A New Multi-objective Evolutionary Algorithm for Inter-Cloud Service Composition

Li Liu[1], Shuxian Gu[1], Dongmei Fu[1], Miao Zhang[2], Rajkumar Buyya[3]
[1]School of Automation and Electrical EngineeringUniversity of Science and Technology Beijing, China
[2]School of Information and Electronics, Beijing Institute of Technology, Beijing, China
[3]The University of Melbourne, Australia
liuli@ustb.edu.cn
*Corresponding author: Li Liu

Received June 9, 2017; revised August 24, 2017; accepted September 15, 2017; published January 31, 2018

Abstract

Service composition in the Inter-Cloud raises new challenges that are caused by the different Quality of Service (QoS) requirements of the users, which are served by different geo-distributed Cloud providers. This paper aims to explore how to select and compose such services while considering how to reach high efficiency on cost and response time, low network latency, and high reliability across multiple Cloud providers. A new hybrid multi-objective evolutionary algorithm to perform the above task called LS-NSGA-II-DE is proposed, in which the differential evolution (DE) algorithm uses the adaptive mutation operator and crossover operator to replace the those of the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) to get the better convergence and diversity. At the same time, a Local Search (LS) method is performed for the Non-dominated solution set $F\{1\}$ in each generation to improve the distribution of the $F\{1\}$. The simulation results show that our proposed algorithm performs well in terms of the solution distribution and convergence, and in addition, the optimality ability and scalability are better compared with those of the other algorithms.

Keywords: Inter-Cloud; Service Composition; Multi-objective Evolutionary Algorithm; Differential Evolution Algorithm

1. Introduction

Cloud computing is a paradigm for providing web services [1]. However, with the recent growth of application requirements, the limited resources of the single Cloud have been unable to meet the needs of users. While the Inter-Cloud could provide a large number of Cloud service classes, each is composed of many service instances with the same function but different QoS levels. The Inter-Cloud has two paradigms: the first paradigm is that multiple Cloud providers (CPs) form an alliance to work together to provide services, and the second paradigm is that the users choose their own needs from the independent CP [2].

In order to solve the complex requirements of the users while satisfying the service level agreement (SLA), which is established through negotiation between users and CPs, most services have to be combined into a single service [3]. For example, a user who wants to perform the data analysis needs the data mining services (provided by Google), the data storage services (provided by Amazon EC2), and the data processing services (provided by Oracle).

However the service composition in the Inter-Cloud faces the challenges of identifying and selecting suitable services instances and service coordination optimization while considering how to reach high efficiency on the cost and response time, low network latency, and high reliability, especially when the service broker intends to deploy the application in multiple different geo-distributed CPs. It is difficult to find the optimal configuration scheme when the size of the search space grows exponentially with the increasing numbers of service instances. Meanwhile the service composition in the Inter-Cloud must optimize multiple different conflicting QoS objectives under several constraints at the same time, which increases the complexity.

In this paper, a new hybrid multi-objective evolutionary algorithm, LS-NSGA-II-DE, to optimize the Inter-Cloud service composition that faces the aforementioned challenges is proposed. Our main contributions can be summarized as follows:

▪ In our algorithm, the mutation and crossover operators of the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [4] are replaced by those of the adaptive differential evolution (DE) algorithm [5], which combines two mutation strategies and changes the evolutionary parameters adaptively in the process of generation. So we can search for the optimal solution more efficient in the solution space and make the algorithm has better convergence and diversity.

▪ The Local Search (LS) is performed for the Non-dominated solution set $F\{1\}$ in each generation.We set the critical distance changed adaptively according to the number of the individuals in $F\{1\}$ to improve the distribution of the $F\{1\}$ dynamically.

▪ We simulate the Inter-Cloud environment, where each CP provides the same or different service classes, and we consider the latency between the sequential service instances provided by the different CP.

The remainder of this paper is organized as follows: Section 2 introduces the related work about the service composition. Section 3 presents an overview of the service composition architecture, optimization objective and QoS model. Section 4 introduces the NSGA-II and DE algorithm and the realization of the LS-NSGA-II-DE algorithm. Section 5 and 6 analyzes the performance of the proposed algorithm compared with those of the single-objective and multi-objective evolutionary algorithms through experiments, which shows that our algorithm is effective and scalable. In the end, we conclude this paper and describe the future work in Section 7.

2. Related Work

In recent years, service composition and resource schedule in cloud computing have been well studied. Some evolutionary algorithms have been applied to the service composition and resource schedule problem.

In [6] and [7], the authors use a single-objective genetic algorithm (GA) to solve the service composition problem. Yu et al. [8] present an ant colony optimization algorithm for web service composition (ACO-WSC), which attempts to select CP combinations that are feasible and use the minimum number of the CPs. A traffic engineering based adaptive approach to maximize the energy-efficiency for Real-Time-Service Data Centers is proposed in [9], which improves the energy consumption of the servers by 25% compared to the state of the art improvement on average in the entire data center. This paper [10] proposes and tests an energy-efficient adaptive resource scheduler, which is capable to effectively adapt to both synthetic and the networked Fog platform with real world input traffic, various mobility conditions and settings. The authors in [11] present a hybrid approach called FUGE that is based on GA and fuzzy theory that aims to settle the job scheduling in cloud computing, which improves over 45% in terms of execution time, cost and average degree of imbalance than standard GA. However, they all optimize each objective individually or assign the weights to each objective to solve the multi-objective problems, while they can’t optimize the multi-objective simultaneously and the weight parameter is not easy to determine, which may lose many excellent solutions. So the optimization capacity of the above algorithms is not good enough compared with multi-objective algorithms.

In [12] and [13], two popular multi-objective evolutionary algorithms are presented to solve the QoS-aware service composition problem, namely, MOPSO (Multi-Objective Particle Swarm Optimization) and NSGA-II. The authors in [14] propose a new multi-objective genetic algorithm, which uses the approach of data dimension reduction to reduce the number of optimization objectives and decrease the complexity of the problem. The authors in [15] propose a fuzzy multi-objective genetic algorithm (FMOGA) based on fuzzy QoS attributes in which the fuzzy weights are obtained through the Fuzzy Analytical Hierarchy Process (FAHP) method. Thus it can express the user's QoS preferences to achieve the QoS-based Cloud service composition. The paper in [16] utilizes a combination of the multi-objective evolutionary algorithm and AHP, which is applied to the Crowding distance calculation, to solve the Cloud service composition optimization problem. Thus, it can precisely meet the users’ preferences. However, the above algorithms do not simulate the Inter-Cloud service composition very well, and the distribution of the Pareto front of these algorithms is not good enough. In this paper, we assume that the CPs provide the same or different service classes, and we consider the latency between the sequential service instances provided by the different CP, which can simulate the true Inter-Cloud environment well. Additionally, we use LS for the Non-dominated solution set $F\{1\}$ in each generation to improve the distribution.

