer programs may read and parse them, but with little or no semantic information associated with these technologies, the computer programs can do little to reason and infer knowledge about the Web services.

Current research trend is to add semantics to the Web services framework to facilitate the discovery, invocation, composition, and execution monitoring of Web services. Web services with explicit semantic annotation are called semantic Web services (SWS). Several projects are underway to try to reach such a goal. For example, OWL-S (previously DAML-S from OWL Services Coalition) uses OWL based ontology for describing Web services. METEOR-S [20] follows the way that relates concepts in WSDL to DAML+OIL ontologies in Web services description, and then provides an interface to UDDI that allows querying based on ontological concepts. The internet reasoning service (IRS-II) [14] is a semantic Web services framework, which allows applications to semantically describe and execute Web services. IRS-II is base on the UPML framework [15]. The web service modeling framework (WSMF) [6] provides a model for describing the various aspects related to Web services. Its main goal is to fully enable e-commerce by applying Semantic Web technology to Web services.

In our vision, with the maturing of semantic Web services technologies, there willbe a proliferation of public and/or private registries for hosting and querying semantic Web services based on specific ontologies. Currently, there are many public and private UDDI registries advertising numerous similar Web services with different QoS. For example, GenBank, XEMBL, and OmniGene all provide similar Web services with different quality of services. There are two chal-

H. Wang (✉) · Y.-Q. Zhang · R. Sunderraman
Department of Computer Science, Georgia State University,
Atlanta, GA 30302, USA
E-mail: hwang17@student.gsu.edu, yzhang,raj@cs.gsu.edu
Tel.: +1-404-6510682
Fax: +1-404-4639912

lenges existing for automatic discovery and invocation of Web services. One is the efficient location of service registries advertising requested Web services and the other is the efficient retrieval of the requested services from these registries with the highest quality of service (QoS). The semantic Web services technologies that we mentioned above can be exploited to solve the first challenge. For the second challenge, we believe that the QoS of semantic Web services should cover both functional and non-functional properties. Functional properties include the input, output, conditional output, pre-condition, access condition, and the effect of service. These functional properties can be characterized as the capability of the service [1]. Non-functional properties include the availability, accessibility, integrity, performance, reliability, regulatory, security, response time and cost of the Web service.

Several matchmaking schemes have already been proposed to match the service requestor's requirements with service provider's advertisements [16,23]. These schemes basically try to solve the capability matching problem. Here, we must be aware that on the one hand, the degree of capability matching and non-functional properties are all fuzzy, and on the other hand, different application domains have different requirements on non-functional properties. As a consequence, it is not flexible to use classical mathematical modeling methods to evaluate the QoS of semantic Web services. Although there are several existing QoS models [3,7–9,11, 17,18,21,26], none of them are suitable for the requirements considered in this paper. These QoS models are based on precise QoS metrics and specific application domains. They cannot handle fuzzy and uncertain QoS metrics.

In this paper, we propose a framework called *soft semantic Web services agent* (soft SWS agent) to provide high QoS semantic Web services based on specific domain ontology. The soft SWS agent could solve the aforementioned two challenges effectively and efficiently. The soft SWS agent itself should be implemented as a semantic Web service and comprises of six components:

(a) Registries crawler, (b) Repository, (c) Inquiry server, (d) Publish server, (e) Agent Communication server, and (f) Intelligent inference engine. The core of the soft SWS agent is the intelligent inference engine (IIE). It uses soft computing technologies to evaluate the entire QoS of semantic Web services using both functional and non-functional properties. In this paper, we use semantic Web services for bioinformatics as a case study. We employ fuzzy neural networks with genetic algorithms (GA) for the IIE component of our soft SWS agent. The case study illustrates the practicability of soft computing methodology for handling fuzzy and uncertain linguistic semantic Web Services information. For example, capability of a Web service is fuzzy. It is unreasonable to use crisp values to describe it. So we can use several linguistic variables such as a "little bit low" and "a little bit high" to express the capability of services.

The paper is organized as follows. In Sect. 2, we present the necessary background of the QoS model, semantic Web services, and soft computing methodology. In Sect. 3, we provide the architecture of the extensible soft SWS agent. In Sect. 4, we present the design of the fuzzy neural network with GA and simulation results. In Sect. 5, we present related work, and finally, in Sect. 6, we present conclusions and possibilities for future research.

2 Background

This section details the background material related to this research. We cover traditional Web services, semantic Web, semantic Web services, soft computing methodology, and the QoS model.

