Multi-objective Optimization Model with AHP Decision-making for Cloud Service Composition

Li Liu[1] and Miao Zhang[1]
[1] School of Automation and Electrical Engineering, University of Science and Technology Beijing
Beijing 100083, China
[e-mail:liuli@ustb.edu.cn, s20130926@xs.ustb.edu.cn]
*Corresponding author: Miao Zhang

Received April 4, 2015; revised June 25, 2015; accepted July 15, 2015; published September 30, 2015

Abstract

Cloud services are required to be composed as a single service to fulfill the workflow applications. Service composition in Cloud raises new challenges caused by the diversity of users with different QoS requirements and vague preferences, as well as the development of cloud computing having geographically distributed characteristics. So the selection of the best service composition is a complex problem and it faces trade-off among various QoS criteria. In this paper, we propose a Cloud service composition approach based on evolutionary algorithms, i.e., NSGA-II and MOPSO. We utilize the combination of multi-objective evolutionary approaches and Decision-Making method (AHP) to solve Cloud service composition optimization problem. The weights generated from AHP are applied to the Crowding Distance calculations of the above two evolutionary algorithms. Our algorithm beats single-objective algorithms on the optimization ability. And compared with general multi-objective algorithms, it is able to precisely capture the users' preferences. The results of the simulation also show that our approach can achieve a better scalability.

Keywords: Multi-Cloud, Services Composition, Multi-objective Evolutionary Algorithm, AHP

This research 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).

1. Introduction

Cloud computing is a new paradigm that shares resources such as infrastructures, platforms, online applications as web services to cloud users, and recently many different web services are published and available in cloud data centers [1]. There are three types of services: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) [2]. In order to address the complexity of users' requirements, most cloud services providers have to be coordinated to host their services. For example, SaaS providers can either utilize PaaS offerings (such as Google AppEngine) to develop applications, or rent virtual machines from IaaS providers (such as Amazon EC2) to deploy their own system environment. These selected multiple cloud services will be composed as one unified service to host their requirements.

The Cloud environment has a variety of service classes, and each workflow task can be completed by a specific service class. Each service class contains a number of service instances which have the same function but different QoS levels and prices. For example, Amazon EC2 can provide eight different QoS level service instances. Several service instances need to be composed to complete a workflow task, and the aggregate QoS of composed service also should meet the service level agreement (SLA) established through negotiation between users and service providers. SLAs are generally defined as end-to-end QoS requirements such as response time, reliability, cost and so on. In order to satisfy different SLA, the most suitable Cloud services need to be selected and deployed for optimization.

The process of Cloud service composition is very complicated, which consists of service discovery, service selection, and service coordination optimization. It is difficult to find the optimal configuration scheme from massive candidates of the composed service, and the size of the search space will grow up exponentially with increasing of service instances. So service composition in Cloud raises new challenges caused by the diversity of users with different QoS requirements and vague preferences, together with the development of cloud computing having a geographically distributed manner. Furthermore, different conflicting QoS objectives need to be optimized under several constraints at the same time increasing complexity.

Some evolutionary algorithms have been applied in Cloud service composition problem. In [3] and [4], single-objective genetic algorithm is used to solve service composition problem. They use the method of single-objective optimization to solve the multi-objective problems by assigning weights to each objective. But optimization capacity of this approach is not good enough compared with multi-objective evolutionary algorithms. There are two popular multi-objective evolutionary algorithms, namely NSGA-II (Non-dominated Sorting Genetic Algorithm-II) and MOPSO (Multi–Objective Particle Swarm Optimization). The two above algorithms are used to solve QoS-aware service composition problem both in [5] and [6]. A new multi-objective genetic algorithm proposed in [7] and [8], which uses the approach of data dimension reduction to reduce the number of optimization objectives, and decrease the complexity of the problem. The optimization capabilities of multi-objective evolutionary algorithms are better than single-objective algorithms. But in the case of users having no idea about the exact weights of each objective, the general multi-objective evolutionary algorithms just can give the Pareto-optimal set, rather than giving an exact solution. In [9], authors combine evolutionary algorithms and fuzzy logic method for composition optimization. This allows users to express their semantic requirement which brings a great comfort to them compared to systems that force users to assign exact weights for all preferences. However, it just applies fuzzy Logic in the final obtained Pareto-optimal set, which would miss some excellent solutions.

In this paper, we apply combination of multi-objective evolutionary approaches and Decision-Making method (AHP) for composition optimization. AHP is used to calculate the Crowding Distance instead of Euclidian Distance, and finds most satisfied solution in Pareto-individuals. Our main contributions can be summarized as follows: (1) we find the Pareto front composition solutions using different multi-objective algorithms (NSGA-II and MOPSO) and perform a comparative study between them for the Cloud service composition problem to determine which one will be suitable. (2) we apply decision-making method AHP to calculate the weights of user preference, and the weights are used to calculate the Crowding Distance for the multi-objective evolution algorithms. Finally, the simulation results demonstrate that the service composition solution obtained by our approach can meet users' preferences better.

The remainder of this paper is organized as follows: in the following section, an overview of the service composition architecture and the problem definitions are presented. Section 3 introduces a decision-making method for finding optimal solution in Pareto-individuals named AHP. Two evolutionary algorithms named NSGA-II and MOPSO for service composition are given in Section 4. In Section 5, several simulations of our algorithms are evaluated to compare with other existing approaches. We conclude the paper with a discussion and a description of future work in Section 6.

2. Problem Formulation

2.1 Service Composition Architecture

A workflow is depicted as a Directed Acyclic Graph (DAG) $G=(V,E)$, where $V=\{t_1,...,t_n\}$ and E is the vertices and edges of the graph, respectively. Each vertex represents a task t and there are n tasks in the workflow. The edges represent the precedence of different tasks. A directed edge from t_x to t_y, $x,y \in N$ means that t_y can't execute until t_x is completed [10]. **Fig. 1** shows an example of the workflow, each node represents a task, the arcs show the data transfer between nodes. There are 8 tasks: $\{t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8\}$ and 8 specific service classes to complete each task, $\{S_1,S_2,S_3,S_4,S_5,S_6,S_7,S_8\}$. Each service class S_i represents a type of service including a set of atomic service s_{ij} .The atomic service is a service instance offered by cloud provider, and different service instances operate at different QoS levels.

