QoS-driven Global Optimization Approach for Large-scale Web Services Composition

Minghui Wu[1,2], Xianghui Xiong[1,2], Jing Ying[1,2], Canghong Jin[2], Chunyan Yu[3]
[1] Department of Computer Science and Engineering, Zhejiang University City College, Hangzhou, P.R. China
[2] College of Computer Science and Technology, Zhejiang University, Hangzhou, P.R. China
[3]College of Mathematics and Computer Science, Fuzhou University, P.R. China
Email: {minghuiwu}@cs.zju.edu.cn

***Abstract*—One of the aims of SOA is to compose atomic web services into a powerful composite service. QoS based selection approaches are used to choose the best solution among candidate services with the same functionality. Due to the increasing scale of the candidate services and demands for real-time in some specific application domains, the rapid convergent algorithm for large-scale web service composition is especially important, but rare work has been done to solve the problem. This paper describes the Web services composition model and constructs the web service selection mathematical model. According to these models, service composition problem can be considered as Single-Objective Multi-Constraints optimization problem. We propose a new algorithm named GAELS (Genetic Algorithm Embedded Local Searching), which uses the strategies of enhanced initial population and mutation with local searching, to speed up the convergence. Finally, the in-depth experimental results show that the GAELS algorithm can get the non-inferior solution more quickly and more adaptively than simple genetic algorithm in large-scale web service composition.**

***Index Terms*—SOA; web services composition; QoS global optimization; genetic algorithm; local searching**

I. INTRODUCTION

SOC (Service-Oriented Computing) is a new subject across several domains, such as computer and information technology, business management, business consultation, etc. A lot of simple and convenient services have appeared in recent years (e.g., Google Map). Being one of the most important technologies of SOC, web services are autonomous systems identified by URIs which can be advertised, located, and accessed through messages encoded according to XML-based standard (e.g., SOAP, WSDL and UDDI) and transmitted using Internet Protocols [1]. Web services distributed in various locations can be integrated into a composite service with more powerful function. Through services composition, resources could be reused and we could implement a complicated functionality rapidly at lower cost and faster, which can bring incredible social and economic efficiency.

A composite service is assembled by several tasks to accomplish a mission. In Internet there are maybe many available web services with various QoS (Quality of Service) providing the same functionality to a specific task. So a selection needs to be made. According to the definition of ISO 8402 [2] and ITU [3], QoS can be defined as some nonfunctional attributes, such as price, response time, availability and reputation etc. More about QoS can refer to [4]. During the composition, there are demands for QoS constraints to be met and criterions for QoS optimization. For example, as a constraint condition, response time can not exceed 2ms, and as an optimization criterion, the lower price the better. Therefore, under user's QoS constraint and basic functionality claim, web service composition has to search for an optimal set of services to construct a composite service. And how to make a fast selection among a large number of candidate services is the main research goal of the paper.

QoS-driven web services selection is a NP-hard problem [5]. Applied to this problem, SGA (Simple Genetic Algorithm) has a good time consumed performance, but on-going research works mainly aim to small scale of candidate services. If the scale is large, SGA could not satisfy the time consumed performance requirement, especially for those real-time or interactive systems. This paper proposes a method named GAELS (Genetic Algorithm Embedded Local Searching) which applies the policies of enhanced initial population and mutation with local searching to speed up the convergence of genetic algorithm and reduce the time consumed for large-scale composition. The empirical evaluation based on simulation demonstrates that GAELS can get the result faster than SGA and has a better adaptability to the increasing of composition scale.

The remainder of this paper is organized as follows: the section 2 provides some related works of web services composition, and the section 3 describes the web services composition model. Then in section 4, we introduce the multi-dimensions QoS model. A particular depiction of the QoS-driven web service selection mathematical model and algorithm has been given in section 5, and in section 6 there is a contrastive empirical evaluation. Finally, section 7 concludes this paper and prospects the future work.

II. RELATED WORKS

The web services composition model makes an important role in the web service composition problem. A service process may have multiple execution routes, and the same abstract service may be bound to different

doi:10.4304/jcp.6.7.1452-1460

candidate services after each execution route doing the optimization in its own scope. Zeng [1] uses hot path, the most frequently executed plan containing the abstract service, to handle this problem. But for those plans which are not hot, the binding method might dissatisfy the constraints. Moreover, multiple execution routes need to call the optimization module several times, which will bring extra time consumed. To solve this problem, Yu [6] proposes an approach using integrative optimization. It could avoid the dissatisfaction of constraints and use the optimization module for only once.

