Journal of Computational Information Systems 6:8(2010) 2617-2622
Available at http://www.Jofcis.com

# A Dynamic Web Services Composition Algorithm Based on the Combination of Ant Colony Algorithm and Genetic Algorithm

Zongkai YANG, Chaowang SHANG†, Qingtang LIU, Chengling ZHAO

*Research Center for Education Information on Technology,Central China Normal University*
*Wuhan 430079, China*

**Abstract**

Web services composition has received much interest to support business-to-business or enterprise application integration. Web services with the same functions and different QoS are increasing with the proliferation of SOA（Services Oriented Architecture）. It is needed to select a best composition schema from numerous schemas in order to maximize the satisfaction and meet the QoS requirements of users. In this paper, we propose a dynamic web services composition algorithm (ACAGA_WSC) based on the combination of the ant colony algorithm and the genetic algorithm. ACAGA_WSC transforms the problem of selecting optimal execution schema for composite web service into selection of the optimal path in the weighted directed acyclic graph. The experimental results indicated the validity and efficiency of ACAGA_WSC.

*Keywords:* Web Services; QoS; Web Services Composition; Ant colony Algorithm; Genetic Algorithm

## 1. Introduction

Web services composition is used to compose the existed web services to get a new value-added function, which has become one of the most features of web services [1]. With the number of available web services increasing, lots of web services provide overlapping or identical functionality and different Quality of Services (QoS). A choice needs to be made to determine which web services are to participate in a given composite web service.

Many proposals of web services composition method have been presented in recent years. In [2], the authors use the integer programming algorithm. [3] regards the web services dynamic selection as a multiple-choice knapsack problem, and uses the exhaustive method and dynamic programming for solution. But the paper does not give a concrete solution of dynamic programming and the solving process is complex. Rainer .B et al. [4] propose to use the heuristics to obtain the optimal solution and has a good execution efficiency, yet its testing data can only meet a user constraint of 68%. [5] regards web services dynamic selection as a Markov decision-making process that provides optimal QoS of web services composition. R.Aggarwal et al.[6] employs a web service dependence and user profile based service selection constraint calculation model, however, the model has a low efficiency under the dynamic real-time environment with a large number of web services. Genetic algorithm and immune algorithm provide good models to solve the difficult combinatorial optimization problems and have good performances in web services composition. However, for the genetic algorithm based method [7], the character of “prematurity” obstructs the algorithm from further improving, and it also makes us incapable of getting the best composite service. Gao et al. [8] introduces an immune algorithm based method,

† Corresponding author.
*Email address*: shangchaowang200650@yahoo.com.cn (Chaowang SHANG)


August, 2010

whereas, due to underutilization of feedback information, the massive useless redundant iterations result in low solution efficiency of the algorithm.

The current approaches can solve some key issues in web services composition and give great illuminations to this paper, however, none of them gives an effective solution to address the issues of low efficiency in the large solution space. In this paper, we present an ant colony algorithm [9] and genetic algorithm [10] combination based algorithm (ACAGA_WSC). By means of genetic algorithm, ACAGA_WSC overcomes the shortcomings of the ant colony algorithm, and achieves better efficiency and converging speed. Experiment shows that the new algorithm gives better performance for the services composition problem.

The remainder of this paper is organized as follows. Next section gives short introduction of the requirements of web services composition. Section 3 discusses the ant colony algorithm based web services composition model. Section 4 describes design of ACAGA_WSC. In section 5 we evaluate our algorithm by an experiment. Finally, Section 6 concludes the paper.

## 2. Requirements of Web Services Composition

### *2.1. Model of Web Services Composition*

The model of the composite web service is shown in Fig.1. Let *CWS* be a composite web service consists of *m* service tasks. CWS can be expressed as: $CWS=\{ S_1, S_2.. ,S_m \}$, where $S_i$ $(1 \le i \le m)$ is the $i_{th}$ service task.