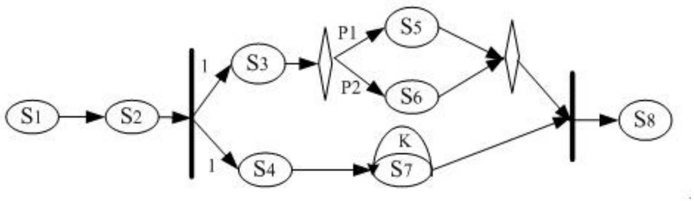


Fig. 1. A Workflow example

We have proposed a multiple services composition architecture for cloud workflow application, detailed in **Fig. 2**. The service requester submits initial goal descriptions of service implementation which is a series of abstract service components for workflow tasks, and presents the corresponding QoS level requirements to the Cloud Service Coordinator using a SLA [11]. The Cloud Service Coordinator deployed four service components (showed as Fig. 2) and it acts as an external, independent broker that can help users construct composited services. Then Service Discovery identifies all available candidate concrete services (published by service file) for a specific functionality. Next the Capability Planning evaluates the QoS capability of each service instance included in a service file and transforms the service request into cloud combination and service composition problem. Then the Service Selection picks the most appropriate Clouds and identifies the best service for service composition planning. Finally, the Service Composition obtains a composition plan, which is composed of a service composition sequence that can satisfy the requester's goal, and creates the composite service process from the selected services using a combinatorial optimization method.

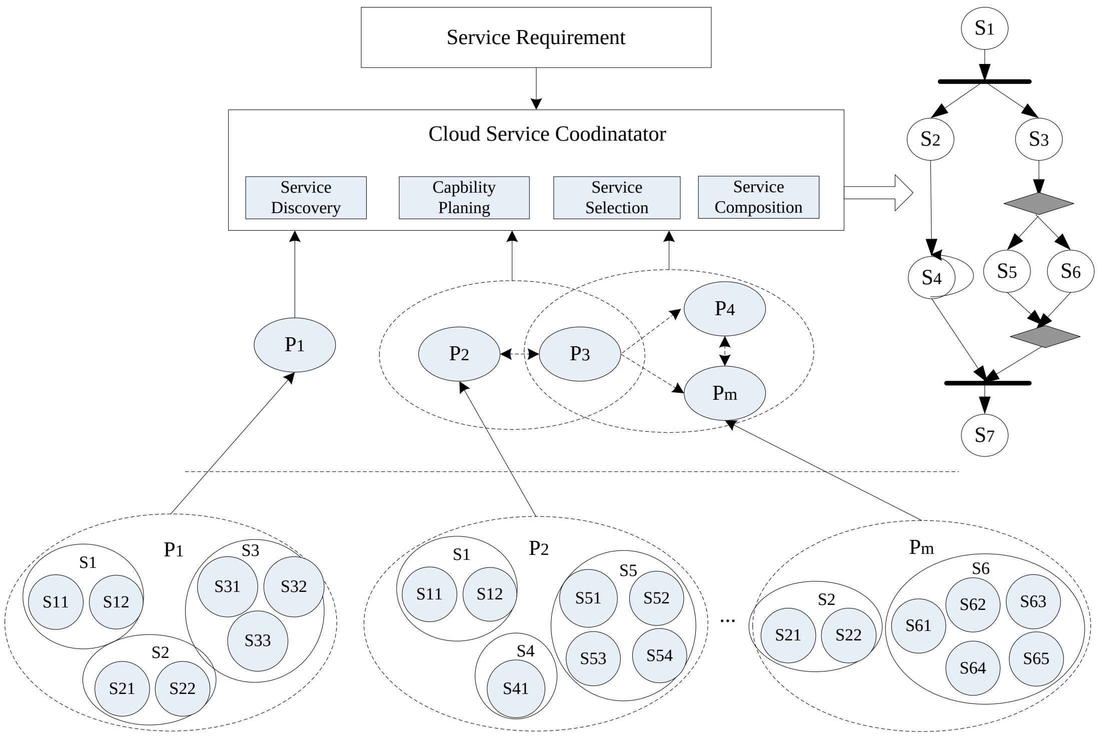


Fig. 2. Architecture of Service Composition.

2.2 QoS Models

The common SLAs for service composition are defined as the criterion of response time, reliability, and cost of an application. In order to judge whether a composed services satisfies the given SLAs, it is required to examine its end-to-end QoSs by aggregating QoS measures of individual services. There are four types of composition structures: sequential, parallel, conditional and loop. The aggregate functions for QoS computation of composed service are shown in Table 1. Where T_i is the response time, R_i is the reliability, and C_i is the cost for the service instance. When the service instances are from different Cloud Provider, there must exists network delay, so the aggregate service response time contains the response time of each service and the network latency between services. The latency between services *L(i)* is measurable and predictable which is not the focus of this paper.

Table 1. QoS Aggregate functions

QoS Criteria	**Sequential**	**Conditional**	**Parallel**	**Loop**
Response time (ms)	$T=\sum_{i=1}^{N} T_i + \sum_{i=2}^{N} L(i)$	$T=\sum_{i=1}^{N} p_i * T_i$	$T=\max_{i=1}^{N} T_i$	$T=k*T_i$
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 ($)	$C=\sum_{i=1}^{N} C_i$	$C=\sum_{i=1}^{N} p_i * C_i$	$C=\sum_{i=1}^{N} C_i$	$C=k*C_i$

2.3 Problem Definition

The optimization goal of cloud service composition is to find a solution with better performance and meet the users' QoS requirements in a vast search space. Thus, for a given workflow application, how to select the suitable service for each task and compose these services to meet QoS requirements specified in SLA is a challenge.

The final objective in our work is to find an optimized service composition which is able to maximize the reliability and minimize the deployment cost and time, while satisfying the users' constraints. The reliability, cost and time is also considered as user's constraint for each Cloud service. There are challenges to find an optimal composition as these objectives can conflict with each other. We find a way to combine multi-objective evolutionary algorithms and Decision-Making method (AHP) for composition optimization. AHP is applied on calculating the Crowding Distance of evolutionary algorithms. Now, we give some definitions and description of notations we will use in the flowing sections.

Definition 1 (Non-dominated solution): Supposing there are two solutions π_1 and π_2, considering all optimization objectives, if π_1 is higher than π_2 at least in one objective, and the other objectives of both π_1 and π_2 are all equal, π_1 dominates π_2; if π_1 is not dominated by any other solutions, π_1 is non-dominated solutions, also known as the Pareto solution.

