A Modified Method of Ant Colony Optimization for Web Service

Sheng Guojun*
Department of Information Management
Dalian Neusoft Information Institute
Dalian, China
shengguojun@neusoft.edu.cn
* Corresponding Author

Lu Yanxia
Department of Information Management
Dalian Neusoft Information Institute
Dalian, China
luyanxia@neusoft.edu.cn

Wang Jingshu
Department of Information Management
Dalian Neusoft Information Institute
Dalian, China
wangjingshu@neusoft.edu.cn

Zhou Dongzhao
Department of Information Management
Dalian Neusoft Information Institute
Dalian, China
zhoudongzhao@neusoft.edu.cn

Abstract—Web service composition optimization is a typical NP-hard problem to which the Ant Colony Optimization algorithm is applied appropriately for its excellent distributed computing capability and strong robustness. In this paper, we propose a new modified ant colony optimization algorithm called MACS and try to apply it in the problem of Web service composition optimization. The MACS algorithm employs both a non-linear dynamic parameter of the pseudorandom proportion selection rule and a random-weighted route selection method to control the behavior of the ant colony. Besides, the algorithm uses a five-dimensional quality vector and the fitness function to evaluate the ant solutions. Each ant updates the pheromone according to the quality of the solution it built, the pheromone variation range is limited in a max-min interval, so the evolution ability of the ant colony can be evidently improved with these measures. In the final experimental part, a novel algorithm performance evaluation method called APEI is presented. These concepts and methods provide a new thinking for application researches of WSC problem. Experimental results show that the MACS algorithm can achieve better performance than traditional ACO algorithms in WSC optimization. Some useful conclusions are obtained through the analysis and explanation of the experimental data, which lay a solid foundation for further researches.

Keywords-Web service composition; quality of service; modified ant colony optimization; dynamic pseudorandom proportion selection paramete; algorithm performance evaluation index

I. INTRODUCTION

Web services[1] are software systems identified by URIs which can support interoperable machine-to-machine interaction over a network. Other systems interact with the Web service using XML based standards (e.g., SOAP, WSDL, and UDDI) and Internet protocols. Web service has become the core technology of new generation distributed processing systems. The function of a single Web service is very limited, WSC(Web Service Composition) can combine simple services into more complex new services so as to meet the demands of users. It can also make the system more adaptive in dynamic network environment. As a promising software reuse mechanism, WSC is given more and more attention by both the business and the academic communities, many researches on WSC have come forth continuously in recent years. In particular, the creation of value-added services by composition of existing ones is gaining a significant momentum[2]. Nowadays, the business environment among enterprises becomes more complex and changing, for each workflow activity there may exist many alternative Web services that provide overlapping or identical functionality but with different Quality of Service (QoS), thus a selection and composition method should be made to determine which services are to participate in a composite service, this is a typical NP-hard problem. It has become a key problem in WSC optimization field about how to select suitable ones from massive composition schemes, which possesses important theoretical significance and practical value. The rest of this paper is organized as follows. Section 2 gives a brief introduction to the related works. Section 3 describes the Web service composition problem. Section 4 gives a concrete exposition on the MACS algorithm. Section 5 presents the experiment results of the MACS and its related algorithms. Section 6 concludes this paper.

II. RELATED WORKS

Many methods were proposed to solve the Web service selection problem. Shangguang Wang et al.[3] proposed a fast service selection method by using fuzzy logic to control the service selection process and used the mixed integer programming to assist users in obtaining suitable services. R. Dinesh Kumar et al. [4] proposed a web service selection model using Analytic Hierarchy Process which was used to select the best web service based on QoS Constraints. Xiaoqin FAN et al.[5] designed a reliable service selection algorithm based on an improved Markov decision process (IMDP) for reliable service composition, and proposed an effective method to

convert the service composition of IMDP into solving a non-homogeneous linear equation set. Yanping Chen et al. [6] proposed a new algorithm, called Mixed Intelligent Optimization Algorithm (MIOA) to optimize service selection with multiple QoS constraints by using both Maximum Entropy Method and Social Cognitive Optimization theory. Ludwig,S.A[7] proposed a meta-heuristic method based on Particle Swarm Optimization to address the Web service composition optimization problem, in which several workflow requests could be processed simultaneously. In [8], a novel QoS-based service composition technique was proposed to work out a multi-objective optimization problem. A dynamic Web service composition and selection method was presented in [9]. The method determined the web services to be invoked during runtime to build a composite web service, and used a finite state machine model to describe the permitted invocation sequences of the web service operations. In [10], a novel web service selection approach with QoS and improved trust management was presented. The approach only considered the feedbacks of consumers with similar preferences to deal with the meaningless feedbacks. Zongkai YANG et al.[11] proposed a comprehensive service composition with ant colony algorithm and the genetic algorithm to meet the maxim satisfaction of QoS requirements of users. In[12], a semantic composer was proposed to find optimal composition length based on ant colony optimization method. Based on the above works, we propose an ACO-based heuristic approach and try to apply it to the problem of the Web service composition optimization.

III. DESCRIPTION OF THE WEB SERVICE COMPOSITION PROBLEM

Definition 1: Quality of Service (QoS) is described as a 5-tuple $QoS=(t,c,a,rel,rep)$,where t is the service response time, c is the one-time cost of calling a service, a is the service availability, that is, the ratio of times of successful executions to total execution times , rel is the reliability that refers to the ratio of service operating time to total service operating time, and $rep=(\sum_{j=1}^{n}e_j)/n$ is the mean value of n-times evaluation score $(e_1,e_2,...,e_j,...,e_n)$ to a service.

