Vol 6 • Number 2 April - Sep 2015 pp. 285-293 **Impact Factor: 2.5** **Available at** www.csjournals.com

Dynamic Web Services Composition using Optimization Approach

Kirit J. Modi[1], Sanjay Garg[2]

[12]CSE Department, Institute of Technology, Nirma University, Ahmedabad, India

kiritmodi@gmail.com, gargsv@gmail.com

Abstract: In recent time, large number of web services is published over the web. Several issues are faced by end users i.e. when single services is unable to fulfil the requirement, when more than one services provide the same functionality or when there is a need of optimal solution. Dynamic web services composition facilitates users to generate value added services from the existing one at runtime. Use of QoS parameters with web services plays vital role to select the most relevant solution when same functionality provided by multiple candidate solutions. Genetic programming along with QoS-based service composition provides optimal solution with assurance. In this context, our work presents an optimization based approach for dynamic web service composition that takes into account the quality of service model. The proposed approach utilizes a Genetic Algorithm to generate the optimal composition.

Keywords: Dynamic web services composition, Genetic Algorithm, Optimization, QoS, Web services

Introduction

A web service is a new concept to perform distributed programming over the web. It is defined as a loosely coupled reusable software component accessible over the internet. It facilitates integration of applications both within and across organizations by avoiding problems of different platform and heterogeneous programming language. There are many advantages and benefits by using the web service based applications as easy and fast deployment, inter-operability, just in time integration and reduced complexity by encapsulation. Web services include functional features that indicate the input/output and non-functional features such as cost, availability, execution rate, and reputation. The Web service model consists of three main entities i.e. service provider, service registry and service consumer. Figure 1 shows the traditional Web service model [1].

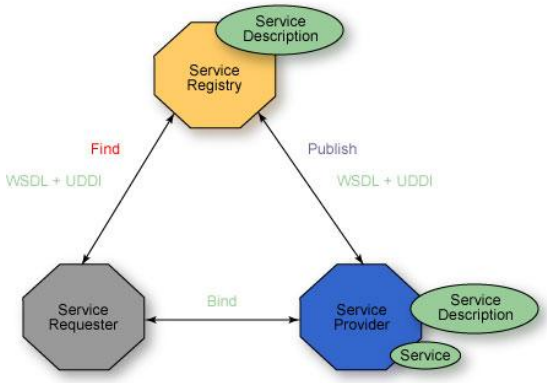

Figure 1. Web service model

As shown in figure 1, a service provider publishes services to the UDDI by generating a web Service description Language (WSDL) file which describes the web services. Once a requestor has queried the UDDI and found the suitable web service, the WSDL file of that service cloud be downloaded and use it to generate SOAP messages to interact with the web service. Sometimes, single web services cannot satisfy the need of the user which raises the necessity to combine the functionality of several web services. Service composition is the process of combining existing services, available on the web, to provide value-added services as a composite service. Composite service combines the functionalities of other services to get desired output. This combination consists of a set of workflow like structures. Finding a composite service which satisfies client's QoSconstraints is time consuming optimization problem. Optimization approach based on genetic algorithm could be an effective approach to solve the above mentioned problem.

The rest of the paper is structured as follows: Section 2 discusses the related work. Section 3 presents Qos model and composite web services. Section 4 represents the proposed work for web service composition based

on optimization approach. Section 5 demonstrates the experimental work and section 6 presents the conclusion and future work.

Related Work

Following section represents related work in the of dynamic web service composition using QoS and Optimization approach.

Genetic Algorithm (GA) provides optimal web service composition, more suitable to handle generic QoS attributes. Paper [5] shows comparison of GA with integer programming for small and large scale Web services. But one drawback of this approach is that when the number of concrete services is small, integer programming performs well than GAs. In [6], GA and case base reasoning is used to generate web service composition. Case based reasoning technique is used for the work flow selection.

In [7], method finds the composition plan that satisfies user constraints efficiently in less time but not tested on any real data sets. New approach is proposed to combine GA with ant colony, GA with Tabu search, GA with Particle Swarm Optimization gives good performance and large search space [8], but it doesn't provide QoS support. In [9], Fuzzy based Genetic algorithm system enables user to participate in the process of web service composition. But this approach requires additional time for fuzzy component.