2.1 Traditional Web services

Web services are modular, self-describing, and self-contained applications that are accessible over the internet [4]. The core components of the Web services infrastructure are XML based standards like SOAP, WSDL, and UDDI. SOAP is the standard messaging protocol for Web services. SOAP messages consist of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. WSDL is an XML format to describe Web services as collections of communication endpoints that can exchange certain messages. A complete WSDL service description provides two pieces of information: an application-level service description (or abstract interface), and the specific protocol-dependent details that users must follow to access the service at a specified concrete service endpoint. The UDDI specifications offer users a unified and systematic way to find service providers through a centralized registry of services that is roughly equivalent to an automated online "phone directory" of Web services. UDDI provides two basic specifications that define a service registry's structure and operation. One is a definition of the information to provide about each service and how to encode it and the other is a publish and query API for the registry that describes how this information can be published and accessed.

2.2 Semantic Web

The current Web is just a collection of documents, which are human readable but not machine processable. In order to remedy this disadvantage, the concept of semantic Web is proposed to add semantics to the Web to facilitate the information finding, extracting, representing, interpreting and maintaining. "The semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [13]. The core concept of semantic Web is ontology. "Ontology is a set of knowledge terms, including the vocabulary, the semantic interconnections, and some simple rules of inference and logic for some particular topic" [10]. There are many semantic Web technologies available today, such as RDF, RDFS, DAML+OIL and OWL. The description logics are used as the inference mechanism for current semantic Web technologies. There are some drawbacks in the description logics [14]. It cannot handle fuzziness and uncertainty associated with concept membership. The current research trend is to combine soft computing with semantic Web [5, 12, 22].

2.3 Semantic Web services

The industry is proposing Web services to transform the Web from "passive state" – repository of static documents to "positive state" – repository of dynamic services. Unfortunately, the current Web services standards are not semantic oriented. They are awkward for service discovery, invocation, composition, and monitoring. So it is natural to combine the semantic Web with Web services, the so-called semantic Web services. Several projects have been initiated to design the framework for semantic Web services such as OWL-S, IRS-II, WSMF and METEOR-S.

For example, OWL-S 1.0 that is based on OWL is the upper ontology for services. It has three subontologies: service profile, service model and service grounding. The service profile tells "what the service does"; that is, it gives the types of information needed by a service-seeking agent to determine whether the service meets its needs. The service model tells "how the service works"; that is, it describes what happens when the service is carried out. A service grounding specifies the details of how an agent can access a service. Typically a grounding will specify a communication protocol, message formats, and other service-specific details such as port numbers used in contacting the service. In addition, the grounding must specify, for each abstract type specified in the ServiceModel, an unambiguous way of exchanging data elements of that type with the service.

2.4 Soft computing methodology

"Soft computing differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth, and approximations" [25]. The principal constituents of soft computing are fuzzy logic, neural networks, and generic algorithms. More and more technologies will join into the soft computing framework in the near future. Fuzzy logic is primarily concerned with handling imprecision and uncertainty, neural computing focuses on simulating human being's learning process, and genetic algorithms simulate the natural selection and evolutionary processes to perform randomized global search. Each component of soft computing is complementary to each other. Using combinations of several technologies such as fuzzy-neural systems will generally get better solutions.

2.5 QoS model

Different applications generally have different requirements of QoS dimensions. Rommel [17] and Stalk and Hout [21] investigate the features with which successful companies assert themselves in the competitive world markets. Their result showed that success is based on three essential dimensions: time, cost and quality. Garvin [8] associates eight dimensions with quality, including performance and reliability. Software systems quality of service has been extensively studied in [9, 11, 26]. For middleware systems, Frlund and Koisinen [7] present a set of practical dimensions for distributed object systems reliability and performance, which include time to repair (TTR), time to failure (TTF), availability, failure masking, and server failure.

In this paper, we construct a QoS model for semantic Web services. It is composed of the following dimensions: capability, response time, and trustworthiness. In order to be more precise, we give our definitions of the three dimensions as follows:

(a) The *capability* of a semantic Web service can be defined as the degree to which its functional properties match with the required functional properties of the semantic Web service requestor;
(b) The *response time* of a semantic Web service represents the time that elapses between service requests arrival and the completion of that service request. Response time is the sum of waiting time and actual processing time;
(c) The *trustworthiness* of a semantic Web services is the extent to which it is consistent, reliable, competent, and honest.

3 Architecture of extensible soft SWS agent

The extensible soft SWS agent can provide high QoS semantic Web services based on specific ontology. The extensible SWS agent uses centralized client/server architecture internally. But itself can also be and should be implemented as a semantic Web service based on specific service ontology. The extensible soft SWS agent comprises of six components: (a) Registries Crawler; (b) SWS Repository; (c) Inquiry Server; (d) Publish Server; (e) Agent Communication Server; (f) Intelligent Inference Engine. The high level architecture of the extensible soft SWS agent is shown in Fig. 1. Each of the components is described next.