Some approaches of web service selection are based on semantic web [7-9], and others are basis of QoS attribute computing [1, 5, 6, 10-16]. The former has difficulties in global QoS evaluation. At present there are many approaches base on QoS attribute computing. Zeng [1] proposes two ways, local optimization and global planning by using integer programming. Local optimization will obtain optimal candidate services in one abstract service scope without considering constraints across multiple abstract services and contributions to optimize global QoS criterions. Consequently, the binding of abstract services is independent with each other. Unless QoS of service is changed or unavailable, the service binding do not need to redo. The time consumed will be stable if we keep identical scale of service composition. However, unlike global planning which can get better global QoS, local optimization could not follow global criterion. Nevertheless, global optimization also has its defect: the time cost will grow proportionally with the amount of execution routes for the same scale of services. Yu [6] defines the service composition problem as a Multi-dimensions Multi-choice 0-1 knapsack problem (MMKP) in combinatorial model and a Multi-constraint optimal path (MOP) problem in graph model, and solves it by using Integer Programming. Li [11] proposes a mapping framework to construct Service Overlay Network (SON), and translates web service composition problem to single constraint path selection problem. In the end, Dijistra shortest path algorithm could give an optimal selection. Ko [15] applies an intelligent algorithm mixed Simulated Annealing (SA) and Tabu Search (TS) to service composition. Ye [10] implements reusing of service composition scenario by Case-Based Reasoning (CBR), which reduces the cost and time consumed of composition. The strongpoint of QoS attribute computing based on integer programming lies on the maturation of theory and plenty of approaches. But it claims that the QoS constraints and QoS criterions should be linear while there are many non-linear QoS attributes just like availability. Simple Genetic Algorithm is also been applied to web service selection [5, 13, 16]. In contrast to Integer Programming, it has no requirement on whether the QoS constraints and objective function are linear or not, which extends the field of application rooting from Genetic Algorithm's dependence to problem areas. On the other hand, Genetic Algorithm has better time consumed than Integer Programming as the increasing of composition scale [13].

The rapid development of web services promotes the expanding of service scale, so the research of large scale web service composition is necessary. The previous work did not pay attention to a large number of execution plans selection. For example, the number of candidate plans is from 10^5 to 10^{20} [5, 12, 13, 16]. For real-time large scale web service composition problem, which has many execution plans resulting from a lot of abstract services and candidate services, the key point is how to get non-inferior solution fast. Simple Genetic Algorithm will cost a lot of time to find a solution in huge search field. GAELS proposed in this paper improves the efficiency of search algorithm and the adaptability to real-time large scale web service composition.

. WEB SERVICES COMPOSITION MODEL

A. Basic Definition

Definition 1: (Abstract Service). Abstract service has function descriptions without implementation and standard service interfaces across different service providers. An abstract service is corresponding to a workflow task.

Definition 2: (Service Instance). Service instances are concrete services published by service providers. They could give the function implementation specified by abstract services. And some service instances may have the same function, but different QoS.

Definition 3: (Candidate Relationship). While the function of several service instances $S_1,...,S_n$ are consisted with the function description of abstract service T, we state that service instances $S_1,...,S_n$ and abstract service T have the candidate relationship. We also say that $S_1,...,S_n$ is the candidate services of T which is labeled as $S_i \in T \quad (i=1...n)$.

Definition 4: (Service Function Graph). Constructing abstract services as a workflow to fulfill user's requirement in functionality will obtain a service function graph. In the service function graph, we have two additional special abstract services treated as Start label and End label, which have no functional meaning. An example is given as Fig.2.

Definition 5: (Service Selection Graph). Shown as Fig.3, all the service instances corresponding to abstract services in Service Function Graph of Fig.2 have been discovered, and a service function graph appears. In the process of service discovering, we will discard service instances that do not meet local constraints. Consequently, local constraints are already met in service selection graph, and we will not take account of local constraints in the following discussion.

B. Web Service Composition Procedure

We abstract web service composition procedure as three phases:

Firstly, compose abstract services as a service function graph to meet user's functionality demand in abstract hierarchy.

Second, discover all the service instances which have candidate relationships with abstract services. And transform service function graph to service selection graph.

Last, select a service chain linking the start node and end node in service selection graph, and the chain could keep the QoS constraints.

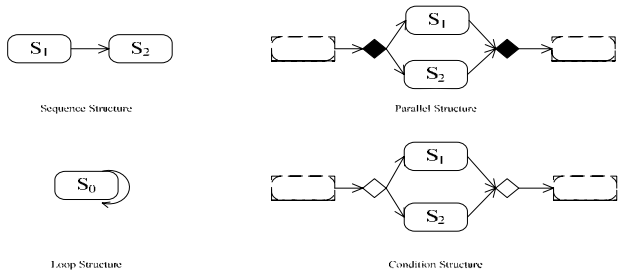

Figure1. Service flow graph

In service function graph and service selection graph, the composite structure among nodes could be separated as Fig.1 shows:

- Sequence Structure: S_1 is the frontal service of S_2, and S_2 is the successive service of S_1. The finish of S_1 will trigger S_2.
- Parallel Structure: S_1 and S_2 will be triggered at the same time, and successive service will start after both of them finished.
- Loop Structure: service S_1 will execute itself for several loops.
- Condition Structure: service S_1 will be triggered in probability p_1, and service S_2 in probability p_2, and $p_1 + p_2 = 1$. Moreover, once either of them finish, successive service will be triggered.

All other structures could be woven by these four basic structures. In the following discussion, loop structure will not appear again, because we could handle loop structure using unlooping method which quickly converges to a sub-optimal solution [5], and unlooping method is in accord with our emphasis goal.

Special for service function graph, we state two conceptions as following [6].

Definition 6: (Execution Route). In the service function graph, execution route is a passageway between start node and end node, and only include one spur track for every condition structure. If there are *k* execution routes with a probability $\rho_i (i = 1...k)$ in a service function graph, then $\sum_{i=1}^{k} \rho_i = 1$.

In Fig.2, there are two execution routes:

$ER_1 : (S_1, S_2, S_3, S_4, S_6)$, with probability ρ_1

$ER_2 : (S_1, S_2, S_3, S_5, S_6)$, with probability ρ_2

and $\rho_1 + \rho_2 = 1$.

