A Broker-Based Architecture for Quality- Driven Web Services Composition using a Hybrid Genetic Algorithm

Kavya Johny
Department of Computer Science, Viswajyothi College of Engineering and Technology, Vazhakulam, Kerala

ABSTRACT
Web services are popular framework for integrating distributed applications and enterprise business processes by individual service components. Several system issues must be considered when integrating distributed Web services into a business process. Web service composition consists of combining web services to offer more complex services. The QoS driven broker based architectures are used for the selection of web services that satisfies the QoS requirements (QoS constraints) of consumers. In the web service selection there might be some web service implementations that are dependent on each other. The web service selection will be better if we also consider those constraints during web service selection. This paper proposes a broker based architecture for quality based web service composition that takes into account the mutual constraints between the web service implementations.

General Terms
Web Services Composition, Web Service Selection, Genetic Algorithms.

Keywords
Web Services, Quality of Service, Broker Based Architecture, Hybrid Genetic Algorithm.

1. INTRODUCTION
Web services provide ready-to-use functionalities through fixed interfaces for other applications by hiding the implementation details and they are commonly used in real world applications. There exist a huge number of web services that can handle particular requests. With the increasing number of available web services, maintaining these services and searching for the ones that satisfy a given requirement has become an important problem. In order to handle a complex request, a combination of more than one service is required. This process of combining web services to achieve complex tasks is called Web Service Composition (WSC).

The Web service composition includes three steps: 1) composite Web service specification, 2). selection of the component Web services, and 3) execution of the composite Web services. The Service providers provide web services with the QoS specification. The consumers specify the service that is to be served with their own QoS requirements.

Table 1 outlines some possible QWS metrics that consider when discovering relevant services.

	Parameter	Description
1	Response Time	Time taken to send a request and receive a response
2	Availability	Number of services available
3	Throughput	Total number of invocations for a given period of time
4	Likelihood of success	Number of response/number of request messages
5	Reliability	The number of error free messages in total messages
6	Compliance	The measure of ability of a Web service to respond to Web Services Interoperability (WS-I)Basic Profile
7	Latency	Time to process a given request

So the need to discover and select the appropriate web services becomes more important during the second step of web services composition. Web service consumers need tools to search for suitable services. This poses challenges not only in discovery mechanisms and guaranteeing high quality services.

The Quality assessment of web service is used for obtaining high-quality results [1]. Web service QoS requirements affects the performance of web services. Often, unresolved QoS issues [2][3][4] cause critical transactional applications to suffer from unacceptable levels of performance degradation.

Many researchers have studied on the adaptive and dynamic service composition problem. Previously, a framework for quality-driven web service composition [5] was proposed that selects the web services based upon the QoS requirements of the requestors. S.Majitha et al proposed a framework for reputation-based semantic service discovery [6]. Diego and Maria [7] proposed an extended Web service architecture to support QoS management. There are many web service discovery models that contain UDDI to accommodate the QoS information and a management system to build and maintain service reputations and a discovery agent to facilitate service discovery. Different approaches various optimal web service selection problems have been proposed in the previous years. There are many algorithms for web service selection in which selection is referring to the QoS requirements without considering the constraints between them. The study on the optimal web service selection problem with constraints remains open. Thus, genetic algorithms might be efficient and effective for solving the problem. The hybrid genetic algorithm performs a constrained web service selection for the web services composition problem. This improves the quality of solutions.

2. THE BROKER ARCHITECTURE

2.1 WF-Modeller

The First step is done by the WF-Modeller. The requestor inputs a service description to the Broker referring to the required final Web service [8]. The WF-Modeller returns a workflow as a result. It returns a set of activities where each activity is complemented by a set of semantic annotations, to describe its functionalities and capabilities.

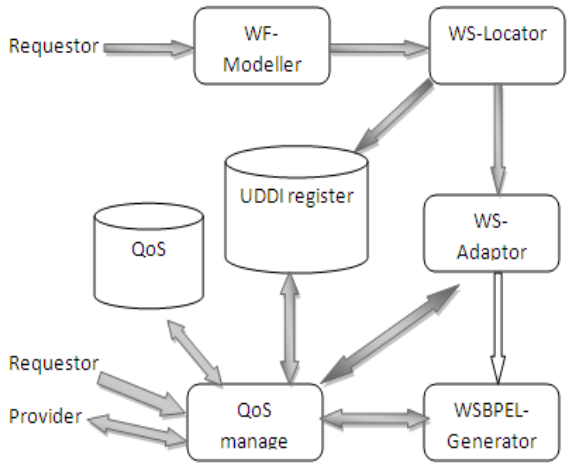

Fig 1: QoS Broker Architecture

2.2 WS-Locator

Based upon the workflow that has been generated, the WS-Locator will identify one or more Web services for each WF activity. It searches the UDDI registry to find similar web services for each workflow activity. It exploits both UDDI search functionality, and semantics annotations to perform the assignment.

