A Hybrid Algorithm Combining Ant Colony Algorithm and Genetic Algorithm for Dynamic Web Service Composition

Chen-Yang Zhao*,[1], Jun- Ling Wang[2], Jie Qin[1] and Wen-Qiang Zhang[1]

[1]*College of Information Science and Engineering, Henan University of Technology, Zhengzhou, P.R. China, 450000*

[2]*College of Science, Henan University of Technology, Zhengzhou, P.R. China, 450000*

Abstract: Growing numbers of web services that offer identical functionality but differ in non-functional properties are emerging on the network, to the need to select them to form a composite service to meet user's requirements has become one research hotspot. Web service selection methods are an attempt to to find optimal solutions for users. However, because each user's personal preference is different and web services are massive and dynamic, it is hard to find optimal solution. Therefore, a Hybrid Algorithm combining Ant Colony Algorithm and Genetic Algorithm for web service composition is proposed in this paper. The global optimization problem in web service composition is firstly transformed to the problem of finding an optimal path in the weighted directed acyclic graph with certain QoS (Quality of Service) constrains. And then an improved ant colony algorithm and an improved genetic algorithm are used alternately in the hybrid algorithm. Improved ant colony algorithm is used to achieve the non-dominated solution sets. Using the sets as the initial population sets, improved genetic algorithm is performed to assist ant colony algorithm to obtain the optimal solution. Experimental results demonstrate the validity and efficiency of the proposed algorithm.

Keywords: Ant colony algorithm, genetic algorithm, QoS utility value, service composition.

1. INTRODUCTION

Web services are self-described software entities which can be advertised, located, and used over the Internet [1]. Based on user's requirements, service selection mechanism is used for selecting service candidates from service classes and composing service candidates into a new value-added composite service. With the growing number of the available web services with identical functionality published online, the non-functional properties of web services are crucial for web service composition to satisfy the user's requirements. Moreover, the values of the non-functional properties may vary under the dynamic environment, so it is necessary to adapt the changes for web service composition. Therefore, in recent years, many researchers have focused on determining which web services participate in a given composite web service.

There are many researches on web service composition with various methods. In literature [2], the Integer Linear Programming is used for web service composition. In literature [3], the web service composition is formulated as a Multi-choice Multi-dimension Knapsack Problem. Unfortunately, the concrete solution under the dynamic environment is not given. In literature [4], one reliable web service composition algorithm based on the Markov Decision Process is designed. However, this algorithm is too complex. In literature [5], the web service composition is regarded as a Constrained Shortest Path, however, it is difficult to build model and solve the shortest path. Generally, the mentioned approaches above hold higher complexity, although they can definitely find better solutions.

Because of good models and good performances of Heuristic Algorithms, they are popular in web services composition recently. Ant Colony Algorithm [6], as one type of heuristic algorithm, not only has the characteristic of parallelism, positive feedback and heuristic search, but also can adapt the dynamic nature of web services. In literature [7], the method combining Optimal Chaos and Ant Colony Algorithm for web service composition is proposed, which uses the randomness of Chaos Variable to help the ant colony algorithm to make an optimal search. In literature [8], the web service selection with QoS global optimality is transformed into a Multi-objective Multi-choice QoS-aware web services composition optimization and the Pareto Optimality is achieved according to ant colony algorithm. In literature [9], one Multi-pheromone and Dynamically Updating Ant Colony Optimization Algorithm is proposed, which consists of the global optimizing algorithm and the local optimizing algorithm. Similar research based on ant colony algorithm can be found in literatures [10-15]. Genetic algorithm has the characteristic of global searching ability, which also performs well in web service composition. In literature [16], service composition problem is mapped into a Solution Space, and then genetic algorithm is utilized to get solution. In literature [17], service composition problem is transferred as a Linear Programming Problem, and an improved genetic algorithm is used to search optimum composite service. In literature [18], multi-population genetic algorithm is used in service composition.

Although ant colony algorithm has many advantages, it also has the lack of global search ability and is subjected to stagnation. Genetic algorithm has the characteristic of global searching ability, which can overcome the shortcomings of the ant colony algorithm. Meanwhile, ant colony algorithm can also overcome the shortcomings of genetic algorithm which has the lack of positive feedback and low convergence speed. In literature [19-21], many service composition methods based on the two algorithms are proposed. These methods are proved to be effective, but further works also needs to be done in order to get shorter computation time or better user's satisfaction. Therefore, in this paper, a Hybrid Algorithm combining Ant Colony Algorithm and Genetic Algorithm (*HA_ACAGA*) for web service composition is proposed. This algorithm makes use of the advantages of two heuristic algorithms to mutually overcome their shortcomings. The characteristic of the hybrid algorithm is roughly described as follows. Firstly, the traditional ant colony algorithm is improved and the Non-dominated Solution Set is introduced. Secondly, the fitness function of chromosome in genetic algorithm is improved. Thirdly, improved ant colony algorithm and genetic algorithm are used alternately. In the hybrid algorithm, The Non-dominated Solution Sets achieved by improved ant colony algorithm become the population sets of genetic algorithm, which is performed to assist ant colony algorithm to obtain the optimal solution. Experimental results demonstrate that the proposed hybrid algorithm can achieve better efficiency and convergence speed.

The rest of the paper is organized as follows. Section 2 firstly introduces the web service composition model and problem description. Section 3 describes the design of hybrid algorithm proposed in the paper. In Section 4, some experiments are established to evaluate the proposed algorithm. Finally, a brief conclusion and future work are presented in Section 5.

2. SERVICE COMPOSITION MODEL AND PROBLEM DESCRIPTION

2.1. Service Composition Model

Service selection implementations are mainly considered from two key factors: functional requirements and non-functional requirements of users. Non-functional requirements of users have recently received increased attention in the service selection, because of the different personalized service requirements of the users. This paper mainly focuses on user's non-functional requirements and supposes user's functional requirements have been satisfied. Meanwhile, only sequential service composition model is considered in the paper, because other models, such as parallel and loop, can be transformed to the sequential model by appropriate process [9].