Approach proposed in [10] transforms the problem of optimal execution schema selection for composite web service into the optimal path selection in the weighted directed acyclic graph. But in this approach search space is limited. In [11], hybrid GA and tabu search approach is proposed for QoS based web service composition. Hybrid GA gives better performance on well tuned parameters. For optimisation of service composition, an approach proposed in [12] uses QoS and semantic similarities between input and output of web services, but it is not tested on any real datasets.

QoS model

We have considered execution duration, availability, reputation, execution cost and successful execution rate as QoS parameters which are defined in table 1 as follows.

QoS Parameter	QoS Description
Price	Fee received by service provider from service requester for each execution.
Execution duration	Difference between the time when a request is sent and time when the result are received
Reputation	Average reputation score of a web service evaluated by the Clients
Success execution rate	It is the percentage of requests which a web service performs successfully when web service is available and accessible
Availability	It is the degree that a web service is accessible and ready for immediate use

Table 1.QoS Parameters

The above QoS parameters need to be normalized for appropriate evaluation. The QoS parameters are divided into two categories [7]. First, negative values such as execution duration and execution cost higher the value of negative property lower the quality they serves .Second positive values, such as availability, reputation and successful execution rate. If higher the value of positive property higher the quality they serves. Positive and negative QoS properties are normalized for appropriate evaluation. Equations (1) and (2) are used to normalize positive and negative properties, respectively which are proposed by [7].

$$Qnrm = \begin{cases} \dfrac{q - qmin}{qmax - qmin} & qmax - qmin \neq 0 \\ 1 & qmax - qmin = 0 \end{cases} \quad (1)$$

$$Qnrm = \begin{cases} \frac{qmax - q}{qmax - qmin} & qmax - qmin \neq 0 \\ 1 & qmax - qmin = 0 \end{cases} \quad (2)$$

Where *Qnrm* isnormalized QoS for each web service calculated by equation (1) and (2) for positive and negative QoS parameter respectively. *qmax is* maximum QoS value among all web services in repository or data set. *qmin*is minimum QoS value among all web services in repository or data set.

The local value of each web service calculated by equation (3), proposed by [7]

$$\text{Qlocal} = \sum_{i=0}^{Q} wiqi \quad (3)$$

Where *wi* isith candidate web service and *qi* is QoS value of ith service.

Calculation of aggregate QoS value for service composition

For each composition plan, we required to calculate aggregate QoS value of all participating services.There is some predefined aggregation functions used to compute the QoS value of each composite service. The aggregation value of web service composition is calculated based on its workflow pattern. We have utilised serial pattern based aggregation function proposed in [7].

We have obtained the fitness of each composition plan using aggregated QoS value of each QoS property and user specified QoS constraints. Therefore, we required to find aggregated QoS value of each composition plan or composite service.

Optimization approach

An Optimization algorithm is intended to discover optimized web service composition plan based on user QoS requirements. We have improved fitness function of genetic algorithm as an optimization approach. The genetic algorithm process is presented in the form of flow chart as shown in figure 2, which is proposed by [7].

In the beginning, user sends request with QoS constraints. Each service available in the repository has its QoS parameters call it q. QoS parameters are divided in to positive and negative parameters. The equation 1 and 2 are used to normalize parameters, while equation 3 is used to calculate the local value of each web services and sort it in the descending order.

