Journal of Ubiquitous Systems & Pervasive Networks
Volume 11, No. 2 (2019) pp. 15-21

Optimizing Web Service Composition with Graphplan and Fuzzy Control

Guodong Fan, Ming Zhu, Xiaoliu Cui

College of Computer Science and Technology, Shandong University of Technology, Zibo, China

Abstract

With the development of cloud computing, more and more applications are posted online to provide services for users. Since user needs can be complex, an individual service will not able to meet the requirement. Web Service Composition composes multiple web services together to fulfil the complicated user requirement. While searching an optimal composition with both functional and non-functional requirements still is a challenging problem that needs to be addressed. QoS-aware web service composition is an NP-hard problem. To solve this problem, we design a system which combines GraphPlan with Fuzzy Control algorithm. Fuzzy Control is employed to generate overall QoS according to user preferences. In the forward phase of Graphplan, less competitive services are pruned according to the overall QoS. In the backward phase, services are selected according to functional goals and their overall QoS. Furthermore, case study and are performed, and the experimental results show that our approach improves the quality of service composition significantly compared with ordinary and Skyline approach.

Keywords: *fuzzy control, cloud computing, GraphPlan, QoS-aware service composition*

1. Introduction

In the era of cloud computing, users can access resources on demand through the network and pay for usage. At present, most of the cloud computing systems in the industry generally adopt the web service method in remote communication [1]. Web services are self-described software entities that can be published, located and invoked on the web [2]. Web services could help to satisfy specific customer needs as well as lower expenses on purchasing and maintaining local computing resources [3]. With the increasing complexity of user needs, an individual service would fail to meet the requirement. Web service composition involves the combination of many web services together to produce a more complex and useful service. Quality of Service (QoS) is used to assess the ability of a service provider to meet user needs. To maximize user satisfaction, researchers introduce QoS to web service composition, which refers to composition achieves desired functionality as well as optimizes QoS values. However, the QoS-aware web service composition has been considered as an NP-hard problem [4]. QoS-aware service composition can be formulated as single-objective and multiple-objective optimization problems [5].

To solve the problems, a number of research initiatives are proposed. These include Skyline operator [6], GraphPlan model [7], and others [8, 9]. However, Skyline cannot find Top-k web services that better meet user preferences and may fail to find a solution because available services may be filtered. Ordinary GraphPlan considers all possible combinations without filtering any service in the forward expand phase and cannot choose optimal solution in the backward search phase.

As a continuation of research [6, 7], we propose to utilize Fuzzy Control together with GraphPlan for QoS-aware web service composition. Fuzzy logic usually is considered as a ranking algorithm for selecting the best service in a pool of web services [10]. However, very few research efforts have been contributed in using this approach with GraphPlan in QoS-aware service composition. By using Fuzzy Control, some services with worse QoS in the forward expand and backward search phases of GraphPlan are filtered according to some criteria, which could reduce the searching space of services in the compositions. The experimental results show that the proposed method would get better services, response time and throughput, when compared with the ordinary and the Skyline approach. Especially, the main contributions of this paper are listed as follows:

(1) We create three fuzzy set for response time, throughput and overall QoS, respectively. The fuzzy Control is used to calculate the overall QoS of each service, and the users' preferences are modeled as *IF-THEN* rules.
(2) We use the overall QoS in the two phases of the Graphplan. In the forward phase, less competitive services are pruned according to the overall QoS. In the backward phase, services are selected according to functional goals and their overall QoS.

The paper is organized as follows. Section 2 describes preliminary knowledge. The architecture and algorithms are

* Ming Zhu. Tel.: +86-15689080816
E-mail: zhu_ming@sdut.edu.cn ,

DOI: 10.5383/JUSPN.11.02.003

introduced in Section 3. We demonstrate a case study in Section 4 to further explain our approach. We present experimental results in Section 5. Related work to this research is presented in section 6, and the conclusion is drawn in Section 7.

2. Preliminaries

In this section, we introduce necessary knowledge of QoS-aware web service composition and Fuzzy Control.

2.1. *The QoS-aware web service composition*

definition 2.1. A web service w is defined as a tuple (w_{in}, w_{out}, w_{QoS}) with the following components: w_{in} is a finite set of typed input parameters of w; w_{out} is a finite set of typed output parameters of w; w_{QoS} is a finite set of QoS values of w.
definition 2.2. A web service composition problem can be used with a tuple(S, C_{in}, C_{out}, Q) with the following components: S is a finite set of services; C_{in} is a finite set of typed input parameters; C_{out} is finite set of typed output parameters; Q is a finite set of quality criteria.
Plug-in matching degree is used to match services: two services can be connected if the input of a service is a subset of the output of the other service. This semantic model, borrowed from [11], is consistent with many proposed web service composition approaches.