Definition 2 (Pareto-Optimal front): That is a surface that formed by Pareto-Optimal solutions in space. The Pareto front solutions are not only dominated by the others within the Pareto-Optimal front but also dominated by the others out of the Pareto-Optimal front.

We firstly provide the definitions of the terms and notions in **Table 2** for future references.

Table 2. Notations

Notation	**Description**
T	the response time of the composed service
R	the reliability of the composed service
C	the cost of the composed service
M	number of optimized objectives
N	the size of population in NSGA-II, and the number of particles of a swarm in MOPSO
Q_i^j	the j-th QoS criterion of individual i
pop_g	the population of the generation *g* in NSGA-II
p_c	the crossover probability in NSGA-II
p_m	the mutation probability in NSGA-II

$g_{\max}$	the max generation in NSGA-II
P^t	the particle swarm in the *t* iteration in MOPSO
x_i^t	the particle *i* in the *t* iteration in MOPSO
v_i^t	the velocity for particle *i* in MOPSO
$Pbest^t$	the personal optimal particle in the *t* iteration in MOPSO
$Gbest^t$	the groups optimal particle in the *t* iteration in MOPSO
$I_{\max}$	the max iteration in MOPSO
$\omega_{\max}$ $\omega_{\min}$	the maximum of inertia weight in MOPSO the minimum of inertia weight in MOPSO
c_1, c_2	learning factors for the local optimal particle and group optimal particle in MOPSO

3. AHP Decision-Making Method

There are multiple QoS criteria that need to be optimized in service composition. Most unskilled users usually have vague preferences to be unable to accurately assign weights for each QoS objectives. Several Pareto Front solutions will be generated from the evolutionary algorithms, so we have to find a way to filter them according to users' preference.

Decision-making methods have been effectively applied for determining the weight of QoS criteria, such as CRITIC [12], AHP [13], entropy method [14], TOPSIS method [15], and so on. All these methods can find a suitable solution for users in these Pareto front solutions without knowing the exact weights for each QoS objective. In this paper, we apply the weights generated from AHP to Crowding Distance of evolutionary algorithm.

One way to optimize multi-objective service composition problem is to assign weights for each objective based on users' preference. But in most cases, users just vaguely comprehend the relative importance of the two criteria, and they cannot make decision just based on the relative importance of ratios. The AHP approach has been proposed to handle the similar decision situations.

The AHP is a multi-criteria decision-making approach which can be used to solve complex decision problems [16]. The pertinent data are derived by using a set of pairwise comparisons. These comparisons are used to obtain the weights of QoS criteria, and the relative performance measures of the alternatives in terms of each individual decision criterion. If the comparisons are not perfectly consistent, the AHP provides a mechanism for improving consistency.

As an example which QoS criteria includes response time, cost, reliability and availability, the hierarchy structure is shown in **Fig. 3**.

3.1 Assign weight to each metric

Users adopt a pairwise comparison mechanism to determine the relative priority of each metric, and construct pairwise comparison matrix A. For example, we compare the relative importance between response time and cost, that is to determine which one has a greater impact to the user's preference, and how much the degree of influence will be. Usually integer 1-9 is used for pairwise comparison: 1 means the equal importance, 9 means the highest level of importance. After the pairwise comparison, matrix $A=(a_{ij})_{m\times m}$ (*m* is the number of metrics) is as Eq.1.

$$A = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \tag{1}$$

where $a_{ii} = 1, a_{ji} = {}^{1}\!/_{a_{ij}}$. The weights for each metric are given by the right eigenvector ω corresponding to the highest eigenvalue $\lambda_{\max}(A)$, the weight ω derived by matrix A, and obtained by the equation $A\omega = \lambda_{\max}(A) * \omega, \omega = (\omega_1, ..., \omega_m)$.

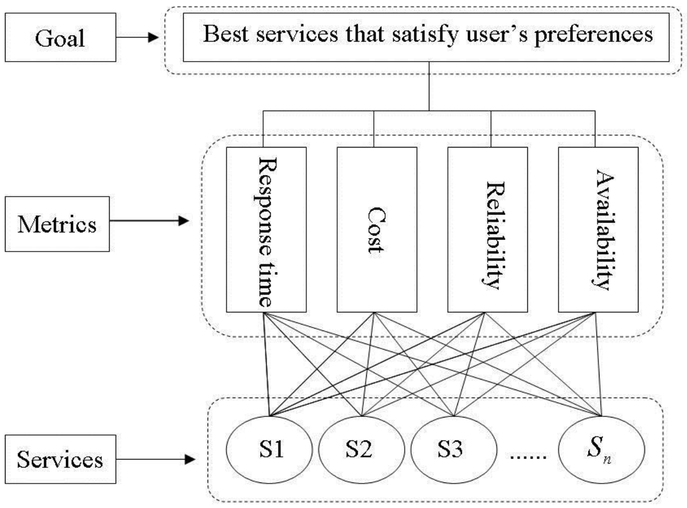


Fig. 3. The AHP hierarchy model of service selection.

3.2 Consistency check

The purpose of consistency check is for testing coordination of the important degree between each metric. We have to avoid the contradiction situation, i.e. for a user, A is more important than B, B is more important than C, and C is more important than A. For example, it is inconsistent in the case where a service user thinks that availability is strongly more important than reliability, and reliability is more important than cost, however the cost is more significant than availability to this user. The consistency index (CI) is shown as Eq.2.

$$CI = \frac{\lambda_{\max} - n}{n-1} \tag{2}$$

where n is the number of metrics. The consistency ratio (CR) is calculated as $CR = {}^{CI}\!/_{RI}$, indicates whether the evaluations are sufficiently consistent. The RI is obtained from the random consistency index table. If $CR < 0.1$, the consistency rate is acceptable; Otherwise, matrix A needs to be revised to be calculated again. Finally, the weight of each trust metric can be obtained.

4. Multi-Objective Optimization of Service Composition

Evolutionary algorithms have been effectively applied for solving optimization problems.

Among them, the NSGA-II and MOPSO outperform the other genetic optimization algorithms.

4.1 Service composition optimization with NSGA-II

Genetic Algorithm (GA) is a heuristic approach which contains selection, crossover and mutation operations to iteratively find near-optimal solutions in large search spaces [7][17]. NSGA-II is the most popular evolutionary approach to solve multi-objective optimization problem. The Non-Dominated sort and Crowding Distance are the cores of NSGA-II algorithm. The basic operations of NSGA-II are similar to GA, but in the selection process, NSGA-II uses non-domination sort and crowding distance to sort the individuals in population.