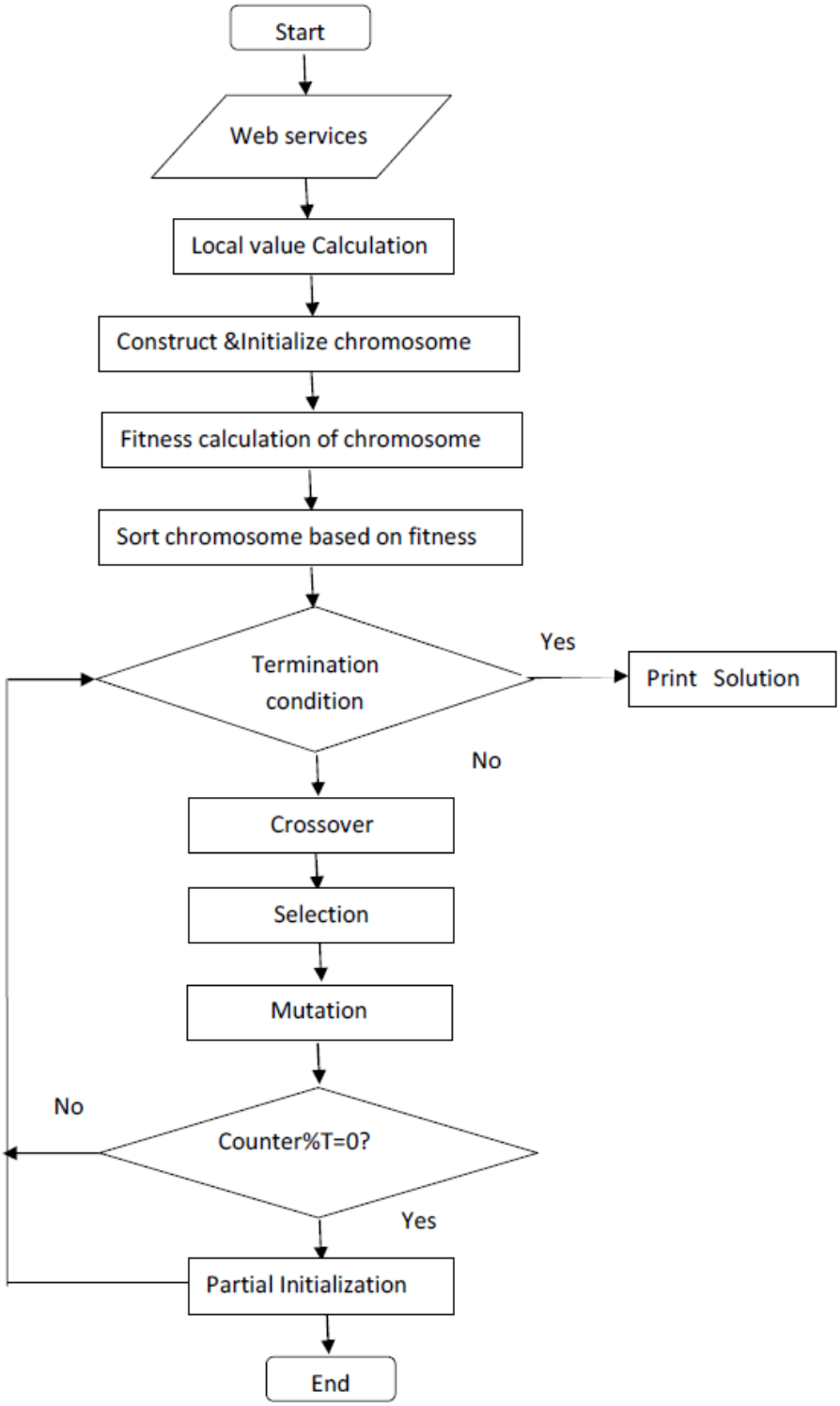

Figure 2: GA based optimization approach

Generate and initialize chromosome based on m tasks and n candidate web services. To generate chromosome, we take m tasks as number of genes in chromosome and each task have n candidate web services. 20% of chromosomes are taken from 20% of good services which contains high local value whereas the remaining 80% are taken randomly from n web services. Chromosome structure is depicted in Figure 3 which is proposed by [5]

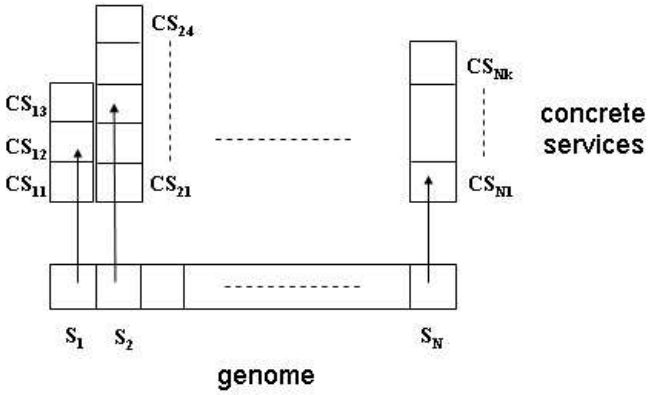

Figure 3. Chromosome structure

To calculate the fitness function, we need to find aggregation of QoS parameters for each web services in chromosome or composition plan. After Aggregation calculation for each chromosome, fitness function is calculated using equation (5) and (6) proposed by [7] where D1 is used for negative values and D2 is used for positive values respectively.

