Computers and Electrical Engineering xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

# A genetic-based approach to web service composition in geo-distributed cloud environment ☆

Dandan Wang, Yang Yang, Zhenqiang Mi *

*School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China*

ARTICLE INFO

*Article history:*
Received 14 May 2014
Received in revised form 13 October 2014
Accepted 15 October 2014
Available online xxxx

*Keywords:*
Web service
Service composition
Cloud
Geo-distributed datacenters

ABSTRACT

Independent fine-grain web services can be integrated to a value-added coarse-grain service through service composition technologies in Service Oriented Architecture. With the advent of cloud computing, more and more web services in cloud may provide the same function but differ in performance. In addition, the development of cloud computing presents a geographically distributed manner, which elevates the impact of the network on the QoS of composited web services. Therefore, a significant research problem in service composition is how to select the best candidate service from a set of functionally equivalent services in terms of a service level agreement (SLA). In this paper, we propose a composition model that takes both QoS of services and cloud network environment into consideration. We also propose a web service composition approach based on genetic algorithm for geo-distributed cloud and service providers who want to minimize the SLA violations.



## 1. Introduction

Cloud computing as a revolutionary technology is changing the entire IT ecosystem, and all aspects of our lives. It brings not only the technical change, but also profound influence on enterprise business applications and business models. Applications are delivered as services over the Internet in cloud environment [1]. Today, users are increasingly accustomed to using the Internet to gain software resources in the form of web services. Web services are self-describing software applications that provide certain functions independently from underlying implementation technologies [2,3]. Through service composition technologies, loosely-coupled services that are independent from each other can be integrated into complex and value-added composited services as long as each component service's interface specification is subject to standard protocols.

The system architecture for service composition in cloud environment is shown in Fig. 1. The cloud architecture includes three layers: software layer, platform layer and infrastructure layer. A user sends composition requests to brokers for utilizing composited web services. The software layer includes brokers and web services. The brokers, which can be centralized or distributed, manage all services that are offered to users by SaaS providers. Web services are registered to brokers by service providers in order to be discovered. The composition engine in platform layer communicates with the brokers to discover candidate services according to the user's request. Based on the candidate services discovered, composition engine generates an execution plan which satisfies user's QoS requirements. The infrastructure layer controls the actual resource allocation in terms of the execution plan generated in platform layer.

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Associate Editor Dr. Ziya Arnavut.

* Corresponding author. Tel.: +86 82373756.
*E-mail addresses:* wdd_ustb@163.com (D. Wang), yyang@ustb.edu.cn (Y. Yang), mizq@ustb.edu.cn (Z. Mi).

http://dx.doi.org/10.1016/j.compeleceng.2014.10.008
0045-7906/

Please cite this article in press as: Wang D et al. A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electr Eng (2014), http://dx.doi.org/10.1016/j.compeleceng.2014.10.008

QoS of web services refers to various nonfunctional characteristics such as response time, throughput, availability, and reliability [4]. Given an abstract representation of a composition request, a number of candidate services that provide the same function but differ in QoS can be obtained. Composition engine need to select the best candidate service from a set of functionally equivalent services according to the QoS. The work in [5,6] uses combinational model to find the optimal selection of component services. The authors use linear programming technique which is best suited for small scale problems. But with the increasing scale of problems, the complexity of this method increases exponentially. The work of Mahammad Alrifai et al. addresses the problem by combining global optimization with local selection methods [7,8]. By decomposing the optimization problem into small sub-problems, their approach is able to solve the problem in a distributed manner. The work in [9] extends the methods above. The authors present a strategy to further reduce the search space by examining only the subsets of candidate services since the number of candidate services for a composition may be too large. The above papers do not solve the composition problem in the cloud context. Thus, the distributed network environment is not considered in these papers.

Cloud computing is an increasingly popular computing paradigm. It has become a reliable foundation for a wide array of enterprise and end-user applications [10]. Tao et al. investigate the composition problem of various cloud resource including software and hardware service with multiple objectives and constraints [11]. With the increasing number of cloud service users worldwide, major cloud service providers have been deploying and operating geographically dispersed datacenters to serve the globally distributed cloud users. At the same time, cloud services continue to grow rapidly. The resource capacity of a datacenter is limited, so distributing the load to global datacenters will be effective in providing stable services [12,13]. More and more web services are deployed on geo-distributed cloud datacenters and are offered all over the world. Cloud datacenters depend on networks to connect with each other and cloud users. Network environment has influence on the performance of composited services cross datacenters. QoS of network is a noticeable parameter of service composition. To avoid SLA violations, the network performance has been attracting more and more attention during service composition. In [14], the authors investigate the composition of QoS-aware network communication path across large scale multi-domain networks. They reduce the problem to k-MCOP (k multi-constrained optimal path) problem via domain graph expansion technique and developed a fast search heuristic. Klein et al. [15] propose a generic model towards network-aware service composition in the cloud. The authors estimate the network latency between arbitrary network locations of services or users and propose a network-aware selection algorithm to find services that will result in low latency. However, their work only focuses on response time. Other QoS criteria are not considered in this paper.