Services are connected either in sequence or in flow control. Services in sequence are represented as (w_1; w_2; ...; w_n). Services in a flow control are represented as ($w_1||w_2||...||w_n$). Some quality criteria for a Web service w and their aggregated values are listed as below:

- Response time(R): the interval between the receipt of the end of transmission of an inquiry message and the beginning of the transmission of a response message to the station originating the inquiry.

$$\mathrm{R}(w_1; w_2; \ldots; w_n) = \sum R(w_i) \quad (1)$$
$$\mathrm{R}(w_1||w_2|| \ldots ||w_n) = \max\mathrm{R}(w_i) \quad (2)$$

- Throughput (T): the average rate of successful message delivery over a communication channel,e.g., .,10 successful invocations per second.

$$\mathrm{T}(w_1; w_2; \ldots; w_n) = \min\mathrm{T}(w_i) \quad (3)$$
$$\mathrm{T}(w_1||w_2|| \ldots ||w_n) = \min\mathrm{T}(w_i) \quad (4)$$

2.2. *The Fuzzy Control*

2.2.1. Fuzzy set and number

Fuzzy set and possibility theory was first proposed by Zadeh [12] and has been used to deal with vagueness or ambiguity information. It's known as a branch of mathematics currently. The fuzzy set defined in Definition 2.3 is represented by a membership function defined on the universe of discourse which maps the value to interval [0, 1]. A fuzzy number is a convex, normalized fuzzy set whose membership function is at least segmentally continuous and has the functional value μA(x) = 1 at precisely one element. A triangular fuzzy number can be defined by a triplet (a, b, c) with the following membership function specified in Definition 2.4.
definition 2.3. Let A be a mapping from domain X to [0,1], that is: A : X → [0, 1], x →A(x), where A is called fuzzy set on X, and A(x) is called membership function of fuzzy set A.

definition 2.4. Triangular fuzzy number is represented by the fuzzy set A= (a, b, c), where $a < b < c$ is defined on **R**. the function has the following form:

$$\mu(x) = \begin{cases} \dfrac{x-a}{b-a}, if\ a < x \le b \\ \dfrac{x-c}{b-c}, if\ b < x \le c \\ 0, otherwise \end{cases}$$

2.2.1. Fuzzy System

A fuzzy system mainly includes the following three parts shown in figure 1.

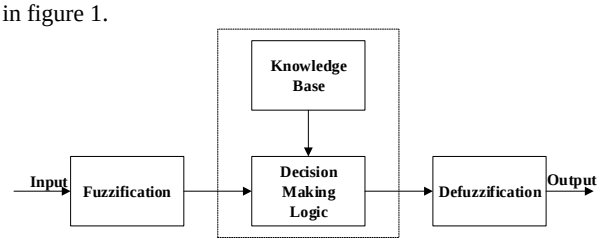

Fig. 1. The Diagram of Fuzzy Control System.

Firstly, the Fuzzification part converts the crisp input information from the actual system into the fuzzy values for each input fuzzy set. Secondly, the Decision-Making-Logic part determines how the fuzzy logic operations are performed with the *IF-THEN* rules based on the Knowlege Base. Thirdly, the defuzzification part maps the fuzzy number that the fuzzy system outputs to a crisp value. There are many different ways to transform a fuzzy number into a crisp value [13], such as COG (Center Of Gravity), RCOM (Random Choice Of Maximum), FOM (First of Maximum), QM (Quality Method), EQM (Extended Quality Method), and COA (Center Of Area).

3. Architecture and Algorithm

In this section, we introduce the architecture of the system illustrated in figure 2. There are three major modules listed as follows.

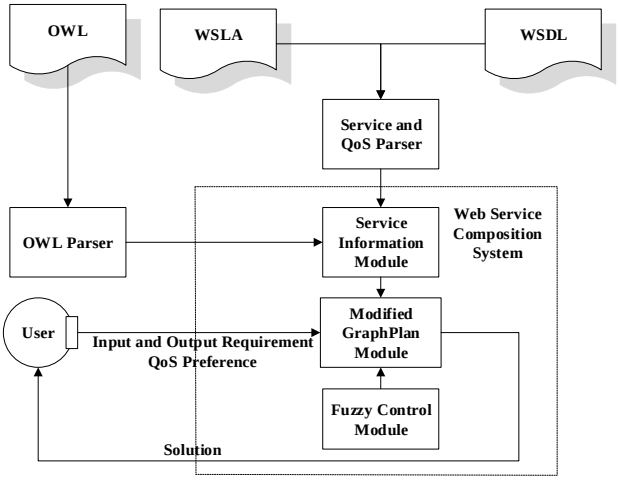

Fig. 2. System Architecture.