As shown in Fig. (**1**), let $ASC = \{S_1, S_2, ..., S_n\}$ be an abstract composite service which consists of *n* abstract single service. Where, *T* is the starting point, *E* is the end point, $S_i (1 \le i \le n)$ is the *i*-th abstract service. Each abstract service S_i has one corresponding service class C_i which includes *m* concrete service candidates with identical function but different QoS. Each concrete service $s_{i,j} (1 \le i \le n, 1 \le j \le m)$ has a *r*-dimensional QoS attribute vector $(Q_1(s_{i,j}), Q_2(s_{i,j}), ..., Q_r(s_{i,j}))$. There is an arc $(s_{i,v}, s_{j,u})$ between each concrete service $s_{i,v}$ in C_i and each concrete service $s_{j,u}$ in C_j. Different arcs hold different weights. So the problem of service composition can be transformed to the problem of finding an optimal path in the weighted directed acyclic graph. It is crucial that the composite service should satisfy certain QoS constrains, such as response time, price or reputation, and etc. If a path, which is a concrete composite service, meets the user's QoS requirements, the path is called the solution. Given s_i and s_j both are the solutions, for any QoS attribute type $k(1 \le k \le r)$, if $Q_k(s_i) \ge Q_k(s_j)$, then s_i dominates s_j or s_j is dominated by s_i, which means that solution s_i is better than solution s_j. Given a set, if any solution in the set is not dominated by other solutions in the set, it is called the set as *Non-Dominated Solution Set* (*NDSS*). Exhaustive method can be used for finding optimal solution, but the method always leads a large amount of computation, even for smaller *m* and *n*. In fact, this problem has been proved to be an *NP-hard* problem.

2.2. QoS Attributes of Web Service

QoS attributes of web service are constituted by many types, such as response time, price, reputation and etc. It is needed to map multi-dimension attributes into one single real value to enable sorting or ranking web service candidates. Because QoS attributes have diverse dimensions and ranges, the paper normalizes all QoS attributes in the same range between [0, 1]. QoS attributes can be divided into two subsets according to their characteristics: positive attribute and negative attributes. Positive attribute means higher QoS value will bring better service quality, such as reputation and availability. While negative attribute means higher attribute value will cause worse service quality, such as response time or price. The normalization rules of QoS attributes are shown in formula (1). Here, $Q_k^{\max}(S_i)$ indicates the maximum value of the *k*-th QoS attribute among all concrete service candidates that belong to service class C_i, and $Q_k^{\min}(S_i)$ indicates the minimum value of the *k*-th QoS attribute among all concrete service candidates that belong to service class C_i. Naturally, $Q_k'(s_{i,j}) \in [0,1]$.

2.3. User's QoS Utility Value

Supposing there are *r* QoS attributes type, let w_k be the weight of each attribute type for representing user's preference, and let $\sum_{k=1}^{r} w_k = 1$. Here, supposing the number of negative attributes is *h*, then, the number of positive attributes is (*r-h+1*). For concrete service $s_{i,j}$, the corresponding user's QoS utility value, denoted by $F(s_{i,j})$, can be calculated by formula (2).

The objective of service composition is to achieve one service composition solution which maximizes the user's

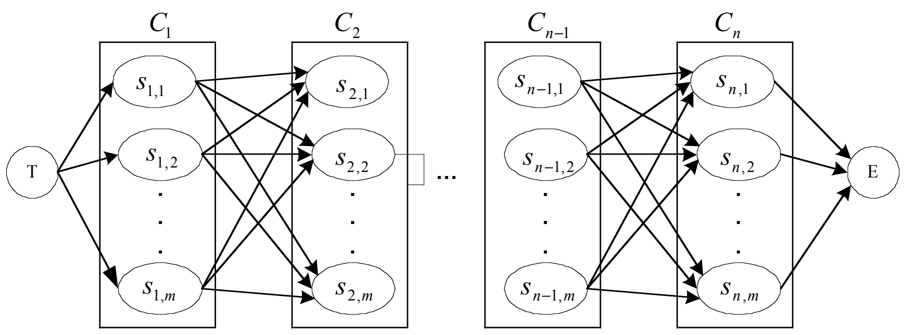


Fig. (1). The Service Composition Mode.

Table 1. Examples of QoS aggregation rules.

Aggregation Type	Rule	Examples
Summation	$Q_i(CS)=\sum_{k=1}^{n} Q_i(s_{k,j})$	Response time or Price
	$Q_i(CS)=1/n\sum_{k=1}^{n} Q_i(s_{k,j})$	Reputation
Multiplication	$Q_i(CS)=\prod_{k=1}^{n} Q_i(s_{k,j})$	Availability
Minimum	$Q_i(CS)=\min_{k=1}^{n} Q_i(s_{k,j})$	Throughput

$$Q_k^{'}(s_{i,j})=\begin{cases}\dfrac{Q_k^{\max}(S_i)-Q_k(s_{i,j})}{Q_k^{\max}(S_i)-Q_k^{\min}(S_i)} & if(QoSType=negative\ and\ Q_k^{\max}(S_i)\neq Q_k^{\min}(S_i))\\ \dfrac{Q_k(s_{i,j})-Q_k^{\min}(S_i)}{Q_k^{\max}(S_i)-Q_k^{\min}(S_i)} & if(QoSTypepositiveve\ and\ Q_k^{\max}(S_i)\neq Q_k^{\min}(S_i))\\ 1 & if(Q_k^{\max}(S_i)=Q_k^{\min}(S_i))\end{cases} \quad (1)$$

$$F(s_{i,j})=\sum_{k=1}^{h}\frac{Q_k^{\max}(S_i)-Q_k(s_{i,j})}{Q_k^{\max}(S_i)-Q_k^{\min}(S_i)}\times w_k+\sum_{k=h+1}^{r}\frac{Q_k(s_{i,j})-Q_k^{\min}(S_i)}{Q_k^{\max}(S_i)-Q_k^{\min}(S_i)}\times w_k \quad (2)$$

