Genetic Algorithm based QoS-aware Service Composition in Multi-Cloud

Miao Zhang
School of Automation and Electrical Engineering
University of Science and Technology Beijing. Beijing, China

Li Liu*
School of Automation and Electrical Engineering
University of Science and Technology Beijing
Beijing, China
e-mail: liuli@ustb.edu.cn

Songtao Liu
Beijing AVIC Information Technology Company
Aviation Industry Corporation of China
Beijing, China

***Abstract*—Cloud computing as a widely used computing platform can provide a number of services for customers in a pay-as-you-go fashion. Enabling further growing and complex needs of users, service of different independent cloud provider should be composed to deliver uniform Quality of Service (QoS) as a single request. An open and valid question is how to select services as a partner chain and optimize the service compositions in order to satisfy both functional and non-functional requirements across multiple Cloud services. It is a NP-hard problem and faces trade-off among various QoS criteria. In this paper, a service composition model is presented considering the geo-distributed Multi-Cloud environment. Furthermore a Genetic Algorithm (GA) with improved crossover and mutation operator is proposed for QoS-aware service composition which allows users to select the optimized composition solution according to their preference. Experiment results show that this algorithm can improve the solution optimality and accelerate convergence speed.**

Keywords—Multi-Cloud, Services Composition, Genetic Algorithm

I. INTRODUCTION

Cloud computing is internet-based computing platform, where resources such as infrastructures, platforms, online applications can be shared as web services to cloud users, and recently many different web services are published and available in cloud data centers[1]. Since there are a number of cloud services, and a single Cloud service usually cannot satisfy complex needs of users. These factors emphasize the usage of multiple clouds (i.e. Multi-Cloud) in order to achieve better QoS. The term Multi-Cloud denotes the usage of multiple, independent clouds by a client or a service [2]. So how to select services in Multi-Cloud to compose an optimization service under various QoS constraints is a challenge problem.

Driven by the increasing customer requirement for flexibility, the number of service delivery model offered by cloud providers are expected to increase in the years to come [3]. There are more complex scenarios in cloud service, i.e., where multiple service providers, belonging to a single or multiple legal entities, collaborate and mutually interact in order to provision resources to an end-user. Moreover, bundling services that combine several services into a single offering are typical for cloud environments. For example, the data mining services packaged as the integrated cloud service，Google BigQuery or Cloud9 Analytics, and the database services packaged as Amazon DB and Quickbase [4]. The considered hybrid resource provisioning is a scenario where a consumer at the SaaS abstraction level consumes infrastructure resources from other infrastructure providers (IaaS). The number of candidate combinations will also go up exponentially with increasing service instances number. And different service instances with same functionally equivalent have different QoS levels in the cloud. How to execute services composition in Multi-Cloud environment and achieve high service utility customized for client requests without complicated processes has become an important research issue.

Service composition can be defined as the process which creates new services by combining and linking existing services via optimized orchestrator strategy [5]. In order to promptly achieve consumer requirements, this composition of services in Multi-Cloud should be carried out in a dynamic and automated manner. Furthermore, an efficient Cloud service composition method should select the highest level QoS and the cheapest services. It is challenging to efficiently find a composition solution in Multi-Cloud environment for the reason that it involves not only service composition but also the supply chain combination optimization under the constraint of many optimal objectives. This problem is NP-hard and Genetic Algorithm (GA) is naturally for solving these problems, however, GA also sometime finds some inferior solutions when the search space of problem is very huge, like in Cloud environment. We propose a GA with improved crossover and mutation based on elitism for QoS-aware service composition in Multi-Cloud environment which could prevent premature convergence efficiently.