Definition 7: (Execution Plan). For execution route $(S_1, ..., S_n)$, we define $(T_1, ..., T_n)$ as a execution plan of execution route $(S_1, ..., S_n)$, if $T_i \in S_i \quad (i = 1...n)$, here T_i denotes service instance, S_i represents abstract service. According to the definition, an execution plan is an executable service chain which meets user's requirement.

In Fig.3, execution plans of execution route ER_1 could be: $(S_{11}, S_{21}, S_{31}, S_{41}, S_{61}), ..., (S_{12}, S_{23}, S_{33}, S_{42}, S_{63})$.

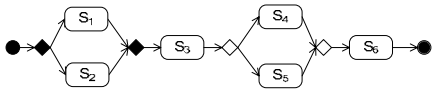

Figure2. Service function graph

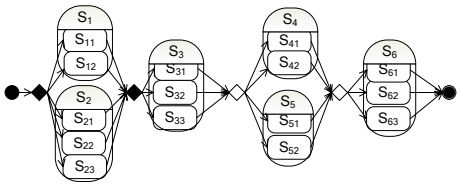

Figure.3 Service selection graph

Ⅳ. MULTI-DIMENSIONS QOS MODEL

A. Multi-dimensions QoS Model

QoS is a series of non-function attributes, such as reputation, price, availability, duration time. During the service composition, all services instances belong to the same abstract service have identity functionality but different QoS, so QoS is the only criterion while binding abstract service in this situation. In this paper will just use three attributes: reputation, price and availability into account. Other attributes could be imported and will not impact our results. Due to article [1] has already given an explicit depiction of definition and calculation methods of QoS attribute, we will not give a duplicate description.

B. QoS Attribute Standardization

Because different QoS values have different value range, it is unfair to calculate these values directly. In our paper, we should standardize them before computing. For example, value scope of service price is [2000, 5000], but the value of reputation may only range from 1 to 10. So QoS attribute standardization is needed.

In this paper, we state the *jth* service instances as S_{ij} of abstract service S_i, and the QoS attributes of S_{ij} is $(q_{j1}, ..., q_{jn})$. QoS attributes can be separated into positive attributes and negative attributes. The positive attribute, as availability and reputation, means the higher value the better. Negative attribute, as price and response time, is contrast with positive one. So we standardize positive attribute with formula (1) and standardize negative attribute with formula (2)

$$q_{jk} = \frac{q_{jk} - \alpha_k}{\sigma_k} \quad 1 \le k \le n \tag{1}$$

$$q_{jk} = 1 - \frac{q_{jk} - \alpha_k}{\sigma_k} \quad 1 \le k \le n \tag{2}$$

Where α_k, σ_k are the average and standard deviation of the QoS values for all candidates of kth QoS attribute.

C. QoS of Execution Plan (Composite Service)

Execution plan $EP(S_1,...,S_n)$ is a powerful service composed by a series of other services. Like a normal service, execution plan has its own QoS. The QoS calculation formula of execution plan is displayed in Table 1.

TABLE I.
QOS CALCULATION FORMULAS

QoS attribute	Formula
Price	$q_{price} = \sum_{i=1}^{n} q'_{price}(i)$
Reputation	$q_{reputation} = \frac{1}{n}\sum_{i=1}^{n} q'_{reputation}(i)$
Availability	$q_{availability} = \prod_{i=1}^{n} q'_{availability}(i)$
Others	Formulas decided by attribute

D. QoS of Execution Route

In execution route $ER(S_i,...,S_n)$, the candidates of abstract service $S_i(1 \le i \le n)$ are $S_{i1},...,S_{il}$, and QoS of the jth service instance S_{ij} is $(q_{j1},...,q_{jm})$, so the QoS calculation formulas are as follows:

$$Q^{repu} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{i_l} x_{ij} \cdot q_{ij}^{repu}}{n} \tag{3}$$

$$Q^{price} = \sum_{i=1}^{n}\sum_{j=1}^{i_l} x_{ij} \cdot q_{ij}^{price} \tag{4}$$

$$Q^{reliability} = \prod_{i=1}^{n}\sum_{j=1}^{i_l} x_{ij} \cdot q_{ij}^{reliability} \tag{5}$$

When the service S_{ij} is selected as an instance of the abstract service S_i, we set $x_{ij} = 1$, otherwise $x_{ij} = 0$. The i_l denotes the amount of candidate services of the abstract service S_i. Other QoS calculation formula of execution route could be imported by the same way.

E. Utility Function

A service has multi-dimensions QoS value $(q_i,...,q_j)$, so we need to consider all the QoS in web service selection process. But multi-dimensions value is not easy for comparison. Thus we propose a utility function, mapping multi-dimensions value into a single function value which is a scalar quantity, to give a comprehensive reference for our comparison. The definition of utility function is as follows:

$$UF(s) = \sum_{i=1}^{n} \omega_i \cdot q_i' \quad \text{and} \quad \sum_{i=1}^{n} \omega_i = 1 \tag{6}$$

Where ω_i denotes the weight of the ith QoS attribute which reflects their importance. q_i is the standard attribute value.

V. QOS-DRIVEN WEB SERVICE SELECTION MATHEMATICAL MODEL AND ALGORITHM

A. Web Service Selection Mathematical Model

QoS-driven web service selection problem could be mapped into single constraint multi objectives optimization problem in mathematics. We will first give web service selection mathematical model for single execution route, and the model for multi execution routes will be displayed.

A.1 Selection Model for Single Execution Route

An execution route could make multi execution plans when abstract service is bound to different candidate services. So web service selection problem has to choose the optimal execution plan among all execution plans. In mathematics, it could map into a single constraint multi objectives optimization problem.