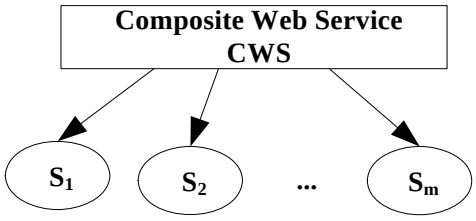



Fig.1 Model of Composite Service

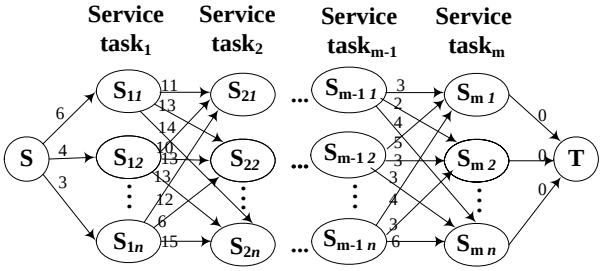



Fig.2 Directed Acyclic Graph Composed by Candidate Services

There are *n* candidate atomic services with the same functions and different QoS for each service task. As shown in Fig.2, problem of web services composition can be solved by a directed acyclic graph and there will be $n^m$ alternative services composition schemas. Implementation of optimal web services composition can be transformed to the problem of finding an optimal path in the directed acyclic graph.

It is crucial that the composite service should satisfy certain QoS constrains in practical applications, such as time, cost, reliability, availability and reputation, etc. We use QoS to evaluate the web services and the execution schema of composite web service. The composite web service composing from the atomic web services on the optimal path should best meet the QoS requirements of users.

There will be a large computing to find the optimal solution when we use exhaustive method even if *m* and *n* have a low value. For the services composition which has a depth of *d* and a complexity of $O( n^{m^d} )$, the long-time computing is unacceptable.

### *2.2. Problem Description of Web Services Composition*

In the directed acyclic graph, except for the start point *S* and target point *T*, we can represent the candidate atomic web services as nodes, the path from a candidate atomic web service in one service task to the candidate atomic web service in the other task as an edge, and QoS of the selected atomic web service as weight of the edge in the graph. Therefore, problem of web services composition can be abstracted to a weighted directed acyclic graph. The composite web service can be denoted by $G= (V, A, QoS)$, where $V=\{S, v_1, v_2, v_3,\ldots,v_n,\ldots,T\}$ represents the set of atomic web services, $v_i$ $(i=1, 2,\ldots,n\ldots)$ stands for the nodes

of atomic web service and $A=\{(v_i,v_j)|v_i,v_j \in V,\ i \neq j\}$ stands for the set of arcs. $QoS_i$ represents the QoS evaluation index of web service $v_i$, it can be used to determine whether $v_i$ is the optimal candidate atomic web service in the service selection process for composition.

Therefore, the process of services composition can be expressed as: finding a path which has the optimal sum of QoS value from the start point *S* to the target point *T* in the weighted directed acyclic graph.

## 3. Ant Colony Algorithm Based Web Services Composition Model

In ant colony algorithm based web services composition model, the appointed start point *S* is set as the ants nest and target point *T* is set as the food source. Then, problem of web services composition can be abstracted to finding a path from the ants nest to the food source.

We use $\tau_0$ to denote the initial pheromone concentration value on each path and $\tau_{ij}(t)$ to denote the pheromone concentration value on path(i,j) at the $t_{th}$ time. Then the pheromone concentration on the path(i,j) at $(t+1)_{th}$ time can be expressed as:

$$\tau_{ij}(t+1) = \rho \bullet \tau_{ij}(t) + \sum_{k=1}^{l} \Delta\tau_{ij}^{k} \tag{1}$$

In formula (1), constant $\rho$ stands for the volatile coefficient of pheromone that between 0 and 1, $\Delta\tau_{ij}^{k}$ stands for the pheromone concentration increment between $t_{th}$ time and $(t+1)_{th}$ time on path(i,j) for the $k_{th}$ ant and *l* stands for the number of ants. In web services composition problem, $\Delta\tau_{ij}^{k}$ is directly proportional to the QoS of atomic web services.