3.1 Registries crawler

As we pointed out before, the current UDDI registry only supports keyword-based search for the Web services description. Under the Semantic Web environment, UDDI registry must be extended to be ontology-compatible which supports semantic matching of semantic Web services' capabilities. One possible way is to map the OWL-S service profiles into current UDDI registry's data structure. Semantic Web service providers will publish the service profiles of semantic Web services in the public or private specific service ontology-oriented UDDI registries or directly on their semantic Web sites. The specific ontology based semantic Web services registries crawler has two tasks:

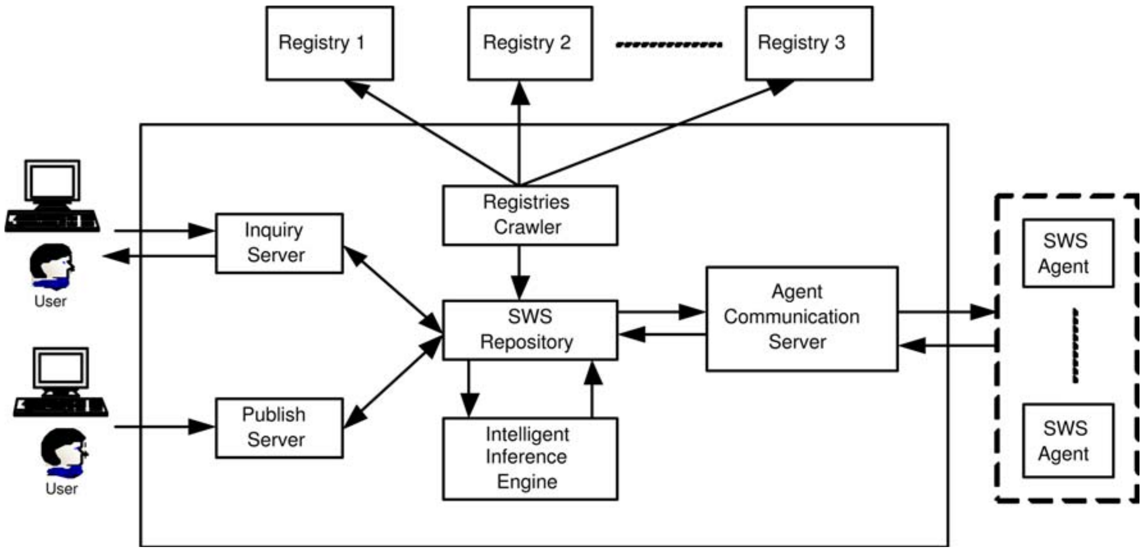

Fig. 1 Architecture of the extensible soft SWS agent

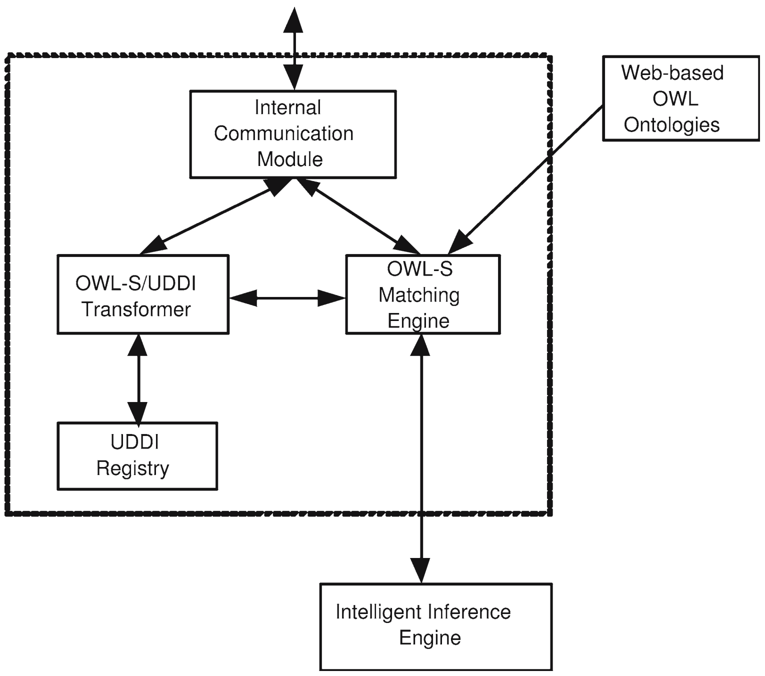

Fig. 2 Architecture of the SWS repository

(a) Accessing these public and private specific service ontology-oriented UDDI registries using UDDI query API to fetch the service profiles, transforming them into the format supported by our repository, and storing them into the repository using the publish API of our repository;
(b) Crawling the semantic Web sites hosting the specific ontology based semantic Web services directly to get the service profiles, transforming them into the format supported by the repository, and storing them into repository using the publish API for the repository.