In cloud environment, it is a challenge to search for an optimal and feasible composition path efficiently because the problem of service composition is an NP-complete problem. Cloud applications usually involve a large number of components and there are many candidate services for each component. With the increase of the number of components, the number of composition paths increases exponentially. So it is impractical to traverse all the composition paths in search space when the flow of composition is large. Furthermore, an important issue in cloud computing is the need for providers to guarantee the service level agreements (SLAs) established with users [13]. Cloud providers derive their profits from the margin between the operational cost of infrastructure and the revenue generated from users. Therefore, cloud providers are interested in maximizing profit and ensuring QoS for users to enhance their reputation in the marketplace. They are looking into solutions that can minimize the SLA violation. In this paper, we propose a new approach towards web service composition in geo-distributed cloud environment. Our main contributions can be summarized as follows:

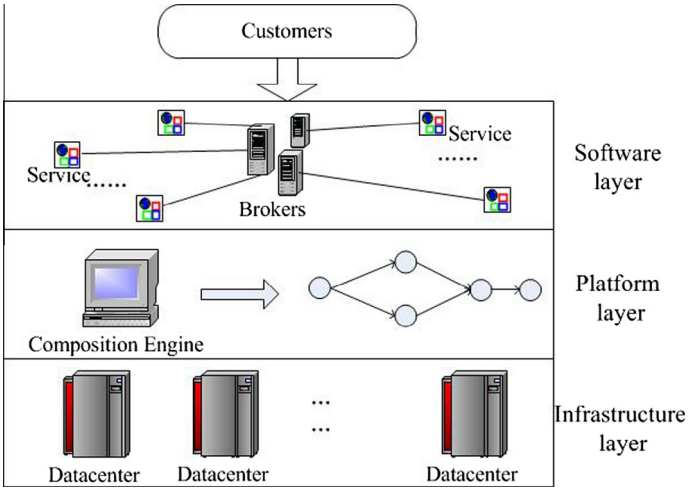


**Fig. 1.** Composition architecture in cloud environment.

Please cite this article in press as: Wang D et al. A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electr Eng (2014), http://dx.doi.org/10.1016/j.compeleceng.2014.10.008

1. We first specify a realistic QoS-based composition model that allows us to consider the distributed network environment. Compared with most existing work, our model considers not only the QoS of services but also the QoS of network. In addition, our model can adapt to scenarios of multiple QoS criteria. It is more scalable than some related works [8,16] which only focus on certain QoS criterion. We also present an approach to calculate the QoS of composited service in cloud computing.
2. Some related methods use graph theory to solve the network-aware composition problem [17], which leads to the exponential growth of computation time. In order to reduce the complexity, a heuristic composition algorithm based on genetic algorithm to maximize user experience and minimize SLA violation is proposed to solve the problem in this work.
3. Unlike existing genetic algorithms, we use the notion of skyline to generate the initial population, which improve the solution quality and convergence speed. Simulation results show that our algorithm performs well in terms of solution feasibility, optimality and scalability with respect to different parameters.

The rest of this paper is organized as follows: In Section 2, we introduce the problem and describe our model for composition. Our genetic-based algorithm for service composition is presented in Section 3. Section 4 shows the simulation results and performance analysis. Finally, Section 5 gives conclusions and an outlook on possible continuations of our work.

## 2. Composition model

The following definitions are used in this article.

**Definition 1** (*Atomic service*). Atomic service is an independent unit to solve a particular task in a service computing system. Atomic services are published to brokers by service providers in order to be discovered.

**Definition 2** (*Service set*). A service set is a collection of atomic services with the same function but different QoS levels.

Fig. 2 gives a conceptual overview of web service composition. Given a composition request, a composition process that defines a workflow of components is designed.

1. *Service Discovery:* A service set for each task in the composition process is discovered according to the functional description of atomic services.
2. *Service Selection:* Given the non-functional description of services and the QoS of network, the service selection strategy selects atomic services from service sets.

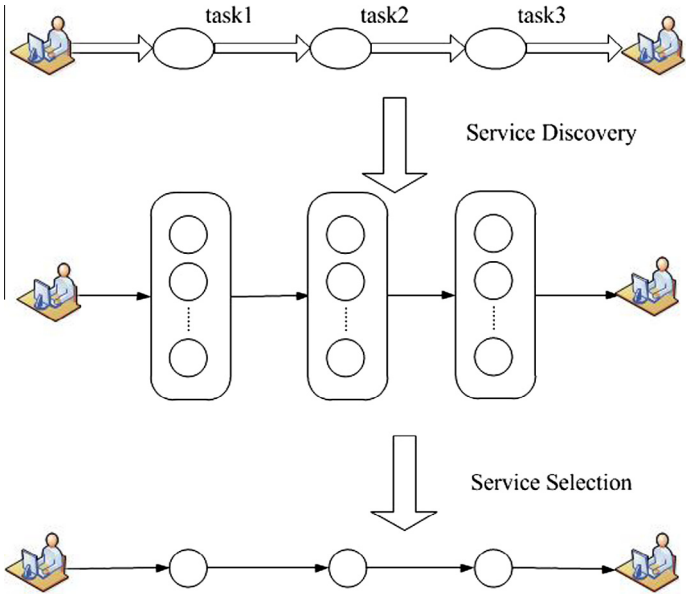


**Fig. 2.** Conceptual overview of web service composition.

Please cite this article in press as: Wang D et al. A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electr Eng (2014), http://dx.doi.org/10.1016/j.compeleceng.2014.10.008

### 2.1. Service level agreement

In SOA, a service level agreement (SLA) is a legal contract between service provider and user. It is defined upon a workflow instance as its end-to-end QoS requirements such as throughput, latency and cost (e.g., resource utilization fees) [2]. In this paper, an SLA is currently defined with end-to-end response time, price, availability and reputation of composited services.

### 2.2. QoS-based web services

QoS describes the non-functional properties. QoS of atomic services can be provided by providers, computed based on execution and monitored by the users, or collected via users' feedback in terms of the characteristic of each QoS criterion [18]. In this paper, we focus on four typical QoS criteria shown in Table 1.