3. Service Composition Model

Before the introduction of the system models and the proposed methods, we first provide the definitions of the terms and notions in **Table 1** for future references.

Table 1. The Notation Description

Notation	Description	Notation	Description
T	Response time	NDR	The Non-dominated rank
R	Reliability	Q	The normalized QoS value
C	Cost	q	The non-normalized QoS value
M	The number of the optimization objectives ($m \in (1,2,\dots,M)$)	$F\{r\}$	The set of the individuals whose NDR is r
N	The size of the population	CD	The Crowding distance
pop_G	The population in the G-th generation	F	The Mutation probability of the LS-NSGA-II-DE
$x_{i,G}$	The i-th individual in pop_G. ($pop_G = [x_{1,G}.., x_{i,G}, \dots, x_{N,G}]$)	CR	The Crossover probability of the LS-NSGA-II-DE
G_{max}	The max number of the generation G		

3.1 Service Composition Architecture

The architecture of the service composition is shown in **Fig. 1**, The user submits a service composition requirement, which requires 10 service classes S= $\{S^1, S^2, S^3, S^4, S^5, S^6, S^7, S^8, S^9, S^{10}\}$ to complete the user's tasks. They are provided by 10 different CPs and different CP can provide the same or different service classes, Such as CP1 can provide 4 service class: S^1,S^3,S^6,S^{10}, CP2 can provide 3 service class S^2,S^4,S^7,…, CP10 can provide 4 service class: S^3,S^6,S^7,S^9. Each service class provided by each CP contains 10 service instances that have the same functionality but different QoS levels. We can select the service instances from the different CPs. For example, we can select the service instance of the service class S^1 from four CPs: CP1, CP3, CP6 and CP9.

The Cloud service coordinator [17] has four service components (shown in **Fig. 1)**, and it acts as an independent broker, which can help the users select and compose the service instances. First, the Service Discovery identifies all of the available service instances according to the specific functional requirements. Next, the Capability Planning evaluates the QoS capability of the candidate service instances. After then, the Service Selection uses the optimization method to choose the most appropriate CP and service instances according to the non-functional requirement. Finally, the Service Composition combines the selected service instances to form the service sequence, for example the service sequence $(s^1_{1,3}, s^2_{2,1}, s^3_{7,2}, s^4_{8,7}, s^5_{9,4}, s^6_{3,9}, s^7_{5,8}, s^8_{7,5}, s^9_{10,3}, s^{10}_{4,6})$, to satisfy the user's QoS requirements [18]. Where $s^9_{10,3}$ represents the third service instance of the 9-*th* service class provided by the 10-*th* CP.

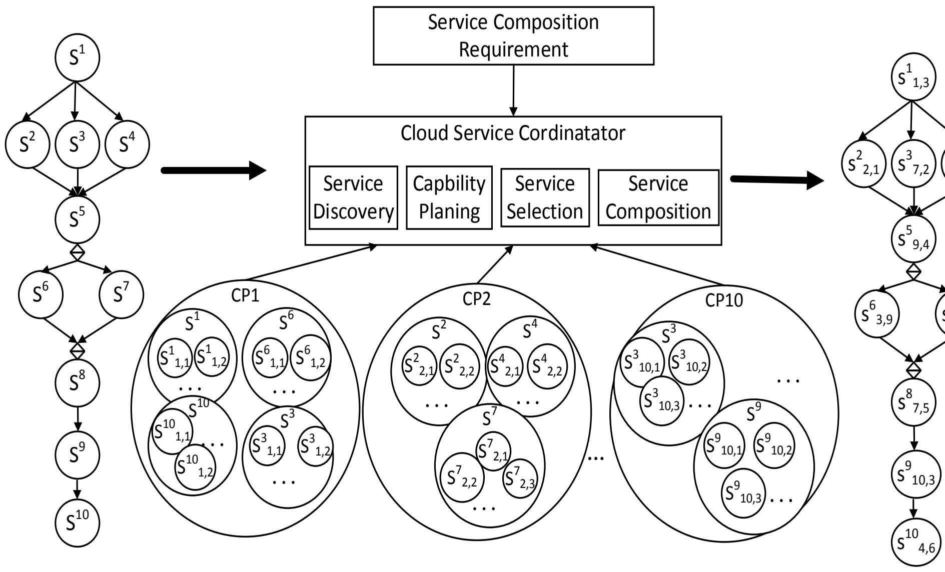


Fig. 1. The architecture of the Service Composition in the Inter-Cloud

3.2 Optimization Objective

The optimization objective of the service composition is to find the solution that has better performance and meet the user's QoS requirements over a vast search space. The common SLA for service composition are defined as QoS criteria. Here, we optimize three objectives, which are to minimize both the cost C ($) and response time T (ms) and maximize the reliability R (%) while satisfying the user's constraints $T_0 = 2000ms,\ C_0 = 800\$$, and $R_0 = 15\%$, which obtained by several experiments. A penalty function is used to optimize the constrained problem.

$$\begin{cases} Minimize \quad T \\ Minimize \quad C \\ Maximize \quad R \\ Subject\ to\colon\ T \le T_0, C \le C_{0,} R \ge R_0 \end{cases} \tag{1}$$

The reliability is the positive criteria, its increase is beneficial for users, while the response time and cost is negative criteria, its decrease is well. The optimization process aims to promote the increase of the positive criteria and the decrease of negative criteria.