Definition 2: Web Service Instance(WSI) refers to a web service with explicit URL which can be invoked directly. It can be formally defined as a 6-tuple $\mathrm{WSI}=(in,\ out,\ precondition,\ effect,\ QoS,\ Url)$, where in and out are sets of input parameters and output parameters respectively, precondition is the condition that has to be satisfied before the invocation of the web service, and effect is the new state generated after the invocation of the web service, the combination of these parameters is called IOPE, which determines the functional attributes of the service instance, QoS and Url are non-functional attributes of the service instance. Definition 3: Abstract Web Service(AWS) :

$AWS=\{WSI_1,WSI_2,...,WSI_i,...,WSI_n\}$

is a set of Web service instances with the same functional properties and different non-functional properties, $|AWS|$ denotes the number of the Web service instances included in the AWS .

Definition 4: Composite Service (CS) is described as a 6-tuple $CS=(S,R,QoS,C,W,f)$, where $S=\{AWS_1,AWS_2,\ldots,AWS_d,\ldots,AWS_n\}$ denotes the collection of abstract web services , $|CS.S|$ refers to the size of the collection, $R\subset S\times S$ represents the relation set among abstract web services, QoS is the quality vector of the composite service, C={g(t), g(c), g(a), g(rel), g(rep) } is the quality constraints that have to be met, W={w1,w2,…,w5} is the weight vector of the elements in QoS and it must satisfy the condition $\sum_{i=1}^{5}w_i=1$, and f is the function which is employed to evaluate the composite service quality.

Definition 5: Composite Service Instance (CSI) :

Select a concrete Web service instance WSI_d^i from each abstract service AWS_d of the composite service to replace the abstract service, the formation of the executable composite service scheme is called as a CSI, where WSI_d^i denotes the i-th alternative Web service instance in AWS_d , $i\in\{1,2,...,|AWS_d|\}$. One CS and its CSIs follow the relation of one-to-many correspondence. The function $f(<QoS_1,...QoS_i...,QoS_n>,<w_1,w_2,...,w_5>)$ is employed to evaluate the quality of the composite service in order to select the optimal solution or the suboptimal solutions that meet customer's needs. Each element in $<QoS_1,...QoS_i...,QoS_n>$ refers to the quality vector of one WSI in the composite service, and $w_1,w_2,...,w_5$ are the weights of the corresponding elements in the QoS respectively. Main process control activities defined in WS-BPEL[13] are as follows: sequence, flow, switch, and while. Several component services can be aggregated into one composite service through process control activities. The computational model of the composite service is shown in Table 1,where n is the number of all abstract services in the composite service, c is the number of abstract Web services in each core path of one flow-control activity, b is the number of branches of one switch-control activity or one flow-control activity, p_i is the probability for the i-th branch of one switch-control activity to be run and it must meet the following condition $\sum_{i=1}^{b}p_i=1$, l refers to the loop times of one while-control activity. Take each branch of the non-sequential service composition as the sequential sub-composition and then process them in turn recursively. Sequential service composition is the basis of other types of service compositions. The Web service selection process in this study can be represented with a directed graph $G=<V,E>$ (the sequential service composition as an example), as shown in Fig. 1. V is the set of the graph

vertices, $V = (\bigcup_{i=1}^{m} AWS_i) \cup \{S, L\}$,where S and L are the virtual start node and the virtual end node respectively, which are additionally added to graph G, E is the set of all edges in graph G,

$$E = (\bigcup_{i=1}^{n_1} < S, WSI_1^i >) \cup (\bigcup_{i=1}^{n_m} < WSI_m^i, L >)$$
$$\cup (\bigcup_{d=1}^{m-1} \bigcup_{i=1,k=1}^{n_d, n_{d+1}} < WSI_d^i, WSI_{d+1}^k >)$$

where $< WSI_d^i, WSI_{d+1}^k >$ is the ordered couple of two vertices that represents one directed edge from the i-th Web service instance, namely WSI_d^i in AWS_d , to theadjacent k-th Web service instance, namely WSI_{d+1}^k in AWS_{d+1} , $i \in \{1, 2, ..., n_d\}$ and $k \in \{1, 2, ..., n_{d+1}\}$, n_d and n_{d+1} are the number of Web service instances included in AWS_d and AWS_{d+1} respectively. As shown in Fig. 1, the in-degree and the out-degree of vertice WSI_d^i are as follows.

$\deg_{in}(WSI_d^i) = |AWS_{d-1}|$, $\deg_{out}(WSI_d^i) = |AWS_{d+1}|$,

where $d = \{2, 3, ..., m-1\}$, $i = \{1, 2, ..., |AWS_d|\}$.

TABLE I. THE QOS MODEL OF COMPOSITE SERVICES

	sequence	flow	switch	while
T	$\Sigma_{i=1}^{n} T_i$	$\Sigma_{i=1}^{c} Tcp_i$	$\Sigma_{i=1}^{b} p_i \times T_i$	$l \times \Sigma_{i=1}^{n} T_i$
C	$\Sigma_{i=1}^{n} C_i$	$\Sigma_{i=1}^{n} C_i$	$\Sigma_{i=1}^{b} p_i \times C_i$	$l \times \Sigma_{i=1}^{n} C_i$
A	$\Pi_{i=1}^{n} A_i$	$\Pi_{i=1}^{b} A_i$	$\Sigma_{i=1}^{b} p_i \times A_i$	$(\Pi_{i=1}^{n} A_i)^l$
Rel	$\Pi_{i=1}^{n} Rel_i$	$\Pi_{i=1}^{b} Rel_i$	$\Sigma_{i=1}^{b} p_i \times Rel_i$	$(\Pi_{i=1}^{n} Rel_i)^l$
Rep	$(\Sigma_{i=1}^{n} Rep_i) / n$	$(\Sigma_{i=1}^{b} Rep_i) / b$	$(\Sigma_{i=1}^{b} p_i \times Rep_i) / b$	$(\Sigma_{i=1}^{n} Rel_i) / n$