Several approaches [5] also have been proposed to solve service composition problem, most of them only consider the composition in the application level, but we should consider composition in both application level and user's preference in cloud environment. And they are just useful for small-scale composition problem, when introduced to cloud environment, it may incurs performance degradation. There are many approaches to solve cloud service composition, especially the evolutionary algorithms like GA [6], PSO and ACO which are heuristic approaches to iteratively find near-optimal solution in large search spaces. In [7], the reliability-driven (RD) reputation was presented, which is time dependent, and can be used to evaluate the reliability of a resource. The authors proposed a look ahead genetic

*Corresponding Author: liuli@ ustb.edu.cn

978-1-5090-0089-0/15 $31.00 © 2015 IEEE
DOI 10.1109/CIC.2015.23

algorithm (LAGA) which utilizes the RD reputation and multi-objective model to optimize both the makespan and the reliability of a workflow application. The intelligent algorithms like PSO algorithm and ant colony algorithm are also adapted to solve service composition problem. Paper [8] presents a graph-based PSO technique which simultaneously determines the optimal workflow and the optimal Web services to be included in the composition based on their QoS properties. An ACO (Ant Colony Optimization algorithm) was proposed for QoS Based Web Services Composition problems in [9].

In this paper, (1) A service composition model is presented considering the geo-distributed Multi-Cloud environment; (2) A heuristic service composition algorithm using new mutation and crossover operator to improve general GA is proposed, which allow users to select the optimized composition solution according to their preference; (3) Simulation results show that our proposed GA can improve the solution optimality and convergence speed compared with existing GA.

II. PROBLEM FORMULATION

In the view of Cloud service composition mechanism, the system models include: Cloud provider (CP), applications(service requirements)(R), QoS criterion and constraints, types of composite service profile (CSF). Different CP partner needs to collaborate to form supply chain to satisfy the service requirements completely as it cannot provide all the services. We assume that each CP can provide more than one services and each service has one or more CP. Services in different commercial independent cloud platforms can be composed via mutual communication to satisfy complex needs of customer.

A. Definitions

In QoS-aware service composition, the cloud user submits a set of abstract services to be composed, which define an application requirement for executing a task [10]. As well, the corresponding QoS requirements for execution constraints of this application are submitted to Cloud service coordinator.

Define $R = \{R_1, \dots R_i, \dots R_n\}$ $(1 \le i \le n)$ is a set of n service requirements of customer. Each requirement R_i can be accomplish by a set of abstract services $\{S_i | 1 \le i \le l\}$with a certain sequence. In most cases, one abstract service can satisfy a requirement.

$Cons = \{cons^1, \dots, cons^\alpha, \dots, cons^N\}$, $cons^\alpha (1 \le \alpha \le N)$ is a constraint value over QoS criterion Q^α required by cloud customer, and Nis the number of QoS constraints. Let $cons_i = \{cons_i^1, \dots, cons_i^\alpha, \dots, cons_i^N\} (1 \le \alpha \le N)$ represent a set of constraints for requirement R_i, where $cons_i^\alpha$ is the constraint value over Q^α of requirement R_i. Cons which is defined by end users specifies QoS expectation about the composite solutions.

A Multi-Cloud is a set of clouds, such as $P = \{P_1, \dots P_k, \dots P_m\}$, where each P_k *(1≤k≤m)* is an independent CP. m is the number of CP. CP publishes a set of web services,$P_k = \{S_1^k, \dots S_i^k, \dots S_l^k\}$, where S_i^k $(1 \le i \le l)$ is a service functionality class (abstract service) from CP k. A service class include different services instance which can satisfy user's request. CP often specifies their published services in a standard OWL-S specification service file, which denote a service functionality classes. l is the number of service classes in Pk. Each service classes S_i^k is a collection of different atomic service (concrete service instance) $s_{i,j}^k$with different QoS level while have the same functionality. Each service instance $s_{i,j}^k$ is associated with a QoS criterion Q_{ij}. $Q = \{Q^1, \dots, Q^\alpha, \dots, Q^N\}$ $(Q^\alpha, 1 \le \alpha \le N)$ is a QoS criterion of service and N is the number of QoS criteria that a service holds. Let $Q_{ij}^k = \{Q_{ij}^{k,1}, \dots, Q_{ij}^{k,\alpha}, \dots, Q_{ij}^{k,N}\} (1 \le \alpha \le N)$ denote the vector of the QoS criterion for service instance $s_{i,j}^k$ offered by CP_k.