### 2.3. Location-aware web services

There may be multiple atomic services that are deployed on different datacenters providing the same function. In addition, one application may be offered in more than one datacenters, we consider them as multiple atomic services providing the same function in this paper. The degree of distribution of the atomic services has influence on the QoS of a composited service. For example, compared with two atomic services deployed on the same datacenter, if two atomic services are deployed on datacenters located in Asian and Europe, the network delay between them is a noticeable parameter when they communicate with each other.

The performance of the network is vital to the performance of distributed composited services. We can distinguish the network delay into two kinds: network delay between services and network delay between service and user.

Network delay between services, which is noted by variable $dt_1$ in this paper, is mainly determined by the geographical location of the datacenters that services are deployed on. The delay between datacenters is measurable and predictable because that the number of datacenters for certain cloud provider is limited and stable. Cloud provider can storage the network delay between datacenters in cache for facilitating usage. The network delay between service and user, which is noted by variable $dt_2$ in this paper, is mainly determined by the network environment between them. It can be obtained from the feedback of network and the information of execution monitoring. Many studies deal with the measure of point-to-point QoS of network, such as [19,20], which is not the focus of this paper.

### 2.4. Composited service

SLA is often used as contractual basis between customers and providers on the expected QoS level. In order to judge whether a composited service satisfies a given SLA or not, it is required to examine its end-to-end QoS by aggregating QoS of atomic services and QoS of network. The QoS of a composited service is relevant to the structure of composition path. Fig. 3 shows three composition structures: sequential, parallel and conditional. QoS computation of sequential structure provides a basis for QoS computation of other structures. Similar to most works [14,18], aggregation functions for QoS computation of composited services are illustrated in Table 2. Where $T^i$ represents the aggregated response time of the i-th sequential branch.

### 2.5. Problem statement

The goal of web service composition in geo-distributed cloud environment is to find one composition path that the overall performance is optimal and the user's QoS requirements are satisfied. Therefore, for a given composition request, the problem is how to select an appropriate service for each task so that:

1. the user experience can be optimized;
2. the QoS requirements described in SLA can be satisfied.

**Table 1**
Typical QoS criteria of atomic services.

| QoS criterion | Unit | Description |
|---|---|---|
| Response time (*st*) | ms | The execution duration between the moment when a request is arrived and the moment when the result is obtained |
| Availability (*sa*) | percent | The probability that a service is accessible |
| Price (*sp*) | dollar | The money that the requester has to pay to the service provider for the use of service |
| Reputation (*sr*) | percent | A measure of services' trustworthiness |

Please cite this article in press as: Wang D et al. A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electr Eng (2014), http://dx.doi.org/10.1016/j.compeleceng.2014.10.008

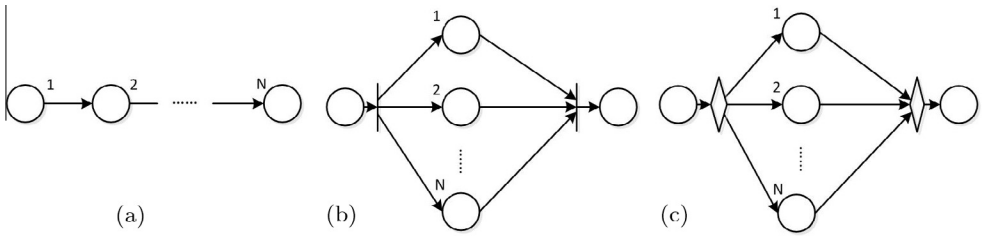


**Fig. 3.** Composition structure. (a) Sequential; (b) parallel; (c) conditional.

**Table 2**
Aggregation functions for QoS computation.

| | Response time | Price | Availability | Reputation |
|---|---|---|---|---|
| Sequential | $t = \sum_{i=1}^{N} st^i + \sum_{i=1}^{N-1} dt_1^i + \sum_{i=1}^{2} dt_2^i$ | $p = \sum_{i=1}^{N} sp^i$ | $a = \prod_{i=1}^{N} sa^i$ | $r = \mathrm{Avg}_{i=1,2,\ldots,N} sr^i$ |
| Parallel | $t = \max_{i=1,2,\ldots,N} T^i$ | $p = \sum_{i=1}^{N} sp^i$ | $a = \prod_{i=1}^{N} sa^i$ | $r = \mathrm{Avg}_{i=1,2,\ldots,N} sr^i$ |
| Conditional | $t = \mathrm{Avg}_{i=1,2,\ldots,N} T^i$ | $p = \mathrm{Avg}_{i=1,2,\ldots,N} sp^i$ | $a = \mathrm{Avg}_{i=1,2,\ldots,N} sa^i$ | $r = \mathrm{Avg}_{i=1,2,\ldots,N} sr^i$ |

## 3. Service composition algorithm

### 3.1. The design of algorithm

Our approach is based on Genetic Algorithm. There are many well-known heuristic search methods, such as Tabu Search, Simulated Annealing and Genetic Algorithm. They have been applied to solve many different problems, and proven to be more effective compared with their tailored counterparts. Each of these methods has their unique properties thus is suitable in certain environment. In our proposed work, we choose Genetic Algorithm simply because Genetic Algorithm is more suitable for our composition model. Firstly, Genetic Algorithm is population-based, whereas Tabu Search and Simulated Annealing are individual-based [21,22]. Population-based methods are more suitable for our composition model compared with the individual-based methods. Considering the QoS requirements described in SLA, we make some adjustments on the generation of initial population, which increases the proportion of feasible and excellent gene in the population. The proposed method can increase the solution's feasible rate to a great extent. Secondly, the optimization of the parameters for Genetic Algorithm is simpler than other algorithms under our proposed model. Given the complexity and variety of cloud environment, the search method for service composition need to be easy to configure. The optimization of the parameters for other methods is sometime more complicated. For example, the performance of Simulated Annealing highly depends on the initial temperature, the cooling rate, the transition probability, etc. [21].