To ensure that all of the objectives are converted to the minimized problems and all between [0,1], the QoS criteria need to be normalized first. The normalization formulations of negative and positive criteria are as Eq.2 and Eq.3, respectively. Where the $q_{x_{i,G}}^m$ and $Q_{x_{i,G}}^m$ are the non-normalized and normalized QoS value of the individual $x_{i,G}$ for the objective m, the q_{max}^m and q_{min}^m are the max and min non-normalized QoS values for the objective m.

$$Q_{x_{i,G}}^m = (q_{x_{i,G}}^m - q_{min}^m)/(q_{max}^m - q_{min}^m) \tag{2}$$

$$Q_{x_{i,G}}^m = (q_{max}^m - q_{x_{i,G}}^m)/(q_{max}^m - q_{min}^m) \tag{3}$$

3.3 QoS Model

To obtain the objective value of the composed service instances, it is required to calculate its end-to-end QoS value by aggregating QoS measures. There are three types of composition structures in **Fig. 1**: sequential, parallel and conditional, which are calculated by **Table 2** [16], where $S = \{S^1, \dots, S^k, \dots, S^K\}$ are the composed service classes set, $1 \leq k \leq K$. T_k is the response time, R_k is the reliability, C_k is the cost, and p_k is the probability for the service instance of the k-th service class S^k. Because the service instances could come from different geo-distributed CPs, the data transmission between the sequential service instances will have a network latency, which makes the response time T contain T_k and the network latency L_k. The latency between each CP is generated randomly from 100ms to 200ms, where if two sequential service instances are provided by the same CP, then the latency between them is 0.

Table 2. The QoS aggregate functions

QoS Criteria	**Sequential**	**Conditional**	**Parallel**
Response Time	$T = \sum_{k=1}^{K} T_k + \sum_{k=2}^{K} L_k$	$T = \sum_{k=1}^{K} p_k * T_k$	$T = max_{k=1}^{K} T_k$
Reliability	$R = \prod_{k=1}^{K} R_k$	$R = \sum_{k=1}^{K} p_k * R_k$	$R = min_{k=1}^{K} R_k$
Cost	$C = \sum_{k=1}^{K} C_k$	$C = \sum_{k=1}^{K} p_k * C_k$	$C = \sum_{k=1}^{K} C_k$

4. Multi-objective Evolutionary Algorithm

A new hybrid multi-objective evolutionary algorithm with local search LS called LS-NSGA-II-DE is proposed based on the NSGA-II and the adaptive DE algorithm to solve the service composition problem in Inter-Cloud. Some of the definitions applicable to our multi-objective algorithm are as follows, and the process of the NSGA-II and DE algorithm is introduced in section 4.1 and 4.2.

Definition 1 (Pareto optimal solution): The dominance rule means that if there are two solutions x_1 and x_2 for the minimization problem, if the fitness meets the condition: $\forall m \in (1,2,\dots,M)$: $f_m(x_1) \leq f_m(x_2) \wedge \exists m \in (1,2,\dots,M)$: $f_m(x_1) < f_m(x_2)$, then x_1 dominates x_2. If x_1 is not dominated by any other solutions, then x_1 is a Non-dominated solution, which is also known as the Pareto optimal solution.

Definition 2 (Non-dominated set and Pareto front): The set $F\{1\}$ composed of the Pareto optimal solutions is called the Non-dominated set. The surface that is formed by Pareto optimal solutions is called the Pareto front. The Pareto front solutions cannot be dominated by the others within or out of its surface.

4.1 The Process of the NSGA-II

The NSGA-II has three important steps: the Non-dominated Sorting, Crowding distance calculation and Elitist strategy.

Non-dominated Sorting: According to the dominance rule in Definition 1, we assign the Non-dominated rank (*NDR*) to each individual. The lower rank has the priority to enter to the next generation.

Fig. 2 is an example of the Non-dominated Sorting with two QoS optimization objectives: minimizing response time and cost. We assign the Non-dominated rank *NDR* to six individuals. The individuals *A*, *B* and *C* are in the first rank, which are the Pareto optimal solutions. Individuals *D* and *E* are in the second rank because they are dominated only by the individuals in the first rank. At the last, individual *F* is located at the third rank.

Algorithm 1 gives the process of the Non-dominated sorting, here the pop_G,$x_{i,G}$,Q, NDR and $F(r)$ have been illustrated in **Table 1**.

Algorithm1: ***Non-domination Sorting***

```
Input: $pop_G$; Output: $NDR_{pop_G}$, $F\{r\}$,
1: Initialize $r = 1$;
2: while size($pop_G$)≠0 do // Loop until all individuals have been assigned $NDR$
3:    for i=1:size($pop_G$) do
4:       for j=1:size($pop_G$) do
5:            dom_less=0;dom_equal=0;
6:            for m=1:M do
7:                 if $Q^m_{x_{i,G}} < Q^m_{x_{j,G}}$ then
8:                      dom_less= dom_less+1;
9:                 else $Q^m_{x_{i,G}} = Q^m_{x_{j,G}}$ then
10:                      dom_ equal = dom_ equal +1;
11:                end if
12:            end for
13:            $F\{r\} = []$;
14:            if dom_less=0 & dom_ equal≠M then
15:                 dominated_number ($x_{i,G}$) =dominated_number ($x_{i,G}$) +1; //Calculate the number of individuals who dominate $x_{i,G}$
16:            end if
17:      end for
18:      if dominated_number($x_{i,G}$)=0 then
19:            $NDR_{x_{i,G}}$=r;$F\{r\} = [F\{r\},\ x_{i,G}]$; //Join the individual $x_{i,G}$ into $F\{r\}$
20:      end if
21:   end for
22:   $pop_G = pop_G - F\{r\}$; // The individuals who have not been ranked
23:   r=r+1;
24: end while
```

Crowding distance Calculation: In the process of optimization, excellent individuals will occur increasingly more and more, and thus, the algorithm must determine which individuals can enter to the next generation. The Crowding distance (*CD*) is used to choose the individuals who have the same *NDR*. The basic idea of the *CD* is to calculate the Euclidean distance between the two neighbors of the individual with the same *NDR* for all of the QoS objectives. The *CD* of the two boundary individuals is set to infinity [4]. The individual with a greater *CD* will be selected for the next generation when two individuals have the same *NDR*.