The process of service composition in Multi-Cloud is to find a set of service instances s_{ij} and bound them to abstract services to complete a application. The solve of service composition can be denoted as $CS = \{CS_1, \dots, CS_j, \dots, CS_n\}$, CS_j is a service composition profile, which is composed of the available services needed by R_i.

B. Aggregated QoS Model of Service Composed

Every task in a cloud should be executed by a special service class [12]. The service set uses the basic types of composite service patterns. Atomic services (service instances) are connected by different structures in a composite service. Generally, there are four service composition structures: Sequential, AND (parallel), XOR (conditional), Loop.

We can simplify cloud service composition with the basic structures into an abstract service, and use the above four structures to calculate the aggregation QoS value of the composite service. The commonly used SLAs defined with QoS criteria include response time, reliability, cost, etc. The QoS of composition service should be examined by aggregate measuring the QoS criteria of individual atomic service to decide whether it satisfies desired SLA. Obviously, there may be multiple atomic services deployed on different CPs providing the same function. So the network latency between services should be considered in computing the aggregate response time. The latency between services noted by $L(i)$ is measurable and predictable which is not the focus of this paper. The aggregated response time of i^{th} sequential branch is represented by T_i . Similar to the work [6], the aggregation QoS of composition service (CS) can be

TABLE I. QOS AGGREGATE FUNCTIONS

QoS Criteria	Sequential	Conditional	Parallel	Loop
Response time	$T = \sum_{i=1}^{N} T_i + \sum_{i=1}^{N} L(i)$	$T = \sum_{i=1}^{N} p_i * T_i$	$T = max_{i=1}^{N} T_i$	$T = k * T_i$
Reliability	$R = \prod_{i=1}^{N} R_i$	$R = \sum_{i=1}^{N} p_i * R_i$	$R = min_{i=1}^{N} R_i$	$R = (R_i)^k$
Cost	$C = \sum_{i=1}^{N} C_i$	$C = \sum_{i=1}^{N} p_i * C_i$	$C = \sum_{i=1}^{N} C_i$	$C = k * C_i$

calculated as Table1.

III. Optimization of Cloud Service Composition

In order to satisfy differentiated SLAs (defined for different user), application broker is required to find the optimal combinations of service instances to satisfy users' QoS metrics. Service composition optimization may trade off among conflicting QoS objectives, such as the response time, reliability and cost. For example, to improve the response time of service by deploying expensive service instance, however, this is against another objective to reduce cost [11].

A. Genetic Algorithm

Genetic Algorithm (GA) is a heuristic approach which iteratively finds near-optimal solutions in large search spaces. It introduces the biological evolution principle "survival of the fittest" into encoding of chromosome, and uses selection, crossover and mutation operations to filtrate individuals, thus make sure that the individuals which have good fitness value are retained and poor one are eliminated [12]. The new generation of individuals inherits the information of the previous generation, and is better than it. These operations repeat circularly until the user's QoS is satisfied or reaches the maximum number of iterations[6].

In the selection step, it generates a predefined number of chromosomes to form the initial generation. The chromosomes are ordered by their fitness value. Then a rank-based roulette wheel selection scheme is used to implement the selection operation.

In the crossover step, it randomly chooses some pairs of the chromosomes to form the offspring. For each pair, it randomly generates a cut-off point, and divides the parents into top and bottom parts. The bottom parts of the parent chromosomes are reordered. This ensures that the newly generated offspring are valid. Fig.1 shows a crossover operation. The mutation operation refers to select an arbitrary individual and mutate an arbitrary point in the chromosome to generate a better individual. Fig. 2 shows a mutation operation.