First of all, all the abstract services in an execution route are topological sequenced, each abstract service has its own id i. Then we do the same things to all candidate services of each abstract service, and every candidate has an id j. We assume that abstract service $S_i(1 \le i \le n)$ in execution route $ER(S_i,...,S_n)$ has candidate services $(S_{i1},...,S_{il})$, and i_l is the amount of service instances included in abstract service S_i. Then, the corresponding single objective multi-constraints optimization problem is:

1) Objective function

$$MAX \quad f(ER) = UF(ER) = \sum_{i=1}^{n} \omega_i \cdot Q_i' \tag{7}$$

Where $Q_i(1 \le i \le n)$ is the ith QoS attribute value of execution route ER, derives from QoS calculation formula of execution route for standardization QoS. ω_i denotes the weight.

2) Global QoS constraint

Regarding that QoS values in global QoS constraints are actual data in application, so all the QoS values are not standardized in this part. Global QoS constraints can be divided into two groups.

- Single selection constraint

In the web service selection process, there is only one candidate service of each abstract service can be selected into composition flow. So single selection constraint can formalize as below:

$$\forall i \in n \quad , \quad \sum_{j=1}^{i_l} x_{ij} = 1 \tag{8}$$

Where n denotes the number of abstract services in execution route. When jth candidate service in abstract

service i is selected, $x_{ij}=1$; Otherwise, $x_{ij}=0$, i_l is the amount of service instances of abstract service i.

- QoS value constraint

QoS value constraints are constraints proposed by users, such as service price don't go beyond 100$, service availability don't be under 99.9%. If there are h QoS value constraints $C^k, 1 \le k \le h$, and k denotes the kth QoS attribute , then

$$\forall k, \quad 1 \le k \le h \quad ; \quad Q_k \le C^k \tag{9}$$

Where Q_k is the QoS value with attribute k of execution route i.

A.2 QoS-driven Web Service Selection Mathematical Model for Service Function Graph

There are may be multiple execution routes in a service function graph. For example in service function graph of Fig.4, There are three execution routes exist: $path\ 1:(S_1,S_2,S_4,S_5,S_7)$, $path\ 2:(S_1,S_3,S_6,S_8,S_{10})$ and $path\ 3:(S_1,S_3,S_6,S_9,S_{10})$. Web service selection of service function graph with multiple execution routes is more close to the application in reality. The problem can map into single objective multi-constraints optimization problem in mathematics.

A.2.1 The Difficulty of Web Service Selection Problem of Service Function Graph

In the service function graph, each execution route after optimization will get a best execution plan among its route, then for multiple execution routes will obtain multiple optimal execution plans. And it will bring two problems: 1) how to merge multiple optimal execution plans, 2) the emergence will need extra resource consumed. Refer to service function graph of Fig.4, abstract service S_1 is bound to candidate service S_{1i} in the optimal execution plan after optimization of $path1$. While S_{1j} will be bound to S_1 in optimization result of execution route $path2$. So which one should be selected, S_{1i} or S_{1j}? [1] proposes a method hot-path to deal with this problem. The hot path chooses the candidate service most frequently selected before when binding conflicts exist; and if conflicts don't exist, it will chose the candidate service selected by its execution route optimization. But this approach takes two weak points: first, sometime the actual execution plan may not conform to the hot path, and the execution may break the QoS constraints. Second, hot path indeed need to execute service selection module as many times as the potential execution routes, and this will consume extra resource.

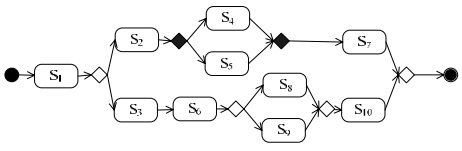

Figure.4 Service function graph with multiple execution routes

This paper will use the approach multiple execution routes comprehensive optimization to address the problem.

A.2.2 Selection Model of Service Function Graph with Multiple Execution Routes

Abstract web service selection problem of multiple execution routes into single objective multiple constraints optimization problem, then we could comprehensive optimize multiple execution routes, and there will be only one binding for each abstract service and only one time to execute the optimization module. Overall, this approach can avoid the two weak points above totally.

First, topological sequencing all the abstract services and service instances, and every abstract service and service instance will have an unique id.

The execution routes of service function graph are $ER_1, ER_2, ..., ER_s$, and execution probability of $ER_i (1 \le i \le s)$ is ξ_i, where $\sum_{i=1}^{s} \xi_i = 1$.

1) Objective function

$$MAX \quad f(SFG) = \sum_{i=1}^{s} \xi_i \cdot f(ER_i) \tag{10}$$

Where $f(ER_i)$ is defined by formula (7).

2) Global QoS constraints

- Selection constraints

It is the same as formula (8).

- QoS value constraints

In order to meet the QoS value constraints strictly, we enforce every execution route to satisfy the constraint requirements. If there are h QoS value constraint $C^k, 1 \le k \le h$, and k denotes the kth QoS attribute, then

$$\forall i,k \quad 1 \le i \le s \quad 1 \le k \le h \quad ; \quad Q_k^i \le C^k \tag{11}$$

Where Q_k^i is the kth QoS attribute value of the execution route i.

B. QoS-driven Web Service Selection Algorithm

Being a NP-hard problem, QoS driven web service selections exhausts traditional algorithm numerous time and even more as increasing of scale, while Simple Genetic Algorithm behaves good adaptation [3,15].