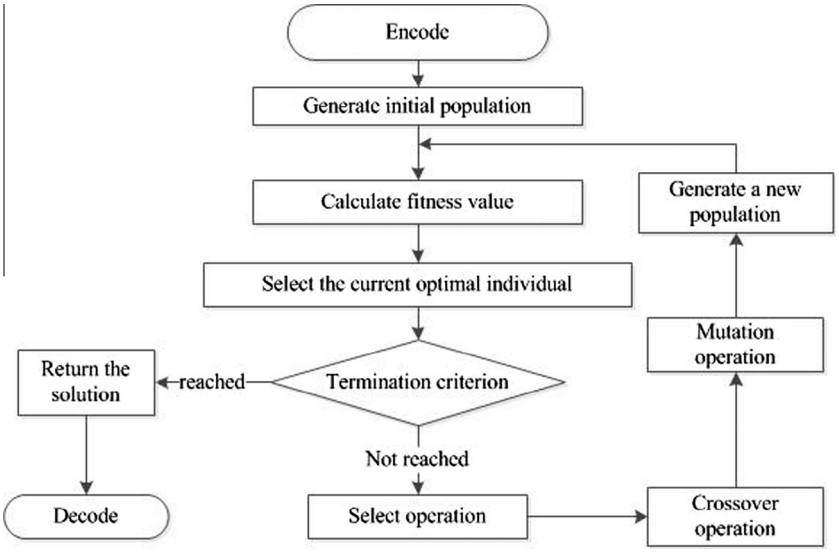


**Fig. 4.** Flowchart of the algorithm.

Please cite this article in press as: Wang D et al. A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electr Eng (2014), http://dx.doi.org/10.1016/j.compeleceng.2014.10.008

Fig. 4 shows the flowchart of our algorithm. Given the encoded mode, the algorithm first generates an initial population. Then it tries to find a near-optimal solution iteratively. In every iteration, fitness values of individuals are computed to evaluate every solution, and new population is generated by selection, crossover and mutation operations. The process above is repeated until the convergence criterion is satisfied. In the following sections, every part of the algorithm will be explained in detail.

### 3.2. Fitness function

For each individual, we evaluate its user experience and assign it a fitness value based on SLA. We can distinguish two types of QoS criteria: positive criteria and negative criteria. The increase of values for positive criteria is beneficial for users, such as availability and reputation. The decrease of values for negative criteria is beneficial for users, such as time and price. The fitness function must promote the increase of positive criteria and the decrease of negative criteria. In addition, the fitness function needs to reflect users' preference. Users have preference to different QoS criteria. For example, some users prefer service with high availability to short response time. In our approach, weights are assigned to QoS criteria to represent users' preference. The weight is determined by users and described in SLA.

To compute the fitness value, QoS criteria need to be normalized first for a uniform measurement of multiple criteria independent of units. The normalization formulations of negative and positive criteria are as (1) and (2), respectively.

$$\zeta_i^-(CS) = \frac{Sq_i^- - q_i^-(CS)}{Sq_i^-} \tag{1}$$

$$\zeta_i^+(CS) = \frac{q_i^+(CS) - Sq_i^+}{Sq_i^+} \tag{2}$$

where $\zeta_i^-$ and $\zeta_i^+$ are the normalized values of the $i$-th QoS criterion of the composited service $CS$. $q_i$ represents the $i$-th QoS criterion of $CS$ and $Sq_i$ is the $i$-th QoS constraints defined in SLA.

Generally, in order to avoid SLA violation, it is better to maximize the values of positive criteria and to minimize the values of negative criteria. In addition, users' preference to different criteria should be considered in fitness computation. We use parameter $\alpha_i(\sum_{i=1}^{o}\alpha_i = 1)$ to reflect the user's preference. The larger the $\alpha_i$ , the higher the priority level of the corresponding criterion. Based on the aforementioned description, the fitness function is shown below.

$$f(CS) = \sum_{i=1}^{o}\alpha_i \times \zeta_i(CS) \tag{3}$$

where the number of QoS criteria is $o$. In the evolution process, the fitness function can help to maximize positive criteria and minimize negative criteria. In addition, it is helpful to eliminate individuals that do not meet SLAs. The goal of our algorithm is to find a composited service with the largest fitness value.

### 3.3. Encoding

In genetic algorithms, genomes represent the possible choices available in the problem. Possible solutions are encoded with genomes. In our approach, we encode composited services as genomes. Each gene in the genome encodes the atomic service for each task. For the example in Fig. 5, the shown genome is corresponding to a composited service that consists of four tasks. Each gene represents the chosen atomic service from each service set. For example, atomic service $s_{1i}$ is chosen to implement task1. In this example, the workflow consists of four tasks, so the genome consists of four genes.

### 3.4. Initial population

Generally, the initial population of genetic algorithm is generated randomly. In order to improve the solution quality and convergence speed, we generate one fifth of the initial population based on the notion of skyline [23] and four fifths of the initial population in random.

**Definition 3** (*Domination*). For two atomic services $s_1, s_2$ in a service set, $s_1$ is said to dominate $s_2$ when both of the following conditions are satisfied.