In order to utilize the global information and guarantee limitation of residual pheromone accumulation, we employ the ant quantity model. The pheromone concentration increment can be denoted by:

$$\Delta\tau_{ij}^{k}(t,t+1) = f(t)/C \tag{2}$$

Where *C* is a constant, $f(t)$ stands for the QoS measurement index of selected atomic web services for the composite web service, which can be expressed as:

$$f(t) = \sum_{i=1}^{h}\sum_{j=1}^{m} w_i Q_{ij} \tag{3}$$

In formula (3), $w_i$ denotes the weight value of the $i_{th}$ QoS attribute of one atomic web service, $0 \le w_i \le 1$, $\sum_{i=1}^{h} w_i = 1$, $Q_{ij}$ denotes the $i_{th}$ QoS attribute value of the atomic web service selected by the $j_{th}$ service task for the composite service. *h* stands for the number of QoS attributes and *m* stands for the number of tasks in the composite web service.

The selection rule of next node $node_u$ at the $t_{th}$ time for the $k_{th}$ ant can be expressed as:

$$s = \arg\max\{[\tau_{iu}(t)]^{\alpha}[\eta_{iu}(t)]^{\beta}\},\ u \in allowed_k(i),\ q \le q_0 \tag{4}$$

In formula (4), *q* is a random number between 0 to 1, $q_0$ stands for the selective factor which satisfies $0 \le q_0 \le 1$, $\eta_{iu}(t)$ stands for the heuristic factor between $node_i$ to $node_u$ in the $t_{th}$ cycle. $\alpha$ stands for the relative importance of residual information, $\beta$ stands for the relative importance the heuristic factor.

$\eta_{iu}(t)$ can be expressed as $\eta_{iu}(t) = k \bullet f(t)$ in ACAGA_WSC, where *k* is a constant.

$allowed_k(i) = \{node_1, node_2, ..., node_m\} - tabu_k$ represents the set of target that will be selected by the $k_{th}$ ant; $tabu_k$ is a tabu list that records the nodes already be selected.

Then probability for the $k_{th}$ ant to choose $node_j$ as the next target at the $t_{th}$ time can be denoted by:

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{j \in allowed_k(i)} [\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}},\quad j \in allowed_k(i),\ q > q_0 \tag{5}$$

## 4. Algorithm Design of ACAGA_WSC

Selection of the four parameters: $\alpha, \beta, \rho$ , and $q_0$ has great effect on the ant colony algorithm, however, the ant colony algorithm does not give theoretical guidance on how to select the four parameters but gives some experiential values. For example, in the ant colony algorithm based QoS multicast of network information transmission problem, the values of $\alpha, \beta$ are usually set between 0.05 and 0.2, the value of $\rho$ is usually set as 0.1 and $q_0$ is set as 0.9. Yet, these parameters are not always suitable to solution of other problems, such as web services composition.

Genetic algorithm is an efficient search method based on principles of population genetics, i.e. mating, chromosome crossover, gene mutation, and natural selection. In ACAGA_WSC, we use genetic algorithm to set the key parameters of the ant colony algorithm to obtain the great efficiency.

Basic flow of ACAGA_WSC is as follows:

**Step1.** Initialization

Randomly generate twenty 28 bit coding chromosomes.

**Step2.** Nodes retrenchment

Retrench unqualified nodes or edges in the weighted directed acyclic graph.

**Step3.** Chromosomes selection

Randomly select 4 chromosomes from the 20 chromosomes.

**Step4.** Fitness value computation

Each chromosome corresponds with a set of $(\alpha, \beta, \rho, q_0)$ parameter combination for the ant colony algorithm. Put *l* ants on the start point *S*, set pheromone concentration value $\tau_0$ = const (a constant). Find a QoS based superior path by means of the state transition rule, pheromone concentration local update rule and pheromone concentration global update rule. Select the optimum fitness value by comparison of paths that the *l* ants have gone through, set a variable *number* to record the times for obtaining the optimum path (with the highest value of fitness value) for ants, repeat **Step4** until the 4 chromosomes get the fitness value respectively.