Service Information Module. This module reads the inputs such as functional, non-functional, and ontology information of web services generated by Service, QoS and OWL Parser. Once, all the information of web services is retrieved, it is stored and can be used by the system.

Fuzzy Control Module. This module allows users to choose different fuzzy rules used in service compositions by their preferences. For example, if a user demands high priority

on response time of the service composition, the system sets corresponding configuration with higher weight for time. Besides, evaluation rules and the process of Fuzzy Control are defined in this module. In Section 3.1, detailed information of this module is provided.

Modified GraphPlan Module. This module is modified based on GraphPlan by fitting Fuzzy Control into it. It is the key module that is used to find solutions of QoS-aware services compositions. It receives the input information such as user requirements, web services information, and fuzzy configuration defined by users. Then, by using modified GraphPlan and Fuzzy Control, it performs the forward expand to construct a search graph, and carries out backward search to retrieve a solution. If it cannot find the goal concepts of service compositions, it will switch the ordinary approach to generate solutions. Detailed explanation of this module is provided in Section 3.2.

3.1. *Fuzzy Control*

Based on the fuzzy system in section 2.2, the Fuzzy Control used in this paper consists of three aspects as below.

Fuzzification. Fuzzification is the process of decomposing a system input and/or output into one or more fuzzy sets. Many types of cures can be used, but triangular shaped membership function is one of the most common. Each fuzzy set spans a region of input(or output) value graphed with the membership. Any particular input is interpreted from this fuzzy set and a degree of membership is interpreted. There are three types of crisp variables, response time, throughput, and overall QoS value. An overall QoS value is used to measure QoS of web services according to the *IF-THEN* rules defined in the following paragraph. The higher value of overall QoS indicates the better quality of the service. Variables are mapped to degrees of membership for linguistic terms using the triangular membership functions. The response time is fuzzified with the linguistic terms: "long", "medium" and "short", the throughput is fuzzified with the linguistic terms: "high", "medium" and "low", and the overall QoS is fuzzified with the linguistic terms: "good", "middle" and "bad".

IF-THEN rules. Fuzzy *IF-THEN* rules allow an effective way to evaluate approximations of desired attribute values. Using these rules, the output fuzzy variables are obtained through fuzzy reasoning. The *IF* part contains various QoS standard member functions of a web service, and the *Then* part contains one of the member functions that reflects the overall QoS status of the services. An example of *IF-THEN* rules is defined as follows: IF response IS poor AND throughput IS not good, THEN overall QoS IS bad;

Defuzzification. The output variable is a fuzzy variable, but the output result requires a crisp value. We use the center of gravity (COG) method to defuzzify QoS and obtain an crisp output result.

To illustrate the process of Fuzzy Control, an example of calculating Overall QoS by using it is provided in Section 4.

3.2. Modified GraphPlan

The Graphplan approach is modified by adding service prune in the forward phase and selecting optimal candidate services in the backward phase, according to functional requirements and Overall QoS. Therefore, a local optimal solution is obtained in polynomial time. A planning graph contains two kinds of layers: the proposition (*P*) layers contain concepts and action (*A*) layers contain services. Layers of the planning graph form an alternative sequence of proposition layers and action layers. The *prunedPool* represents a set of services that are pruned. The *selectedServices* represents a set of selected service in the current layer.

In each forward layer, we search the services whose input concepts are all contained in *P* layer and exclude the services in *prunedPool*, then, calculate service's overall QoS value by Fuzzy Control module and prune the service below the threshold (Algorithm 1 line 3-7). Finally, standard forward expand is executed to obtain the goal. Repeat this process until goal is contained or a fixed layer is reached.

In the backward search phase, we loop from the last layer to the first layer. In each layer, we select services that cover most concepts and with best overall QoS value (Algorithm 2 line 6-13). This ensures that each selection is optimal in current state.

Algorithm 1 Service Prune

Input：*serviceLayer, prunedPool, threshold*;

Output: *serviceLayer, prunedPool*;

1. *serviceLayer* ← *serviceLayer* \ *prunedPool*
2. **for** each *service* in *serviceLayer* **do**
3. *caculateQoS(service)*
4. **if** *service.getScore* < *threshold* **then**
5. *serviceLayer* ← *serviceLayer* \ *service*
6. *prunedPool* ← *prunedPool* ∪ *service*
7. **end if**
8. **end for**
9. **return** *serviceLayer*, *prunedPool*

Algorithm 2 Service Select

Input：*serviceLayer, goal*;

Output: *selectedServices*;