2.3 WS-Adaptor

The requestor specifies QoS requirements along with the request. From the set of services for each workflow activity obtained from the WS-Locator it will select the best web services satisfying requestor's QoS constraints and preference for every task to take part in the composition.

2.4 QoS Manager

The QoS property values provided by service providers are finally verified by the QoS-Broker. The QoS-Manager will refine these values in QoSDB according to the user feedbacks to reflect more accurate values.

2.5 WSBPEL Generator.

This is the last step. The results returned by the QoS Adaptor are translated into a WSBPEL document.

3. THE HYBRID GENETIC ALGORITHM

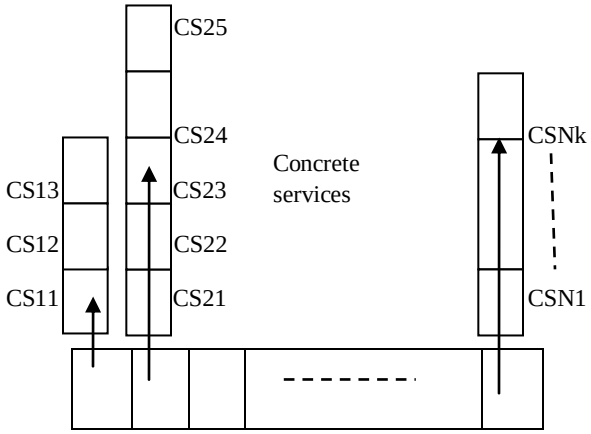

Fig 2: Genetic Encoding

The Genetic algorithm works on a number of individuals which are found to be feasible solutions. Each individual have a fitness value which represents a measure of feasibility. Fig 2 shows the genetic coding of web services where CSij represents concrete web services. The hybrid genetic algorithm, an extended form of normal genetic algorithms, provides an optimal web service selection. There may be mutual constraints between some web service implementations. The mutual constraints are dependency constraints and conflict constraints. In the web service selection, both dependency constraints and conflict constraints must be considered. The hybrid genetic algorithm performs web service composition considering both these constraints.

3.1 Dependency Constraint

For a web service, if an implementation is selected, then a different implementation must be selected for another web service. For instance, a travel booking web service can be built by aggregating a flight booking web service, a car rental web service, a travel insurance web service, an accommodation booking web service, a payment web service, and an itinerary planning Web service. When building a travel booking web

service, if we select a particular travel insurance web service that only accepts payments made by Master cards, then we must select a payment web service that accepts Master cards. This kind of constraints is called dependency constraint.

3.2 Conflict Constraint

The conflict constraint is, sometimes when an implementation for one web service is selected, a set of implementations for another web service must be excluded in the web services composition. When building a travel booking web service, if we select a particular flight booking web service implementation that does not accept deposits made by Master cards, then we must not select an implementation for the payment web service that supports Master cards. This type of constraints is called conflict constraint.

3.3 Local Optimizer

The hybrid genetic algorithm [9] uses a local optimizer to improve the individuals in the population and utilizes a knowledge based crossover operator. The local optimizer is used at the beginning of the genetic algorithm. The individuals in the initial population are randomly generated at first, and at the end of each generation the local optimizer is used to improve the individuals in the population. The local optimizer maximizes the overall QoS value and also minimizes the number of constraint violations of an individual.

For each concrete web service of set of web services, it will calculate the fitness value of the new web service selection plan. It will check systematically all the concrete web services one by one to see if there exists an alternative concrete web service that gives the individual a better fitness value. If a better one exists then it will replace the present one with the best web service. Thus local optimizer optimizes individuals in a population.

3.4 Crossover Operation

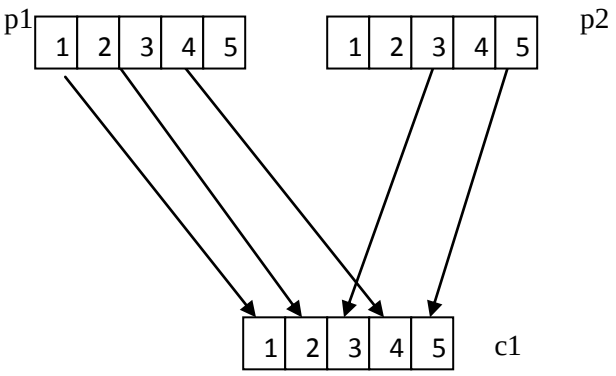

Fig. 3 Crossover Operation

The Hybrid genetic algorithm optimizes individuals using local optimizer and selects the best individuals to generate a new child using crossover operator. There are two types of crossover operators: one-point crossover and combining crossovers. The combining crossover combines the two solutions as shown in Fig 3. To generate c1, the crossover operator firstly identifies all the concrete web services in parent p1 that do not violate any constraints, and then copies them to child c1. The rest concrete web service selections in c1 are copied from p2.