The *CD* is calculated by Eq. 4. Where the $F\{r\}_i$ means the *i*-th individual in the $F\{r\}$. For example if $F\{r\} = [x_{1,G}, x_{3,G}, x_{5,G}, x_{8,G}]$, then the $F\{r\}_3$ is the individual $x_{5,G}$. The $CD^m_{F\{r\}_i}$ means the *CD* of the individual $F\{r\}_i$ for the objective m, and the $CD_{F\{r\}_i}$ is the sum of $CD^m_{F\{r\}_i}$ for all of the objectives. The Q^m_{max} and Q^m_{min} are the max and min normalized QoS values for the objective m. Algorithm 2 gives the calculation process of the *CD* of the population pop_G.

$$CD^m_{F\{r\}_i} = \sqrt{\left(\frac{Q^m_{F\{r\}_{i+1}} - Q^m_{F\{r\}_{i-1}}}{Q^m_{max} - Q^m_{min}}\right)^2} \tag{4}$$

Algorithm 2: ***Crowding Distance Calculation***

```
Input: $pop_G, F\{r\}$; Output: $CD_{pop_G}$
1: Initialize $r = 1$, $CD_{pop_G}=0$;
2: while size($pop_G$)$\neq 0$ do
3:   $s$=size($F\{r\}$);
4:   for $m = 1{:}M$ do
5:       ($Q^m_{F\{r\}}$,index)= sort($Q^m_{F\{r\}}$,'descend'); //sort $F\{r\}$ in the descending order
6:       $F\{r\}'$= $F\{r\}$(index); //Obtain the $F\{r\}'$ after the sort
7:       $CD^m_{F\{r\}'_1} = \infty$; $CD^m_{F\{r\}'_s} = \infty$;// The CD of the two boundary individuals is set to infinity
8:       for $i$=2:(s-1) do
9:           Calculate $\mathrm{CD}^m_{F\{r\}'_i}$ according to Eq.4;
10:      end for
11:      for $i$=1:s do
12:          $\mathrm{CD}_{F\{r\}_i}=CD_{F\{r\}_i}+CD^m_{F\{r\}_i}$;
13:      end for
14:  end for
15:  $pop_G = pop_G - F\{r\}$; // The individuals whose CD have not been calculated
16:  $r=r+1$;
17: end while
```

Elite strategy: By using the Elite strategy, the NSGA-II can choose better individuals to be retained into the next generation. First, the population P_G is composed of the parent $P1_G$ and the offspring $P2_G$. The size of the population P_G is *2N*. Then, the P_G is sorted to obtain the $F\{1\}$, $\mathrm{F}\{2\}$ and the other solution sets according to the *NDR* and *CD*. The solutions in $F\{1\}$ are the optimal solutions. If the number of the solutions in the $F\{1\}$ is less than *N*, then we choose all of the solutions in the $F\{1\}$ for the next population P_{G+1}. The remaining solutions will be chosen from the solution set $F\{2\}$, $F\{3\}$, etc. until the size of the P_{G+1} is equal to *N* [4]. **Fig. 3** gives the process of the Elite strategy.

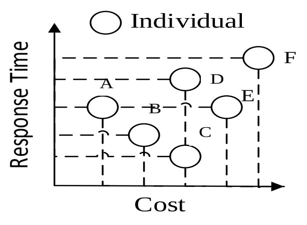


Fig. 2. An example of the Non-dominated Sorting

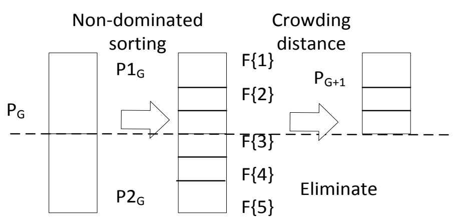


Fig. 3. The process of the elite strategy

4.2 The DE Algorithm

The optimization problem to be solved is $min\,f(x)$, and the process of the DE algorithm is as follows:

Step1: Encoding and initialization: The DE algorithm uses the real coding, and the population size is N, i.e., $pop_G = [x_{1,G}, \dots, x_{i,G}, \dots, x_{N,G}]$. The number of the parameters of $x_{i,G}$ is D, i.e., $x_{i,G} = [x^1_{i,G}, \dots, x^j_{i,G}, \dots, x^D_{i,G}]$, where $x^j_{i,G}$ refers to the j-th parameter of individual $x_{i,G}$. The bounds of $x^j_{i,G}$ are $x^j_{min} \le x^j_{i,G} \le x^j_{max}$. Each parameter of each individual at generation $G = 1$ is generated by Eq. 5:

$$x_{i,1}^{j} = rand(0,1) \times \left(x_{max}^{j} - x_{min}^{j}\right) + x_{min}^{j} \tag{5}$$

Step2: Individual evaluation: Calculate the fitness values $f(x_{i,G})$ of the individual $x_{i,G}$.

Step3: Mutation: Obtain the mutated individual $v_{i,G}$ by performing Eq. 6 for the original individual $x_{i,G}$ [19]. Where the mutation probability factor $F \in [0,2]$, $r1, r2, r3 \in \{1,2,\dots,N\}$ are randomly generated and $r1 \neq r2 \neq r3 \neq i$.

$$v_{i,G} = x_{r1,G} + F \times \left(x_{r2,G} - x_{r3,G}\right) \tag{6}$$

Step 4: Crossover: The j-th parameter of the crossover individual $u_{i,G}$ can be obtained by using the crossover operation between the j-th parameter of $x_{i,G}$ and $v_{i,G}$ by Eq. 7 Where the crossover probability factor is $CR \in [0,1]$, $rand$ is a random number between [0 1], and $\alpha \in (1,2,\dots,D)$ is a random number to ensure that the crossover individual $u_{i,G}$ has at least one parameter that can be obtained from the mutation individual $v_{i,G}$.

$$u_{i,G}^{j} = \begin{cases} v_{i,G}^{j} & if\ \ (rand \leq CR)\ or\ j = \alpha \\ x_{i,G}^{j} & if\ \ (rand > CR)\ and\ \ j \neq \alpha \end{cases} \tag{7}$$

Step5: Selection: The selection operation determines which will enter into the next generation by comparing the fitness value of the $x_{i,G}$ and the fitness value of the $u_{i,G}$

$$x_{i,G+1} = \begin{cases} u_{i,G} & if\ f(u_{i,G}) \leq f(x_{i,G}) \\ x_{i,G} & if\ f(u_{i,G}) > f(x_{i,G}) \end{cases} \tag{8}$$

Step6: Terminal test: If the test reaches the maximum number of generations G_{max}, output the optimal solution; otherwise, turn to Step 2.

4.3 The LS-NSGA-II-DE Algorithm

The LS-NSGA-II-DE algorithm is designed by the combination of the NSGA-II, adaptive DE algorithm and LS.

NSGA-II: The LS-NSGA-II-DE algorithm retains the ability of the NSGA-II, i.e., the fast Non-dominated Sorting, Crowding distance calculation and elite strategy, which improves the efficiency of the optimization and ensures the diversity of the solutions.

The adaptive DE algorithm: The DE algorithm has the advantage to search for the optimal solution in the solution space more efficient. So we use the adaptive mutation and crossover operators of the DE algorithm instead of the those of the NSGA-II [20].