1. **While** *serviceLayer* ≠ Ø && *goal* ≠ Ø **do**
2. *maxCurrentService* ← Ø
3. *maxCurrentSize*, *maxCurrentScore* ← 0
4. **for** each *service* in *serviceLayer* **do**
5. *currentSize* = *count*($service_{out} \cap goal$)
6. **if** *currentSize* > *maxCurrentSize* **then**
7. *maxCurrentService* ← *service*
8. *maxcurrentSize* ←*currentSize*
9. *maxcurrentScore* ←*service.getScore*
10. **else if** *currentSize*= *maxCurrentScore* && *service.getScore* > *maxCurrentScore* **then**
11. *maxCurrentService* ← *service*
12. *maxCurrentScore* ← *service.getScore*
13. **end if**
14. **end for**
15. **if** *maxCurrentService* = 0
16. break
17. **end if**
18. *selectedServices* ←*selectedServices* ∪ *maxCurrentService*
19. *selecteLayer* ←*selecteLayer* \ *service*
20. *goal* ←*goal* \ $maxCurrentService_{out}$
21. **end while**
22. **return** *selectedServices*

4. Case study

In this section, we present a meaningful example to illustrate the proposed algorithms. In this example, service information is

shown in Table 1, which contains input, output concepts, response time and throughput. We calculate the Overall QoS of each service by using fuzzy control and then obtain a solution by using Modified Graphplan. We assume that the user gives inputs A, B, C and output F to find a set of Web service composition.

Table 1. A set of available services.

w_i	Input	Output	Response Time	Throughput
w_1	A, B	D	10	19000
w_2	B, C	E	20	10000
w_3	C, D	E	100	10000
w_4	E	F	400	8000
w_5	C	D	450	1000

4.1. *Fuzzification*

We use triangular shaped membership functions to interpret response time, throughput and overall QoS.

There are three fuzzy sets for response time defined as follows.

TERM good := (0, 1) (250, 0) ;
TERM middle := (0, 0) (250, 1) (500, 0);
TERM poor := (250, 0) (500, 1);

There are three fuzzy sets for throughput defined as follows.

TERM poor := (0, 1) (10000, 0) ;
TERM middle := (0, 0) (10000,1)(20000,0);
TERM good := (10000, 0) (20000, 1);

There are three fuzzy sets for overall QoS defined as follows.

TERM bad := (0,1) (0.5,0);
TERM middle := (0,0) (0.5,1) (1,0);
TERM good := (0.5,0) (1,1);

4.2. *Fuzzy Rule*

The *IF-THEN* rules used in this paper is defined as follows.

RULE 1 : IF response IS poor AND throughput IS not good THEN QoS IS bad;
RULE 2 : IF response IS good and throughput IS NOT poor THEN QoS IS good;
RULE 3 : IF throughput IS good and response IS NOT poor THEN QoS IS good;
RULE 4 : IF response IS middle AND throughput IS middle THEN QoS IS middle;
RULE 5 : IF response IS good AND throughput IS poor THEN QoS IS middle;
RULE 6 : IF response IS poor AND throughput IS good THEN QoS IS middle;
RULE 7 : IF response IS not good AND throughput IS poor THEN QoS IS bad;

4.3. *Defuzzification*

We use the COG method to defuzzify QoS and obtain a crisp output result. To illustrate the process of Fuzzy Control, we use MATLAB to simulate it. For example, Figure 3 depicts the membership function of response time, and Figure 4 shows how the overall QoS is produced from response time and throughput by fuzzy reasoning. Specifically, response time is 100 and throughput is 10000, and then overall QoS is 0.591. The surface view is shown in Figure 5.

4.4. *Obtain Services with Overall QoS*

Table 2. Ranking of Services.

w_i	Overall QoS
w_1	0.794
w_2	0.758
w_3	0.591
w_4	0.409
w_5	0.258

Finally, we obtain services with overall QoS, which are used in modified Graphplan shown in Table 2. Usually the amount of data is large, we only calculate the services used.

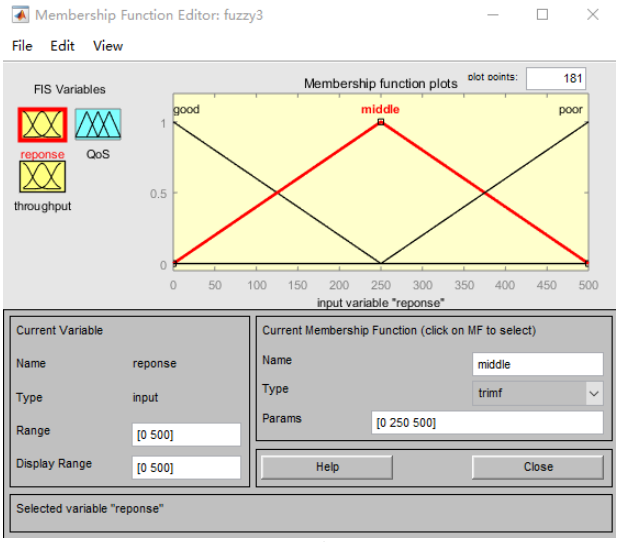

Fig. 3. Membership Function of Response Time.