Genetic Algorithm is proposed by professor John H. Holland and his students in last century, then De Jong and Goldberg's additional work improved it. Right now, it has become one of the most broad and successful intelligent optimization methods. Genetic Algorithm evolves the generation step by step imitating crossover, mutation and selection of biological principle [18]. But the update of Simple Genetic Algorithm's mutation operation is not aggressive, which lead to slow hill-climbing [19]. Applying local searching to mutation operation can improve hill-climbing of Simple Genetic Algorithm and promote convergence performance. In the following, we

will introduce our algorithm GAELS, and compare it with SGA which has been used in web services [5, 13].

B.1 Fitness Function

GAELS and SGA will use the same fitness function, just like $f(SFG)$ in formula (10). Although fitness function in [5, 13] is different, it won't affect the experiment result of SGA and our comparable evaluation.

B.2 Generation of initial population

SGA: Randomly generate individuals of initial population, so the average fitness of initial population is low.

GAELS: Enhanced initial population can accelerate the convergence of population [12]. So as to generate high population average fitness, GAELS adopts the following steps based on the method proposed in [12]:

First, calculate the proportion between the composition number of every execution plan and the sum of all composition of all execution plans. So the more execution plans of the execution route has, the more chromosome number of the route in population is, which could keep the diversity of the initial population.

Second, the value of each chromosome gene is decided by topological number of the selected candidate service. And the method like roulette wheel selection [20] is adopted. The algorithm is as following:

Input: chromosome $S(S_1,...,S_n)$, candidate services of abstract service S_i is $S_{i1},...,S_{il}$, and il denotes the number of candidates included. The selection function is $f(S_{ij})$, just like formula (6).

Output: the coding $(t_1,...,t_n)$ of chromosome S

Begin:

(a) *For* $i=1$ *to* n

(b) *set* $sumf(S_i)=0$

(c) *For* $j=1$ *to* i_l

(d) *caculate* $f(S_{ij})$

(e) $sumf(S_i) = sumf(S_i) + f(S_{ij})$

(f) *End For*

(g) *For* $j=1$ *to* i_l

(h) *set* $P_{ij} = f(S_{ij}) \Big/ sumf(S_i)$

(i) *End For*

(j) *set the roulette wheel*

(k) *generate a random number and select*

(l) *End For*

The steps (c), (d), (e), (f) calculate the sum of selection function values of candidates for abstract service S_{ij}. And the steps (g), (h), (i) obtain the probability of every service instance to be selected. Then the following steps (j), (k) apply the method roulette wheel selection to give value j of the ith gene position of the chromosome, which means abstract service S_i is bound to service instance S_{ij}.The algorithm above promises that candidate service with better QoS also has the higher probability to be selected.

Last, repeat the first and second step until obtain enough individuals.

B.3 Structure of Chromosome

GAELS and SGA adopt the same chromosome structure. For a service function graph with n abstract services, the gene position number of chromosome is also n, and the i $(1 \le i \le n)$ gene position is corresponding to the ith abstract service in topological sequencing. The value range of i gene position is integer in $[1, i_l]$, and i_l denotes the amount of service instances belonging to abstract service i $(1 \le i \le n)$. The structure is like [5, 13].

B.4 Crossover

In this step, we randomly select two parent chromosomes to do two-point crossover operation, and we will have a new child chromosome. Because the new one may break the global QoS constraint, it needs to be checked for verification. If the new chromosome fails to meet constraint, we will discard this chromosome, and repeat the process until find a new chromosome fulfilling constraints. GAELS has the same crossover operation as SGA.

B.5 Mutation

The mutation algorithm of SGA and GAELS just as following:

Input: the chromosome coding $S(t_1,...,t_n)$ to mutate, and available value of the ith chromosome gene position range from 1 to i_l.

Output: the chromosome coding $S(t_1,...,t_n)$ after mutation.

SGA Mutation:

(a) *random generate a mutation gene locus* i

(b) *generate a random number* i_{random} *in* $[1, i_l]$

(c) *set* $t_i^{'} = i_{random}$

(d) *set* $t_j^{'} = t_j \quad (\forall j \ne i)$

GAELS Mutation:

(a) *random generate a mutation gene locus* i

(b) *For* $j=1$ *to* i_l

(c) *set* $S_{temp}(t_1,...,t_{i-1}, j, t_{i+1},...,t_n)$

(d) *caculate* $FitnessTemp[j] = UF(S_{temp})$

(e) *End For*

(f) *select* $\max\ FitnessTmep[j] = FitnessTemp[i_{opt}]$

(g) *set* $t_i^{'} = i_{opt}$

(h) *set* $t_j^{'} = t_j \quad (\forall j \ne i)$

In SGA mutation the step (a) produces the gene locus value i to mutate, step (b) and (c) regards the random integer i_{random} as the gene locus value after mutation; In GAELS mutation steps (b), (c), (d) and (e) obtain chromosome fitness array $FitnessTemp[i_l]$ after all

available value of gene locus i mutating by themselves, then in the (f) step selects gene locus value i_{opt} with the max fitness value. It is clear that fitness of new individual after mutation in SGA is not high, while GAELS mutation enhances the fitness of individual after mutation, which could accelerate the convergence speed of generation. In the end, QoS constraints checking must be imposed to the new individual after mutation, and the mutation need to redo until meeting the QoS constraint.

B.6 Selection

GAELS and SGA both use optimal individual preserved and roulette-wheel selection method [20]. After selection operation to the candidate population, new generation will go back to the constant scale.