(1) Considering that the DE algorithm has many different mutation strategies [21]. Where DE/rand/1/bin in Eq. 9 has strong global search ability but a slow convergence speed and the DE/best/1/bin in Eq. 10 has a fast convergence speed but could easily fall into a local optimum [22], we combine their better characteristics by Eq. 11. Where $\beta = G/G_{max}$. β changes from 0 to 1 in the process of the generation, thus the proportion of $x_{r1,G}$ decreases gradually, while the proportion of $x_{best,G}$ increases gradually, which causes the diversity and convergence speed of the algorithm to be balanced and enhanced.

DE/rand/1/bin: $$v_{i,G} = x_{r1,G} + F \times \left(x_{r2,G} - x_{r3,G}\right) \tag{9}$$

DE/best/1/bin: $$v_{i,G} = x_{best,G} + F \times \left(x_{r1,G} - x_{r2,G}\right) \tag{10}$$

$$v_{i,G} = (1-\beta) \times x_{r1,G} + \beta \times x_{best,G} + F \times \left(x_{r2,G} - x_{r3,G}\right) \tag{11}$$

(2) Simultaneously, we use the adaptive mutation and crossover operators, as shown in Eq. 12 and Eq. 13. Where $F_{max} = 0.9$, $F_{min} = 0.4$, $CR_{max} = 0.9$ and $CR_{min} = 0.3$ [23]. The values of the F and CR become smaller in the process of the generation. At the beginning of the generation, the F and CR values are large, which can improve the diversity to avoid premature. At a later time, the F and CR values are smaller, which is helpful to converge faster to the optimal solution.

$$F = F_{max} - (F_{max} - F_{min}) \times \frac{G}{G_{max}} \tag{12}$$

$$CR = CR_{max} - (CR_{max} - CR_{min}) \times \frac{G}{G_{max}} \tag{13}$$

Local Search LS: The NSGA-II uses the Elitist strategy to select the individual which has a lower *NDR* and a higher *CD* for the next generation. However, the distance of two individual with a larger *CD* may be not larger, so an individual with a poor distribution can be retained, as illustrated in **Fig. 4 (a).** The *CD* of the individuals b and c is larger, and thus, they could be retained together, but the distance between them is small; thus, the ideal situation should retain only one of them. Individuals e and f are the same case.

Next, we will use the LS to improve the distribution of the $F\{1\}$ in the NSGA-II-DE algorithm. The LS involves three important steps: The calculation of the critical distance, the determination of the adjacent individual and eliminating an adjacent individual.

Step1: The calculation of the critical distance: First, the Non-dominated set $F\{1\}$ is sorted by one of the objectives $m\,(m \in (1{:}M))$ in a descending order. Then, the critical distance is calculated by Eq. 14. Where the distance $dist_{i,j}$ between two individuals $x_{i,G}$ and $x_{j,G}$ is calculated by Eq. 15. $|F\{1\}|$ is the number of the Non-dominated individuals in the current generation, and $dist_{1,n}$ is the distance between the two boundary individuals $F\{1\}_1$ and $F\{1\}_n$. In the evolutionary process, δ will be changed adaptively according to the $|F\{1\}|$, which can ensure to maintain the uniformity of the Pareto front dynamically.

$$\delta = dist_{1,n}/(2 \times (|F\{1\}| - 1)) \tag{14}$$

$$dist_{i,j} = \sum_{m=1}^{M} \sqrt{\left(Q^{m}_{F\{1\}_i} - Q^{m}_{F\{1\}_j}\right)^2} \tag{15}$$

Step2: Determination of the adjacent individuals: If the distance between the individuals $F\{1\}_i$ and $F\{1\}_{i+1}$ is equal to or less than δ, i.e., $dist_{i,i+1} \le \delta$, then the individuals $F\{1\}_i$ and $F\{1\}_{i+1}$ are the adjacent individuals.

Step3: Eliminating one adjacent individual: First, we will use the individuals $F\{1\}_{i-1}$ and $F\{1\}_{i+2}$ to generate a new individual $x_{j,G}$ according to Eq. 16, which is in the middle between $F\{1\}_{i-1}$ and $F\{1\}_{i+2}$. Then, we eliminate one individual of the $F\{1\}_i$ and $F\{1\}_{i+1}$ which is far from the $x_{j,G}$ and retain another individual. Algorithm 3 gives the process of the LS.

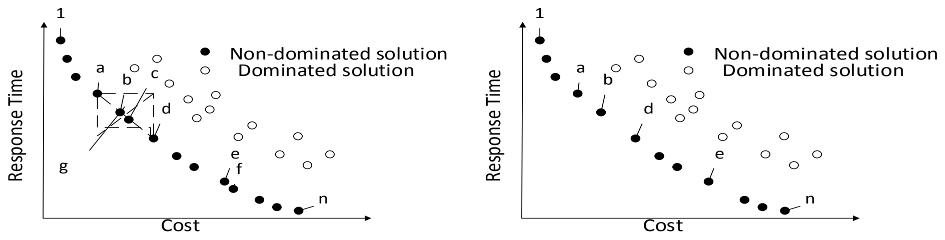
$$Q_{x_{j,G}} = 0.5 \times Q_{F\{1\}_{i-1}} + 0.5 \times Q_{F\{1\}_{i+2}} \tag{16}$$

Algorithm 3 The Process of the LS

Input:The $F\{1\}$; **Output:** The $F\{1\}$ *after performing the LS*

```
1: Calculate $\delta$ according to Eq. 14;
2: ($Q^m_{F\{1\}}$, index)= sort($Q^m_{F\{1\}}$,'descend');//Sort the $F\{1\}$
3: $F\{1\} = F\{1\}(index)$; // Get the $F\{1\}$ after the sort
4: for $i = 2{:}(size(F\{1\}) - 2)$ do // $F\{1\}$ perform the LS
5:   if $dist_{i,i+1} \le \delta$ then //Find the adjacent individuals
6:       $Q_{x_{j,G}} = 0.5 \times Q_{F\{1\}_{i-1}} + 0.5 \times Q_{F\{1\}_{i+2}}$;
7:       if $dist_{i,j} \ge dist_{i+1,j}$ then
8:          $F\{1\} = F\{1\}[1{:}i-1 \;\; i+1{:}size(F\{1\})]$; //remove the individual $F\{r\}_i$
9:       else $dist_{i,j} < dist_{i+1,j}$ then
10:         $F\{1\} = F\{1\}[1{:}i \;\; i+2{:}size(F\{1\})]$; //remove the individual $F\{r\}_{i+1}$
11:      end if
12:   end if
13: end for
```

According to the process of LS, there are two pairs of adjacent individuals who meet the LS rules that can be found in **Fig. 4 (a)**, i.e., (b,c) and (e,f) Then, individuals c and f are removed. The result of the improved distribution is shown in **Fig. 4 (b)**.