**Step5.** Selection, crossover and mutation operation of chromosomes

Compare the values of optimum fitness and *number* of each chromosome, sort the 4 selected chromosomes by bubbling up method. Higher value of fitness means better path and better chromosome. Equal fitness values of two chromosomes denote the equal path. Then we can compare the value of *number* (the times for selecting the path). Higher value of *number* means higher hit rate of choosing the optimum path, that is, the chromosome with higher value of *number* is better than the other one.

After the sort operation of chromosomes, it is the crossover and mutation operation of the genetic algorithm. In ACAGA_WSC, select the 2 better chromosomes from the 4 selected chromosomes as father and perform crossover and mutation operations on them with a definite probability. Replace the other two inferior chromosomes with the two new child chromosomes. Reorganize the 4 chromosomes, place them back to the original 20 chromosomes and complete a cycle.

**Step6.** Repeat **Step3**, **Step4** and **Step5** until the cycle termination conditions are fulfilled

After multiple cycles, we will get the chromosome which has the best performance and contains the optimal $(\alpha, \beta, \rho, q_0)$ parameter combination for the ant colony algorithm. On the other hand, we can also search for the optimal path which meets the requirement of QoS and has the least cost during the cycles.

## 5. Experimentation

In order to show the performance of ACAGA_WSC, we conducted a comparison experiment. The experiment was performed on a personal computer with 1.8 GHz CPU and 512 MB RAM. Generally, cost and time are two primary factors that customers are concerned about [7]. The performance is measured by the execution schema with optimal time under a given cost constraint (*CC*). Thus the optimal services composition schema must satisfy the following expressions:

$$\begin{cases} QoS_{time} = \max \sum_{i=1}^{m} 1/time_i \\ QoS_{\cos t} = \sum_{i=1}^{m} \cos t_i \le CC \end{cases} \quad (6)$$

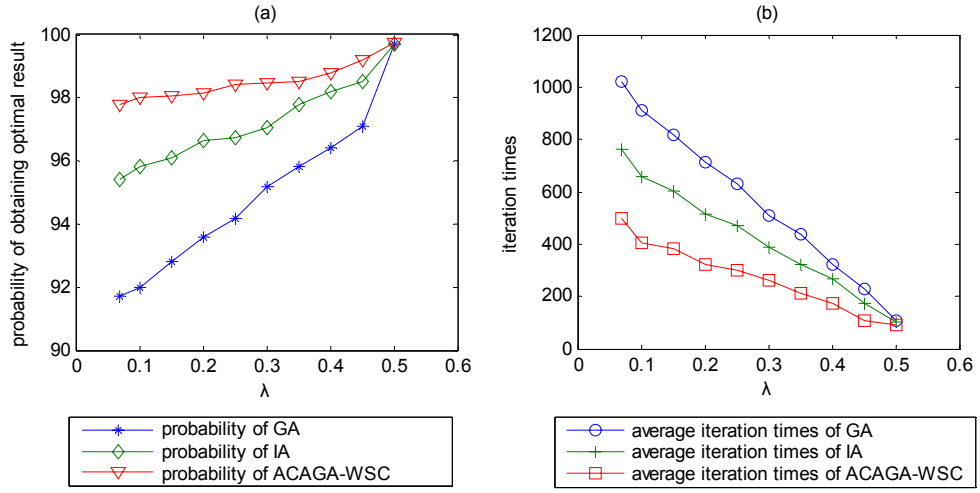
In the experiment, we create a composite service with 10 tasks and each task has 35 candidate services. The minimum cost path can be obtained by adding the cost value of atomic web services each of who has the minimum cost in the corresponding service task. The maximum cost path can be obtained by adding the cost value of atomic web services each of who has the maximum cost in the corresponding service task.