3.5 Mutation Operation

Mutation is also used to perform constraint less web service composition as in Fig 4. Mutation is the process of selecting one web service randomly and replacing it with a better one. There are five kinds of mutation operators such as one-point mutation, biased one-point mutation, K-means mutation, cluster addition and cluster removal [10].

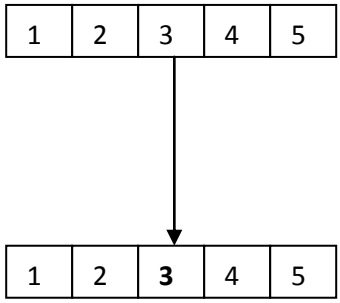

Fig. 4 Mutation Operation

The constraint violation is removed in such a way that, if the current concrete web service selection violates any constraints, we check through all the alternatives and select the alternative that gives the best fitness value. If the current concrete web service selection does not violate any constraint, then we check the alternatives one by one in the decreasing order of their weighted QoS value. We replace the current concrete web service selection with the better alternative concrete web service immediately. In order to improve the computation time of the local Optimizer, we sort the concrete web services only once at the beginning of the genetic algorithm.

Begin
Initialize population with random candidate solutions;
Optimize each individual with local optimizer
Evaluate each candidate with its fitness value.
Repeat
Select parents from the population;
Crossover is performed by randomly selecting the atomic services without any constraints and generates a child with those atomic services of parents.
Mutate the resulting children by selecting a web-service and replace with another one.
Evaluate the children.
Replace the parents with the children if it provides best fitness value.
Select individuals for the next generation
Until termination-condition is satisfied
End

4. CONCLUSION

With the emerging role of web services in business processes, the requirement of composing and executing them has begun to draw high attention, and today the need to find the optimal web services composition for the business processes is a challenging issue. This paper proposes the broker based architecture that makes a quality driven web service composition considering

both the dependency and conflict constraints. The hybrid genetic algorithm is scalable and its computation time does not change significantly with the increase in the number of abstract web services. This will provide a better web service selection for each workflow activity and thereby provide an optimal web service composition according to the user QoS requirements.

5. REFERENCES

[1] Eyhab Al-Masri and Qusay H. Mahmoud,"Toward Quality Driven Web Service Discovery," IEEE Computer Society, IT Pro May/June 2008.

[2] Daniel A. Menasce, "QoS Issues in Web services," IEEE Internet Computing, December 2002.

[3] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for web services composition," IEEE Transactions on Software Engineering, vol. 30, no. 5, pp. 311–327, May 2004.

[4] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end QoS constraints," ACM Trans. on Web, vol. 1,no. 1, p. 6, 2007.

[5] S.M. Babamir, S. karimi and M.R. Shishehchi, "A Broker-Based Architecture for Quality-Driven Web Services Composition," IEEE computational Intelligence and software engineering (CISE), International conference, December 2010.

[6] S. Majitha, A. Shaikhali, O Rana, and D. Walker. "Reputation based semantic service Discovery," In Proc. Of the 13 th IEEE Intl Workshops on Enabling Technologies Infrastructures for collaborative Enterprises (WETICE), pp.297-302, Modena, Italy, 2004.

[7] D. Garcia and M. Toledo, "A web service Architecture providing QoS Management", Web Congress, LA-Web '06. Fourth Latin American, pp.189-198, 2006.

[8] B. Carminati, E. Ferrari, P.C. K. Hung, "Web Service Composition: A Security Perspective", International Workshop on Challenges in Web Information Retrieval and Integration, Tokyo, Japan April 08-April 09.

[9] Maolin Tang and Lifeng Ai, "A Hybrid Genetic Algorithm for the Optimal Constrained Web Service Selection Problem in Web Service Composition", Evolutionary Computation (CEC), 2010 IEEE Congress, July 2010.

[10] Petra Kudova ," Clustering Genetic Algorithm", 18th International Workshop on Database and Expert Systems Applications, 2007 IEEE