$$D1 = \sum wd\ (\frac{aggd}{cond} - 1) If\ aggd >= cond \textbf{(4)}$$

$$D2 = \sum wd (\frac{aggd}{cond} - 1) If\ aggd <= cond \ \textbf{(5)}$$

$$\text{Fitness} = D1 - D2 \ \textbf{(6)}$$

Where *cond* isQoS constraints for d^{th}QoS property, aggdisaggregated value of the d^{th}QoS property of a composition plan and *d*is index of QoS property.

After calculating D1 and D2, use equation (6) to calculate fitness of each chromosome and to sort each one. Repeat the loop for genetic algorithm process until an optimized composition plan is not achieved. Crossover, selection, mutation and partial initialization operations are main phases of the genetic algorithm process. Crossover performs integration of two or more parent chromosomes and generates one or more child chromosomes. According to [7], we defined two kinds of crossover. In crossover type 1, genes of a child are inherited from parents alternatively. In crossover type 2, a certain percentage of genes are inherited from one parent and the other genes are inherited from the other parent.

To perform crossover, 20% of good chromosomes with good fitness are integrated with each other by crossover type 1 and for the remaining 80%, each chromosome is integrated with a randomly selected chromosome belonging to 20% of chromosomes with good fitness function by crossover type 2. To perform selection, roulette wheel selection method is utilized. Constant number of good chromosomes from the previous step are chosen and replaced with worst chromosomes. This results in preservation of good chromosomes but accelerates the convergence of the algorithm to the local optimum. To escape from local optimum we apply mutation. In Mutation, some genes of some chromosomes that are selected randomly will change with probability of pm. where pm is mutation probability is important to get rid from local optima.

A. Mapping of web services composition problem into Genetic approach

Mapping process of web services composition problem into genetic approach is described in table 2.

Genetic algorithm terms	**Terms in web service composition process**
Gene	Task or abstract web service called **m. m** can map 1 to n candidate web services. 1 to n **Gene(m)**
Chromosome	Collection of genes is called Composition plan. It contains **m** tasks or genes. 1 to n | 1 to n | 1 to n Gene 1 Gene 2 Gene m
Population	Collection of composition plan is called population.
Initialization	20% of composition plans are initialized with best web services which have higher Local value. Remaining composition plans are initialized with randomly from number web services.
Fitness function	Fitness of composition plan is calculated based on aggregated QoS and user QoS constraints.
Crossover	One point and two point crossover methods are used. Here crossover probability is 0.8.
Selection	Best composition plans from population are selected and replaced with worst composition plans in population
Mutation	Some composition plans are selected randomly from population and mutation is performed by interchanging or reversing of genes. Here mutation probability is 0.2.
Partial initialization	Some composition plans which have poor fitness are initialized again.

Table 2. Mapping process of web services composition problem into Genetic approach

Experimental Work

We have tested our approach on randomly generated web services repository and web service challenge (WSC) test sets [14]. All experiments are performed on the same setup defined by [7]. For effectiveness, our experiments were executed 5 times and then took average values of them.

A. Setup

For the experimental purpose, we have considered an Eclipse as an Integrated Development Environment with Java 1.6 Software Development Kit (SDK) which can run on Pentium 2.2 GHz dual core with 3 GB of main memory.

The Genetic programming approach required some control parameters. For our experimental results, we used the control parameters, which are summarized below:

- Population size: 50
- Crossover probability: 0.8
- Mutation probability: 0.2
- Random selection for mutation
- Roulette wheel selection method for selection

B. Results

First experiment was performed with 30, 40 and 50 tasks. For each task we have considered 30, 40 and 50 services. We use stopping criteria as maximum number of generations in which fitness remains constant. Here we have taken 10 generation as stopping criteria. We have considered user QoS parameters as proposed by [7] (Availability=0.89, Reputation=0.61, Successful execution rate=0.78, cost=0.25, Execution Duration=12ms). If we have aggregation value of plan is equal to (0.90, 0.54, 0.82, 0.12, 9.20) then our approach will select this plan for further process because all QoS parameters are satisfied except Reputation.