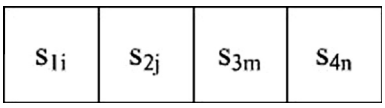


**Fig. 5.** Example of genome.

Please cite this article in press as: Wang D et al. A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electr Eng (2014), http://dx.doi.org/10.1016/j.compeleceng.2014.10.008

$$\left\{\forall\left(q_{i,1}^{-}, \quad q_{i,2}^{-}\right)|q_{i,1}^{-} \leqslant q_{i,2}^{-}\right\} \wedge \left\{\forall\left(q_{i,1}^{+}, \quad q_{i,2}^{+}\right)|q_{i,1}^{+} \geqslant q_{i,2}^{+}\right\} \tag{4}$$

$$\left\{\exists\left(q_{i,1}^{-}, \quad q_{i,2}^{-}\right)|q_{i,1}^{-} < q_{i,2}^{-}\right\} \vee \left\{\exists\left(q_{i,1}^{+}, \quad q_{i,2}^{+}\right)|q_{i,1}^{+} > q_{i,2}^{+}\right\} \tag{5}$$

where $q_{i,j}^{-}$ is the $i$-th negative QoS criterion of atomic service $s_j$, $q_{i,j}^{+}$ is the $i$-th positive QoS criterion of atomic service $s_j$.

Fig. 6 shows the example of Definition 3. For simplicity, we only use two QoS criteria of atomic services in this example. They are availability and reputation which are both positive criteria. According to the definition, atomic service d dominates atomic service $b$. Atomic service $f$ is not dominated by any other atomic services. In other words, $f$ is non-dominated.

**Definition 4** (*Skyline set*). Skyline set is a subset of service set. Skyline set comprises the atomic services in a service set that are non-dominated.

For example, the skyline set of Fig. 6 is $\{f,e,c\}$.

The skyline operator provide a better understanding of the trade-offs between QoS criteria. It is important for applications involving multi-criteria decision making [24]. In our algorithm, one fifth individuals of initial population are generated based on skyline operator. Atomic service from each skyline set is selected to be encoded as gene. These selected genes are encoded to a genome of a composited service.

The computation cost for skyline can be expensive if the number of atomic services per set is too large. Some researchers have proposed efficient algorithms to address this problem [24,25]. More importantly, the process of determining the skyline set does not need to be conducted online during service composition time. Brokers maintain a list of skyline set of each service set which can be updated when there are changes of atomic services. Therefore, the computation cost of skyline operator has no influence on the process time of service composition.

### 3.5. Selection operator

The strategy of roulette-wheel selection is used to select individuals to be reproduced via mutation and crossover. The probability that an individual with fitness value $f_k$ is chosen from the population is computed as follows.

$$p_k = \frac{f_k}{\sum_{j=1}^{N} f_j} \tag{6}$$

where $N$ is the number of individuals in the current population. $f_j$ is the fitness value of the $j$-th individual. Roulette-wheel selection ensures that better individuals have higher chance to be chosen and better genes have higher probability to be inherited.

### 3.6. Crossover operator

In the design of crossover operator, some pairs of the matching genomes are randomly chosen to be combined to generate offspring. For each parent genome, it is divided into two parts by a cut-off point. Then the tail parts are exchanged. A typical crossover operator is depicted in Fig. 7.

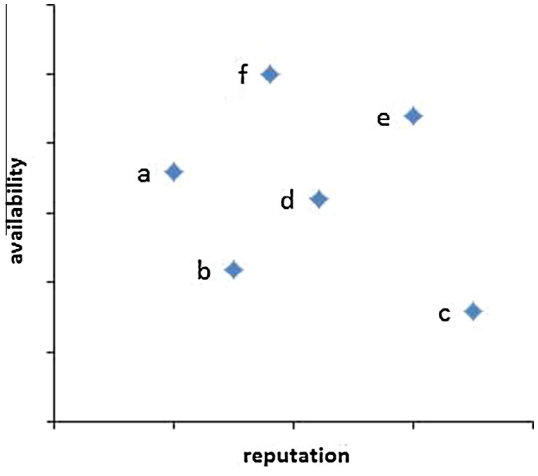


**Fig. 6.** Example of domination.

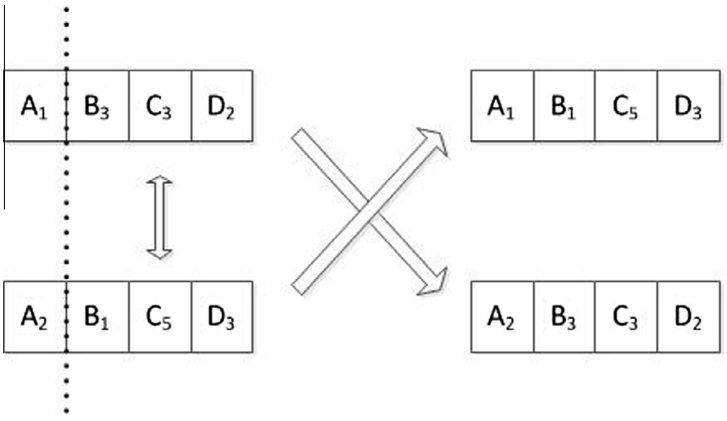


**Fig. 7.** Example of crossover operator.

### 3.7. Mutation operator

In order to avoid local optimum, mutation operator is applied to change offspring slightly in a random way. The mutation operator for a genome randomly selects a gene and randomly replaces it with an atomic service in the corresponding service set.

## 4. Simulation and evaluation

In this section, we present simulations of our model and algorithm: (a) model evaluation, in terms of solutions' fitness values generated under different models, (b) feasible rate, in terms of whether a solution satisfies the QoS constraints defined in SLA or not, (c) optimality, in terms of the solutions' fitness values, and (d) scalability, in terms of the computation time required to find a solution.

### 4.1. Simulation setup

All simulation results were measured with a Sun Java SE 7 VM running on Windows 7 PC with an AMD 4.1 GHz CPU and 8GB memory space.