VI. Empirical Study

A. Objective of Experimental Study

Aiming to real-time large-scale web service composition problem, we propose the algorithm GAELS, which adopt two strategies: enhanced initial population and mutation with local searching, for the purpose of get a composite service with no-inferior QoS. In order to verify the performance of the strategies in use, we will compare three algorithms SGA, GAELSWEIP and GAELS, where GAELSWEIP is almost identical with GAELS except for without enhanced initial population strategy. We must lay emphasis on that our empirical study is specific to time performance of large scale composition.

B. Experiment Environment and Parameter Setting

The experiment is run on Windows XP with AMD@Athlon, 64*2 Dual-Core processor TK-57 (1.9G), 1GB RAM. And program language is JAVA with IDE Eclipse 3.2.

Genetic Algorithm is set up by the following parameters:

cross rate: $\alpha = 0.9$, mutation rate $\beta = 0.2$, max evolving generation of population $\wp_{\max} = 1000$, population convergence parameter $\kappa = \frac{Fit_{\max} - Fit_{even}}{Fit_{\max}}$, and $Fit_{\max}$ denotes max individual fitness value of population, Fit_{even} denotes population average fitness .

The halt criterion of algorithm is:

SGA: As population parameter $\kappa \le 0.05$ or population evolving generation is equal to $\wp_{\max}$, halt.

GAELS, GAELSWEIP: As max individual fitness of population surpass that of SGA while SGA's halting, or population evolving generation is equal to $\wp_{\max}$, halt.

Halt criterion above can ensure that convergence result of GAELS and GAELSEIP is not inferior to that of SGA, when GAELS and GAELSEIP stop normally (evolving generation less than $\wp_{\max}$). Even they are abnormal, still can halt, but time consuming will be greatly bad than outweigh SGA.

We will take into account three QoS attributes: availability, price, reputation. Availability is the random number in interval [0.0, 1.0], and price is the random number in [50, 2000], while reputation random generates from interval [0.5, 1.0]. There still exist global QoS constraints for all attributes, and the weight of each attribute is 1/3.

The structure of service function graph in Fig.4 contains 10 abstract services. When the number of abstract services is 20, then structure will duplicate structure of Fig.4 in series. In our experiment, the number of abstract services will increase iteration of 10, and the structure is iteration in series also.

C. Study Result and Analysis

We fix the number of service instances of each abstract service as 50. Then compare the run time of three algorithms under identity QoS data, with number of abstract service varying from 10 to 90 at step 10. So as to reflect the empirical result have nothing to do with QoS data, we generate 50 groups different QoS data under each service scale. Fig.5 and Fig.6 show the experiment result.

From Fig.6, we can note that as the expanding of abstract service, convergence time of GAELS grows slowly while SGA and GAELSWEIP increase rapidly. In the other hand, time cost of GAELS is less than the other two, under the premix of getting a better result than SGA and GAELSWEIP. Then combining with Fig.5, we can note that GAELS behaves the merits above all the time under 50 groups random QoS data, and the merits is independent with QoS data.

In the evaluation, we notice that time consumed of GAELS and GAELSWEIP is very high in some cases, especially for small number of service instances, at these cases, the evolving generations of GAELS and GAELSWEIP when halt arrive at our threshold value $\wp_{\max}$, we name it "storm". When storm happens, which is an unstable phenomenon, evolving generations of GAELS and GAELSWEIP is much larger than SGA, in order to exceed the max fitness value of SGA. So as to research the "storm", we fix the number of abstract services as 50, and increase the number of service instances from 10 to 100 at step 10, then watch what happens. Moreover, for the purpose of excluding interference of QoS data, we generate 50 groups of random QoS data and take the average of 100 running times under each scale. Fig.7 plots the result. We can get that storm times decrease greatly as the increasing of service instances, and when the number of service instance is 30, storm times of GAELS and GEALESWEIP are only 0.92 and 0.64; and they will become 0.16 and 0.12 as the number of service instances is 40. Furthermore, there is almost no storm when the number of service instances is more than 50.

From all the experiment result, we could conclude that along with expanding of abstract service, convergence time of SGA and GAELSWEIP increase quickly while GAELS performs the opposite case in the condition of holding a better time cost, so GAELS show more

suitable. But when number of service instances is too small, GAELS and GAELSWEIP will have fluctuations, which is the storm phenomenon. And as the rising of service instances, storm times reduce rapidly till 0. In brief, while the scale of abstract service and candidate services is large, GAELS behaves a better efficiency than that of [5, 13] when applying to service composition.