In this paper, genomes (chromosome) represent the possible solution for service composition. Three genes in a chromosome encode a service instance. The first gene represents the independent CP that provides the service, and the following two genes represent the service instance from a service class. For example in Fig.3, the shown chromosome is a composited service that consists of 8 service classes to implement 8 tasks. Each three genes represent the chosen service instance from each service class. Such as service $s^1_{1,13}$ is the thirteenth service instance picked from service class S_1 and offered by CP P_1. In this example, the composited services are $\{s^1_{1,13}, s^7_{2,4}, s^8_{3,6}, s^3_{4,13}, s^4_{5,12}, s^7_{6,11}, s^6_{7,11}, s^5_{8,19}\}$.

Algorithm 1 shows the evolution process in GA, pop_g denotes the population of the generation g, μ is the number of individuals in a population. γ is the length of individual. p_c is the crossover probability, and p_m is the mutation probability.

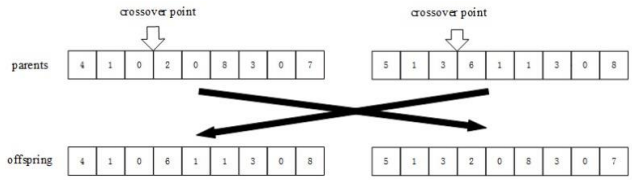


Fig. 1. The crossover operator of GA

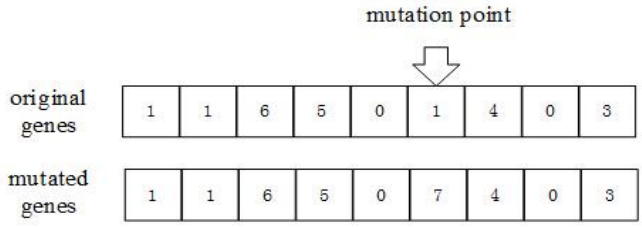


Fig. 2. The mutation operator of GA

Algorithm 1 Evolution Process in GA

Input: g_{max}, μ, γ, p_c, p_m

Output: *best population* p_b

1. *Set the maximum generation* g_{max}

2. *Initial population* pop_g *with* μ *individuals, and the length of each individual is* γ.

3. **For** g=1:g_{max}

3.1. *calculate the fitness of individuals in* pop_g *: f* (pop_g)

3.2. pop_s=*selection*(pop_g, *f* (pop_g))

3.3. pop_c=*crossover*(pop_s, p_c)

3.4. pop_m=*mutation*(pop_c, p_m)

3.5. pop_{g+1}=pop_m

End

B. Fitness function

After crossover and mutation operators, GA will evaluate the chromosomes using fitness function. Most end-users not only prefer service with high reliability, but also short cost and response time. The fitness function must to satisfy all these preferences.

QoS criteria should be normalized for the different measurement units of them. $Q^k_{ij} = \{Q^{k,1}_{ij}, \dots, Q^{k,\alpha}_{ij}, \dots, Q^{k,N}_{ij}\}$ is the vector of the QoS criteria of service instance s^k_{ij} from CP P_k. $Q^\alpha_{i,max}$ is the maximum of QoS criteria α for all service instance in class i, and $Q^\alpha_{i,min}$ is the minimum. The normalized QoS criteria of Q'^k_{ij} can be calculated as equation 1.

$$Q'^k_{ij} = \frac{Q^k_{ij} - Q^\alpha_{i,-}}{Q^\alpha_{i,+} - Q^\alpha_{i,-}} \quad (1)$$

For the positive QoS criteria to be maximized, the ideal value $Q^\alpha_{i,+} = Q^\alpha_{i,max}$, $Q^\alpha_{i,-} = Q^\alpha_{i,min}$, such as reliability, while for the negative criteria to be minimized, the ideal value $Q^\alpha_{i,+} = Q^\alpha_{i,min}$, $Q^\alpha_{i,-} = Q^\alpha_{i,max}$ such as time and cost.