*CC* can be expressed as:

$$CC = cost_{min} + \lambda (cost_{max} - cost_{min}) \quad (7)$$

In formula (7), $\lambda$ stands for the constraint factor and $\lambda$ = [0,1]. The closer $\lambda$ to 0, the more restrictive the constraint is. Comparison of ACAGA_WSC with the genetic algorithm given by Ref [7] and immune algorithm given by Ref [8] is shown in Fig.3. We set 20 to the amount of ants, 500 to the cycle number for the path searching of ants, 100 to the iteration number of the genetic algorithm, 0.6 to the crossover factor, 0.05 to the mutation factor and 20 to the number of chromosome, where 4 chromosomes will be randomly selected for each generation.

From (a) in Fig.3, we can see that, ACAGA_WSC has higher probability in obtaining the optimal result than GA based method given by Ref. [7] and IA based method given by Ref. [8] under the same constraint factor value. From (b) in Fig.3, we can see that ACAGA_WSC is less in number of iteration times in obtaining the optimal result under the same constraint factor value than other two methods. The experiment indicates that, the presented algorithm gives better performance, and can quickly and effectively find the optimal (or near optimal) solution for QoS based dynamic web services composition problem.




(a) Comparison of the Probability of Obtaining Optimal Result (b) Comparison of Average Iteration Times

Fig.3 Comparison between the Three Algorithms

## 6. Conclusion

With the proliferation of web services providing the same functionality, researches about practicability and adaptive capability of web services selection mechanism have gained considerable momentums.

This paper presents a dynamic web services composition algorithm based on the optimal path method of the weighted directed acyclic graph. Experiment results show that the algorithm can accelerate the speed of service selection, and bring great benefit to the application of web services composition.

There are two directions in our future work: one is to search for the method of implementing run-time

fault-tolerance since component services may become unavailable or QoS of the component service may change significantly in dynamic environment, and the other one is to enhance the flexibility of ACAGA_WSC based on fuzzy control principle to implement environmental dynamic adaptivity.

**Acknowledgment**

This paper is supported by NSFC of China (NO.60673010), partly supported by the Key Projects in the National Science & Technology Pillar Program (NO. 2006BAJ07B06, NO.2009BAH51B03), the Important Projects in the Scientific Innovation of Colleges and Universities (NO.708065), and the Natural Science Foundation of HuBei Province (NO. 2009CDA135).

**References**

[1] Nakamura Kazuto, et al. Value-based dynamic composition and evaluation of Web Services. IPSJ SIG Technical Reports, 2006, 9-16.

[2] Zeng Liangzhao, Benatallah.B, et al. QoS-aware middle-ware for Web Services composition. IEEE Transactions on Software & Engineering, Vol 30, 2004, 311–327.

[3] Yu Tao,Lin Kwei-Jay, “Service selection algorithms for Web services with end-to-end QoS constraints,” Proc. Of the IEEE International Conference on e-Commerce Technology (CEC), pp. 129-136, 2004.

[4] Rainer .B, Michael .S, et al, “Heuristics for QoS-aware Web Service Composition,” Proc. Of the 4th IEEE International Conference on Web Services(ICWS 2006), pp. 72-82, 2006.

[5] Pistore.M,et al. Automated composition of web service by planning at the knowledge level.IJCAI2005,1252-1259.

[6] R.Aggarwal, K.Verma, j.Miller, and W.Milnor, “Constraint driven web service composition in METEOR-S,” Proc. Of the IEEE international Conference on Services Computing (SCC’04), pp. 23-30, 2004.

[7] Zhang L , Li B , Chao T , et al, “Requirements Driven Dynamic Business Process Composition for Web Services Solutions,” Journal of Grid Computing, vol. 2, no. 2, pp. 121-140, 2004.

[8] Gao Y, Na J, et al, “Immune Algorithm for selecting optimum services in Web Services composition,” Wuhan University Journal of Natural Sciences, vol. 11, no. 1, pp. 221–225, 2006.

[9] Dortgo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem [J]. IEEE Trans. On Evolutionary Computation, 1997,1(1): 53-66.

[10] Holland J H, Adaptation in natural and artificial systems [M].Michigan: The University of Michigan Press, 1975.