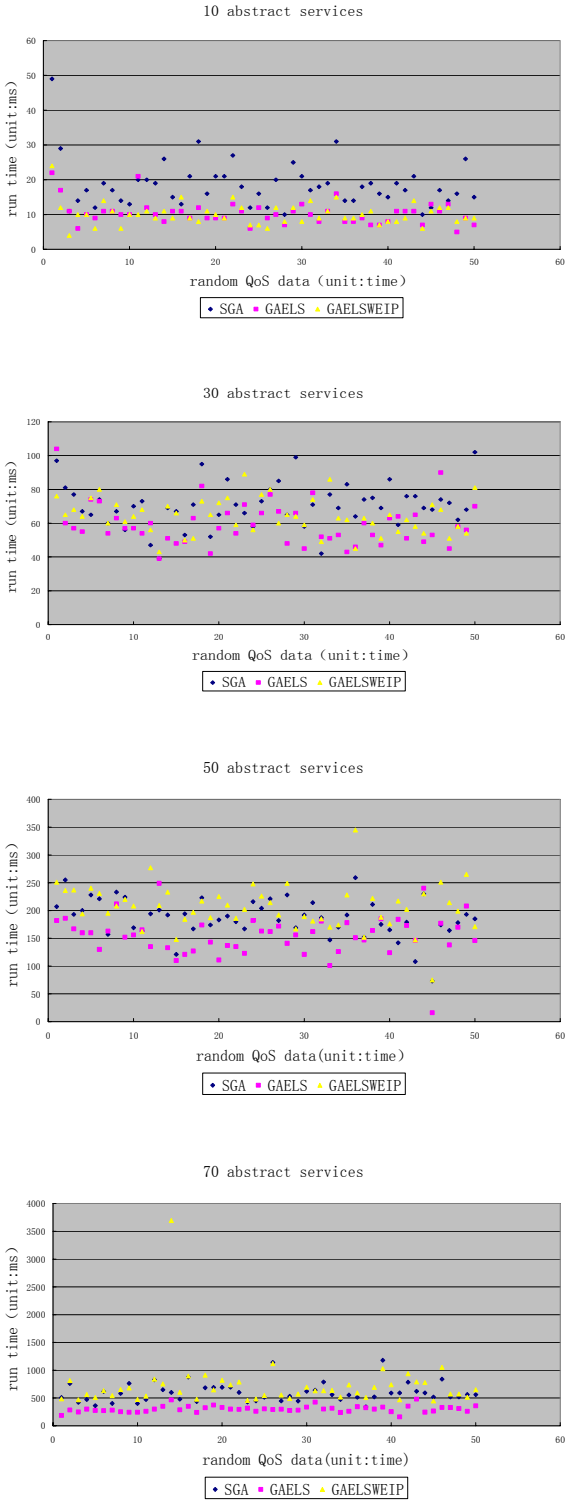

Figure.5 Convergence time comparison while abstract service increasing

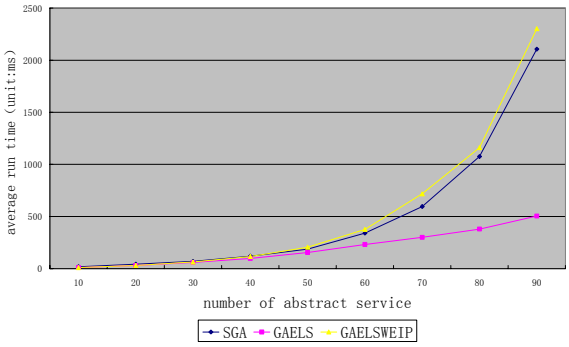

Figure.6 Average convergence time comparision as abstract service increasing

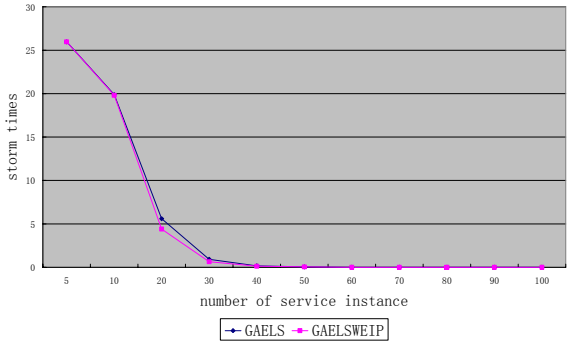

Figure.7 Strom times movement under 50 random QoS data as service instance increasing

Ⅶ. Conclusion and Future Work

Services composition has a broad prospect in the future. And large-scale services composition is the problem we must face during the development of SOA.

Our work at present mainly focuses on constructing a platform for SOA application development: SMICE (Semantic Model-driven Integrated Construction Environment). Research of services composition and selection with a large number of abstract services and complex workflow is an important part in this project. At the early stage we had developed a QoS driven service composition ontology framework [21], which provides a basis for web service composition and selection. Our work in this paper mainly lays on the following two aspects: 1) Develop a multi-dimensions QoS model and web service composition model in detail and map it into single objective multi-constraints problem. 2) Propose a large scale web service composition oriented algorithm GAELS, which uses enhanced initial population and mutation with local search strategies to accelerate convergence. And empirical study shows that when apply to large scale services composition problem, GAELS can obtain the approximate optimal solution more quickly than SGA, and perform more suitable and effective to the increasing of composition scale.

In the future, we will improve algorithm GAELS in details, particularly the strategy of mutation with local search, which spend too much time on search all the candidates of abstract service. Maybe the time waste can be shorten by applying searching with rank and cash principle. On the other hand, how to combine service

semantics and QoS attribute in services composition is the other goal of our work.