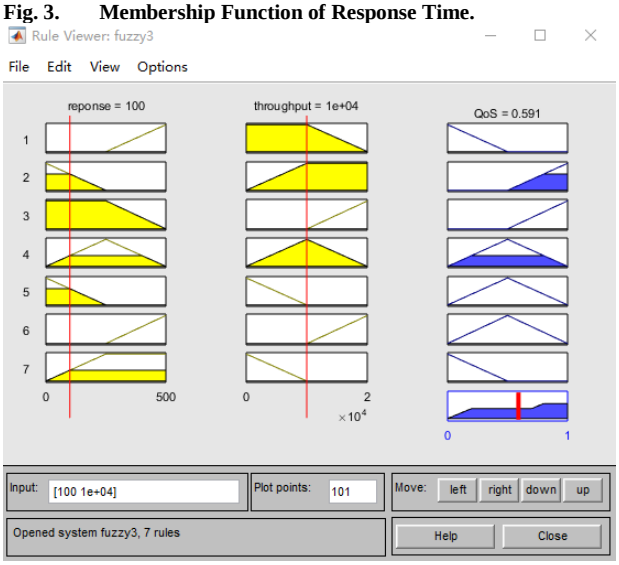

Fig. 4. Rule.

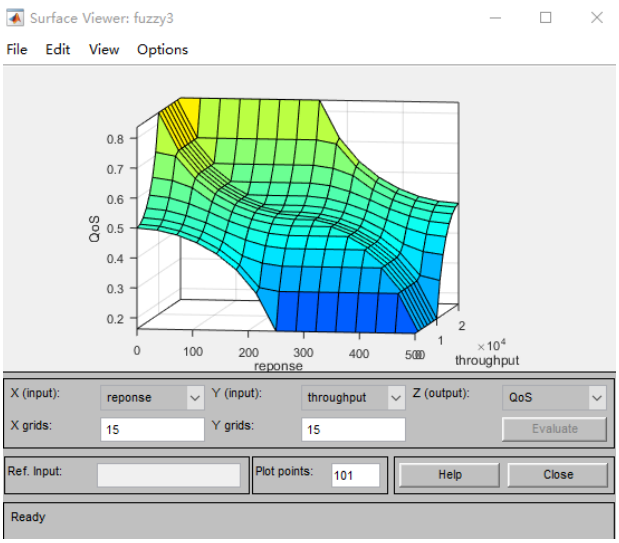

Fig. 5. Surface View.

4.4. *Obtain a Solution Using Modified Graphplan*

After the calculation of the Overall QoS, the solution is obtained by using the Modified Graphplan.

The forward expand phase builds the planning graph from the initial state. The proposed approach prunes less competitive services, such as w_5. Then the subsequent processes of w_5, represented by dotted lines in the Figure 6, are virtually no longer available.

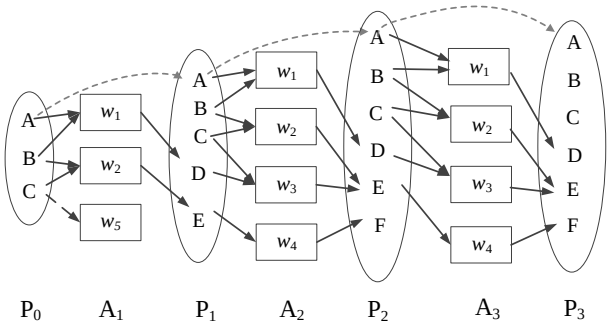

Fig. 6. Forward Phase of Modified Graphplan.

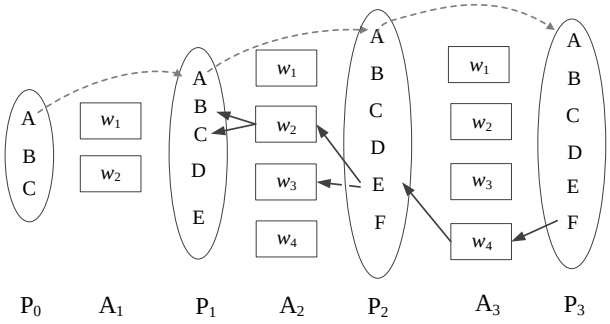

Fig. 7. Backward Phase of Modified Graphplan.

In the process of backward search phase, services are selected, which loop from the last layer to the first layer shown in Figure 7 according functional and non-functional requirements.