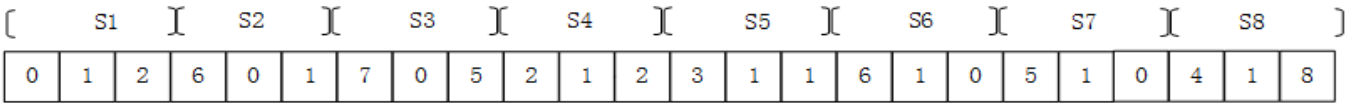


Fig. 3. An example of Chromosome in Multi-Cloud

And QoS values of composited service CS_m is calculated based on Table 1, $\mathrm{Q(CS)_m} = \{\mathrm{Q(CS)_m^1}, \dots, \mathrm{Q(CS)_m^\alpha}, \dots, \mathrm{Q(CS)_m^N}\}$. The weights ω_α determined by users are assigned to QoS criteria to represent users' preference in fitness function. The fitness of an individual is calculated as equation 2.

$$f(CS_m) = \sum_{\alpha=1}^{N} \omega_\alpha * Q'(CS)_m^\alpha - \sigma * E(m) \quad (2)$$

where σ is the penalty coefficient, and E(m) is an error term which is 0 for the feasible solutions, and the sum of all the amounts by which the constraints are violated for infeasible solutions, and it has been normalized, too.

C. Improved crossover and mutation process

Using Genetic Algorithm to solve optimization problem of service composition differs from continuous optimization problems, due to the change of fitness in chromosome is not continuous with gene modifications. So one gene change in an excellent chromosome may lead to a substantial decline of fitness value, and it is likely this outstanding solution can't be found again within the specified iterations. And a simple crossover or mutation may deteriorate the whole population, and the situation would be worse when the number of instance is tremendous, like in Cloud environment, and lead to premature convergence. So how to lead the population evolve in a correct direction is a key issue. We improve the crossover operator (See Algorithm 2) turning it to increase the fitness value to prohibit the poor crossover and mutation.`

ALGORITHM 2 IMPROVED CROSSOVER PROCESS

Input: population pop_s, p_c

Output: population pop_c

1. *oldpop*=pop_s

2. *set p which is a random decimal within the range [0,1].*
set s which is a random crossover point in parents.

3. For i=1: μ

If $p < p_c$

$pop_s(i,:) = [pop_s(i,1:s), pop_s(i+1, s+1:\gamma)]$;

$pop_s(i+1,:) = [pop_s(i+1,1:s),\ pop_s(i,s+1:\gamma)]$;

End

If $f(newpop(i,:)) > f(oldpop(i,:))$

$pop_c(i,:)$=$pop_s(i,:)$;

Else

$pop_c(i,:)$= $oldpop(i,:)$;

End

End

The chromosome is randomly changed in mutation operator of general GA, which does not care about the individual's fitness. So in some cases, a mutation operation is likely to make an outstanding individual becomes infeasible solution. An improved mutation operator is presented as Algorithm 3. The mutation process is recognized effective that only when the fitness value of the chromosome after the mutation operation is higher than the before.

ALGORITHM 3 IMPROVED MUTATION PROCESS

Input: population pop_c, p_m

Output: population pop_m

1. *oldpop*=pop_c

2. *set p which is a random decimal within the range [0,1].*
set o which is a random mutation point in parents.
k is a random number.

3. For i=1: μ

If p<p_m

$pop_c(i,o) = k$;

End

If $f(pop_c(i,:)) > f(oldpop(i,:))$

$pop_m(i,:)$=$pop_c(i,:)$;

Else

$pop_m(i,:)$= $oldpop(i,:)$;

End

End

IV. EVALUATION AND RESULTS

Compared with general GA, the performance of our proposed GA is evaluated in Multi-Cloud environment. The data of web service is from QWS Dataset (2.0) developed by Eyhab Al-Masri and Dr. Qusay H. Mahmoud. We consider three QoS criteria, $Q = \{response\ time(ms), reliability(\%), cost(\$)\}$. All the experiments are performed on computers with Inter Core i5-4570S CPU(2.9GHz and 8G RAM).