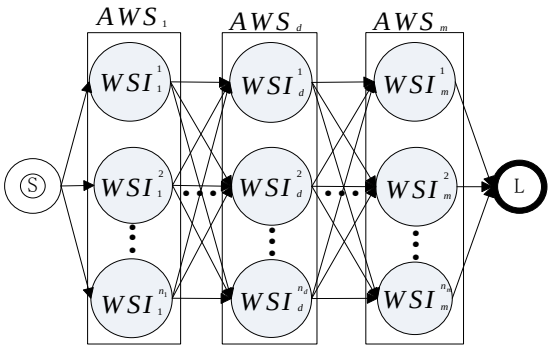

Figure 1. The directed graph representation of the Web service selection problem

IV. THE MODIFIED ANT COLONY SYSTEM(MACS)

A. *The basic concepts of the MACS algorithm*

Definition 6: Ant is described as a 9-tuple $A = (k, C, c, tl, P, f(P), P^*, fv^*, P_L)$, where $k \in \mathrm{N}^+$ is the identity of the ant, $C \in \mathrm{N}^+$ is the number of the total iterations, and $c \in [1, C]$ refers to the current iteration number which increase monotonously with the step 1, tl is the tabu list of the ant keeping a record of the visited nodes. $P = \{p_1, p_2, \ldots, p_d \ldots, p_n\}$ is the solution vector built by the ant after it has finished one complete iteration, and the state of the tabu list at this moment is called $full(tl)$, that is , $(tl = P) \Leftrightarrow full(tl)$. p_d is the d-dimensional position of the solution vector , $f(P)$ is the evaluation function (fitness function) for the solution vector P , P^* is the optimal solution vector of the ant in its historical iterations, namely the best-so-far solution, fv^* is the evaluation value of P^* ,that is $fv^* = f(P^*)$, P_L is the solution vector constructed by the ant in the last iteration, and $P_L = null$ for the first iteration.

Definition 7: Ant Colony(AC) is described as a 4-tuple:

$$AC = (AS, P_g^*, fv_g^*, ACD)$$

where $AS = \{A_1, A_2, \ldots A_k ..., A_m\}$ is a set of ants and $\forall i, j \in [1, m]$, $A_i.C = A_j.C$, $A_i.f(P) = A_j.f(P)$ .That is, all ants in the same AC have the same total evolutionary cycles, the same evaluation function and the same solution vector length. P_g^* is the optimal solution vector of the ant colony in its evolutionary history, fv_g^* is the corresponding evaluation value of P_g^* , $fv_g^* = f(P_g^*)$, ACD is the ant colony diversity.

Definition 8: Solution Vector Distance(SVD) refers to the ratio of the counts of different edges between $A_i.P$ and $A_j.P$ to the total solution vector dimension after ant A_i and ant A_j finished building their solutions.

$$SVD_{i,j} = \frac{1}{D} \sum_{d=1}^{D} iff(s_{i,d} \neq s_{j,d}, 1, 0) \quad (1)$$

where D is the dimension of the solution vector, $SVD_{i,j} \in [0,1]$ denotes the distance between $A_i.P$ and $A_j.P$, the smaller the value of $SVD_{i,j}$ is , the closer the two solution vectors are. It is called an overlap between $A_i.P$ and $A_j.P$ if $SVD_{i,j} = 0$, which is an equivalence relation to satisfy the reflexivity, symmetry and transitivity properties on the set of solution vectors.

Definition 9: Ant Colony Diversity(ACD):

$$ACD = (\sum_{i=1}^{m} SVD_{i,g}) / m \quad (2)$$

where m is the number of ants in the ant colony , $SVD_{i,g}$ is the solution vector distance between $A_i.P$ and the global optimal solution vector P_g^* , and

$ACD \in [0,1]$. The greater the value of ACD , the stronger the colony diversity. All the ants in the ant colony completely overlap in the global optimal solution vector P_g^* when the condition $ACD = 0$ is met.

B. Design of the fitness function of the MACS Algorithm

Assume that the user requirements for the quality of the composite service CS are as follows:

$$QC = \{t \le T, c \le C, a \ge A, rel \ge REL, rep \ge REP\},$$

where T,C,A,REL and REP are values given by user in advance. Then the mathematical model of Web service composition optimization problem is as follows:

$$\max f = \sum_{i=1}^{5} (Q_i^{'} \times w_i)$$

$$s.t. \begin{cases} g(t) = \left(1 - t^{'} / T^{'}\right) \ge 0 \\ g(c) = \left(1 - c^{'} / C^{'}\right) \ge 0 \\ g(a) = \left(a^{'} / A^{'} - 1\right) \ge 0 \\ g(rel) = \left(rel^{'} / REL - 1\right) \ge 0 \\ g(rep) = \left(rep^{'} / REP^{'} - 1\right) \ge 0 \end{cases}$$

where Q_i is one quality attribute and $g(Q_i)$ denotes one constraint. The MACS algorithm takes f as the objective function for evaluating the quality of the current solution vector built by one ant. Moreover, each quality attributes which is used in the function f need to be normalized as a dimensionless variable. In this study, the 5-dimensional quality model described in Definition 1 is applied to describe the quality of a Web service. Different QoS attributes may have different units of measurement, and for those cost-oriented quality attributes such as time and cost, it follows: the greater the value of the quality attribute, the lower the evaluation value of the QoS, thus the values of quality attributes need to be dimensionless unitary as follows :

$$Q_{d,i}^{'} = Q_{d,i} / \sum_{j=1}^{m_d} Q_{j,i} \quad (3)$$

$$Q_{d,i}^{'} = 1 - Q_{d,i} / \sum_{j=1}^{m_d} Q_{j,i} \quad (4)$$

where $Q_{d,i}$ is the i-th quality attribute of WSI_d . Formula

proportional to the QoS value. Conversely, for those quality attributes which are inversely proportional to the QoS value , formula (8) is applied. The formula for computing the quality of composition services(the sequential service composition, for example) is :

$f = \sum_{i=1}^{5} (w_i \times Q_i^{'})$ where $Q_i^{'}$ is the i-th quality attribute, and w_i is the corresponding weight of $Q_i^{'}$,the value of Q_i can be computed according to the rules in Definition 5 and Table 1.