(a) Before the LS (b) After the LS

Fig. 4. The individual distribution of the population

The process of the LS-NSGA-II-DE is as follows:

Step1: Initialize a population pop_G that includes N individuals.

Step2: Calculate the fitness and then calculate the Non-dominated rank NDR and the Crowding distance CD of each individual according to the fitness.

Step3: Sort the population, and select the optimal individual based on the NDR and CD

Step4: The differential mutation and crossover operations are performed on the parent population to generate the offspring population, which will combine the parent population to calculate the NDR and CD.

Step5: Sort the combined population based on the NDR and CD. Only the first N individuals can enter into the next generation according to the elite strategy.

Step6: The $F\{1\}$ of the population in the current generation will perform the LS.

Step7: If the generation G reaches G_{max}, output the population, otherwise turn to step 2.

5. Experiments on Multi-objective Benchmark Functions

To evaluate the optimization performance of our proposed algorithm, experiments have been conducted on multi-objective benchmark functions. The experimental environment is the Inter Core CPU i5-4570S (2.9 GHz and 8G RAM).

5.1 The Multi-Objective Benchmark Functions

We select 7 different widely used multi-objective benchmark functions, which are listed in **Table 3** [24], including ZDT1, ZDT3 and UF2 which are unconstrained problems, and Binh2, Srinivas, F1 and CTP1 which are constrained problems. Among all of these 7 functions, UF2, F1 and CTP1 have a much more complicated search space.

Table 3. Description of Benchmark Functions

Function	Nature	Variable	Objectives	Pareto shape
ZDT1	Unconstrained	30	2	Convex
ZDT3	Unconstrained	30	2	Discontinuous
UF2	Unconstrained	10	2	Convex
Binh2	Constrained	2	2	Convex
Srinivas	Constrained	2	2	Concave
F1	Constrained	10	2	Linear
CTP1	Constrained	2	2	Convex

5.2 Compared Algorithms

We evaluate the proposed algorithm LS-NSGA-II-DE and compare it with three common algorithms, NSGA-II [4], MOEA/D [25] and MOPSO [26]. To intuitively show the differences in the distribution, the parameters are set as follows: the size of the population is 50, the max generation is 200.The crossover probability p_c and mutation probability p_m of the NSGA-II and MOEA/D are set to 0.8 and 0.2, respectively, and the external file's size of the MOEA/D is set to 20. The initial learning rate of the MOPSO is c1=c2=2, its inertia weight uses the adaptive method, which is calculated by Eq. 17. Here, w_{max} and w_{min} are set to 0.9 and 0.4 [27].

$$w = w_{max} - (w_{max} - w_{min}) \times \frac{G}{G_{max}} \tag{17}$$

5.3 Evaluated Method

The multi-objective algorithm is aimed at making the Pareto front converge quickly to the true Pareto front (obtained by the exhaustive method) and has a well uniform distribution. Thus, we use the inverted generational distance IGD [28] and Spread to quantify the convergence and uniformity of the Pareto front obtained by the algorithm [29].

IGD is calculated by Eq. 18 and is the average distance between the obtained Pareto front and the true Pareto front, which represents the convergence of the algorithms.

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{18}$$

Where d_i is the Euclidean distance between the obtained solution and the closest solution of the true Pareto front. Here, n is the number of obtained solutions. A smaller IGD indicates that the achieved Pareto front converges to the true Pareto front better [28].

The Spread calculated by Eq. 19 represents the diversity of the Pareto front solutions:

$$Spread = \frac{(d_f + d_l + \sum_{i=1}^{n} |d_i - \bar{d}|}{d_f + d_l + (n-1) \times \bar{d}}) \tag{19}$$

Where d_i and n are the same as in IGD. Here, d_f and d_l are the Euclidean distance of the boundary solutions in the obtained Pareto front set. A smaller Spread means that the solutions perform well in terms of their diversity [29].