The registries crawler should be multithreaded and should be available 24×7. The registries crawler must also be provided the information of highest level specific service ontology before its execution.

3.2 SWS repository

The specific ontology based semantic Web services repository will store service profiles of semantic Web services. The architecture of repository is shown in Fig. 2.

The internal communication module provides the communication interface between the repository and the registries crawler, inquiry server, publish server, and the agent communication server. If a message is an advertisement, the internal communication module sends it to the OWL-S/UDDI transformer that constructs a UDDI service description using information about the service provider and the service name. The result of publishing with the UDDI is a reference ID of the service. This ID combined with the capability description and non-functional properties of the advertisement are sent to the OWL-S matching engine that stores the advertisement for capability matching. If a message is a query, the internal communication module sends the request to the OWL-S matching engine that performs the capability matching. After calculating the degree of capability, the OWL-S matching engine will feed the degree of capability and non-functional properties to the intelligent inference engine to get the entire QoS. The service with highest QoS will be selected. The result of the selection is the advertisement of the providers selected and a reference to the UDDI service record. The combination of UDDI records and advertisements is then sent to the inquiry server. If the required service does not exist, OWL-S matching engine will transfer the query to the agent communication server through the internal communication module. The matching algorithm used by OWL-S matching engine is based on the modified algorithm described in [9]. The modified algorithm considers not only the inputs, outputs, preconditions and effects, but also service name.

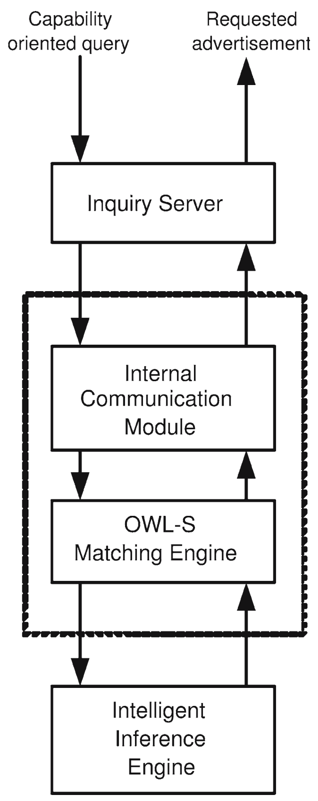
Capability oriented query
Requested advertisement
Inquiry Server
Internal Communication Module
OWL-S Matching Engine
Intelligent Inference Engine

Fig. 3 Capability oriented query

3.3 Inquiry server

The specific ontology based semantic Web services inquiry server provides two kinds of query interface: a programmatic API to other semantic Web services or agents and a Web-based interface for the human user. Both interfaces support keyword oriented query as well as capability oriented searches.

For capability-oriented query, the inquiry server transforms the service request profile into the format supported by the repository such as OWL-S service profile and sends the query message to the internal communication module of the repository. The internal communication module sends the service profile to the OWL-S matching engine and returns back the requested advertisement to the inquiry server and then on to the service requestor. The process is shown in Fig. 3.

For the keyword-oriented queries, the inquiry server will directly send the query string to the internal communication module as a query message and the internal communication module sends the query string to the UDDI Registry and returns back the requested UDDI records to the inquiry server and then on to the service requestor. The process is shown in Fig. 4.

We use SOAP as a communication protocol between service requestors and the inquiry server.

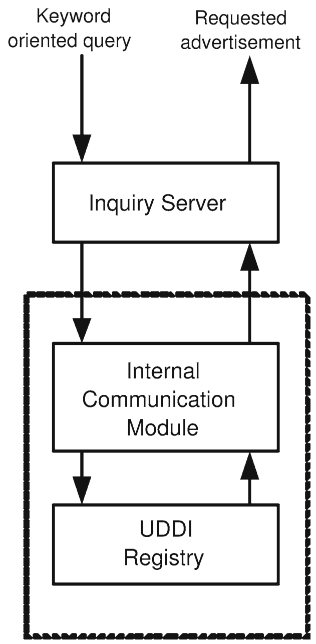

Fig. 4 Keyword oriented query

3.4 Publish server

The specific ontology based semantic Web services publish server provides the publishing service for other agents and human users. It has two kinds of interface. One is the programmatic API to other semantic Web services or agents and another is for the human user that is Web-based. The publish server will transform the service advertisement into the