In the process of NSGA-II, a population is initialized with *N* individuals firstly, then the rank of non-domination sort and the crowding distance for each individual are calculated. Binary tournament selection is used to select offsprings. The individual with higher non-domination rank and longer Crowding Distance has more probability to be selected. Then the offsprings goes in to genetic operations, i.e. crossover and mutation. After that, the population contains parent population and offsprings which are sorted based on non-domination sort and crowding distance. Only the best *N* individuals are selected into the next iteration population.

Algorithm 1 shows the optimization process in NSGA-II, a rank-based roulette wheel selection scheme is used to implement the selection operation.

Algorithm 1 EVOLUTION PROCESS IN NSGA-II

```
Input: $g_{\max}$ , $p_c$ , $p_m$ ,N,M
Output: best population $pop_b$
1. Set $g_{\max}$ //set the maximum generation.
2. Initializing a population $pop_g$ randomly.
3. FOR g=1: $g_{\max}$ {
    3.1 $NDS_g$ =non-domination-sort ( $pop_g$ )// rank individual by non-dominant sort.
    3.2 $CD_g$ = crowding-distance-calculation ( $pop_g$ ) // calculate crowding distance of individuals with same $NDS_g$ .
    3.3 offsprings= selection ( $pop_g$ , $NDS_g$ , $CD_g$ ) // generate a new population according to non-dominant rank and crowding distance.
    3.4 offspringc= crossover (offsprings, $p_c$ ) //execute crossover operation to population.
    3.5 offspringm= mutation (offsprings, $p_m$ ) // execute mutation operation to population.
    3.6 popcomb= $pop_g$ + offspringm //combine the former population and the generated population by evolutionary operation.
    3.7 $NDS_{popcomb}$ =non-domination-sort (popcomb) // rank each individual according to non-dominant sort for the combined population.
    3.8 $CD_{popcomb}$ = crowding-distance-calculation (popcomb) // calculate crowding distance of individuals with same $NDS_g$ for the combined population.
    3.9 popcombs= selection (popcomb) // get a new population according to non-dominant rank and crowding distance.
    3.10 $pop_{g+1}$ = popcombs//get the next generation population.
    3.11 g=g+1}
END
```

A. Non-Dominated sort

In the search space of multi-objective service composition, there exists a set of solutions which are superior to the others with consideration of all QoS criteria. These solutions are known as Pareto-optimal solutions or non-dominated solutions. We assign ranks to all solutions in population based on domination relationship. The solution with higher rank dominates the lower one.

Fig. 4 shows an example of non-dominated sort with two QoS criteria. In this example, we need to maximize reliability and minimize cost. We examine non-dominated sort rank for six individuals as shown in this figure.Individuals A, B, C are in the first rank, which are called non-dominated individuals or Pareto-Optimal solutions. Solutions D, E are in the second rank, which are dominated by the first rank class individuals. Similarly, solution F is in the third rank.

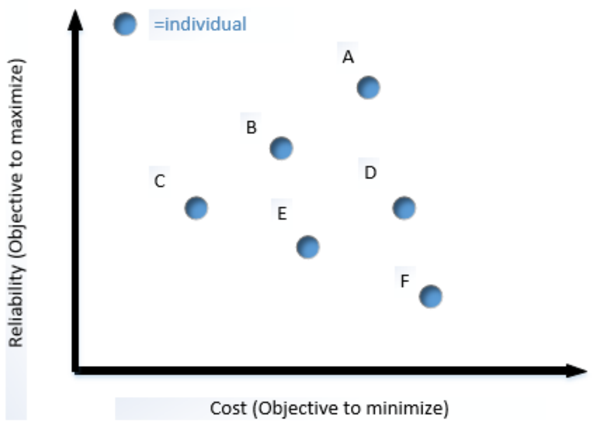


Fig. 4. An example domination ranking

Algorithm 2 shows the non-dominated sort process in NSGA-II. $pop_g(r)$ is individuals whose rank is r. $rank(i)$ is the rank for individual i. The QoS values of composed service are calculated as in **Table1**.

Algorithm 2 NON-DOMINATED SORT

```
Input: population $pop_g$
Output: ranks for all individuals
1. Set r=1
2. WHILE size( $pop_g$ )≠0; // loop until each individual has been assigned non-dominated rank.
3. FOR i=1: size( $pop_g$ )
   FOR j=1: size( $pop_g$ )
     dom_less=0; dom_more=0; dom_equal=0;
       FOR k=1:M   //comparing every optimization objective for each individual.
         IF $Q_i^k < Q_j^k$
           dom_less= dom_less+1;
```

```
            ELSEIF $Q_i^k = Q_j^k$
              dom_ more = dom_ equal +1;
            ELSE
              dom_ equal = dom_ more +1;
           END
         END
       IF dom_less=0 and dom_ equal≠M
         dominated_number(i)=dominated_number(i)+1; // calculate the number of individuals which dominate individual i.
       ELSEIF dom_ more =0 and dom_ equal≠M
     dominate_indi(i)= dominate_indi(i)+individual (j);//get the individuals  dominated by individual i;
       END
  END
     IF dominated_number(i)=0
      rank(i)=r;
     END
  END
     $pop_g = pop_g - pop_g(r)$ //get individuals that have not been assigned rank;
    r=r+1;
END
```

B. Crowding distance

Crowding distance is used to sort individuals with the same rank, and determine the probability entering into the next generation for each individual. The basic idea of the crowing distance is calculating the Euclidian distance between each pair of neighbor individuals in a rank class for all QoS criteria. Crowing distances of the individuals in the boundary are always set as infinite [18].The individual with higher rank has greater probability to be selected. When the two individuals are in the same non-dominated rank, the one with greater crowding distance should be selected, which means less individuals around it.