QoS utility value and meets user's requirement. As for certain user u, let the service composition solution for the user be $CS_u=\{s_{1,a},s_{2,b},\ldots,s_{n,m}\}$, then the user's QoS utility value of the composite service solution can be computed by formula (3).

$$F(CS_u)=\sum_{k=1}^{h}\frac{Q_k^{\max}-Q_k(CS_u)}{Q_k^{\max}-Q_k^{\min}}\times w_k+\sum_{k=h+1}^{r}\frac{Q_k(CS_u)-Q_k^{\min}}{Q_k^{\max}-Q_k^{\min}}\times w_k \quad (3)$$

Here, $Q_k(CS_u)$ represents the aggregated value of k-th QoS attribute in the solution, $Q_k^{\max}$ and $Q_k^{\min}$ represent the maximum and minimum aggregated value of k-th QoS attribute in all possible solutions, respectively. The aggregated rules of QoS attributes for the composite service can be found in the following subsection.

2.4. QoS Aggregated Rules of Composite Service

The QoS attribute value of a composite service is decided by the QoS attribute values of its component services as well as the composition model used (*e.g.* sequential, parallel and conditional). The paper also focuses on sequential composition model, because other model may be transformed to the sequential composition model.

The QoS vector for a composite service CS is defined as $(Q_1(CS),Q_2(CS),\ldots,Q_r(CS))$, where, $Q_i(CS)$ represents the estimated value of i-th QoS attribute of CS and can be aggregated from the expected QoS values of its component services. The paper considers three types of QoS aggregate rules: summation, multiplication and minimum relation. Table **1** shows examples of these aggregation rules.

2.5. Service Composition Problem Description

Based on the descriptions above, the objective of service composition can be considered into an optimization problem with constraints which is shown by formula (4).

$$\max F(CS_u)=\sum_{k=1}^{h}\frac{Q_k^{\max}-Q_k(CS_u)}{Q_k^{\max}-Q_k^{\min}}\times w_k+\sum_{k=h+1}^{r}\frac{Q_k(CS_u)-Q_k^{\min}}{Q_k^{\max}-Q_k^{\min}}\times w_k$$
$$s.t.\begin{cases}Q_k(CS_u)\geq q_k(CS_u)\\ Q_k(s_{i,j})\geq q_k(s_{i,j})\end{cases} \quad (4)$$

Here, $Q_k(\mathrm{CS}_u)$ indicates the k-th QoS attribute type of solution and $q_k(\mathrm{CS}_u)$, which is specified by user, indicates the constraint value for the solution. Analogously, $Q_k(s_{i,j})$ indicates the k-th QoS attribute type of $s_{i,j}$ and $q_k(s_{i,j})$ indicates the constraint value for $s_{i,j}$ in the solution.

3. THE DESIGN OF HYBRID ALGORITHM *HA_ACAGA*

3.1. Improved Ant Colony Algorithm in *HA_ACAGA*

Initial pheromone amount on path: the initial pheromone amount of arc $(s_{i,v}, s_{j,u})$ from $s_{i,v}$ in C_i to $s_{j,u}$ in C_j is assigned by formula (5).

$$\tau(s_{i,v}, s_{j,u}, 0) = (\tau^1(s_{i,v}, s_{j,u}, 0), \tau^2(s_{i,v}, s_{j,u}, 0), \ldots, \tau^r(s_{i,v}, s_{j,u}, 0)) \qquad (5)$$

Here, $\tau(s_{i,v}, s_{j,u}, 0)$ denotes pheromone amount of the arc $(s_{i,v}, s_{j,u})$ at the initial time, and $\tau^k(s_{i,v}, s_{j,u}, 0) = w_k \times Q_k(s_{j,u})$ denotes pheromone amount of the k-th QoS attribute type.

Transition probability: Once ant agent x reaches one concrete service node $s_{i,v}$ in C_i, it will select one arc to reach another concrete service node $s_{j,u}$ in C_j, according to certain probability, which is shown by formula (6).

$$P^x_{(s_{i,v}, s_{j,u})} = \frac{[\tau(s_{i,v}, s_{j,u})]^\alpha [\eta(s_{j,u})]^\beta}{\sum_{1 \le \lambda \le m} [\tau(s_{i,v}, s_{j,\lambda})]^\alpha [\eta(s_{j,\lambda})]^\beta} \qquad (6)$$

Here, $\tau(s_{i,v}, s_{j,u}) = \sum_{k=1}^{r} \tau^k(s_{i,v}, s_{j,u})$ is the pheromone amount on arc $(s_{i,v}, s_{j,u})$, which indicates the sum of pheromone amounts on each QoS attribute type. $\eta(s_{j,u}) = \sum_{k=1}^{r} w_k \times Q_k(s_{j,u})$ is the heuristic factor, which indicates the sum of the r QoS attributes value of $s_{j,u}$. α, β are parameters that control the relative weight of pheromone and heuristic factor. Moreover, each ant agent has a tabu-list which records all the visited concrete services. This tabu-list allows the ant agent to backtrack the same path and deposit pheromone on the visited arcs after the ant finds one solution.

Partial pheromone amount update on path: In each iteration, ant agents respectively find paths from the starting point T to the end point E. A path is called as one solution, if the solution meets the user's QoS requirements. If not dominated, the solution will be added into the *NDSS*. Moreover, ant agent deposits pheromone on the visited arcs after it finds one solution. The amount of pheromone is proportional to the quality of the solutions they produced. The more satisfactory to user, the greater the amount of pheromone it deposits on the arcs, which is used to guide the future ant agents towards finding better solutions. If arc $(s_{i,v}, s_{j,u}) \in tabu^x$, then the amount of pheromone deposited by ant agent x on the arc is calculated by formula (7).