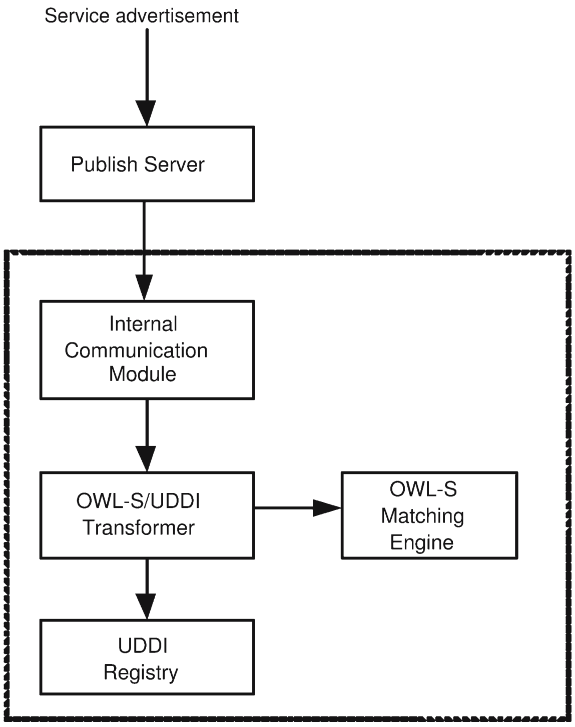

Fig. 5 Publish service advertisement

format supported by the repository such as OWL-S service profile and sends the publish message to the internal communication module. The internal communication module sends the transformed OWL-S service profile to the OWL-S/UDDI transformer. The OWL-S/UDDI transformer will map the OWL-S service profile into UDDI registries data structure, and store the OWL-S service profile and reference ID of service into OWL-S matching engine. The process is shown in Fig. 5.

If the advertised semantic Web services are not in the domain of the soft SWS agent, the internal communication server will transfer the advertisements to the agent communication server which will try to publish the advertisements into other soft SWS agents. SOAP is used as a communication protocol between service publisher and the publisher server.

3.5 Agent communication server

The soft semantic Web services agent communication server uses a certain communication protocol such as knowledge query and manipulation language (KQML) and agent communication language (ACL) to communicate with other soft SWS agents. If the current soft SWS agent cannot fulfill the required services (query and publish), the agent communication server is responsible for transferring the requirements to other soft SWS agents, getting results back, and conveying the results back to the service requestors. The current KQML and ACL should be extended to be ontology-compatible to facilitate the semantic oriented communication.

3.6 Intelligent inference engine

The intelligent inference engine (IIE) is the core of the soft SWS agent. The soft SWS agent is extensible because the IIE uses soft computing methodology to calculate the QoS of the semantic Web services with multidimensional QoS metrics. The IIE gets the degree of capability matching and non-functional properties' values from OWL-S matching engine and returns back the whole QoS to the OWL-S matching engine. In the next section, we show the details of designing an IIE using fuzzy logic, neural networks, and genetic algorithms.

4 An Implementation of intelligent inference engine

This section shows one implementation of IIE based on fuzzy logic, neural networks, and genetic algorithms. A schematic diagram of the four-layered fuzzy neural network is shown in Fig. 6. Nodes in layer one are input nodes representing input linguistic variables. Nodes in layer two are membership nodes. Each membership node is responsible for mapping an input linguistic variable into a possibility distribution for that variable. The rule nodes reside in layer three. The last layer contains the output variable nodes [13].

As we mentioned before, the metrics of QoS of semantic Web services are multidimensional. For illustration of specific ontology based semantic Web services for bioinformatics, we decide to use capability, response time and trustworthiness as our inputs and whole QoS as output. The fuzzy logic system is based on TSK model.

4.1 Input fuzzy sets

Let x represent capability, y represent response time, and z represent trustworthiness. We scale the capability, response time and trustworthiness to [0,10] respectively. The graphical representations of the membership functions of x, y, and z are shown in Fig. 7.

4.2 Fuzzy Rule Bases

Here, we design the fuzzy rule base based on the TSK model. A fuzzy rule is shown below:

IF x is I_1 and y is I_2 and z is I_3

THEN O is $a_{i,1}{*}\, x + a_{i,2}{*}\, y + a_{i,3}{*}\, z + a_{i,4}$.

where, I_1, I_2 and $I_3$are in low, middle, high respectively and i *is the integer* in [1,27]. There are a total of 27 fuzzy rules. The $a_{i,j}$ are consequent parameters which will be obtained by the training phase of fuzzy neural network using genetic algorithms.