In A_3 layer, w_4 is selected according to the functional requirement of F, and E which need to be searched in the next layer is the input concept of w_4. In A_2 layer, w_2 and w_3 have same output E, but the overall QoS of w_2 is better than w_3. So w_2 is added in the solution. w_2 is produced by B and C which are initial concepts. As a result, (w_2; w_4) is the solution. According to Equation 1-4, response time of the composition (w_2; w_4) is 420 and throughput is 8000.

5. Experimental Results

We run experiments on a computer with the following configuration: 1) CPU: Intel(R) Core(TM) i5-7200U 2.50GHz 2.71GHz, 2) RAM: 8.00GB DDR4-2400 3) Hard disk: LITEON T11 256GB, TOSHIBA MQ01ACF050 500GB 4) Operating System: Windows 10 professional 64-bit. The algorithms are implemented in Java.

5.1. *Dataset*

We use TestsetGenerator2009 [14] to generate five datasets for our experiments. Dataset 1 has 1000 services, 3000 concepts and 398 lines of BPEL solution. Dataset 2 has 5000 services, 15000 concepts and 955 lines of BPEL solution. Dataset 3 has 10000 services, 30000 concepts and 1231 lines of BPEL solution. Dataset 4 has 15000 services, 45000 concepts and 1741 lines of BPEL solution. Dataset 5 has 20000 services, 60000 concepts and 1876 lines of BPEL solution.

Table 3. Results with The WSC09 Data Sets.

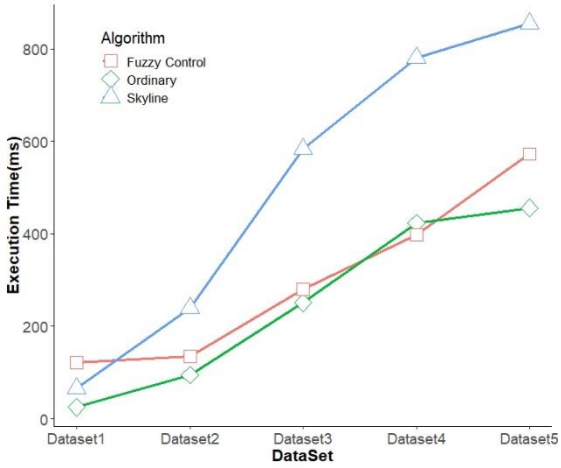

Fig. 8. Forward Phase Execution Time.

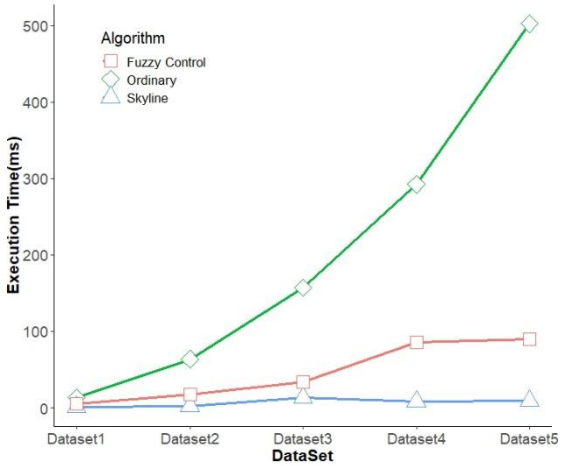

Fig. 9. Backward Phase Execution Time.

5.2. *Performance analysis*

In order to illustrate the effectiveness of our approach, the results of modified GraphPlan with Fuzzy Control is compared with that of the ordinary and Skyline approach. The comparisons are shown in Table 3, where #Services represents the number of services, #Resp represents response time, and #Tpt represents throughput. The final response time and throughput of the combined services are calculated according to the equations in Section 2.1.

Service Number. In the experiment, it can be seen from Table 3 that the Skyline approach obtains the least number of services in the service composition, and the fuzzy algorithm is nearly half the service of the ordinary approach. The ordinary approach has 270 services in dataset5 which contain a large number of redundant nodes.

The Response Time & Throughput. In the experiment, compared with the ordinary and the Skyline approach shown in the Table 3, the Fuzzy approach returns a solution with better response time and throughput.

Execution Time. In the experiment, Figure 8 shows that in the forward phase, using the Skyline algorithm is the longest. Because it is very time-consuming to calculate the

Algorithm	Dataset1 #Services/#Resp/#Tpt	Dataset2 #Services/#Resp/#Tpt	Dataset3 #Services/#Resp/#Tpt	Dataset4 #Services/#Resp/#Tpt	Dataset5 #Services/#Resp/#Tpt
Ordinary	**11/3020/5000**	**50/4460/1000**	**84/7910/1000**	**96/7870/1000**	**270/14860/1000**
Fuzzy	**11/2230/5000**	**29/2050/3000**	**33/3590/2000**	**46/4760/3000**	**48/4200/1000**
Skyline	**9/2360/5000**	**20/3680/2000**	**29/5170/2000**	**42/5420/1000**	**45/4740/1000**

the Skyline Services. It consumes more time to use fuzzy operations, but the fuzzy approach can prune less competitive services, which could significantly reduce execution time. Due to the above reasons, fuzzy approach consumes approximately the same amount of time as ordinary approach. Figure 9 shows that ordinary approach consumes more time than Fuzzy and Skyline approach. The fuzzy approach takes a little more time than the ordinary one.