C. Description of the MACS algorithm

As similar to the MMAS algorithm, the variation range of the pheromone in MACS is limited to an interval of $[\tau_{\min}, \tau_{\max}]$, the initial value of pheromone on each directed edge is set to the maximum pheromone value, namely the $\tau_{\max}$. The MACS algorithm adopts the dynamic pseudo-random proportion selection parameter which is computed in accordance with formula (5):

$$q_0 = 1 - \sqrt{\frac{c \times Q_{\max}}{C \times (1 + C - c)}} \quad (5)$$

where c is the current iteration number of the ant, C is the total number of the iterations of the ant, $Q_{\max}$ is a pre-given constant which is set to 0.999 in this study. Before the ant selects its next hop, it generates a random number $rand$ drawn from the interval of $[0,1]$ beforehand, and compares $rand$ with the value of q_0 , then the ant selects the next node to be visited according to the comparison result. The routing selection algorithm in the MACS is described in Algorithm 1.

Algorithm 1. Ant route selection algorithm of the MACS

Input: the ant A_k ,the directed graph: $G =< V, E >$, two parameters α and β which reflect the relative importance of the pheromone information and the heuristic information respectively.

Output: the nextnode to be visited by A_k

Step1: $q_0 = getDynamicQ0(A_k.c, A_k.C)$, compute the dynamic pseudo-random proportion selection parameter in accordance with formula (5).

Step2: $rand = makeRand()$, generate a random number drawn from the interval of $[0,1]$.

Step3: $if\ (rand > q_0)\{$

select the nextNode to be visited using the traditional ACO algorithm, as described in formula (1) .

return nextNode.

$\}$ *else*$\{$

select the nextNode to be visited using Algorithm 2.

return nextNode.

$\}$

Algorithm 2. Random-weighted route selection method

Input: the ant A_k ,the directed graph: $G =< V, E >$, two parameters α and β

Output: the nextnode to be visited by A_k

Step1: $neighbors_k = getNeighbors(A_k.tl, G)$, obtain all possible neighbors that may be visited according to the tabu list of A_k and the graph topology, each index number of a candidate node in the collection $neighbors_k$ denotes the corresponding position of the node.

Step2: get the position of the specific node in the solution vector that A_k has built at the last cycle, and the position obtained is denoted by $idx(A_k.P_L)$.

Step3: get the position of the specific node in the best solution vector that A_k has built at its historical cycles, and the position obtained is denoted by $idx(A_k.P^*)$.

Step4: get the position of the specific node in the best-so -far solution vector the ant colony has found and the position obtained is denoted by $idx(AC.P_g^*)$.

Step5: calculate the value of the random variable RV^k using the following formulas:

$RV^k = (RV^L + RV^* + RV^g)/3$, where

$RV^L = \tau^\alpha(idx(A_k.P_L)) \times \mu^\beta(idx(A_k.P_L))$,

$RV^* = \tau^\alpha(idx(A_k.P^*)) \times \mu^\beta(idx(A_k.P^*))$,

$RV^g = \tau^\alpha(idx(AC.P_g^*)) \times \mu^\beta(idx(AC.P_g^*))$,

where $\tau(idx(A_k.P_L))$, $\tau(idx(A_k.P^*))$ and $\tau(idx(AC.P_g^*))$ are the pheromones on each of the three edges from the current node to the next nodes $idx(A_k.P_L)$, $idx(A_k.P^*)$, and $idx(AC.P_g^*)$ respectively, $\mu(idx(A_k.P_L))$, $\mu(idx(AC.P^*))$ and $\mu(idx(AC.P_g^*))$ are the corresponding heuristic information, τ^α denotes τ to the power of α, and μ^β is also the case.

Step6: $r = iff(isStagnation(AC.fv_g^*, \eta), rand1, RV^k)$.

Step7: calculate the position of the next hop, namely, the index number of the Web service instance node:

$idx_{next} = idx(A_k.P_L) + 2 \times r \times (idx(A_k.P^*) + idx(AC.P_g^*) - 2 \times idx(A_k.P_L))$.

Step8: $return$ $\text{getRightNode}(idx_{next}, neighbors_k)$.

}

In Step 6 of Algorithm 2, the function $isStagnation(AS.fv_g^*, \eta)$ returns true if the evolution of the ant colony falls into stagnation, and then in this case, the random variable r is set to a random number drawn from the interval $(0,1]$. The state of stagnation means the best-so-far fitness value of the ant colony, that is the fv_g^*, does not change in the last η cycles. In Step7 of Algorithm 2, idx_{next} denotes the position of the next-hop node that A_k will visit, the value of idx_{next} is the sum of $idx(A_k.P_L)$ and a weighted random variable with the weight of $2 \times r$. The variable idx_{next} calculated via Step7 may be a floating-point or negative number, or a number whose absolute value is greater than $|neighbors_k|$, so in Step8, the function $\text{getRightNode}(idx_{next}, neighbors_k)$ is employed to calculate the right value of idx_{next}, that is, to make idx_{next} be an integer that belongs to the collection $\{1,2,...,|neighbors_k|\}$. The main logic of this function is described as follows:

$if(|idx_{next}| > |neighbors_k|)$

$idx_{next} = (idx_{next} \bmod |neighbors_k|)$.