4.3 Design of defuzzier

Suppose, for certain inputs x, y and z, there are m fired fuzzy rules. To calculate the firing strength of jth rule, we use the

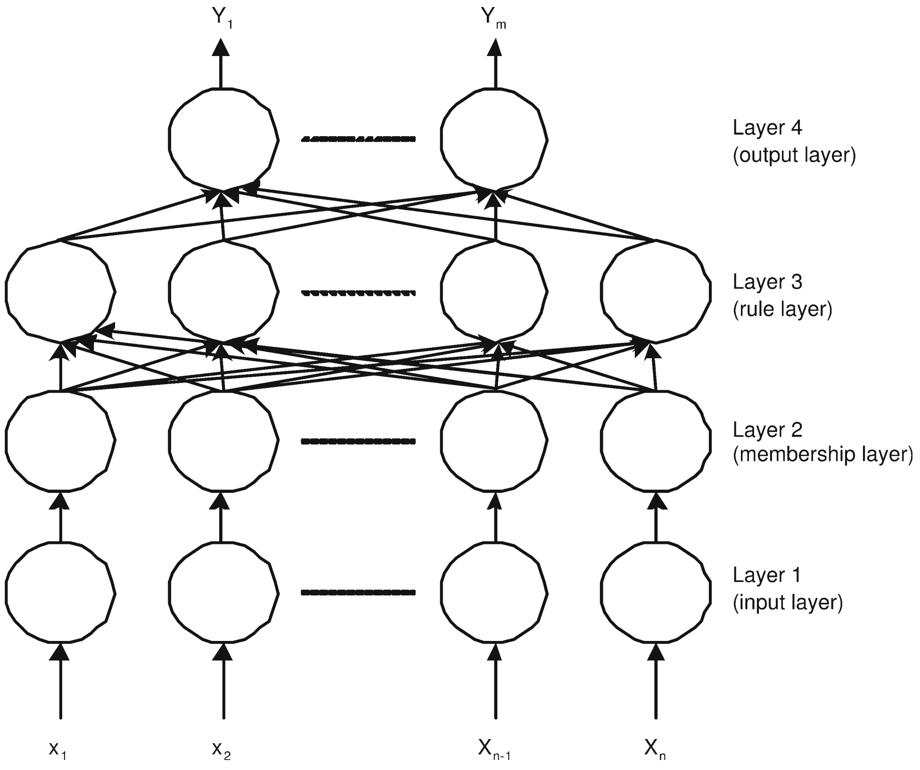

Fig. 6 Schematic diagram of fuzzy neural networks

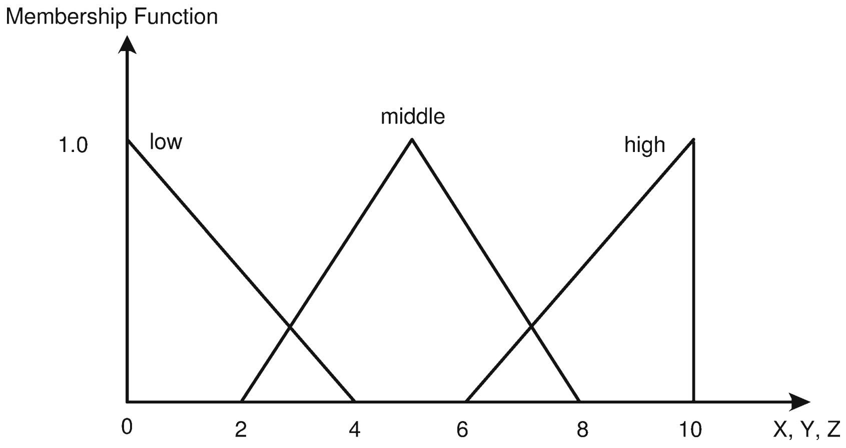

Fig. 7 Membership functions of inputs

formula:

$$W^j = \mu_x(x)^* \mu_y(y)^* \mu_z(z) \tag{1}$$

So, the crisp output is:

$$O = \sum_{j=1^m} \left(a^*_{j,1}x + a^*_{j,2}y + a^*_{j,3}z + a_{j,4}\right) / \sum_{j=1^m} W^j \tag{2}$$

4.4 Genetic algorithms

Genetic algorithm is a model of machine learning that derives its behavior from a metaphor of the processes of evolution in nature. This is done by creation within a machine of a population of individuals represented by chromosomes. Here we use real-coded scheme. Given the range of parameters (coefficients of linear equations in TSK model), the system uses the derivate-free random search-GA to learn to find the near optimal solution by the fitness function through the training data.

(a) Chromosome: The genes of each chromosome are 108 real numbers (there are 108 parameters in the fuzzy rule base) which are initially generated randomly in the given range. So, each chromosome is a vector of 108 real numbers.

(b) Fitness function: The fitness function is defined as

$$E = \frac{1}{2}^{*} \sum_{j=1^{m}} \left(d_j - o_j\right)^2 \tag{3}$$

(c) Elitism: The tournament selection is used in the elitism process.