Here, $\Delta\tau^k(s_{i,v}, s_{j,u}) = C \times w_k \times Q_k(s_{j,u})$ is the pheromone amount deposited by ant agent x on the k-th QoS attribute type. C is a value to tune $\Delta\tau^k(s_{i,v}, s_{j,u})$. Then the pheromone amount on arc $(s_{i,v}, s_{j,u})$ at t+1 time is updated by formula (8).

$$\tau(s_{i,v}, s_{j,u}, t+1) = \begin{cases} (1-\rho)\tau(s_{i,v}, s_{j,u}, t) + \Delta\tau(s_{i,v}, s_{j,u}) \\ \tau_{\max}, \quad if\ \tau(s_{i,v}, s_{j,u}, t+1) \ge \tau_{\max} \\ \tau_{\min}, \quad if\ \tau(s_{i,v}, s_{j,u}, t+1) < \tau_{\min} \end{cases} \qquad (8)$$

Here, $\rho \in [0,1]$ is the pheromone decay coefficient. The smaller ρ is, the less the impact of past solution is. An evaporation mechanism is introduced to pheromone update. Pheromone evaporation allows the ant agent to slowly forget its past history in order to void over-constrained by past decisions. In addition, the amount of pheromone is limited in $[\tau_{\min}, \tau_{\max}]$, where $\tau_{\max}$ is the maximum value of the pheromone amount, and $\tau_{\min}$ is the minimum value. This limit can effectively avoid the algorithm converge to local optimal solution too quickly. Generally, such update is called partial pheromone amount update on path.

Global pheromone amount update on path: Once all the ant agents have visited all nodes, the ant agent, which has found the best solution in the iteration, is selected as the best ant agent. And then extra amount of pheromone is deposited on the path which the best ant agent has found. This strength on the best path can help ant colony to find the optimal solution. Generally, such update is called global pheromone amount update on path, which consists of following three steps:

Step 1: In one iteration, all solutions found by all the ant agents are formed into set S. According to formula (1), the QoS attributes of composition services in S must first be standardized.

Step 2: After standardized, the QoS utility value of each solution is calculated according to formula (3), and then the best ant agent *ba* in the iteration can be found.

Step 3: Extra amount of pheromone is deposited on the path found by ant agent *ba*. Assume arc $(s_{i,v}, s_{j,u})$ is the arc visited by ant agent *ba*, the pheromone amount will be added as shown by formula (9).

Here, $\Delta\tau^k(s_{i,v}, s_{j,u}) = C \times w_k \times |Q_k - q_k^*|$ is the pheromone amount deposited by best ant agent on the k-th QoS attribute

$$\Delta\tau(s_{i,v}, s_{j,u}) = \begin{cases} (\Delta\tau^1(s_{i,v}, s_{j,u}), \Delta\tau^2(s_{i,v}, s_{j,u}), \ldots, \Delta\tau^k(s_{i,v}, s_{j,u})) & (s_{i,v}, s_{j,u}) \in tabu^x \\ 0 & (s_{i,v}, s_{j,u}) \notin tabu^x \end{cases} \qquad (7)$$

$$\Delta\tau(s_{i,v}, s_{j,u}) = \begin{cases} \{\Delta\tau^1(s_{i,v}, s_{j,u}), \Delta\tau^2(s_{i,v}, s_{j,u}), \ldots, \Delta\tau^r(s_{i,v}, s_{j,u})\} & (s_{i,v}, s_{j,u}) \in tabu^{ba} \\ 0 & (s_{i,v}, s_{j,u}) \notin tabu^{ba} \end{cases} \qquad (9)$$

type. Here $(Q_1, Q_2, ..., Q_r)$ is the actual QoS attribute value vector of the best solution, and $(q_1^*, q_2^*, ..., q_r^*)$ is the ideal QoS attribute value vector which can be obtained by following method. For each positive or negative QoS attribute, the maximum or minimum QoS value in each service candidate class is firstly obtained, and then ideal QoS value on each type is obtained by QoS aggregation rules. Then the pheromone amount on arc $(s_{i,v}, s_{j,u})$ at t+1 time can be updated by formula (8).

3.2. Improved Genetic Algorithm in *HA_ACAGA*

Fitness function of chromosome: Let fitness function of chromosome c be $F(c) = \sum_{k=1}^{r} w_k \times Q_k(c)$, where, c indicates a solution, r indicates the number of QoS attributes type, $Q_k(c)$ indicates the value of the k-th QoS attribute type and w_k indicates the weight of the k-th QoS attribute type.

Selection of initial population: Compare the values of optimum fitness for each solution in *NDSS*, select the top N solutions as the initial population, each solution is a chromosome. Each gene in chromosome is a concrete web service.

Selection operator: Select the M better chromosomes from the N chromosomes as father by bubbling up method.

Crossover operator: Select randomly a gene as the crossing point for two chromosomes, and then exchange the genes from crossing point to end point, while remaining genes form start point to crossing point.

Mutation operator: Select randomly a gene as the mutation point, and then select a new single gene to replace the original gene.

3.3. Algorithm Design of *HA_ACAGA*

As show in Fig. (**2**), the basic flow of *HA_ACAGA* proposed in the paper can be described as following:

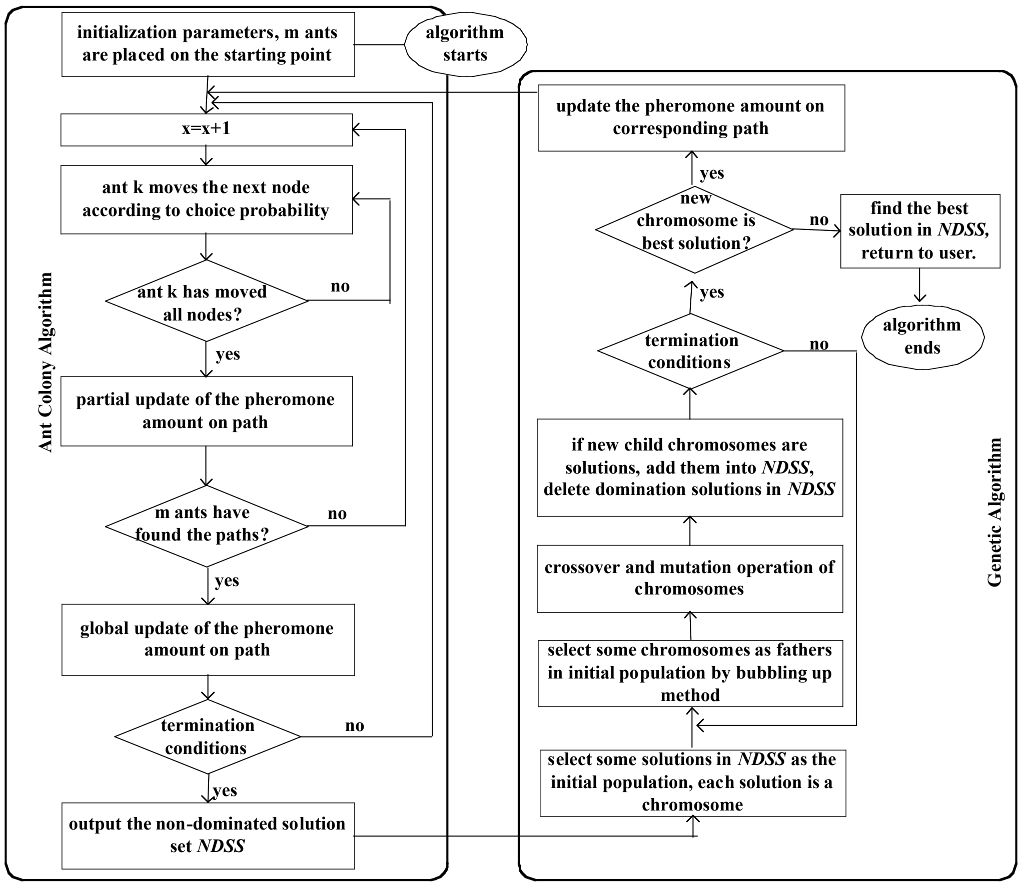
Step 1: Receive the description d from the user, which consists of the needed service type and the corresponding QoS constraints. Analyze d and then translate it into an abstract composite service a. Translate a into the sequential model m in the logic level. For each service candidate class, concrete services, which satisfy with QoS constraints, have been selected.

Step 2: Algorithm starts. Set *NDSS* is empty. l ant agents with tabu-lists are placed on the initial node of the weighted directed acyclic graph in one iteration. Initialize the pheromone amount on each arc. Moreover, use x to control the number of ants and set $x = 0$.

Step 3: Set $x = x + 1$, if $x > 1$, go to step 7.

Step 4: Ant agent x selects and moves to the next concrete service node according to the transition probability. And then adds this chosen concrete service into its tabu list.

Step 5: Go to step 4, if ant agent x does not reach the end



Ant Colony Algorithm
initialization parameters, m ants are placed on the starting point
algorithm starts
x=x+1
ant k moves the next node according to choice probability
ant k has moved all nodes?
no
yes
partial update of the pheromone amount on path
m ants have found the paths?
no
yes
global update of the pheromone amount on path
termination conditions
no
yes
output the non-dominated solution set NDSS
Genetic Algorithm
update the pheromone amount on corresponding path
yes
new chromosome is best solution?
no
find the best solution in NDSS, return to user.
algorithm ends
yes
termination conditions
no
if new child chromosomes are solutions, add them into NDSS, delete domination solutions in NDSS
crossover and mutation operation of chromosomes
select some chromosomes as fathers in initial population by bubbling up method
select some solutions in NDSS as the initial population, each solution is a chromosome

Fig. (2). The flow of Hybrid Algorithm combing ant colony algorithm and genetic algorithm.

point, that is, the service selection of the ant agent does not finish.

Step 6: Adjust the pheromone amount on the path visited by the ant agent *x*, according to local pheromone update strategy. Add the path *p* into *NDSS*, if *p* is the solution and cannot be dominated by any solution in *NDSS*. Go to step 3.

Step 7: Find the best solution in the iteration according to the user's QoS utility value and adjust the pheromone amount on the corresponding path according to global pheromone update strategy.

Step 8: Find the best solution from *NDSS* according to the user's *QoS* utility value and record the solution. Then compare the best solution with the former two best solutions. If changed, set $x = 0$, go to step 3, which means carrying on new iteration; otherwise, go to step 9.

Step 9: Compare the values of optimum fitness for each solution in *NDSS*, select the top 20 solutions as the initial population, each solution is a chromosome. Select 10 better chromosomes from the 20 chromosomes as father and perform crossover and mutation operations on them with a definite probability. If the new child chromosomes are solution, add them into *NDSS*. Then delete solutions which are dominated, that is, replace inferior chromosomes with better chromosomes.

Step 10: Find the best solution form *NDSS* according to the user's *QoS* utility value and record the solution. Then compare the best solution with the former two best solutions. If changed, repeat step 9, which means carrying on new iteration; otherwise, go to next step.

Step 11: Adjust the pheromone amount on the corresponding path according to global pheromone update strategy, if the best solution is new child chromosome then, set $x = 0$, go to step 3; otherwise, return the best solution to user, algorithm ends.

4. EXPERIMENTS RESULTS AND ANALYSIS

In order to show the validity and effectiveness of the *HA_ACAGA*, the paper makes some comparisons between *HA_ACAGA* and two other algorithms from closeness to optimal result and computation time. One is *Chaos_ACA* which combines optimal chaos and ant colony algorithm [7]. Another is *ACAGA* which combines ant colony algorithm and genetic algorithm [19]. Experiments are conducted on a computer with Intel(R) Core(TM) i3 2.4GHz processor, 2GB of RAM and Windows 7 operating system.