$if(idx_{next} < 0)$

$idx_{next} = |neighbors_k| + idx_{next}$.

$if(idx_{next} < \lfloor idx_{next} \rfloor + 0.5)$

$idx_{next} = \lfloor idx_{next} \rfloor$.

$if(idx_{next} > \lfloor idx_{next} \rfloor + 0.5)$

$idx_{next} = \lceil index_{next} \rceil$.

$if(idx_{next} = \lfloor idx_{next} \rfloor + 0.5)$

$idx_{next} = iff(R_{0,1} = 0, \lfloor idx_{next} \rfloor, \lceil idx_{next} \rceil)$.

The random variable $R_{0,1}$ is the result of performing the Bernoulli experiment one time with the probability 0.5, that is $P\{R_{0,1} = k\} = p^k(1-p)^{1-k}, k = 0,1, (p = 0.5)$.

The MACS algorithm is described as follows:

Algorithm 3. Modified Ant Colony System

Input: m : the population size of the ant colony,

C: the largest evolution cycles of the ant colony,

n: the number of the abstract Web services in CS ,namely, the length of the solution vector built by one ant,

m_d : the number of the candidate Web service instances in AWS_d, $d \in \{1,2...n\}$,

QoS : the 5-dimensional QoS Vector of one Web service instance in AWS_d,

$CS.W$: the weight vector corresponding to the QoS vector of the composite service,

Output: the optimal or suboptimal solution in the ant colony solution space,

Step1: initialize the directed graph $G =< V, E >$ according to n and m_d, then set the initial pheromone on each edge of G to the value of $\tau_{\max}$, and set the initial heuristic information of each node of G to the value of μ_{init} ,which will keep unchanged during the whole iterative process, the calculation method of μ_{init} is as follows:

$$\mu_{init} = \frac{Q}{\sqrt{t^2 + c^2 + 1/a^2 + 1/rel^2 + 1/rep^2}} \quad (6)$$

Where the five parameters t, c, a, rel, rep construct the 5-dimensional QoS vector of WSI_i, Q is a pre-given constant greater than 0. Set the value of the current cycle to zero, generate m ants for the ant colony.

Step2: initialize the state of each ant in the ant colony, and place all ants to the start node of the directed graph, the tabu list of each ant contains only one node, namely the start node at this time.

Set the value of c to $c+1$.

Step3: do:

$for\ k=1\ \ to\ \ m$

A_k gets the next node to be visited using Algorithm 1.

A_k visits the next node and add the node to its tabu list.

A_k performs the local pheromone updating.

$endfor$.

$until$: each ant has finished building a complete solution, that is, its tabu list is full.

Step4: $for\ k=1\ \ to\ \ m$

Evaluate the quality of the solution built by A_k using $f(A_k.P)$.

Update the historical optimal solution of A_k according to the result of the evaluation.

Set the value of $A_k.P_L$ to $A_k.P$.

$endfor$.

Step5: select the global best ant A_{best} by comparing all the solutions built by the ants.

$AC.P_g^* = A_{best}.P^*$, $AC.fv_g^* = A_{best}.fv^*$.

Step6: A_{best} performs the global pheromone updating.

Step7: $if\ (c \le C)$, go to Step2.

Step8: output $AC.P_g^*$ and $AC.fv_g^*$.

In Step3 of Algorithm 3, A_k performs the local pheromone updating using the following formula:

$$\begin{cases} \tau_{rs}(c) = (1-\varphi)\times\tau_{rs}(c-1) + \varphi\times\Delta\tau_{rs} \\ \Delta\tau_{rs} = \theta\times A_k.fv^* / n \end{cases} \quad (7)$$

where the ordered couple $<r,s>$ is an edge that A_k is building, the parameter φ is the pheromone evaporation coefficient, θ is a constant on the interval $(0,1]$ which is employed to control the proportion of the pheromone increment and is set to 0.1 in this study, $A_k.fv^*$ is the best-so-far solution that A_k has found till the $c-1$ cycle, n is the size of the solution vector. The local pheromone update mechanism makes the pheromone on each edge of the solution vector decrease gradually to avoid that the ants prematurely converge on the same path.

In Step6, A_{best} applies the following formula to perform the global pheromone updating:

$$\begin{cases} \tau_{ij}(c) = (1-\rho)\times\tau_{ij}(c-1) + \Delta\tau_{ij}^{best} \\ \Delta\tau_{ij}^{best} = \rho\times A_{best}.fv^* \end{cases} \quad (8)$$

Similar to the MMAS, after each ant has constructed a complete path, only the global best ant , that is, the ant with the highest quality of solution , has a chance to update the pheromone on its path. The ordered couple $<i,j>$ is an edge on the solution path of A_{best}, ρ is the pheromone evaporation coefficient that belongs to the interval (0,1], $A_{best}.fv^*$ is the fitness value of the optimal solution that A_{best} has built at the current cycle. The global pheromone update mechanism and the application of the pseudo-random proportional rule make the ants concentrate near the global optimal path , this not only avoids that the ants search for their solutions blindly and randomly but makes the searching process more instructive.