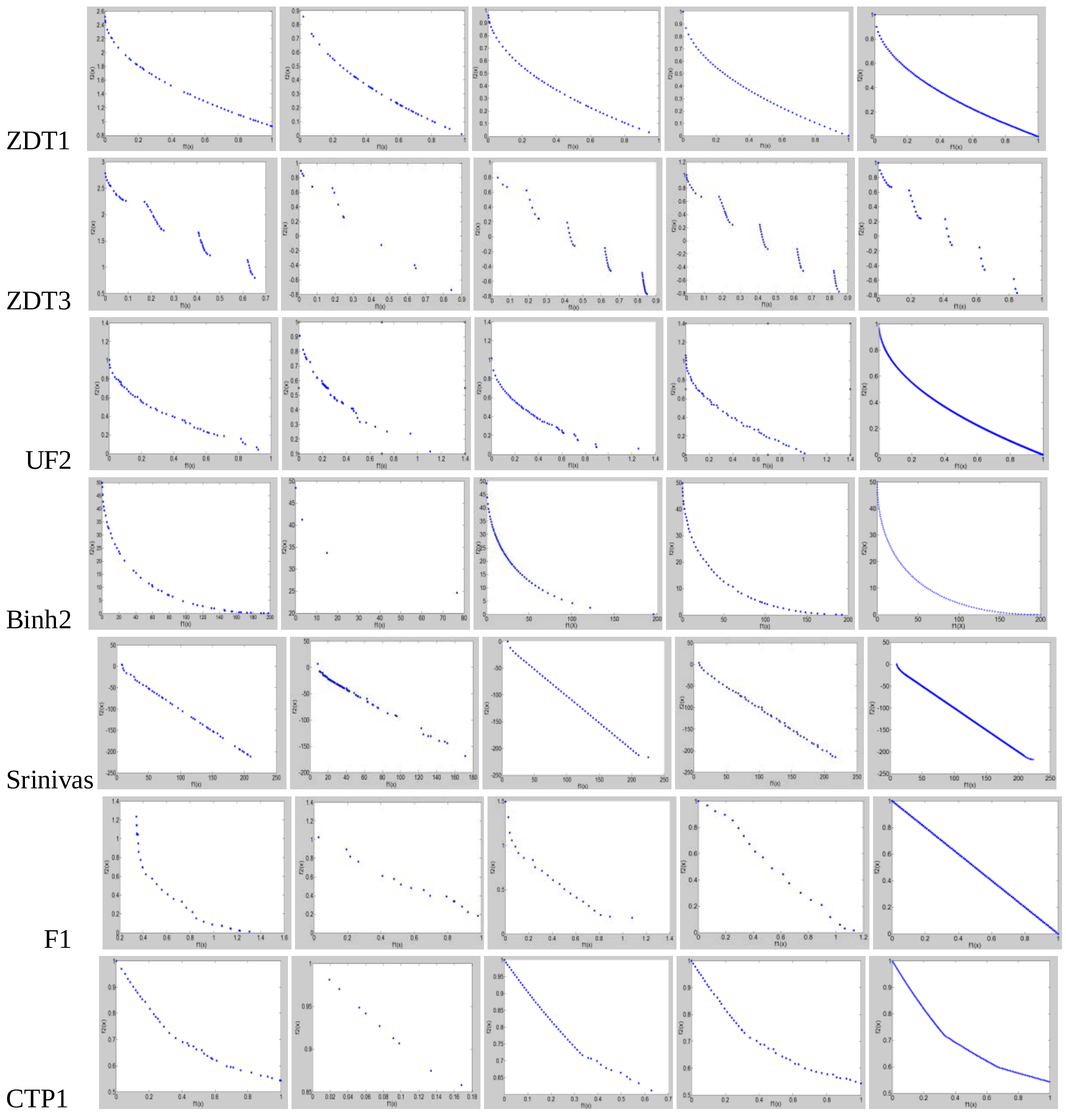
5.4 Results and Analysis

Fig. 5 shows the Pareto front obtained by the four algorithms and the true Pareto front under the 7 benchmark functions. **Table 4** gives the IGD and Spread values obtained by the four algorithms through 30 independent experiments on the 7 benchmark functions. The best values of the IGD and the Spread among these algorithms are highlighted with green and red bold fonts, respectively, for each function. The LS-NSGA-II-DE obtains 6 best values of Spread and 3 best values of IGD. The MOEA/D obtains 1 best values of Spread and 2 best values of IGD. The MOPSO and NSGA-II each obtain 1 best value of IGD.

Therefore, it can be intuitively seen in **Fig. 5** and **Table 4** that the LS-NSGA-II-DE's distribution is better than those of the other algorithms in most cases, and it can converge well to the true Pareto front. Even though the other compared algorithms could obtain satisfactory results for some functions, they do not distribute and converge to the Pareto front well when the search space becomes complicated, such as the UF2, F1 and CTP1.

Table 4. The IGD and Spread obtained byfour different algorithm

Algorithm		**Unconstrained functions**			**Constrained functions**			
		ZDT1	**ZDT3**	**UF2**	**Binh2**	**Srinivas**	**F1**	**CTP1**
NSGA-II	IGD	1.64e-2	6.79e-2	**1.48e-2**	1.16e-1	1.63e-1	4.98e-2	6.42e-3
	Spread	4.14e-1	5.98e-1	4.80e-1	6.48e-1	5.30e-1	3.11e-1	5.94e-1
MOPSO	IGD	**8.14e-3**	2.88e-1	2.02e-2	1.94e-1	1.59e-1	6.65e-2	1.49e-2
	Spread	7.76e-1	8.16e-1	1.75e-0	1.41e-0	1.04e-0	4.39e-1	9.00e-1
MOEA/D	IGD	9.24e-3	**1.94e-2**	1.79e-2	**8.03e-2**	1.67e-1	3.28e-2	5.58e-3
	Spread	2.77e-1	7.15e-1	7.52e-1	9.90e-1	**9.77e-2**	3.65e-1	4.69e-1
LS-NSGA-II-DE	IGD	8.63e-2	2.45e-2	1.68e-2	8.78e-2	**1.43e-1**	**2.84e-2**	**5.01e-3**
	Spread	**2.52e-1**	**3.21e-1**	**2.99e-1**	**4.80e-1**	1.62e-1	**2.24e-1**	**3.13e-1**



(a)NSGA-II (b) MOPSO (c) MOEA/D (d) LS-NSGA-II-DE (e) True Pareto

Fig. 5. The Pareto front obtained by four different algorithm for Benchmark Functions

6 Experiments on Service Composition

To evaluate the optimization performance of the algorithm, we also conduct the experiment on service composition. The experimental environment is the Inter Core CPU i5-4570S (2.9 GHz and 8G RAM). The data of the web services are obtained from the QWS Dataset (2.0) [30]. The simulated problem of the service composition is shown in Section 3.

6.1 Chromosome Encoding

Fig. 6 is the chromosome encoding, which uses the real encoding according to the service composition in Section 3. Each pair of chromosomes represents a service instance of a service class. The data in **Fig. 6** represents the max value, such as the range of the service instances of S^1 is 0-39, which represents 40 service instances from 4 CPs, and so on.

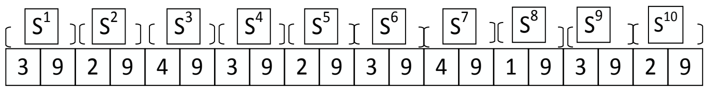


Fig. 6. Chromosome Encoding

6.2 The Compromise Solution

The multi-objective algorithms can obtain the optimal Pareto set, but the user only need one best solution. To solve the problem, we need to select a compromise solution from the Pareto solutions that can best satisfy the user's need. In this paper, the fuzzy set theory [31] is adopted for the multi-objective decision making. The fuzzy function for the minimization problem in Eq. 20 is used to express the satisfaction of each optimal solution.

$$u^m_{x_{i,G}} = \begin{cases} 1 & Q^m_{x_{i,G}} = Q^m_{\min} \\ \dfrac{(Q^m_{\max} - Q^m_{x_{i,G}})}{(Q^m_{\max} - Q^m_{\min})} & Q^m_{\min} < Q^m_{x_{i,G}} < Q^m_{\max} \quad m = 1, 2, .., M \\ 0 & Q^m_{x_{i,G}} = Q^m_{\max} \end{cases} \tag{20}$$

Where the $u^m_{x_{i,G}}$ is the satisfaction of the individual $x_{i,G}$ for the objective m. $u^m_{x_{i,G}} = 0$ means that it's totally dissatisfied, $u_{x_{i,G}} = 1$ means that it's totally satisfied. The average satisfaction for all the objectives is calculated by Eq. 21 .We choose the individual which has the max average satisfaction to be the compromise solution.

$$\tilde{u}_{x_{i,G}} = \frac{1}{M}\sum_{m=1}^{M} u^m_{x_{i,G}} \tag{21}$$

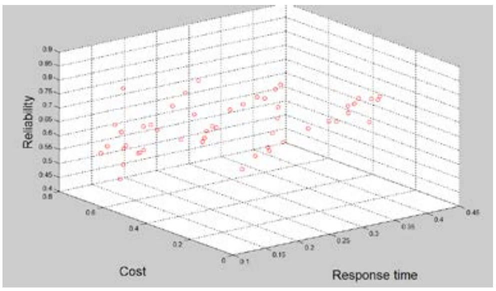
6.3 Results and Analysis

To validate the performance of the proposed algorithm for the service composition, we compare the LS-NSGA-II-DE with the NSGA-II, MOEA/D and MOPSO. The parameters for all of the algorithms are set the same as the experiment on benchmark functions.