A. Experiment setup

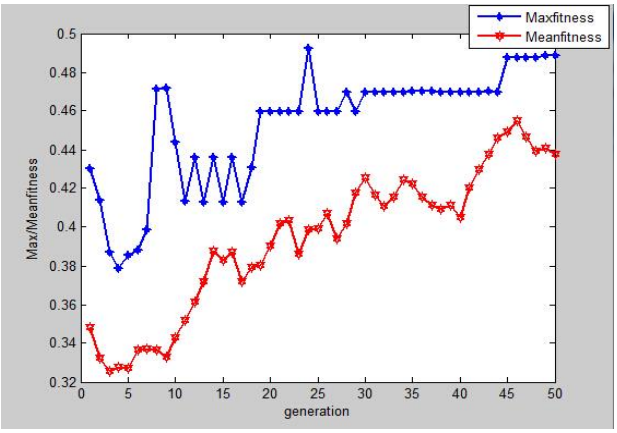
There are 8 service classes, and each of them includes 80 service instances provided by 8 geo-distributed CPs. The $\mathrm{Cons} = \{\mathrm{cons}^1, \mathrm{cons}^2, \mathrm{cons}^3\}$, $\mathrm{cons}^1, \mathrm{cons}^2, \mathrm{cons}^3$ are constraints of response time, the reliability and the cost in order. In reality, QoS constraints defined in SLA are generated through the negotiation between users and service providers. Generally, most QoS constraints were computed as follows:

$$cons^1 = \bar{T} * m + \bar{L} * m \quad (3)$$

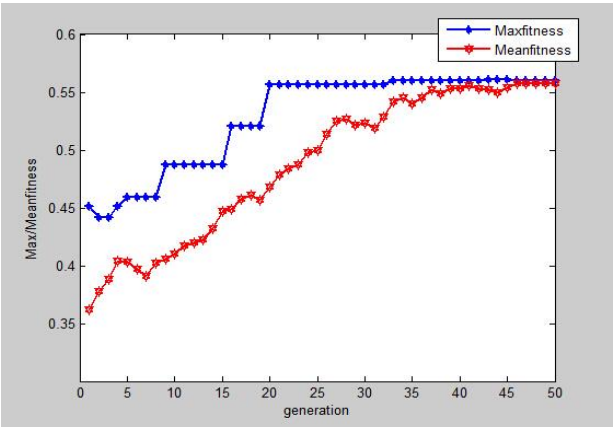
$$cons^2 = \bar{R}\text{^}m \quad (4)$$

$$cons^3 = \bar{C} * m \quad (5)$$

where m is the number of service classes, $\bar{T}$, $\bar{R}$ and $\bar{C}$ is the average response time, average reliability and average cost for all service instance respectively. $\bar{L}$ is the average latency between each two CPs. For two continuous services from the same CP, the latency is zero, and for two



(a)General GA



(b) Improved GA

Fig. 4. Results of general GA and Improved GA

from different CPs, there will be network delay. The latencies (ms) are generated depending on their distance and within the range [100,200]. The probability of crossover p_c is 0.8 and the probability of mutation p_m is 0.2. A rank-based roulette wheel strategy is used for selection.

We configure the evolutionary algorithms to run 50 generation with 50 individuals (or particles), the time complexity of GA is $O(m * (g_{max} + 1))$, where m is the individuals number in a group, and g_{max} is the maximum iteration number, it examines 2550 individuals at most. But the complexity of this problem is $O(\prod_{d=1}^{n} k_d)$, which O is the time complexity of exhaustive approach for this problem, n is the number of workflow tasks and k_d is the number candidate service instances for task d. We can find that, with the increase of the number of task and service instance, the time complexity of this service combination optimization problem is much more than the evolutionary algorithm.