For simulation, the four QoS criteria mentioned above were considered and they gained equal preference from users. QoS of atomic services were generated randomly. The random values of response time in each service set had a Gaussian distribution and a mean value within the range [20,1500]. The ranges of availability, price and reputation were [0.95,1] [2,15] and [0.4,1] in order.

We assumed that the cloud provider owned eight geo-distributed datacenters. The latency between two datacenters was generated between 20 ms and 500 ms depending on their distance so that a good range of realistic values will be covered. For simply, there was only one users' location. The network delay between atomic service and users was also generated between 20 ms and 500 ms depending on their distance.

We then created several QoS vectors of four random values in terms of the number of service sets. Each QoS vector, which is noted by $(Stime, Sprice, Savail, Sreput)$, is corresponding to one SLA-based composition request. The values of QoS vector represent the constraints of response time, price, availability and reputation defined in SLA in order. In reality, QoS constraints defined in SLA are generated through the negotiation between users and service providers. In our simulation, in order to evaluate our approach reasonably and efficiently, we generated the QoS vectors according to the number of service sets and the value ranges of QoS criterion. They were computed as follows:

$$Stime = 760m + 260(m+1) \tag{7}$$

$$Sprice = 8m \tag{8}$$

$$Savail = 0.98^m \tag{9}$$

$$Sreput = 0.72 \tag{10}$$

where m represents the number of service sets.

### 4.2. Evaluation methodology

For the purpose of our simulation, the following approaches were used.

Please cite this article in press as: Wang D et al. A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electr Eng (2014), http://dx.doi.org/10.1016/j.compeleceng.2014.10.008

1. *Exhaustive Search:* It traverses all the possible solutions and finds the optimal composited service that owns the largest fitness value.
2. *Random Selection:* It selects atomic services for each service set at random.
3. *GA_N_NET:* It is the normal generic algorithm with the composition model described in Section 2.
4. *GA_S:* It is our proposed skyline-based generic algorithm with the composition model neglecting the network environment.
5. *GA_S_NET:* It is our proposed skyline-based generic algorithm with the composition model described in Section 2. Unlike GA_S, the composition model of GA_S_NET considers the network delay.

The population sizes for all genetic algorithms were 100. The convergent criterion for all genetic algorithms was that the best fitness value does not improve over the last 30 iterations.

### 4.3. Model evaluation

We created nine composition instances shown in Table 3. The numbers of service sets, atomic services per set were given by parameters m and n. We measured the fitness value, response time, price, availability and reputation by running GA_S and GA_S_NET.

**Table 3**
Instances for model evaluation.

| No. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| *m* | 10 | 10 | 10 | 20 | 20 | 20 | 30 | 30 | 30 |
| *n* | 160 | 320 | 480 | 160 | 320 | 480 | 160 | 320 | 480 |

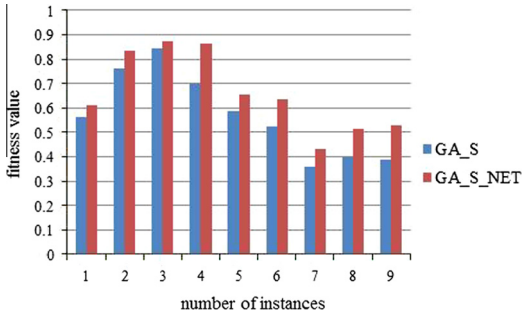


**Fig. 8.** Fitness values generated under different models.

**Table 4**
Results of model evaluation.

| No. | Approach | Response time | Price | Availability | Reputation |
|---|---|---|---|---|---|
| 1 | GA_S | 9416.54 | 47.82 | 0.79 | 0.73 |
| | GA_S_NET | 8820.17 | 41.5 | 0.82 | 0.76 |
| 2 | GA_S | 11085.68 | 36.03 | 0.88 | 0.83 |
| | GA_S_NET | 9534.22 | 32 | 0.84 | 0.84 |
| 3 | GA_S | 7111.08 | 55.38 | 0.77 | 0.82 |
| | GA_S_NET | 5533.69 | 43.99 | 0.78 | 0.82 |
| 4 | GA_S | 14773.65 | 119.78 | 0.68 | 0.75 |
| | GA_S_NET | 13567.53 | 102.38 | 0.72 | 0.85 |
| 5 | GA_S | 18036.58 | 114.24 | 0.73 | 0.76 |
| | GA_S_NET | 17142.77 | 109.41 | 0.75 | 0.78 |
| 6 | GA_S | 18744.49 | 96.62 | 0.68 | 0.72 |
| | GA_S_NET | 17871.48 | 92.54 | 0.67 | 0.8 |
| 7 | GA_S | 27506.59 | 194.93 | 0.6 | 0.74 |
| | GA_S_NET | 26122.96 | 210.27 | 0.62 | 0.76 |
| 8 | GA_S | 29503.66 | 178.94 | 0.58 | 0.69 |
| | GA_S_NET | 28421.91 | 158.69 | 0.57 | 0.82 |
| 9 | GA_S | 28200.15 | 176.01 | 0.57 | 0.74 |
| | GA_S_NET | 24787.31 | 169.64 | 0.57 | 0.79 |

Please cite this article in press as: Wang D et al. A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electr Eng (2014), http://dx.doi.org/10.1016/j.compeleceng.2014.10.008