4.1. Simulation Datasets

The paper has established some datasets, which includes different numbers of service classes and service candidates. In order to verify the effectiveness of proposed algorithm in the paper and cover a variety of situations in the experiment, the paper created two different dataset types. One is the small-scale dataset in which the number *n* of service class changes from 10 to 35, and the number *m* of service candidate from 10 to 35. It includes six group: Group1 (m=10,n=10), Group2 (m=15,n=15), Group3 (m=20,n=20), Group4 (m=25,n=25), Group5 (m=30,n=30), Group6 (m=35,n=35). Another is the large-scale dataset in which the number *n* of service class varies from 40 to 85 and the number *m* of service candidate varies from 55 to 100. It includes ten group: Group1 (m=40,n=55), Group2 (m=45,n=60), Group3 (m=50,n=65), Group4 (m=55,n=70), Group5 (m=60,n=75), Group6 (m=65,n=80), Group7 (m=70,n=85), Group8 (m=75,n=90), Group9 (m=80,n=95), Group10 (m=85,n=100).

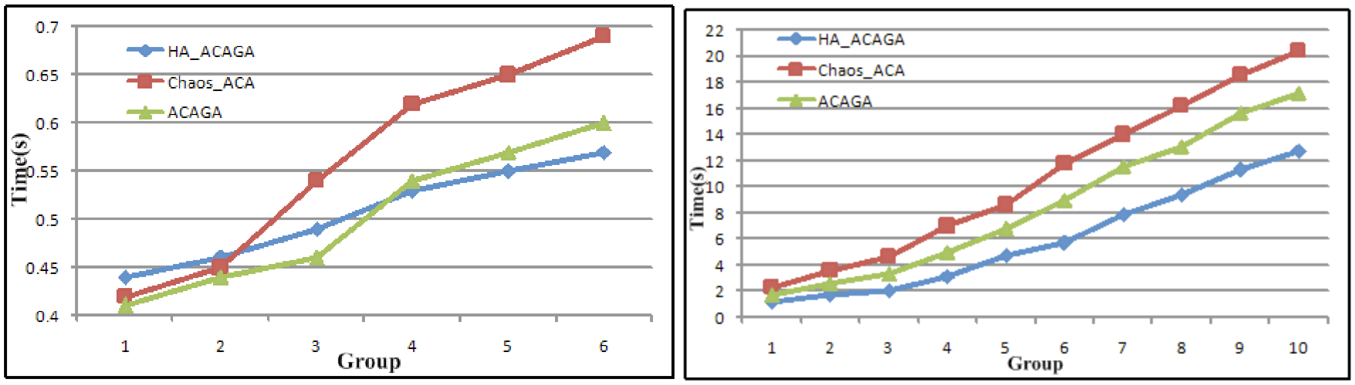
Generally, response time, price, availability and reliability are four primary factors that users are concerned about, so the experiments only consider the four attributes and set the weights as (0.4, 0.2, 0.2, 0.2). The values of response time and price are randomly generated in [0.1,1.0], and the values of availability and reliability are randomly generated in [0.7, 1.0]. The three parameters

α , β and ρ have great effect on the ant colony algorithm. However, ant colony algorithm does not provide specific values for the three parameters. Learned from research results about parameter value selection in service composition, parameter values in the experiments are set as follows: the heuristic factor of pheromone $\alpha = 3$; the expected heuristic factor $\beta = 4$; volatile factor of pheromone amount $\rho = 0.3$; the number of ants is selected by two-thirds of service candidate number.

4.2. Computation Time Comparison

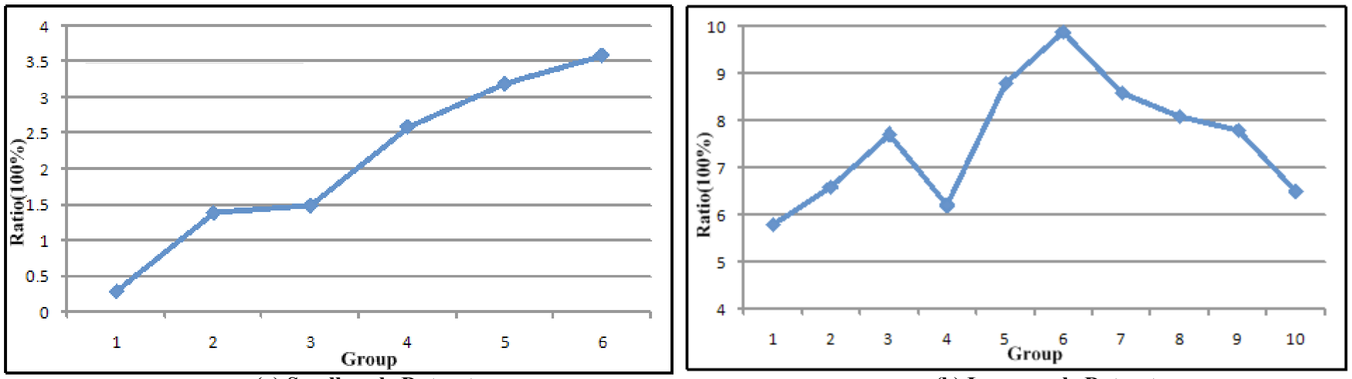
Fig. (**3a**) displays the computation time of three algorithms on the small-scale dataset. Experimental results are average values by repeating experiment to 20 times. As fo*r HA_ACAGA*, *Chaos_ACA* and *ACAGA* have some advantage on computation time when service class number and service candidate number are small. This is because that *HA_ACAGA* method iteratively used genetic algorithm to enhance its global space searching ability. It will cost more computation time comparing to *Chaos_ACA* method which only uses simple chaos variable to make an optimal search and *ACAGA* which only uses genetic algorithm only onceto achieve the better parameters values for the ant colony algorithm. As for *HA_ACAGA*, the smaller the service class number and service candidate number is, the more obvious the impact of iterative genetic algorithm on computation time is. However, as service class number and service candidate number grows from 10 to 35, *Chaos_ACA* and *ACAGA* lose their advantages gradually.