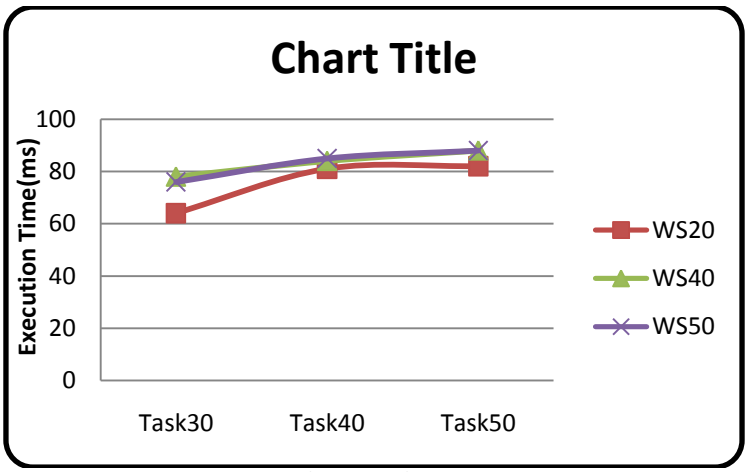

Figure 5. Performance of GA based approach

We have received the execution time is 88 milliseconds which is less than the work represented in [7] as shown in figure 5.

We have compared our work with the approach presented in [13] in which they have used combination of tabu search and simulated annealing. The results of comparison show that our approach is better than algorithm presented in [13] as shown in figure 6.

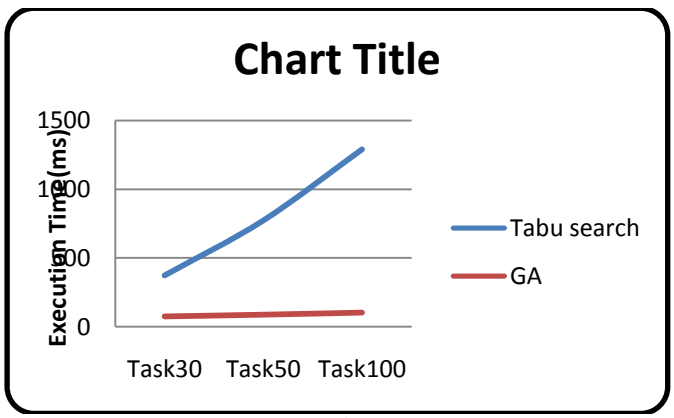

Figure 6. Comparison of GA with Tabu search

We can clearly found that our GA based approach gives better performance than tabu search approach [13] and QoS based approach described in [7].

We have evaluated performance of our approach by changing generations from 50 to 400, by varying tasks from 10 to30. We have taken any number of services but fix for all generations. We have taken 20 web services. Evaluation of composition quality is shown in Figure 7. Here we can clearly say that generation 150 to 400 is best generation in which we get constant fitness. It means there is no need to take more generations for our approach. We also check execution time per generation. We have taken 50 to 400 generations and noted execution time for per generation with 30 tasks. We can say that execution time of our algorithm is better than approach proposed in [12].

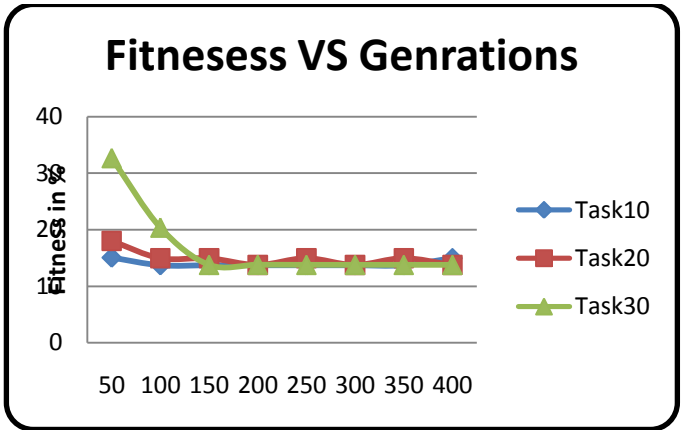

Figure 7. Evaluation of composition Quality

We have used WSC 2007 data set [14] for evaluation of our algorithm where we followedQoS model which uses five QoS parameters includes Execution Duration, availability, reputation, Execution cost and successful execution rate.