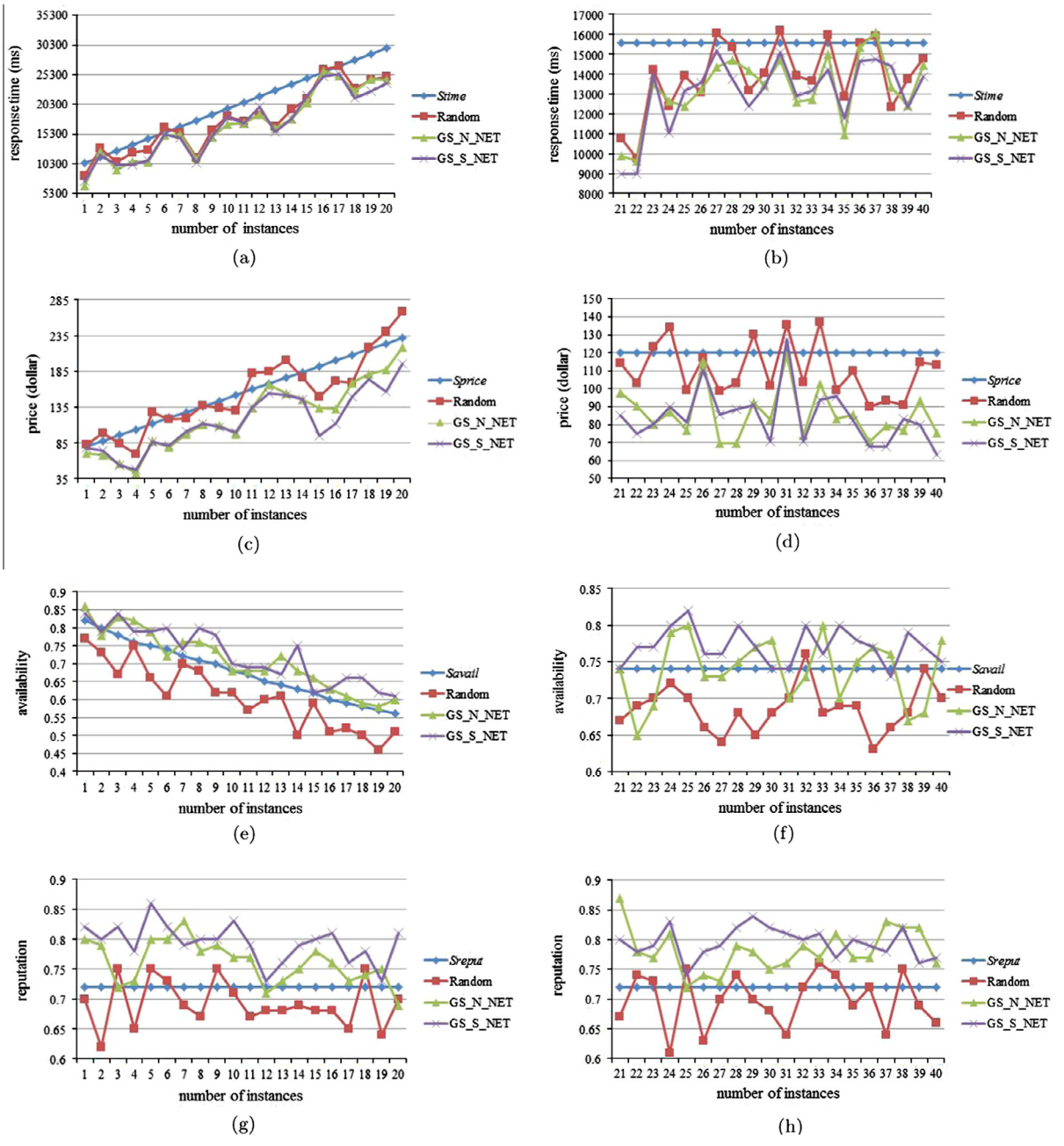


**Fig. 9.** Simulation results of feasible rate.

**Table 5**
Statistical results of feasible evaluation.

| | Random selection | GA_N_NET | GA_S_NET |
|---|---|---|---|
| Number of feasible solution | 1 | 26 | 38 |
| Feasible rate | 2.5% | 65% | 95% |

Fig. 8 shows the result of fitness value. We observe that GA_S_NET can find composited services with higher fitness value compared with GA_S. This is because the former approach considers network delay in the process of population evolution. In Table 4, we can observe that QoS of composited services generated through GA_S_NET is better than that generated through GA_S on the whole. A significant gain of user experience can be achieved when network environment is considered in composition model, which demonstrates that our proposed model is more applicable.

Please cite this article in press as: Wang D et al. A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electr Eng (2014), http://dx.doi.org/10.1016/j.compeleceng.2014.10.008