Theorem 1: In MACS, let $fv_{g\_best}^*$ be the real global best fitness value in the solution space , let ρ be the pheromone evaporation coefficient of the global pheromone updating, and let c be any cycle of the ant colony, then for any edge $<r,s>\in G.E$ it holds that : $\lim_{c\to\infty}\tau_{rs}(c) <= (fv_{g\_best}^* / \rho)$

Proof: In MACS, after any cycle when the best-so-far ant performs the global pheromone updating, the amount of pheromone it added to edge $<r,s>$ is $\Delta\tau_{rs}^{best} = \rho\times A_{best}.fv^*$, as shown in formula (5) , where $\rho\in(0,1]$ and $A_{best}.fv^* <= fv_{g\_best}^*$. So we have:

$\Delta\tau_{rs}^{best} <= fv_{g\_best}^*$. Then at cycle 1, the maximum possible amount of pheromone added is $(1-\rho)\times\tau_0 + fv_{g\_best}^*$ where τ_0 is the initial pheromone on edge $<r,s>$, at cycle 2 it is:

$$(1-\rho)\times((1-\rho)\times\tau_0 + fv_{g\_best}^*) + fv_{g\_best}^* = (1-\rho)^2\times\tau_0 + (1-\rho)\times fv_{g\_best}^* + fv_{g\_best}^*$$

then at cycle c it is :

$$(1-\rho)^c\times\tau_0 + fv_{g\_best}^*\times\sum_{t=0}^{c-1}(1-\rho)^t$$

,where $\sum_{t=0}^{c-1}(1-\rho)^t$ is a geometric sequence with common ratio $(1-\rho)$, so the maximum possible amount of pheromone that added at cycle c is : $(1-\rho)^c\times\tau_0 + fv_{g\_best}^*\times\frac{1-(1-\rho)^c}{\rho}$,

due to $(1-\rho)\in[0,1)$, hence this sum asymptotically converges to $(fv_{g\_best}^*/\rho)$, so $\lim_{c\to\infty}\tau_{rs}(c) <= (fv_{g\_best}^*/\rho)$

V. EXPERIMENTS AND RESULTS ANALYSIS

A. The data sets and the experimental environment

In the experiments, the composite service adopts the sequential model and takes five randomly generated data sets D1, D2, ... D5 as the experimental data sets to test how the MACS algorithm and its related algorithms work, as shown in Table II. For better accuracy and consistency, all data sets and their related data in the experiments are written to files in advance. Each time before running, the algorithms read the data sets from these files as the experimental data which include: the number of dimensions of the composite service, the number of the candidate Web service instances for each abstract service, the 5-dimensional QoS Vector of one Web service instance, the weight vector corresponding to the QoS vector of the composite service. Before the experiments, the real global optimal solutions and their corresponding fitness values are taken from the solution spaces formed by the experimental data sets by using an exhaustive search method in order to make the comparisons more accurate. Assume that the set of all abstract service is

TABLE II. THE DATA SETS FOR THE EXPERIMENTS

data set number	solution vector size	the number of WSIs	solution space size	optimal fitness value
D1	4	49, 32, 27, 39	1651104	1.8516
D2	5	22, 26, 17, 25, 32	7779200	2.3357
D3	6	31, 8, 10, 23, 20, 32	36505600	2.7090
D4	7	19, 11, 12, 17, 19, 20, 11	178218480	2.8973
D5	8	19, 11, 12, 17, 19, 20, 11, 8	1425747840	3.3769

$\{AWS_1, AWS_2, \ldots, AWS_d, \ldots, AWS_n\}$, in Table 2, the "solution vector size" refers to the number of abstract Web Services in one composite service, namely the $|CS.S|$, the "solution space size" is the product of the sizes of all the abstract services, that is ,the $\prod_{d=1}^{n} |AWS_d|$, the "optimal fitness value" refers to the real best fitness value which is taken from the solution space by using the exhaustive search method. In this study, the number of the total cycles is set to 300, and the population size of the ant colony is set to 16. The experimental environment is as follows: HP Proliant DL380 G5 server with Intel(R) Xeon(R) CPU E5450 @ 3.00GHz and 4G RAM, the operating system is Microsoft Windows Server 2003 Enterprise Edition Service pack 2. The development environment are Eclipse SDK Version with JDK1.6.0 and MATLAB7.4.0(R2007a). Each ACO related algorithm is implemented in Java, and the results are written to files in the experimental process, which will be read by a MATLAB program for plotting the diagrams.

B. The experimental evaluation method

Definition 10: Algorithm Performance Evaluation Index (APEI):

$$APEI = \left(\frac{\overline{fv_{g\_n}} + \overline{\overline{fv_{c\_n}}}}{2 \times fv^*_{g\_best}} \right) \times \left(1 + \frac{(t^*_{g\_n})^{\lambda}}{1 + (c^*_{g\_n})^{\gamma}} \right) \quad (9)$$

The formula of APEI consists of two parts , where in the first part, $\overline{fv_{g\_n}} = \frac{1}{n} \sum_{e=1}^{n} fv_{g\_e}$ is the mean of all the final fitness values of n independently repeated experiments, and fv_{g_e} is the final fitness value after one experiment, $\overline{\overline{fv_{c\_n}}} = \frac{1}{n} \sum_{e=1}^{n} \left(\frac{1}{C} \sum_{c=1}^{C} fv_{g\_c} \right)$ is the mean of all mean fitness values in the iteration processes , where fv_{g_c} is one fitness value obtained at one cycle in one fitness value obtained at one cycle in one experiment. $fv^*_{g\_best}$ denotes the real global best fitness value in the solution space. Then the proportion of evolution is denoted by $\left(\overline{fv_{g\_n}} + \overline{\overline{fv_{c\_n}}} \right) \Big/ (2 \times fv^*_{g\_best})$, and the larger value of the proportion means the closer the fitness value to the $fv^*_{g\_best}$ and the stronger the algorithm to be. In the second part of the APEI, $t^*_{g\_n}$ is the total counts of finding the $fv^*_{g\_best}$ in n repeated experiments, and $c^*_{g\_n}$ is the corresponding total cycles that are spent in finding the $fv^*_{g\_best}$. The parameters λ and γ are two nonnegative real numbers which are the impact factors to represent the relative importance of $t^*_{g\_n}$ and $c^*_{g\_n}$ respectively. The second part of the APEI means that, in n independently repeated experiments, the more counts it has found the real best global solution and the less corresponding total cycles it has spent , the better the algorithm is. In the experiments, the parameter λ is set to 1.83, and γ is set to 1.0, these express that we pay more attention to the algorithm's ability of finding the real global best solution.