Algorithm 3 shows the process of calculating the Crowding Distance in NSGA-II. The Crowding Distance CD_i of individual *i* is calculated by Eq. 3.

$$CD_i = \sqrt{\sum_{i=1}^{N} \left(\frac{Q_j^{i+1} - Q_j^{i-1}}{Q_j^{\max} - Q_j^{\min}}\right)^2} \tag{3}$$

Algorithm 3 CROWDING DISTANCE CALCULATION

Input: $pop_g(r)$ *(individuals whose rank are r)*

Output: CD_i *(crowding distance for individual i)*

```
  FOR each rank class individuals $pop_g(r)$
    s=size( $pop_g(r)$ );
      FOR j=1:M
      $CD_1 = \infty, CD_s = \infty$
        FOR i=2:(s-1)
             Calculate $CD_i$ according to Eq.3
        END
      END
    Sort individuals in $pop_g(r)$ based on $CD_i$
    END
```

4.2 Service composition optimization with the MOPSO

Particle Swarm Optimization (PSO) is a popular evolutionary algorithm [19]. The optimization process in PSO is very similar to GA. The PSO first initialize a group of particles from the search space, each particle represents a solution. The position, velocity and fitness are the three indicators of the particles, and fitness functions are defined to calculate the fitness values for all of the criteria. Particle travels in the search space, it update position by tracking personal optimal particle(*Pbest*)and global optimal particle(*Gbest*). *Pbest* is the personal best known position which is the optimal solution in a iteration, while *Gbest* is the best known positions in the previous iterations. Once particle updates its position, the fitness value is calculated, *Pbest* and *Gbest* are updated too. Thus, the whole particle swarm proceeds toward optimal fitness value, and achieves optimization in the search space.

In the multi-objective PSO, there also exist many Pareto-optimal solutions, and we introduce Non-dominated sort and Crowding Distance to find the personal optimal particle and group optimal particle. In the *t*-th iteration, there are *N* particles, the position of particle *i* is $x_i^t = [x_{i,1}^t, x_{i,2}^t, ..., x_{i,M}^t]$,and $v_i^t = [v_{i,1}^t, v_{i,2}^t, ..., v_{i,M}^t]$,the personal optimal positions for all particles are $Pbest = [Pbest_1, Pbest_2, ..., Pbest_N]$, the global optimal particle is $Gbest$, we can calculate the position and velocity for the $t+1$-th iteration using Eq.4 and Eq.5.

$$v_i^{t+1} = \omega^t v_i^t + c_1 r_{1i}^t (Pbest_i - x_i^t) + c_2 r_{2i}^t (Gbest - x_i^t) \tag{4}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{5}$$

where r_{1i}^t and r_{2i}^t are random numbers within (0,1); ω^t is the inertia weight, which represents the inheritance for the previous iteration's speed, $\omega^t = \omega_{\max} - (\omega_{\max} - \omega_{\min}) * \frac{t}{I_{\max}}$, **Algorithm 4** shows the optimization process in MOPSO.

Algorithm 4 EVOLUTION PROCESS IN MOPSO

```
Input: $I_{\max}$, M,N, $\omega_{\max}, \omega_{\min}, c_1, c_2$
Output: Best particle swarm $P_b$
Step1. Set $I_{\max}$, $\omega_{\max}, \omega_{\min}, c_1, c_2$
    Set t=1, $v_i^t = zeros(1, M)$
    Set bounds of velocity and position for individuals
Step2. Initial initializing a particle swarm $P^t$.
    non-dominated-sort ( $P^t$ )//calculate the non-dominate sort rank of individuals
    crowding-distance ( $P^t$ )// calculate the crowding distance of individuals
    get $Pbest$, $Gbest$
Step3. Repeat until t= $I_{\max}$ {
  $\omega^t = \omega_{\max} - (\omega_{\max} - \omega_{\min}) * \frac{t}{I_{\max}}$
  FOR i=1:N
```

```
        Calculate $v_i^{t+1}$ and $x_i^{t+1}$ according to Eq.4 and Eq.5 respectively
     IF $x_i^{t+1}$ better than $Pbest_i$
        $Pbest_i = x_i^{t+1}$
     END
     IF $x_i^{t+1}$ better than $Gbest$
        $Gbest = x_i^{t+1}$
     END
   END
  get a particle swarm with N number particles $P^{t+1}$
  t=t+1
  }
```

The initialization stage is shown in Step 1 and Step 2. In Step 1, we set constant variables and initialize velocity for each particle as 0, positions and velocities are also restricted according to the range of solution space. In Step 2, the algorithm initializes particles scattering randomly in the solution space, and use non-dominated sort and crowding distance to find *Pbest* and *Gbest* of the initialized particle swarm. Step 3 is the iteration stage, which uses Eq. 4 and Eq.5 to generate a new particle swarm to update the *Pbest* and *Gbest* . Then the algorithm gets a new particle swarm for the next iteration. The step 3 is looped until the maximum iteration, and Pareto-optimal solutions are generated.

4.3 AHP based multi-objective evolutionary algorithms

The non-dominated solutions generated by the multi-objective evolutionary algorithms have their own excellent aspects. While without knowing the weights of QoS criteria, it is difficult to find the most preferred solution for users. Most users have some vague idea regarding the optimized objectives. In this case, we can use AHP method to capture the importance relationship of objectives, and find the composition solution which is the most appealing preference to the users.

In most existing works [9], researchers use decision-making methods to find the most preferred solution from the obtained Pareto-optimal set. However, with the increasing of problem size, the solutions in the Pareto-optimal set become unnumbered, and this approach may neglect some excellent solutions. In this paper, we combine AHP method with multi-objective algorithms by applying AHP to calculate Crowding distance instead of using Euclidian distance to sort the individuals. Based on above two multi-objective algorithms and AHP, NSGA-II-AHP and MOPSO-N-AHP are proposed, which can satisfy the users' preferences better. The new crowding distance NCD_i for solution *i* is calculated in Eq.6.

$$NCD_i = \sum_{i=1}^{N} \omega_j * Q_j^{i'} \tag{6}$$

where ω_j is the weight for objective *j* obtained by AHP, and $Q_j^{i'}$ is the quality value of solution *i* for *j*-th objective which has been normalized. The normalized QoS criteria of $Q_j^{i'}$ can be calculated as Eq. 7.

$$Q_j^{i'} = \frac{Q_j^i - Q_j^-}{Q_j^+ - Q_j^-} \tag{7}$$

For the positive QoS criteria to be maximized, the ideal value $Q_j^+ = Q_j^{\max}$, which is the maximum of the *j*-th objective value for all solutions in a population, and $Q_j^- = Q_j^{\min}$ which is the minimum, while for the negative criteria to be minimized, the ideal value $Q_j^+ = Q_j^{\min}, Q_j^- = Q_j^{\max}$.

5. Evaluation

Firstly, we evaluate and compare the performance of NSGA-II and MOPSO in addressing the problem of workflow service composition. Then our method of combination AHP with NSGA-II and MOPSO is evaluated respectively compared with the other existing algorithms.