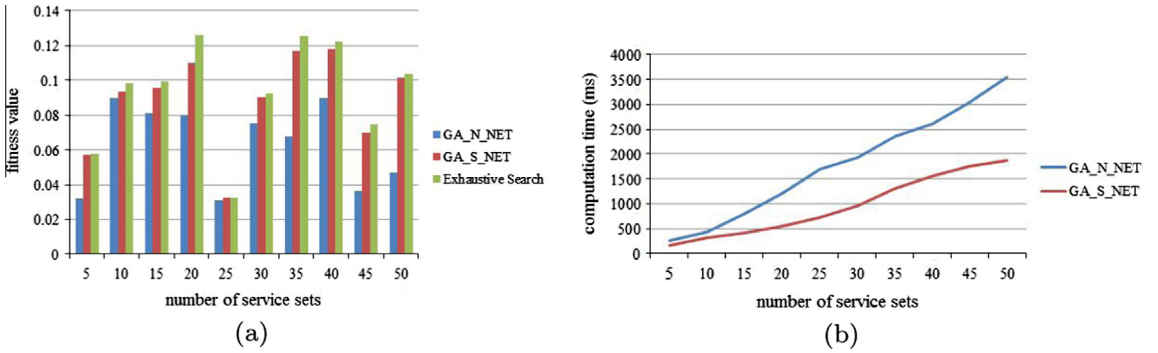


**Fig. 10.** Optimality and scalability versus number of service sets.

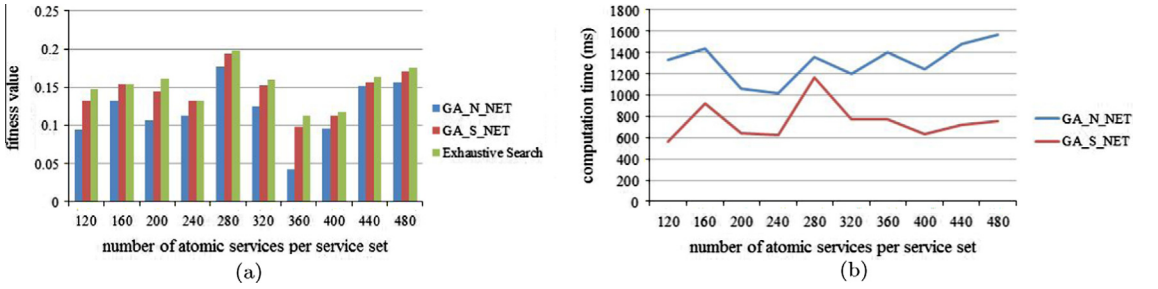


**Fig. 11.** Optimality and scalability versus number of atomic services per set.

### 4.4. Feasible rate

A composited service is feasible if its QoS satisfies the QoS constraints defined in SLA. The validity of an approach can be quantized through feasible rate. In this simulation, we measured the feasible rate of our approach. For this purpose, we created 40 scenarios. In the first 20 scenarios, the number of atomic services per service set was assigned a fix number 320 and the number of service sets ranged from 10 to 29. In the last 20 scenarios, the number of atomic services per service set increased from 40 to 800 at a rate of 40 per instance and the number of service sets was assigned a fix number 15. We measured the feasible rate by running three different approaches: Random Selection, GA_N_NET, and GA_S_NET. Fig. 9 records the QoS values and SLA constraints. Table 5 shows the statistical results of the three approaches' feasible rate obtained from 40 instances.

It can be seen that the feasible rate of Random Selection is very small. Composited services generated by Random Selection could hardly satisfy the SLA constraints, which illustrates the necessity of finding a practicable composition algorithm. In Fig. 9, we can see that all the QoS of composited services generated by GA_S_NET satisfies the SLA constraints except the response time and price of the second solution. In contrast with GA_S_NET, GA_N_NET generates much more solutions that violate the SLA constraints. The main reason leading to above results is that skyline method is used to generate initial population in GA_S_NET. The probability of violating SLA for the initial population generated by GA_S_NET is lower than that for the initial population generated by GA_N_NET. According to the statistical results shown in Table 5, the feasible rate of our algorithm is much higher than that of normal genetic algorithm, which indicates that our proposed composition approach in geo-distributed cloud environment is more feasible.

### 4.5. Optimality and scalability

To evaluate the optimality and scalability of our algorithm we compared the fitness value and computation time of different approaches versus an increasing problem size.

Fig. 10(a) presents the fitness value against an increasing number of service sets. We assigned a fix number 320 to the number of atomic services per service set. The number of service sets ranged from 5 to 50. Fig. 10(b) shows the outcome of scalability evaluation. We measured the computation time under the same settings as for solution optimality.

The results demonstrates that GA_S_NET outperforms GA_N_NET obviously. The fitness values of solutions generated by GA_S_NET are close to that of Exhaustive Search. As the increasing of the number of service sets, all three methods spend increasing time computing. As expected, the computation time of Exhaustive Search is too long to be depicted in figures. So Exhaustive Search is impracticable in real application. Compared with GA_N_NET, GA_S_NET converges much faster. In

Please cite this article in press as: Wang D et al. A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electr Eng (2014), http://dx.doi.org/10.1016/j.compeleceng.2014.10.008

addition, the computation time of GA_S_NET increases more slowly than that of GA_N_NET. GA_S_NET can obtain a near-optimal solution in short time.

In the simulation shown in Fig. 11, we evaluated the performance of approaches with respect to the number of atomic services per service set. Fig. 11(a) presents the fitness value against an increasing number of atomic services per service set. We assigned a fix number 20 to the number of service sets. The number of atomic services per service set ranged from 120 to 480. Fig. 11(b) shows the outcome of scalability evaluation under the same settings.

As shown in Fig. 11(a), GA_S_NET performs better than GA_N_NET with respect to fitness value. It can always generate near-optimal solutions. On the other hand, GA_S_NET need less computation time than GA_N_NET, while it has similar variation tendency of computation time with GA_N_NET as shown in Fig. 11(b). The reason for these results is that GA_S_NET use skyline method by which initial population generated has higher performance in average, which improves the solution quality and convergence speed. Consequently, our approach is more efficient and scalable.

## 5. Conclusion and future work

In this paper, we have addressed the problem of web service composition in geo-distributed cloud environment with SLA constraints. To deal with cases where cloud datacenters are located geographically, we present a QoS-based composition model considering network environment. We have also proposed a skyline-based genetic algorithm to solve the composition problem, simultaneously minimizing SLA violations. Our algorithm beats normal genetic algorithm on the optimality and computation time. The results of the simulation indicate that our model is more applicable and our algorithm can achieve a close to optimal result at the cost of lower computation time. In the current approach, service exceptions and incompatibility during composited service execution are not considered. In our future work, we aim at developing a runtime recovery strategy. This will make our approach more practical and effective.