(d) Crossover: The system will randomly select two parents among the population and then randomly select the number of cross points, and simply exchange the corresponding genes among these two parents to generate a new generation.

(e) Mutation: For each individual in the population, the system will randomly select genes in the chromosome and replace them with randomly generated real numbers in the given range.

4.5 Simulations

In Sect. 3, we just describe the design of the soft SWS agent. We haven't implemented the whole system so the real data is not available. Here, we assume the degree of capability and the values of response time and trustworthiness are already existing.

There are two phases for applying a fuzzy neural network: training and predicting. In the training phase, we use 168 data entries as training data set. Each entry consists of three inputs and one expected output. We tune the performance of the system by adjusting the size of population, the number of generations and probability of crossover and mutation. After the training phase, we use other 168 data entries as testing data set. Table 1 gives the part of prediction results with several parameters for output o.

In Table 1, No. of generations = 100000, No. of populations = 1000, probability of crossover = 0.7, probability of mutation = 0.1. The maximum error of the prediction result is 1.6. The total prediction error for168 entries of testing data set is 25%. In this experiment, the membership functions of degree of capability, response time, and trustworthy are identical just for illustration of the practicability of our proposed method. In real applications, according to the characteristics of the application domain, they may be different. Also, the training data set used in the experiment is not given by the domain experts. In real applications, they should be obtained from the domain experts. We believe that by designing reasonable fuzzy membership functions and choosing reasonable training data set which is based on specific application domain, we can reduce the prediction error further.

Table 1 Prediction result of fuzzy nNeural network

Input *x*	Input *y*	Input *z*	Desired output d	Real output o
0.0	0.0	0.0	0.0	0.0
1.0	4.0	5.0	0.0	0.24
2.0	0.0	4.0	0.0	1.35
3.0	6.0	3.0	0.0	0.53
4.0	2.0	6.0	0.0	0.63
5.0	4.0	7.0	0.0	1.57
6.0	2.0	2.0	3.0	2.80
7.0	5.0	0.0	3.0	2.4
8.0	0.0	8.0	4.0	2.66
9.0	9.0	4.0	2.0	3.3
10.0	0.0	6.0	7.0	5.4

5 Related Work

METEOR-S Web service discovery infrastructure (MWSDI) is an infrastructure of registries for semantic publication and discovery of Web services [24]. MWSDI supports creating registry federation by grouping registries that are mapped to the same node in Registries Ontology. MSWDI is based on the P2P model, so the registries are considered as peers. In our work, the soft SWS agents also can be regarded as peers. MWSDI uses the Registries Ontology to maintain a global view of the registries, associated domains and uses this information during Web service publication and discovery. The limitation of MWSDI is that it supports only capability matching of Web services and does not consider non-functional properties of Web services. The soft SWS agent can be viewed as an enhancement over MWSDI as it provides the service for discovering semantic Web services with the highest whole QoS.

The MWSDI approach annotates WSDL by associating its input and output types to domain specific ontologies and uses UDDI structures to store the mapping of input and output types in WSDL files to domain specific ontologies. It is similar to our work where we use OWL-S ontology directly to enable the semantic description of Web services.

Semantic Web enabled Web services (SWWS) proposes a semantic-oriented service Registry that is similar to our idea. It has five components: Profile Crawler, UDDI Integration Engine, Registry API, Ontology Server and Query Interface. The service modeling ontology is stored in the ontology server. All individual service descriptions are stored as instances of the service description ontology and are also managed by the ontology server. SWWS does not support quality based semantic Web services discovery.

OASIS/ebXML describes architecture of service registry. The registry provides a stable store where information submitted by a submitting organization is made persistent. Such information is used to facilitate ebXML based B2B partnerships and transactions. Submitted content may be XML schema and documents, process descriptions, ebXML Core Components, context descriptions, UML models, etc. It focuses mainly on the registry information model and discusses issues like object replication, object relocation and lifecycle management for forming registry federation. It does not use semantic Web and semantic Web services technologies.

6 Conclusion and future work

In this paper, we discussed the design of an extensible soft SWS agent and gave one implementation of the Intelligent

Inference Engine. The soft SWS agent supports both keyword based discovery as well as capability based discovery of semantic Web services. The primary motivation of our work is to solve two challenges facing current Web services advertising and discovery techniques. One is how to locate the registry hosting required Web service description and another is how to find the required Web service with highest QoS in the located registry. The soft SWS agent can solve both these problems effectively. The soft SWS agent is built upon semantic Web, Web services, and soft computing technologies.