5.1 Simulation setup and data collection

The data of web services are obtained from QWS Dataset (2.0) [20]. We select three attributes as the QoS criteria. Q={response time(ms), reliability(%), cost($)}.The simulated workflow has been shown in **Fig. 1**. All the experiments are performed on computers with Inter Core i5-4570S CPU(2.9GHz and 8G RAM).

In our simulations, $R = \{R_1, R_2, ..., R_8\}$ represents 8 service requirements, and we get 8 service functionality classes $S = \{S_1, S_2, ..., S_8\}$, and each of them includes 80 service instances provided by 8 geographically distributed Cloud Providers (CPs). The $Cons = \{cons^1, cons^2, cons^3\}$, $cons^1, cons^2, cons^3$ are defined as the 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 CPs. In order to evaluate our approach efficiently, we define the QoS constraints according to the number of service sets shown in Eq.8-Eq.10.

$$cons^1 = \overline{T} * m + \overline{L} * m \tag{8}$$

$$cons^2 = (\overline{R})^m \tag{9}$$

$$cons^3 = \overline{C} * m \tag{10}$$

where m is the number of service classes, $\overline{T}$, $\overline{R}$ and $\overline{C}$ is the average response time, average reliability and average cost for the all service instances respectively. $\overline{L}$ is the average latency between each two CPs. For two continuous services from the same CP, the latency is zero. For two services from different CPs, there will be network delay. The latencies (ms) are generated depending on their distance and it is within the range [100,200].

The probability of crossover p_c and mutation p_m in NSGA-II are 0.8 and 0.2 respectively. Our genetic algorithm uses rank-based roulette wheel scheme for selection operation. In the MOPSO, the maximum and minimum of inertia weight $\omega_{\max}, \omega_{\min}$ are set as 0.9 and 0.4, learning factors c_1, c_2 are set as 2.

We configure NSGA-II and MOPSO to run 100 generations with 50 individuals (or particles), the algorithm examines 5050 individuals at most, which is able to find excellent Pareto-optimal solutions within limited time.

5.2 Comparing of NSGA-II and MOPSO

To evaluate the performance of our algorithms, convergence and diversity are normally considered. The distance between the Pareto-front generated by the proposed algorithms and the real Pareto front, should be minimized as zero so that the estimated Pareto-front converges to the real value. In this experiment, we use exhaustive search method to obtain the real Pareto fronts. The performances of the above algorithms are compared in terms of the execution time, IGD (inverse generational distance) and Spread [21].

IGD calculated by Eq.11 determines the convergence of the algorithms, which is the average distance of the obtained solution points from the real Pareto fronts.

$$IGD = \frac{\sqrt{\sum_{i=1}^{N} d_i^{\,2}}}{N} \tag{11}$$

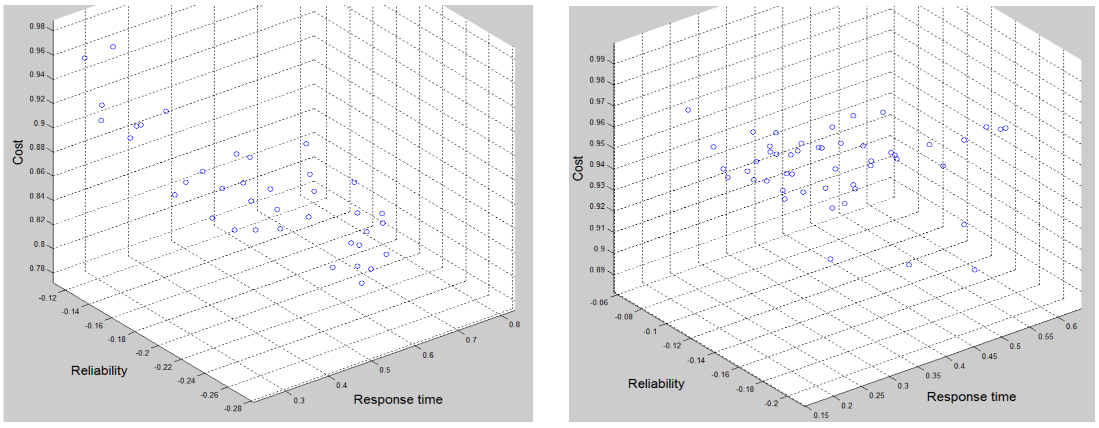
where the d_i is the Euclidean distance between the obtained solution points and the closest point of the real Pareto front and N is the number of the obtained solutions. Hence, $IGD = 0$ indicates that the obtained solutions are in the real Pareto front [9].

Spread in Eq. 12 illustrates the diversity of the algorithms [22], and it is a metric to calculate the broadness.

$$Spread = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1) * \bar{d}} \tag{12}$$

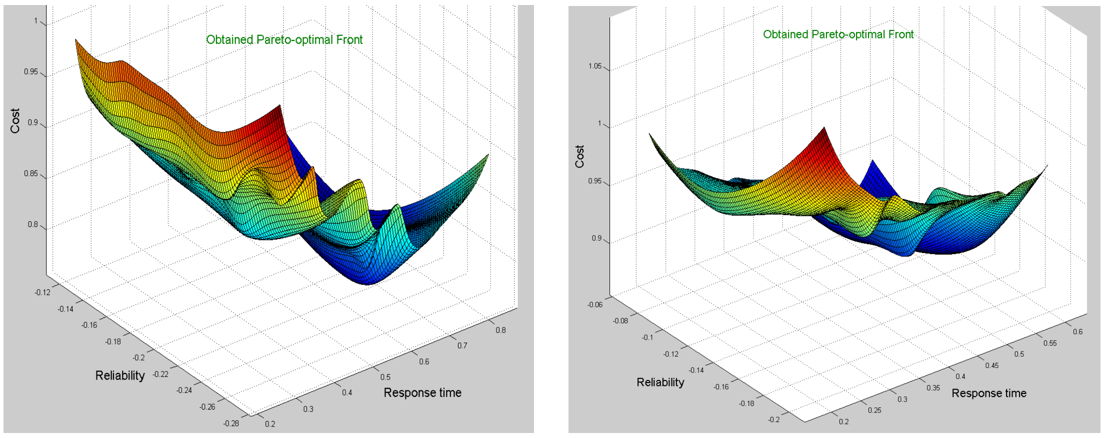
where d_i is the Euclidean distance between two neighbor points obtained. d_f and d_l are the Euclidean distance of the two boundary points in the obtained Pareto front set. $Spread = 0$ means that the obtained Pareto front perform well in diversity.