We got result for WSC -2007[14] test sets which are listed in table 2. In all the tests and runs a valid solution was found, indicating that the algorithm is robust and reliable for different repositories. We got maximum time 404 mille seconds for 10,000 services. So we got improved speed of algorithm. We can say that our approach is valid on real data sets.

Experimental results shows that our approach increase execution speed and give optimal solution effectively. Actually GA for web service composition gives optimal solution but takes too much time for composition but our modified GA based approach give optimal composition in less time.

Test sets	Number of Services	Time(ms)	Fitness
WSC-2007 -1	118	236:251	11.68
WSC-2007 -2	480	191:252	12.5
WSC-2007 -3	1000	263:272	11.17
WSC-2007 -4	981	231:252	11.51
WSC-2007 -5	1964	231:282	13.81
WSC-2007 -6	4000	281:316	10.83
WSC-2007 -7	1964	232:261	12.04
WSC-2007 -8	7942	306:401	12.43
WSC-2007 -9	10000	317:404	11.4

Table 3. Execution time and Fitness value (Minimum: Maximum)

Conclusion and Future work

We have presented in this paper that how can perform the optimized web service composition using genetic algorithm. New techniques for initialization of chromosomes, selection and crossover functions have been proposed. The experimental results show that GA can find suitable composition plan much faster than other random search techniques. From above experiments we can conclude that our approach gives optimized composition plan in reasonable time. We will map our approach with ontology for improving semantic service composition. We will also focus to combine GA with non-evolutionary algorithm to get improved speed and optimized web service composition.

References

[1] McIlraith, Sheila A., Tran Cao Son, and HongleiZeng. "Semantic web services." IEEE intelligent systems 2 (2001): 46-53.

[2] Bucchiarone, Antonio, and StefaniaGnesi. "A survey on services composition languages and models." International Workshop on Web Services–Modeling and Testing (WS-MaTe 2006). 2006..

[3] Zeng, Liangzhao, et al. "Quality driven web services composition." Proceedings of the 12th international conference on World Wide Web. ACM, 2003. .

[4] Dustdar, Schahram, and Wolfgang Schreiner. "A survey on web services composition." International journal of web and grid services 1.1 (2005): 1-30.

[5] Canfora, Gerardo, et al. "An approach for QoS-aware service composition based on genetic algorithms." Proceedings of the 7th annual conference on Genetic and evolutionary computation. ACM, 2005.

[6] Fanjiang, Yong-Yi, et al. "Genetic algorithm for QoS-aware dynamic web services composition." Machine Learning and Cybernetics (ICMLC), 2010 International Conference on. Vol. 6. IEEE, 2010.

[7] Amiri, MahmoodAllameh, and HadiSerajzadeh. "QoS aware web service composition based on genetic algorithm." Telecommunications (IST), 2010 5th International Symposium on. IEEE, 2010.

[8] Wang, Lijuan, Jun Shen, and Jianming Yong. "A survey on bio-inspired algorithms for web service composition." Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on. IEEE, 2012.

[9] Bakhshi, Mahdi, and Mohsen Hashemi. "User-Centric Optimization for Constraint Web Service Composition using a Fuzzy-guided Genetic Algorithm System." arXiv preprint arXiv:1210.3604 (2012).

[10] Yang, Zongkai, et al. "A dynamic web services composition algorithm based on the combination of ant colony algorithm and genetic algorithm." Journal of Computational Information Systems 6.8 (2010): 2617-2622.

[11] Parejo, Jose Antonio, Pablo Fernandez, and Antonio Ruiz Cortés. "Qos-aware services composition using tabu search and hybrid genetic algorithms." Actas de los Talleres de lasJornadas de Ingenieríadel Software y Bases de Datos2.1 (2008): 55-66.

[12] Lécué, Freddy. Optimizing qos-aware semantic web service composition. Springer Berlin Heidelberg, 2009.

[13] Ko, Jong Myoung, Chang Ouk Kim, and Ick-Hyun Kwon. "Quality-of-service oriented web service composition algorithm and planning architecture." Journal of Systems and Software 81.11 (2008): 2079-2090.

[14] Blake, M. Brian, et al. "WSC-07: Evolving the web services challenge." E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, 2007. CEC/EEE 2007. The 9th IEEE International Conference on. IEEE, 2007.