Fig. (**3b**) displays the computation time of three methods on the large-scale dataset. Experimental results are average values by repeating experiment to 20 times. It can be seen that the computation time increases gently when *HA_ACAGA* method is employed. When service class number does not exceed 50 and service candidate number does not exceed 65, computation time of getting final result is no more than two seconds. Even for the most complex situation, in which service class number is 85 and service candidate number is 100, the computation time of *HA_ACAGA* methods is only about 12.8s. By contrast, in the most complex situation, the computation time of *Chaos_ACA* reached about 20.4s and the computation time of *ACAG* reached 17.2s. Such situation can be explained by the nature of *HA_ACAGA* method. By using iterative genetic algorithm and ant colony



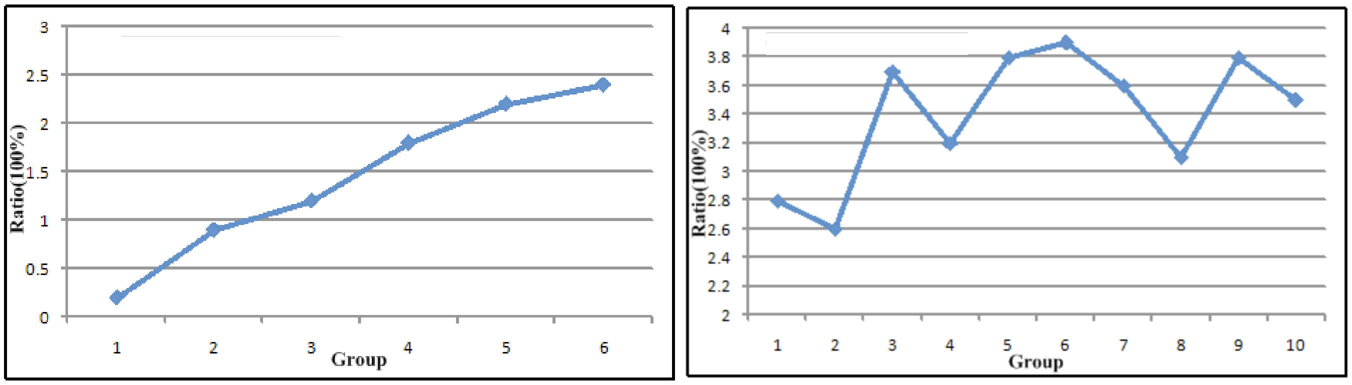
(a) Small-scale Dataset (b) Large-scale Dataset

Fig. (3). Time comparison for small-scale dataset and large-scale dataset.



(a) Small-scale Dataset (b) Large-scale Dataset

Fig. (4). QoS utility value comparison between *HA_ACAGA* and *Chaos_ACA*.



(a) Small-scale Dataset (b) Large-scale Dataset

Fig. (5). QoS utility value comparison between *HA_ACAGA* and *ACAGA*.

algorithm, hybrid algorithm can achieve the optimal solution quickly. Form the Fig. (**3b**), it can also be seen that with the increase of service class number and service candidate number, the advantage of *HA_ACAGA* method become more obvious. So it is worth introducing iterative genetic algorithm on the large-scale dataset.

4.3. QoS Utility Value Comparison

The same datasets are used for *Chaos_ACA*, *ACAGA* and *HA_ACAGA* method in the QoS Utility Value Comparison. The best solutions respectively obtained from *Chaos_ACA* and *ACAGA* method are added to *NDSS* first, and then the QoS Attribute values of solutions in *NDSS* are standardized. Supposing the best QoS utility value obtained from three methods are denoted as u_{Chaos_ACA}, u_{ACAGA} and u_{HA_ACAGA}. In order to compare QoS utility values among these three methods, the paper uses Increasing Ratio which can be computed by formula (10).

$$Ratio_{HA\_ACAGA-Chaos\_ACA} = \frac{u_{HA\_ACAGA} - u_{Chaos\_ACA}}{u_{Chaos\_ACA}} \times 100\% \quad (10)$$

Experimental results also are average values by repeating experiment 20 times. From the experimental results, it is found that the QoS utility value achieved by *HA_ACAGA* method is better than *Chaos_ACA* method in most conditions. Fig. (**4a**) shows the Increasing Ratio of QoS utility value achieved by *HA_ACAGA* method and *Chaos_ACA* on the small-scale dataset. It can be seen that *HA_ACAGA* method is able to get a higher utility value than *Chaos_ACA* method when service class number and service candidate number is very small. The utility value achieved by *HA_ACAGA* method is average 2.1% higher than utility value achieved by *Chaos_ACA* method. Fig. (**4b**) shows the Increasing Ratio of QoS utility value achieved by *HA_ACAGA* method and *Chaos_ACA* on the larger-scale dataset. The QoS utility value achieved by *HA_ACAGA* method is about 6%~10% higher than QoS utility value achieved by *Chaos_ACA* method. Curves show that when service class number and service candidate number increases, advantage of *HA_ACAGA* becomes more obvious. This is because that *HA_ACAGA* method uses genetic algorithm which is more effective than chaos variable in *Chaos_ACA* method, especially under the condition of the large-scale dataset.

Fig. (**5**) shows the comparison of QoS utility value between *HA_ACAGA* method and *ACAGA* method. The ratio is computed by formula (11).

$$Ratio_{HA\_ACAGA-ACAGA} = \frac{u_{HA\_ACAGA} - u_{ACAGA}}{u_{ACAGA}} \times 100\% \qquad (11)$$

It is seen that *HA_ACAGA* method only has a little advantage than *ACAGA* method on QoS utility value. The QoS utility value achieved by *HA_ACAGA* method is on average 1.4% higher than QoS utility value achieved by *ACAGA* method on the small-scale dataset. On the large-scale dataset, the QoS utility value is about 2.8%~3.9% higher. This is because both methods use genetic algorithm and ant colony algorithm as their main process structure, but *ACAGA* method only uses genetic algorithm once while *HA_ACAGA* method iteratively uses genetic algorithm which enhances its global searching ability and better combines the advantages of two algorithms.