It would take a long time to find the optimal Pareto fronts for a relatively large search space through exhaustive search method, so we decrease the size of the search space to test the performance of the two algorithms. We calculate 95% Confidence Interval (CI) for each indicator of two algorithms in 30 independent runs. The smaller *IGD*, the better convergence property the algorithm has, so as the *Spread*. **Fig. 5** and **Fig. 6** have shown the outputs and-the optimal Pareto front obtained from NSGA-II and MOPSO.



(a) Output of NSGA-II (b) Output of MOPSO

Fig. 5. Outputs of NSGA-II and MOPSO



(a) Pareto front of NSGA-II (b) Pareto front of MOPSO

Fig. 6. Pareto front of NSGA-II and MOPSO

As shown in **Table 3**, we measured the averagevalue (Avg), standard deviation (STD) and 95% CI of execution time, IDG and Spread respectively by running NSGA-II and MOPSO. We can see NSGA-II performs better than MOPSO according to execution time, because MOPSO would cost some time to calculate every particle's velocity. It also demonstrates that the NSGA-II has better diversity and the better convergence ability than MOPSO.

Table 3. Comparison of Performance for the two Evolutionary Algorithms

	Execution time (s)			**IGD**			**Spread**		
	Avg	STD	95% CI	Avg	STD	95% CI	Avg	STD	95% CI
NSGA-II	1.359	0.055	[1.333, 1.384]	0.015	0.019	[0.0056,0.0233]	0.563	0.059	[0.536,0.590]
MOPSO	2.577	0.032	[2.562,2.592]	0.027	0.021	[0.0173,0.0365]	0.610	0.078	[0.574,0.647]

5.3 Evaluation of the evolutionary algorithms with AHP

We use the decision-making method AHP to obtain weight of each optimization objective which will be used in the evolutionary algorithms later.

In following experiments, the relative importance value between the response time and the reliability is set to be 3, i.e. $a_{12}=3$, similarly, we set $a_{13}=8$ that represents the relative importance value between the response time and the cost and $a_{23}=5$ that represents the relative importance value between the reliability and the cost. We get $\omega=(0.6612,0.2718,0.0670)^T$, the CI=0.07597, which is less than 0.1.

For the purpose of comparison, the following approaches were used.

1. GA with AHP: It applies AHP to obtain weights of objectives, transforming the multi-objective service composition problem as a single-objective problem, and using the single-objective GA to find the optimal solution.

2. PSO with AHP: It is similar to GA with AHP, combining AHP and single-objective PSO to find the optimal solution.

3. NSGA-II with AHP: It first obtains the Pareto-optimal solution set by running NSGA-II, and then using AHP to find the most satisfied solution.

4. MOPSO with AHP: It is similar to NSGA-II with AHP, while using MOPSO to find Pareto-optimal solution set.

5. NSGA-II-AHP: It is our proposed approach which applies AHP method to calculate Crowding distance of NSGA-II instead of Euclidian distance.

6. MOPSO-AHP: It is our proposed approach which applies AHP method to calculate Crowding distance of MOPSO.

We measured the response time, the reliability and the cost by running the above six algorithms respectively shown as **Table 4**. It can be seen that the optimization ability of multi-objective algorithms are much better than the single-objective GA and PSO with AHP. Compared with the most existing methods, our approaches which apply AHP on calculating the Crowding distance in NSGA-II and MOPSO are able to meet user's preferences better.

Table 4. Performance Comparing of Evolutionary Algorithms with AHP

Approach	**Response time (ms)**		**Reliability (%)**		**Cost($)**	
	Avg	95% CI	Avg	95% CI	Avg	95% CI
GA with AHP	1408.1	[1300.47, 1515.75]	14.1	[10.69, 17.43]	842.2	[831.49, 852.91]
PSO with AHP	1648.7	[1391.36 ,1909.01]	14.4	[10.24, 18.46]	878.8	[844.41,913.22]
NSGA-II with AHP	1224.3	[1130.55,1288.13]	28.7	[27.98 ,29.33]	832.0	[818.45,847.77]
NSGA-II-AHP	1131.4	[1089.92, 1172.84]	24.6	[21.24, 27.95]	830.2	[815.15,845.25]
MOPSO with AHP	1190.1	[1043.57, 1336.53]	19.7	[18.32, 21.04]	837.7	[825.49,849.91]
MOPSO-AHP	1036.4	[980.42 ,1092.42]	19.3	[18.85, 19.79]	846.2	[834.33,858.07]

To evaluate the scalability of our algorithm, we compared the output QoS values of different approaches with an increasing problem size. The QoS criteria of service instances are generated randomly as a Gaussian distribution. The response time (ms) has a mean value of 1000 with standard deviation of 20. The mean value of reliability (%) and cost($) are 90 and 100 respectively, the standard deviation of them are 5 and 10 respectively. We set $\omega = (0.6612, 0.2718, 0.0670)^T$ which is the same as the one in previous experiments, and it means that, for users, the most preferred objective is response time. Population size in NSGA-II and MOPSO are 100, the iterations number are set as 100. We have run each algorithm independently for 30 times under an increasing number of service instances.

Table 5 presents the compared average QoS values of composited services by running different approaches with increasing number of service instances. As shown in **Table 5**, our proposed approaches perform better with the increasing number of service instances.

Table 5. Comparing of different approaches with increasing number of service instance

Number of service instances	**Approach**	**Response time (ms)**	**Reliability (%)**	**Cost($)**
80	NSGA-II with AHP	4522.97	61.856	816.6
	NSGA-II - AHP	4319.14	64.217	895.72
	MOPSO with AHP	5086.96	56.921	931.05
	MOPSO-AHP	4905.4	49.37	936.24
160	NSGA-II with AHP	4569.75	71.409	819.66
	NSGA-II-AHP	4092.56	67.719	859.73
	MOPSO with AHP	5125.20	55.976	915.68
	MOPSO-AHP	4843.79	52.199	934.10
320	NSGA-II with AHP	4145.56	67.934	828.12
	NSGA-II - AHP	4005.05	60.63	889.25

	MOPSO with AHP	4857.51	57.496	912.96
	MOPSO - AHP	4801.62	53.432	906.05
640	NSGA-II with AHP	4232.74	67.663	862.91
	NSGA-II - AHP	4004.95	63.271	852.14
	MOPSO with AHP	4989.97	53.969	909.90
	MOPSO - AHP	4844.12	50.558	888.52
800	NSGA-II with AHP	4196.49	67.62	843.15
	NSGA-II - AHP	3946.19	63.77	833.88
	MOPSO with AHP	5042.77	50.531	847.38
	MOPSO- AHP	4683.68	51.386	901.25