The methods of smoothing the curves of other variables in this study are similar to this. Three ACO relative algorithms are compared with each other in this study, they are as follows: the traditional ACO algorithm with the ant-cycle model which is called

TACO in the following experiments, the MMAS algorithm, and the MACS algorithm that is proposed in this study.

The values of the related parameters are set as follows: the evaporation coefficient of global pheromone algorithm, and the MACS algorithm that is proposed in this study. The values of the related parameters are set as follows: the evaporation coefficient of global pheromone updating ρ is set to 0.1784, the evaporation coefficient of local pheromone updating φ is set to 0.1784, the two parameters α and β are each set to 1.0. In MMAS and MACS, the upper and lower limits of the pheromone are set to 0.999 and 0.001 respectively, the pseudo-random proportional selection parameter in MMAS is set to 0.8125.

C. *The experimental results*

Perform the experiment one time on the data set D4, and record the changing process of the indexes obtained in 300 iterations of the experiment, then plot these indexes with their moving average values. In the TACO, there is no mechanism of limited pheromone, and each ant can update the pheromone, so the pheromone changes in a slightly rising trend. Perform the experiment on D4 independently 150 times, and observe the fluctuation of the final fitness value obtained in each experiment. As shown in Fig .2, the fitness values in multiple repeated experiments are always higher than that of the other two algorithms, and the result keeps relatively steady.

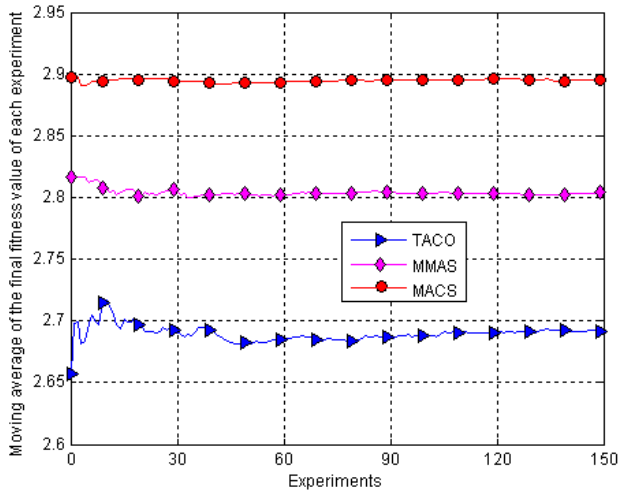

Figure 2. Fitness values in 150 repeated experiments on D4

D. *The comprehensive performance test*

Since all compared algorithms apply random variables, the result of a single experiment may be uncertain and inaccurate. Besides, the data sets used in the experiments are randomly generated with different problem scales, so we perform the experiment on each data set of {D1,D2,…D5} respectively 150 times, and let each experiment iterate for 300 cycles. Record the mean of fitness values, the counts of finding the real best solution and its corresponding total cycles that are spent, then calculate the APEI value based on these indexes by using formula (9)，as listed in Table 3 below. We also compute the standard deviation of the final fitness values that are obtained from the 150-times experiments on different data sets in order to observe the stability of the fitness value evolution. As can be seen from Fig. 3, generally, the deviations between the final fitness values and their means in the MACS are relatively smaller than that of the other two compared algorithms.

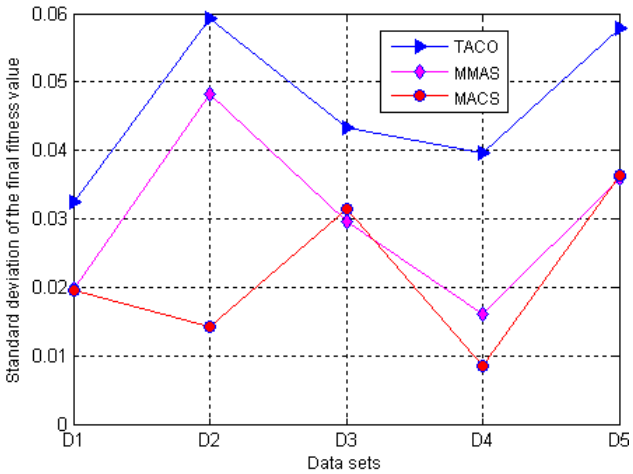

Figure 3. Standard deviation of the final fitness values in 150 repeated experiments on D1~D5

Compute the mean values of APEI on D1~D5 respectively according to the related data listed in Table 3, then we can get that: ACO.APEI=0.9186, MMAS.APEI=1.0200, ACS.APEI =1.2288, as shown in Fig .4. So in the problem of Web service selection, the MACS algorithm demonstrates a better comprehensive performance than the other two compared ACO algorithms, the APEI value of the MACS algorithm is 20.3239% higher than that of the MMAS algorithm.