B. Simulation Results and Analysis

In the first experiment, three QoS criteria gained equal preference from users. The constraints are calculated by equation (3), (4), (5). The goal of service composing is both to maximize the reliability and to minimize the response time and cost under the constraints condition. The number of generation is 100, and the size of population is 50. Fig. 4(a) and Fig.4(b) shows the Maxfitness (the maximum fitness value for each generation) and Meanfitness (the average fitness value for each generation) of general GA and our proposed GA separately. As shown in Fig.4(a), the Meanfitness gradually improves in each generation, and approaches the Maxfitness in certain generations. But we can find that, in some generations, there are high recessions with the Maxfitness and unsatisfied convergence property of Meanfitness. From Fig 4(b), we can see, in each generation, there is no obvious recession with the Maxfitness, although there are small recessions in some generation which may be caused by the selection process, and after a certain number of generations, the Meanfitness converges to the Maxfitness. This demonstrates that the proposed GA can effectively prevent poor crossover and mutation.

The two algorithms are simulated for 30 times respectively, and the QoS criteria of the best individual in each times are gotten. 95% CI (Confidence Interval) for each QoS criterion of the two algorithms are shown in Table 2. The QoS criteria of composited service generated by our improved GA are better than the general GA.

For different users, they have different QoS preferences. Some users may require a relatively high response time property, and others may be very sensitive to cost. Weights can reflect the users' preference. In the second experiment, we set different user's QoS preferences for three cases. In case 1, user prefers to have higher QoS property of response time. In the case 2, user is sensitive to reliability, and in the case 3,the preferred objective is cost. We have simulated the improved GA with the above three cases for 30 times respectively, the average response time, reliability and cost have been shown in Table 3. Under different user's QoS preferences, our proposed GA always can find suitable service composition solutions.

TABLE II. RESULTS OF OUR GA IN DIFFERENT PREFERENCES

	Response time(ms)	Reliability(%)	Cost($)
Case 1	1027.7	25.57	850.4
Case 2	1181.3	31.55	818.2
Case 3	1307.6	22.77	782.8

TABLE III. COMPARIING OF GENERAL GA AND IMPROVED GA

	Response time(ms)		Reliability (%)		Cost($)		Fitness value	
	Average	95%CI	Average	95%CI		Average	95%CI	Average
General GA	1160.8	(1087.6,1234.0)	24.66	(21.38, 27.94)	General GA	1160.8	(1087.6,1234.0)	24.66
Improved GA	1060.8	(1005.9,1115.8)	29.06	(25.62, 32.51)	Improved GA	1060.8	(1005.9,1115.8)	29.06

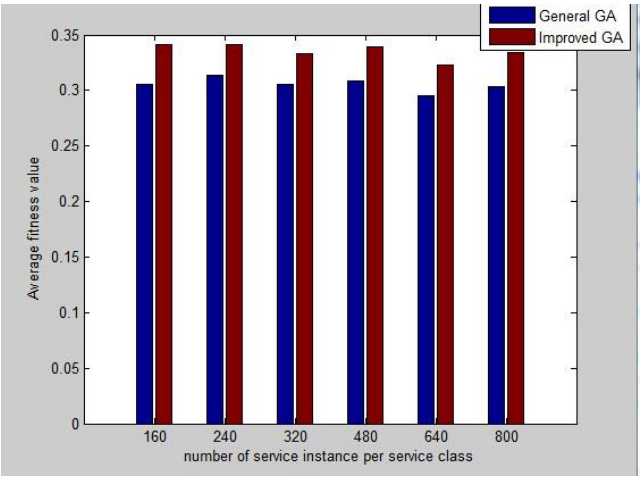


Fig. 5. Average Maxfitness of 30 times with different number of service instances per service class

In order to evaluate the scalability of our improved GA, we have compared the fitness value versus increasing sizes of service class. In this experiment, the QoS values of service instances are generated randomly, the random values of response time (*ms*) have a Gaussian distribution and with a mean value of 1000 and standard deviation of 20. The reliability (%) and cost($) are also Gaussian distribution and have mean value with 95 and 10 respectively, the standard deviation both with 2. Fig.5 shows the fitness value of two algorithms with different number of service instances. The improved GA also performs better than general GA in fitness value with the increasing number of service instance, i.e. our proposed algorithm is more scalable.