6. Conclusion and Future Work

In this paper, we have addressed the problem of service composition in geographically distributed multi-cloud environment with SLA constraints. We present a service composition architecture and QoS models for Cloud Workflow application, which helps us to describe a service composition optimization problem. We have also proposed algorithms which apply the combination of multi-objective evolutionary approaches and Decision-Making method (AHP) to solve the composition optimization problem. Our algorithm beats normal evolutionary algorithms on the optimality and scalability. The results of the simulation indicate that our approach is able to precisely capture the preferences of users. In the current approach, the actual resources deployed in cloud environment are not considered. In our future work, we aim at optimizing the resource configuration via our service composition method. This will make our approach more practical and effective.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and 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, no.6, pp.599–616, Jun. 2009. Article (CrossRef Link).

[2] A. Amato, B. Di Martino, S. Venticinque, "Multi-objective Genetic Algorithm for Multi-cloud Brokering," *Lecture Notes in Computer Science,* pp.55-64, 2014. Article (CrossRef Link).

[3] Z. Ye, X.F. Zhou and Athman Bouguettaya, "Genetic Algorithm Based QoS-Aware Service Compositions in Cloud Computing," *Lecture Notes in Computer Science,* pp. 321-334, 2011. Article (CrossRef Link).

[4] G. Canfora, M. Di Penta, R. Esposito, M. Villani, "An approach for QoS-aware service composition based on genetic algorithms," in *Proc. of the 2005 Conference on Genetic and Evolutionary Computation*, pp. 1069–1075, 2005. Article (CrossRef Link).

[5] Y. Yao and H. Chen, "A Rule-Based Web Service Composition Approach," in *Proc. Of 2010 Sixth International Conference on Autonomic and Autonomous Systems*, pp.150-155, Mar. 2010. Article (CrossRef Link).

[6] J. Cao , X. Sun, X. Zheng, B. Liu, and B. Mao, "Efficient Multi-objective Services Selection Algorithm Based on Particle Swarm Optimization," in *Proc. of 2010 IEEE Asia-Pacific Services Computing Conference,* pp.603-608 ,Dec. 2010. Article (CrossRef Link)

[7] H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "Evolutionary deployment optimization for service-oriented clouds," *Software: Practice and Experience,* vol. 4, no.5 pp. 469 - 493, Mar 2011. Article (CrossRef Link).

[8] H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "E3: A Multiobjective Optimization Framework for SLA-Aware Service Composition," *IEEE Transactions on Services Computing,* vol. 5, no. 3, pp.358-371, 2012. Article (CrossRef Link).

[9] A. V. Dastjerdi, and R. Buyya, "Compatibility-aware Cloud Service Composition Under Fuzzy Preferences of Users," *IEEE Transactions on Cloud Computing*, vol.2, no.1, pp.1-13, 2014. Article (CrossRef Link).

[10] N. Palmer, "Workflow Management Coalition," *in SpringerReference,* 2009. Article (CrossRef Link).

[11] S. Pandey, "Scheduling and management of data intensive application workflows in grid and cloud computing environments," *[D] Melbourne: University of Melbourne*, Department of Computer Science and Software Engineering, 2011. Article (CrossRef Link).

[12] G.A. Klein, R. Calderwood, and Donald Macgregor, "Critical Decision Method for Eliciting Knowledge," *IEEE Transactions on Systems, Man, and Cybernetics*, vol.19, no.3, pp.462-472, 1989. Article (CrossRef Link).

[13] E. Triantaphyllou, and S. H. Mann, "using the analytic hierarchy process for decision making in engineering applications: some challenges," *Inter'l Journal of Industrial Engineering: Applications and Practice*, vol.2, no.1, pp. 35-44, 1995. Article (CrossRef Link).

[14] Z. Zou, Y. Yun and J. Sun, "Entropy method for determination of weight of evaluating in fuzzy synthetic evaluation for water quality assessment indicators," *Journal of Environmental Sciences,* vol.18, no.5, pp.1020-1023, 2006. Article (CrossRef Link).

[15] P.K. Sharma, A. Aggarwal, and R. Gupta, "A expert system for aid in material selection process," in *Proc. of Engineering Management Society Conference on Managing Projects in a Borderless World*, pp.27-31, 1993. Article (CrossRef Link).

[16] L. Liu, X. Yao, L. Qin, M. Zhang, "Ontology-based service matching in cloud computing," in *Proc. of Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on*, pp. 2544 – 2550, 2014. Article (CrossRef Link).

[17] M. Srinivas, L. Patnaik, "Genetic algorithms: A survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994. Article (CrossRef Link).

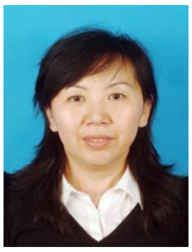
[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation,* vol. 6, no. 2, pp. 182 – 197, 2002. Article (CrossRef Link).

[19] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proc. of ICNN'95 - International Conferenceon Neural Networks*, pp. 1942 – 1948, 1995. Article (CrossRef Link).

[20] http://www.uoguelph.ca/~qmahmoud/qws/ Article (CrossRef Link)

[21] K. Deb, "Multi-objective Optimization Using Evolutionary Algorithms: An Introduction," *Multi-objective Evolutionary Optimization for Product Design and Manufacturing*, pp. 3-34, 2011. Article (CrossRef Link).

[22] A. J. Nebro, F. Luna, B. Dorronsoro, and B. Dorronsoro, "AAbYSS,Adapting Scatter Search to Multi-objective Optimization," *IEEE Transactions on Evolutionary Computation,* vol.12, no.4, pp.439-457, 2008. Article (CrossRef Link).



Li Liu received her Ph.D. degree from University of Science and Technology Beijing, China in 2006. Currently, she is an associate professor and a M. S. supervisor in School of Automation and Electrical Engineering, University of Science and Technology Beijing. Her research interests are cloud computing, service composition and optimization.
Email: liuli@ustb.edu.cn.

Miao Zhang received the B.E. degree in automation from School of Automation and Electrical Engineering, University of Science and Technology Beijing(USTB), China, in the year 2009. He is currently working toward the M.S. degree in USTB. His research interest includes service composition and resource scheduling in Cloud computing.
Email: s20130926@xs.ustb.edu.cn.