In a conclusion, the Fuzzy approach can select the best service in each layer according to the overall QoS value and the functional requirements, and it can prune bad services in the forward phase. Ordinary and Skyline approach can only randomly choose services from a specific collection. The experiments show that Fuzzy Control approach works better than others. In addition, service selection based on user preferences can be modeled with fuzzy *IF-THEN* rules. The limitation of our approach is that the final solution that has local optimization may not necessarily maintain global optimization.

6. Related Work

A great deal of work has been done in the theory and practice of web services. In this section, we discuss automatic web service composition. Then, we review the work related to fuzzy logic decision making in QoS.

Web Ontology Language (OWL) is a semantic web language designed to represent the meaning of terms and relationships between them. OWL expresses preconditions and effects of web services as “concepts” by using ontologies [15, 16, 17]. AI planning algorithms are frequently used to solve the QoS-aware Web service composition problem [7]. Normally, the planning algorithms stop at the first found feasible solution, which corresponds to the shortest plan. Huang et al. [18] proposes an automatic service composition method based on depth-first searching for the Top-k QoS service composition issue based on the MapReduce framework. Discovering and composing non-electronic services based on contexts can be found [19] based on real business. Li et al. [20] propose a novel relational database approach for automated service composition. All possible service combinations are generated beforehand and stored in a relational database. When a user request comes, SQL queries are composed to search in the database and return the best QoS solutions. A non-redundant web services composition search system called NRC is proposed [21], which is based on a two-phase algorithm.

Fuzzy logic is an approximate reasoning technique applied to handle uncertainty and support decision making. Fuzzy logic has been used as supporting tool in the area of service selection, discovery and composition. Avila and Djemame present an adaptation approach which employs fuzzy logic to optimize service selection [22]. More specifically, two fuzzy support systems are proposed in this paper. One for assessing QoS values of the composition in each step. The other is used to determine weights of QoS criteria in the service selection process. Paper [23] proposes a FQ (Fuzzy-QoS) architecture which handles uncertainty of both vague user preferences and service descriptions. Zhao et al. firstly model the multi-objective optimization problem with a weighted Tchebycheff distance, then, they propose a fuzzy preference model to represent preference order of multiple QoS criteria, finally, they present two algorithms for service composition [24]. A new fuzzy hybrid ranking technique is proposed [25], which is based on a linear combination of two new ranking techniques: an objective Fuzzy Distance Correlation Ranking Technique (FDCRT) and a subjective Fuzzy Interval-based Ranking Technique (FSIRT).

7. Conclusion and Future work

In this paper, we propose a method to optimize Web Service Composition with Graphplan and Fuzzy Control. The Fuzzy Control is used to generate the overall QoS according to response time, throughput and user preferences. We use the overall QoS in the two phases of the Graphplan. Less competitive services are pruned out according to their overall QoS, in the forward expand phase, and services are selected according to their overall QoS and function requirements, in the backward search phase. Finally, we compare the experimental results of the proposed method with that of the ordinary and Skyline approach. The comparison indicates that the proposed method has better performance in nonfunctional requirements and execution time.

As future work, instead of doing a simulation experiment on a personal computer, we plan to improve our web service composition system in a number of ways. Firstly, we will build a real-life service composition application to fulfil user requirements. Secondly, we will deploy our system to the cloud platform to improve performance. Thirdly, our system will use the micro-service architecture to improve the scalability. With the above improvements, our web service composition system will be more valuable.

References

[1] A. Bastia, M. Parhi, B. Pattanayak, and M. Patra. Service composition using efficient multi-agents in cloud computing environment. In L. Jain, S. Patnaik, and N. Ichalkaranje, editors, Intelligent Computing, Communication and Devices, volume 308, pages 357 – 370. Springer, New Delhi, 2015. https://doi.org/10.1007/978-81-322-2012-1_37

[2] M. Papazoglou. Web Services: Principles and Technology. Prentice Hall, 2011.