The soft SWS agent could be used in WWW, P2P, or Grid infrastructures. The soft SWS agent is flexible and extensible. With the evolution of soft computing, more and more technologies can be integrated into the soft SWS agent. We used fuzzy neural network with genetic algorithm as our study case. The training time is short and training results are satisfactory. In the future, with the maturing of semantic Web services technology, we plan to implement the whole system and apply it into real applications. We also plan to extend the architecture of the soft SWS agent to compute the entire QoS workflow of semantic Web services to facilitate the composition and monitoring of complex semantic Web services and apply it to semantic Web-based bioinformatics applications.

Acknowledgements The authors would like to appreciate the support by NIH under Grant P20 GM065762. We thank Dr. Harrison for his support and Mr. Tang for his valuable suggestions.

References

1. Ankolekar A, Burstein M, Hobbs J (2002) Daml-s: Web services description for the semantic web. In: Proceeding of the the 1st international semantic web conference (ISWC), Chia, Sardegna, Italy
2. Berners-Lee T, Hendler J, Lassila O (2001) The semantic web. Sci Am 284(5):34–43
3. Clark D, Shenker S, Zhang L (1992) Supporting real-time applications in an integrated services packet network: architecture and mechanism. In: Proceedings of ACM SIGCOMM. pp 14–26
4. Curbera F, Nagy W, Weerawarana S (2001) Web services: why and how. In: Proceedings of the workshop on object-oriented Web services-OOPSLA, Tamba, Florida
5. Ding Z, Peng Y (2004) A probabilistic extension to ontology language owl. In: Proceedings of the Hawai' international conference on system sciences, Big Island, Hawaii
6. Fensel D, Bussler C (2002) The web services modeling framework: wsmf. Electr Commerce: Res Appl 1:113–137
7. Frlund S, Koisinen J (1998) Quality of service specification in distributed object systems. Distrib Sys Eng J 5(4):179–202
8. Garvin DA (1998) Managing quality: the strategic and competitive edge, Free, New York
9. Georgiadis L, Guerin R, Peris V, Sivarajan K (1996) Efficient network qos provisioning based on per node traffic shaping. IEEE/ACM Trans Networking 4(4):482: 501
10. Hendler J (2001) Agents and the semantic web. IEEE Intelli Sys 16(2):30–37
11. Hiltunen M, Schlichting R, Ugarte C, Wong G (2000) Survivability through customization and adapatability: the cactus approach. DARPA Information Survivability Conference and Exposition, pp 294–307
12. Koller D, Levy A, Pfeffer A (1997) P-classic: a tractable probabilistic description logic. In: Proceedingss of AAAI-97:390–397
13. Lee C-H, Hong J-L, Lin Y-C (2003) Type-2 fuzzy neural network systems and learning. Int J Comput Cogniti 1(4):79–90
14. Motta E, Domingue J, Cabral L, Gaspari M (2003) A framework and infrastructure for semantic web services. In: Proceedings of the 2nd international semantic conference (2003), Sanibel island, Florida, USA
15. Omelayenko B, Crubezy M, Fensel D, Benjamins R (2003) Upml: the language and tool support for making the semantic web alive, In: Spinning the semantic web: bringing the WWW to its full potential, MIT Press Cambridge, USA, pp 141–170
16. Paolucci M, Kawamura T, Payne T, Sycara K (2002) Semantic matching of web services capabilities. In: Proceedings of 1st ISWC 2002, Sardinia, Italia
17. Rommel G (1995) Simplicity wins: how Germany's mid-sized industrial companies succeed. Harvard Business School Press, Boston, Mass
18. Sheth A, Cardoso J, Miller J, Kochut K (2002) Qos for service-oriented middleware. In: Proceedings of conference on systems, cybernetics and informatics, Orlando, Florida
19. Sheth A, Ramakrishnan C, Thomas C (2005) Semantics for the semantic web: the implicit, the formal and the powerful. Int J Semantic Web Inf Sys 1(1):1–18
20. Sivashanmugam K, Verma K, Sheth A, Miller J (2003) Adding semantics to web services standards. In: International conference on Web services pp 395–401
21. Stalk G, Hout T (1990) Competing against time: how timebased competition is reshaping global markets, Free, New York
22. Straccia U (1998) A fuzzy description logic. In: Proceedings of 1st AAAI-98, Madison, USA, pp 594–599
23. Sycara K, Klusch M, Widoff S, Lu J (1999) Dynamic service matchmaking among agents in open information environments. SIGMOD Record 28(1):47–53
24. Verma K, Sivashanmugam K, Sheth A, Patil A (2005) Meteors wsdi: A scalable p2p infrastructure of registries for semantic publication and discovery of web services. J Inf Technol Manag (in press)
25. Zadeh L (1994) Fuzzy logic, neural networks, and soft computing. Communi ACM 37:77–84
26. Zinky J, Bakken D, Schantz R (1997) Architectural support for quality of service for corba objects. Theory Pract Obj Sys 3(1):1–20