5. CONCLUSION

This paper presents a Hybrid Algorithm combing Ant Colony Algorithm and Genetic Algorithm for web services composition. The problem of web service composition is firstly transformed to the problem of finding an optimal path in the weighted directed acyclic graph with certain QoS constrains. Then the hybrid algorithm makes full use of advantages of two algorithms to achieve the optimal solution. Experiment results show that the algorithm is more effective and better expansible. The next work is to improve the adaptability of algorithm when services become unavailable or QoS of the services changes in a dynamic environment.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This research is supported by the National High Technology Research and Development Program (Grant No. 2012AA101608), the National Nature Science Foundation (Grant No.U1304606, No. U1304609), Science and Technology Development Project of Henan Province (Grant No. 132102210555), Ph.D. Programs Foundation of Henan University of Technology (No.2012BS060) and Research Foundation for Key Program of Education Bureau of Henan Province (14B520029).

REFERENCES

[1] P. Papazoglou and D. Georgakopoulos, "Serive-Oriented Computing," *Commun. ACM*, vol. 46, no.10, pp. 25-65, 2003.

[2] J. J. -W. Yoo, S. Kumara, D. Lee and S. C. Oh, "A web service composition framework using integer programming with non-functional objectives and constraints," In Proc. 10th IEEE Joint Conference on E-Commerce Technology and 5th Enterprise Computing, E-Commerce and E-Services, Crystal City, 2008, pp. 347-350.

[3] C.S. Liu, D.B. Liu and N. Han. "A novel web service composition algorithm for multiple QoS constraints," *J. Softw.*, vol.7, no.8, pp. 1867-1872, 2012.

[4] X.Q. Fan, C.J. Jiang, J. L. Wang and S.C. Pang, "Random-QoS-aware reliable web service composition," *J. Softw.*, vol. 20, no. 2, pp. 546-556, 2009.

[5] T. Yu and K.J. Lin, "Service selection algorithms for web services with end-to-end QoS constraints," *J. Inform. Syst. e-Business Manag.*, vol.3, no.2, pp. 103-126, 2005.

[6] M. Dortgo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evolut. Comput.*, vol. 1, no. 1, pp. 53-66, 1997.

[7] Y. W. Wang, "Application of Chaos Ant Colony Algorithm in Web Service Composition Based on QoS," In Proc. 2009 International Forum on Information Technology and Applications, Cheng Du, 2009, pp. 225-227.

[8] S. S. Zhao, L. Wang, L. Ma and Z. P. Wen, "An Improved Ant Colony Optimization Algorithm for QoS-Aware Dynamic Web Service Composition," In Proc 2012 Industrial Control and Electronics Engineering, Xi'an, 2012, pp.1998-2001.

[9] Y.M. Xia, B. Cheng, J. L. Chen, X. W. Meng and D. Liu, "Optimizing services composition based on improved ant colony algorithm," *Chin. J. Comput.*, vol. 35, no. 2, pp. 270-281, 2012.

[10] L. Wang and Y.X. He, "A web service composition algorithm based on global QoS optimizing with MOCACO," *Adv. Intell. Softw. Comput.*, vol. 111, pp.79-86, 2011.

[11] Z. J. Wang, Z. Z. Liu, X. F. Zhou and Y. S. Lou, "An approach for composite web service selection based on DGQoS," *J. Adv. Manuf. Tech.*, vol. 56, no. 9-12, pp. 1167-1179, 2011.

[12] X. L. Wang, Z. Jing and H. Z. Yang, "Service selection constraint model and optimization algorithm for web service composition," *Inform. Tech. J.*, vol. 10, no. 5, pp. 1024-1030, 2011.

[13] Q. W. Wu and Q. S. Zhu, "Transactional and QoS-aware dynamic service composition based on ant colony optimization," *Future Gene. Comput. Syst.*, vol. 29, no. 5, pp. 1112-1119, 2013.

[14] J. X. Cao, G. R. Zhu, X. Zheng, B. Liu and F. Dong, "TASS: Transaction assurance in service selection," In Proc 2012 IEEE 19th International Conference on Web Services, Honolulu, 2012, pp.472-479.

[15] H. Jin, H. Zou, F. C. Yang, R. H. Lin and T. P. Shuai, "A novel method for optimizing multi-user service selection," *J. Converg. Inform. Tech.*, vol. 7, no. 5, pp. 296-310, 2012.

[16] L. Zhang and B. Li, "Requirements driven dynamic business process composition for web services solutions," *J. Grid Comput.*, vol. 2, no. 2, pp. 121-140, 2004.

[17] H. Long, "An improved genetic algorithm-based web service composition," *Adv. Mat. Res.*, vol. 225-226, pp.307-310, 2011.

[18] Q. Li, R. L. Dou, F. Z. Chen and G. F. Nan, "A QoS-oriented Web service composition approach based on multi-population genetic algorithm for Internet of things," *Int. J. Computat. Intell. Syst.*, vol. 7, no. (Suppl. 2), pp. 26-34, 2014.

[19] Z. K. Yang, C. W. Shang, Q. T. Liu and C. L. Zhao, "A dynamic web services composition algorithm based on the combination of ant colony algorithm and genetic algorithm," *J. Comput. Inform. Syst.*, vol. 6, no. 8, pp. 2617-2622, 2010.

[20] T. F. Cao, Y. Q. Fu and M. Y. Zhong, "Based Web Service composition with genetic algorithm and ant colony optimization," *J. Comput. Syst. Appl.*, vol. 21, no. 6, pp.81-85, 2012.

[21] H. Jin, H. Zou, F. C. Yang, R. Lin, T. Shuai, X. Zhao, "Large-scale multi-user service selection based on hybrid method," *Adv. Inform. Sci. Serv. Sci.*, vol. 4, no. 8, pp. 361-380, 2012.

Received: September 22, 2014 Revised: November 30, 2014 Accepted: December 02, 2014