[3] A. Vakili and N. J. Navimipour. Comprehensive and systematic review of the service composition mechanisms in the cloud environments. Journal of Network and Computer Applications, 81:24–36, 2017. https://doi.org/10.1016/j.jnca.2017.01.005

[4] G. Canfora, M. Di Penta, R. Esposito, and M.L. Villani. An approach for QoS-aware service composition based on genetic algorithms. In Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, GECCO ’05, pages 1069–1075. ACM, 2005. https://doi.org/10.1145/1068009.1068189

[5] A. Ramrez, J. Antonio Parejo, J. Ral Romero, Sergio Segura, and Antonio Ruiz-Corts. Evolutionary composition of qos-aware webservices. Expert Syst. Appl., 72(C):357–370, 2017. https://doi.org/10.1016/j.eswa.2016.10.047

[6] J. Li, Y. Yan, and D. Lemire. Scaling up web service composition with the skyline operator. In IEEE International Conference on Web Services, ICWS 2016, San Francisco, CA, USA, June 27 - July 2, 2016, pages 147–154, 2016. https://doi.org/10.1109/ICWS.2016.27

[7] Y. Yan and M. Chen. Anytime qos-aware service composition over the graphplan. Service Oriented Computing and Applications, 9(1):1–19, Mar 2015. https://doi.org/10.1007/s11761-013-0134-6

[8] J. Li, Y. Yan, and D. Lemire. A web service composition method based on compact k2-trees. In 2015 IEEE International Conference on Services Computing, pages 403–410, 2015. https://doi.org/10.1109/SCC.2015.62

[9] J. Li, Y. Yan, and D. Lemire. Full solution indexing for top-k web service composition. IEEE Transactions on Services Computing, pages 1–1, 2016.

[10] H. Yahyaoui, K. Almatori, and M. Almulla. A qos-based fuzzy model for ranking real world web services. In 2011 IEEE International Conference on Web Services(ICWS), pages 203–210, 2011.

[11] S. Bleul, T. Weise, and K. Geihs, “The web service challenge - a review on semantic web service composition,” Electronic Communications of the EASST, vol. 17, 2008.

[12] L. A. Zadeh. Fuzzy sets. Information & Control, 8(3):338–353, 1965. https://doi.org/10.1016/S0019-9958(65)90241-X

[13] Werner Van Leekwijck and Etienne E. Kerre. Defuzzification: criteria and classification. Elsevier North-Holland, Inc., 1999. https://doi.org/10.1016/S0165-0114(97)00337-0

[14] WS-Challenge. Testsetgenerator2009 [online]. available: https://code.google.com/p/ wsc-pku-tcs/downloads/list, 2010.

[15] S. Bechhofer, F. Harmelen, J. Hendler, and I.Horrocks. OWLWeb Ontology Language Reference, 2009.

[15] J. K. Lee and M. M. Sohn. The extensible rule markup language. Commun. ACM, 46(5):59–64, May 2003. https://doi.org/10.1145/769800.769802

[16] D. Dou, D. McDermott, and P. Qi. Ontology Translation on the Semantic Web, pages 35–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. https://doi.org/10.1007/978-3-540-30567-5_2

[18] L. T. Huang, S. G. Deng, K. Dai, et al. Automatic Service Composition in Parallel with MapReduce. Acta Electronica Sinica, 2012, 40(7):1397-1403.

[19] Y. Zhang, J. Wang and Y. Yan, "Context-Aware Generic Service Discovery and Service Composition," 2014 IEEE International Conference on Mobile Services, Anchorage, AK, 2014, pp. 132-139. https://doi.org/10.1109/MobServ.2014.27

[20] J. Li, Y. Yan, and D. Lemire. "Full Solution Indexing for top-K Web Service Composition." IEEE Transactions on Services Computing PP.99(1939):1-1.

[21] J. Kwon and D. Lee. "Non-redundant web services composition based on a two-phase algorithm." Data & Knowledge Engineering 71.1(2012):69-91. https://doi.org/10.1016/j.datak.2011.08.002

[22] S. d. G. Avila and K. Djemame, "Fuzzy Logic Based QoS Optimization Mechanism for Service Composition," 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, Redwood City, 2013, pp. 182-191.

[23] I. Sora and D. Todinca. Dealing with fuzzy QoS properties in service composition. In 2015 IEEE 10th Jubilee International Symposium on Applied Computational Intelligence and Informatics, pages 197–202, 2015. https://doi.org/10.1109/SACI.2015.7208198

[24] X. Zhao, L. Shen, X. Peng, and W. Zhao. Toward SLA-constrained service composition: An approach based on a fuzzy linguistic preference model and an evolutionary algorithm. Information Sciences, 316:370–396, 2015. https://doi.org/10.1016/j.ins.2014.11.016

[25] A. Mohammed , H. Yahyaoui , and K. Al-Matori . "A new fuzzy hybrid technique for ranking real world Web services." Knowledge-Based Systems 77:1-15, 2015. https://doi.org/10.1016/j.knosys.2014.12.021