ACKNOWLEDGMENT

This work was supported in part by the National High-Tech Research and Development Plan Foundation (863), China (Grant No.2007AA01Z187) and NSFC (Grant No. 60805042)

REFERENCES

[1] L.Z. Zeng, B. Benatallah, A.H.H Ngu et al, "QoS-Aware Middleware for Web Services Composition", *IEEE Trans. Softw. Eng.*, Vol.30, No.5, 2004, pp. 311-327.
[2] UNI EN ISO 8402 (Part of the ISO 9000-2002), *Quality Vocabulary*.
[3] ITU, *Recommendation E.800 Quality of service and dependability vocabulary*.
[4] J. Cardoso, *Quality of Service and Semantic Composition of Workflows*, University of Georgia, 2002.
[5] M. D. P. G Canfora, R Esposito, M.L Villiani, "A Lightweight Approach for QoS-Aware Service Composition", in *Proceeding of 2nd International Conference on Service Oriented Computing*, 2004, pp. 36~47.
[6] T. Yu, Y. Zhang, K.J. Lin, "Efficient algorithms for Web services selection with end-to-end QoS constraints", *ACM Trans. Web*, Vol.1, No.1, 2007.
[7] C. Zhou, L.T. Chiq, B.S. Lee, "DAML-QoS Ontology for Web Services", in *Proceedings of the 2004 IEEE International Conference on Web Services* (ICWS2004), 2004, pp. 472-479.
[8] A.S. Bilgin, M.P. Singh, "A DAML-Based Repository for QoS-Aware Semantic Web Service Selection", in *Proceedings of the 2004 IEEE International Conference on Web Services* (ICWS2004), 2004, pp. 368~375.
[9] M. Tian, A. Gramm, H. Ritter et al, "Efficient Selection and Monitoring of QoS-Aware Web Services with the WS-QoS Framework", in *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, 2004, pp. 152~158.
[10] X. Ye, R. Mounla, "A Hybrid Approach to QoS-Aware Service Composition", in *Proceeding of the 2008 IEEE International Conference on Web Service* (ICWS2008), 2008, pp. 62~69.
[11] Y. Li, J. Huai, T. Deng et al, "QoS-aware Service Composition in Service Overlay Networks", in *Proceeding of the 2007 IEEE International Conference on Web Service* (ICWS2007), 2007, pp. 703~710.
[12] Y. Ma, C. Zhang, "Quick convergence of genetic algorithm for QoS-driven Web service selection", *Computer Networks*, Vol.52, No.5, 2008, pp. 1093~1104.
[13] G. Canfora, M.D. Penta, R. Esposito et al, "An approach for QoS-aware service composition based on genetic algorithms", in *Proceedings of the 2005 Conference on Genetic and evolutionary Computation*, 2005, pp. 1069~1075.
[14] L. Cao, M. Li, J. Cao, "Using genetic algorithm to implement cost-driven Web service selection", *Multiagent and Grid Systems*, Vol.3, No.1, 2007, pp. 9-17.
[15] J.M. Ko, C.O. Kim, I.H. Kwon, "Quality-of-service oriented Web service composition algorithm and planning architecture", *The Journal Of Systems and Software*, Vol.81, No.11, 2008, pp. 2079~2090.
[16] L.J. Zhang, B. Li, T. Chao et al, "On demand Web services-based business process composition", in *Proceeding of IEEE International Conference on System, Man and Cybernetics* (SMC'03), vol.4, 2003, pp. 4057~4064.
[17] D.Y. Fan, Y.H. Chen, *Probability and Statistics*, China: Zhejiang University Press, 1996.
[18] M.D. Vose, *The Simple Genetic Algorithm: Foundation and Theory*, USA: MIT Press, 1999.
[19] D.W. Wang, J.W. Wang, H.F. Wang et al, *Intelligent Optimization Methods*, China: Higher Education Press, 2007.
[20] C.R. Reeves, J.E. Rowe, *Genetic Algorithms: Principles and Perspectives: A Guide to GA Theory*, Netherlands: Kluwer Academic Publisher, 2002.
[21] M.H. Wu, C.H. Jin, C.Y. Yu, et al, "QoS and Situation Aware Ontology Framework for Dynamic Web Services Composition", in *Proceeding of Proceedings of the 2008 12th International Conference on Computer Supported Cooperative Work in Design* (CSCWD' 08), Vol.1, 2008, pp. 459~464.

Minghui Wu was born in Leping, Jiangxi Province of China in 1976. He received the BS degree in Computer Science and Engineering from Nanchang University in July 1997 and MS degrees in Computer Science and Engineering from Zhejiang University in March 2000. Now he is the Ph.D. candidate in Computer Science of Zhejiang University. Since Dec 2006, he serves as an associate professor of Computer Science at Zhejiang University City College. His major interests include Software Engineering, System Dynamics Modeling, Model-Driven Development, Semantics Web, and Model Checking.

Xianghui Xiong was born in Xiaogan, Hubei Province of China in 1985. He received his BS degree in Mathematics from Ocean University of China in July 2006. Now he is the MS degree candidate in Computer Science and Engineering of Zhejiang University. His research interests include Semantic Web, and Intelligent Optimization Algorithms.

Jing Ying was born in Longyou, Zhejiang Province of P.R.China in 1971. He received the Bs, Ms and PhD degrees in Computer Science from Zhejiang University in 1990, 1992, 1995 respectively. Since Dec 2000, he has been the professor of Computer Science at Zhejiang University. His major interests include Software Engineering, Software Development Methodology, and Software Architecture.

Canghong Jin was born Shaoxing, Zhejiang Province of P.R.China in 1982. He received the MS degree in Computer Science from Zhejiang University in 2008. His major research interests include Service-oriented Software Engineering and Aspect-oriented Software Development.

Chunyan Yu was born in Zhuji, Zhejiang Province of China in 1976. She received her BS degree, MS degree, Ph. D degree in Computer Science and Technology from Zhejiang University in July 1997, March 2000 and June 2004 respectively. Since Aug 2007, she serves as an associate professor of Computer science at Fuzhou University. Her major interests include intelligent algorithm, virtual environment, and complexity adaptive system.