V. Conclusion and Future Work

We present a service composition model considering the geo-distributed Multi-Cloud environment. To address the problem of QoS-aware service composition, we have also proposed a Genetic Algorithm (GA) with improved crossover and mutation operator, which allows users to select the optimized composition solution according to their preference. Experiment results indicate that our algorithm with improved crossover and mutation performs better in optimality and accelerates convergence speed compared with the normal GA, moreover it has higher performance of scalability. So in the large scale optimizing problem, it is very essential to find a proper evolutionary direction for evolutionary algorithms.

Deep explores are planned in the future work. We aim at using different evolutionary multi-objective algorithms like NSGA-II and MOPSO to discover Pareto-solutions addressing the problem of service composition. Our current central work is to adapt different parameters in different stages for evolutionary algorithms to make them evolve in correct directions to accelerate convergence speed and find more excellent solutions.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No. 61370132 No.61472033, No.61272432) and the State High-Tech Development Plan (No.2013AA01A601).

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computing System vol.25, 2009, pp.599–616, doi: 10.1016/j.future.2008.12.001.

[2] N. Grozev and R. Buyya, "Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey", Software: Practice and Experience, ISSN: 0038-0644, Wiley Press, New York, USA, 2013, doi: 10.1002/spe.2168.

[3] L. Liu, M. Zhang, Y. Lin , L. Qin, "A Survey on Workflow Management and Scheduling in Cloud Computing", Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium, 2014, pp.837 – 846, doi: 10.1109/ccgrid.2014.83.

[4] A. Milenkoski, A. Iosup, S. Kounev, K. Sachs, P. Rygielski, J. Ding, et al., "Cloud Usage patterns: A formalism for description of cloud usage scenarios," Technical Report: SPEC_RG_2013-001. SPEG RG Cloud Working Group.

[5] L. Zeng, B. Benatallah, A. Dumas ,P. Leitner, S. Dustdar, "QoSaware middleware for web services composition," IEEE Transactions on Software Engineering, vol. 30, no.5, pp.311–327,2004, doi: 10.1109/tse.2004.11.

[6] Z. Ye, X. Zhou and A. Bouguettaya, "Genetic Algorithm Based QoS-Aware Service Compositions in Cloud Computing," 16th International Conference, 2011, pp. 321-334, doi: 10.1007/978-3-642-20152-3_24.

[7] X. Wang, C. S. Yeo, R. Buyya and J. Su, "Optimizing the Makespan and Reliability for Workflow Applications with Reputation and a Lookahead Genetic Algorithm," Journal Future Generation Computer Systems, vol. 27, no. 8, 2011, pp. 1124-1134, doi: 10.1109/CIS.2010.46, doi: 10.1016/j.future.2011.03.008.

[8] A.S. da Silva, H. Ma, M. Zhang, "A Graph-Based Particle Swarm Optimisation Approach to QoS-Aware Web Service Composition and Selection," Evolutionary Computation (CEC), 2014 IEEE Congress on, 2014, pp. 3127 – 3134, doi: 10.1109/CEC.2014.6900404.

[9] R. D. Sunil, M. U. Kharat, "QoS Based Web Services Composition using Ant Colony Optimization: Mobile Agent Approach," International Journal of Advanced Research in Computer and Communication Engineering, vol. 1, no. 7, September 2012, doi: 10.1016/j.proeng.2012.01.114.

[10] H. Ma, F. Bastani, I. L. Yen, H. Mei, "QoS-Driven Service Composition with Reconfigurable Service," IEEE Transactions on Services Computing, vol.6, no.1 ,pp. 20-34,2013, doi: 10.1109/tsc.2011.21.

[11] F. Rosenberg, P. Celikovic, A. Michlmayr, P. Leitner, and S. Dustdar, "An end-toend approach for QoS-aware service composition," In: Proceedings of 13th IEEE International EDOC Conference, 2009, pp. 1–4, doi: 10.1109/edoc.2009.14.

[12] M. Srinivas, L.Patnaik, "Genetic algorithms: A survey," Computer, 1994, vol. 27, pp. 17–26 , doi: 10.1109/2.294849