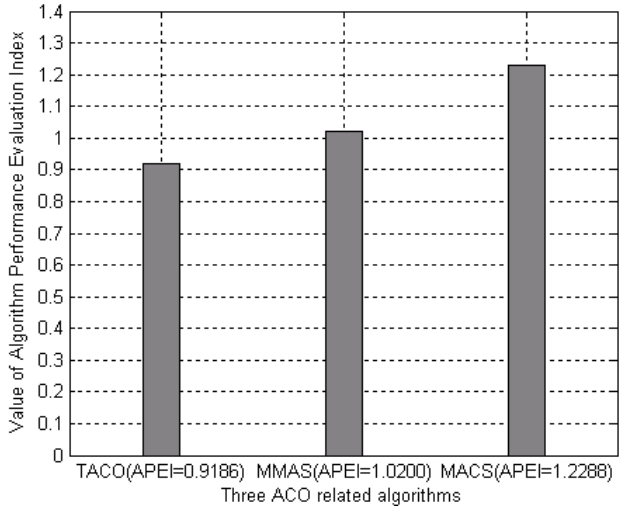

Figure 4. The APEI values in 150 repeated experiments on D1~D5

TABLE III. THE RESULTS OF 150-TIMES REPEATED EXPERIMENTS FOR TACO/MMAS/MACS

dataset	$\overline{fv_{g\_n}}$	$\overline{\overline{fv_{c\_n}}}$	$t^*_{g\_n}$	$c^*_{g\_n}$	APEI	$DAPEI_{TACO}$
D1	1.7269/1.7637/1.8437	1.6785/1.7241/1.7903	0/0/29	0/0/188.1	0.9196/0.9418/1.0667	0/2.12/16.04
D2	2.1268/2.2650/2.3316	2.0603/2.2035/2.3219	2/34/93	129.5/178.8/115.7	0.9086/1.0564/1.3667	0/16.03/48.37
D3	2.4785/2.5160/2.6847	2.4761/2.5152/2.5915	0/13/59	0/196.8/148.2	0.9145/0.9683/1.1783	0/5.85/29.88
D4	2.7456/2.8216/2.8948	2.7237/2.82272/2.8697	0/51/97	0/177.8/132.8	0.9439/1.1172/1.3324	0/17.88/40.82
D5	3.1028/3.2205/3.3617	3.0181/3.2525/3.2983	0/19/56	0/190.4/131.1	0.9063/1.0164/1.1997	0/10.08/31.69

NOTE: $DAPEI_{TACO}$ refers to the APEI difference with TACO

VI. CONCLUSIONS

In this paper we have proposed some useful concepts and methods, such as the multiple solutions random-weighted route selection method, the dynamic pseudo-random proportion selection parameter, and the algorithm performance evaluation index. These concepts and methods provide a new thinking for application researches of WSC optimization problem. Compared with the standard MMAS algorithm, the MACS algorithm proposed in this paper has better comprehensive performance in Web service selection problem. However, whether the MACS algorithm can get good application effect in other optimization problems still needs to be researched.

ACKNOWLEDGMENT

We would like to thank the support of the National Natural Science Foundation of China under grants (No. 61100028, 60803026, 61100028), the Fundamental Research Funds for the Central Universities under grants (No. N110404017); and the National Key Technology R&D Program under grant No.2009BAH43B02, No.2009BAH47B06.

REFERENCES

[1] W3C Working Group. Web services architecture: W3C Working Group Note 11 February 2004. http://www.w3.org/TR/ws-arch/.

[2] Wu Q, Zhu Q. “Transactional and QoS-aware dynamic service composition based on ant colony optimization”. Future Generation Computer Systems,vol. 29(5),pp.1112-1119,2013.

[3] Wang, S.G., Sun, Q.B.,Zou, and H.,Yang,F.C. “Fuzzy Logic Control for Web Service Selection”. Information Technology Journal, vol.11,pp.399-401, 2012.

[4] R. Dinesh Kumar and Dr.G. Zayaraz. “A Qos Aware Quantitative Web Service Selection Model”. International Journal on Computer Science and Engineering(IJCSE), vol.3,pp.1534-1358,2011.

[5] FAN, X.Q.,FANG, X.W., and DING,Z.J. “Indeterminacy-aware service selection for reliable service composition”. Front. Comput. Sci. China, vol.5(1),pp.26–36, 2011.

[6] Chen, Y.P., Zheng,Q.H., and Zhang,J.K. “Mixed intelligent optimization algorithm (MIOA): An algorithm for quality of service (QoS) based web service selection”. International Journal of the Physical Sciences , vol.7,pp.2554 – 2564,2012.

[7] Ludwig,S.A. “Applying Particle Swarm Optimization to Quality-of-Service-Driven Web Service Composition”. Proceedings of Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference , Fukuoka,Japan , 26-29 March, pp. 613- 620 . IEEE, Los Almitos, California,2012.

[8] Zhang W, Chang C K, Feng T, et al. “QoS-based dynamic web service composition with ant colony optimization”.Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual. IEEE,pp.493-502,2010.

[9] Liu H, Zhong F, Ouyang B, et al. “An Approach for QoS-Aware Web Service Composition Based on Improved Genetic Algorithm”. Web Information Systems and Mining, International Conference on IEEE.pp.123-128, 2010.

[10] Zhu,Y.J.,Wen,J.H.,Qin,M.W.,Zhou,G.L. “Web Service Selection Mechanism with QoS and Trust Management”.Journal of Information & Computational Science,vol.8,pp.2327–2334,2011.

[11] Yang Z, Shang C, Liu Q, et al. “A dynamic web services composition algorithm based on the combination of ant colony algorithm and genetic algorithm”. Journal of Computational Information Systems ,vol.6(8),pp. 2617-2622, 2010.

[12] Dhore S R, Kharat M U. “QoS based web services composition using ant colony optimization: mobile agent approach”. International Journal of Advanced Research in Computer and Communication Engineering, vol.1(7) ,pp. 519-527,2012.

[13] OASIS. “Web services business process execution language version 2.0”. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.