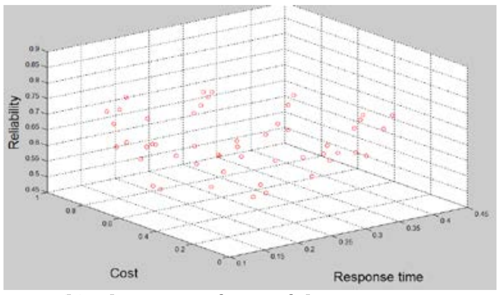
Fig. 7 give the Pareto front obtained by the four different algorithms. **Fig. 8** gives the Box char of the non-normalized QoS value of the compromise solutions obtained by the four algorithms through 30 independent experiments. The optimal vaules are highlighted with red bold font. Table 5 gives the average (Avg) values, Standard deviation (Sd) and 95% Confidence Interval (CI) of the non-normalized QoS value of the compromise solutions and Spread value obtained by the four different algorithms through 30 independent experiments. Fig. 8 and Table 5 shows that the LS-NSGA-II-DE has the better optimization ability, i.e., its average response time is the lowest, its reliability is the highest, and its cost is relatively low compared with the other algorithms. The MOEA/D obtains the lowest average cost, but its average response time and reliability are not the best. And the standard deviation and Confidence interval of the LS-NSGA-II-DE are the lowest, which illustrates that it is more stable than other algorithms. From the Pareto front in Fig. 7 and the Spread value in Table 5, we can also see that the LS-NSGA-II-DE has a better distribution than the other multi-objective algorithms.

Table 5. Comparison of the QoS and Spread values obtained by the different algorithms

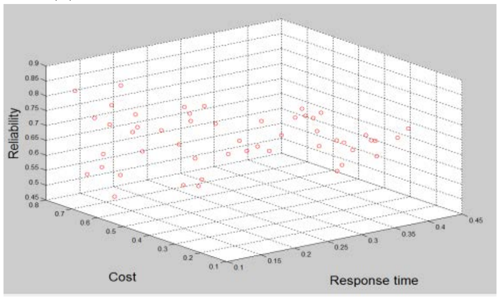
Algorithm	**Response Time(ms)**			**Cost($)**			**Reliability(%)**			**Spread**
	Avg.	Sd	95% CI	Avg.	Sd	95% CI	Avg.	Sd	95% CI	Avg.
NSGA-II	1626.5	52.2	[1607.0, 1646.0]	727.4	17.7	[720.8, 734.0]	18.3	2.4	[17.3, 19.2]	0.653
MOPSO	1665.4	58.5	[1643.2, 1687.5]	743.6	21.5	[735.5, 751.6]	19.3	2.1	[18.5, 20.1]	0.571
MOEA/D	1597.2	49.0	[1579.0, 1615.5]	**706.1**	18.9	[699.0, 713.1]	20.6	2.2	[19.7, 21.4]	0.536
LS-NSGA-II-DE	**1564.4**	**30.6**	**[1556.4, 1579.2]**	712.4	**17.3**	**[706.0, 718.9]**	**22.3**	**1.7**	**[21.6, 22.9]**	**0.480**



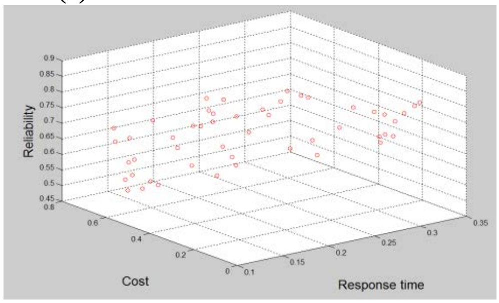
(a)The Pareto front of the NSGA-II



(b) The Pareto front of the MOPSO



(c) The Pareto front of the MOEA/D



(d) The Pareto front of the LS-NSGA-II-DE

Fig. 7. The Pareto front obtained by the different algorithms for service composition

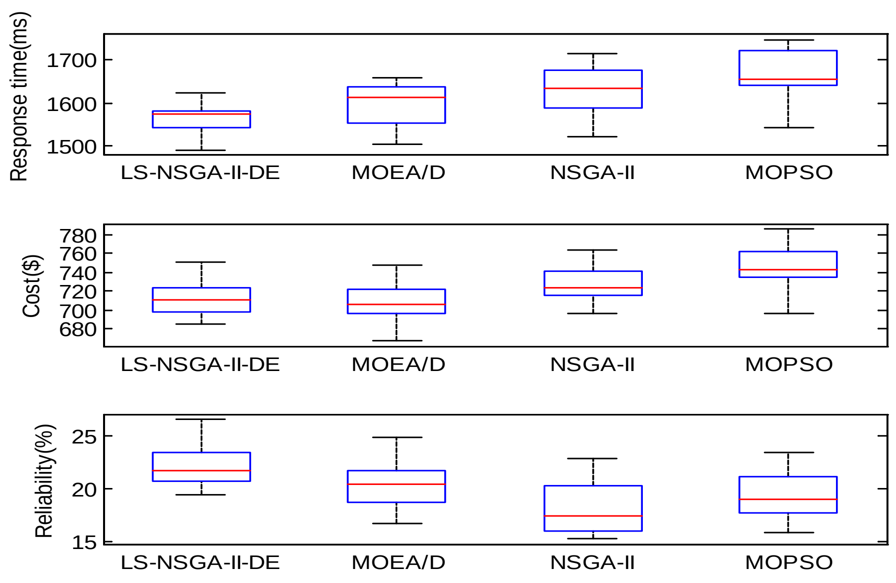


Fig. 8. The QoS values obtained by four the different algorithm

6.4 The Scalability Analysis

To evaluate the scalability of the proposed algorithm, we analyze the performance of each algorithm when the number of the service instances is increasing. The number of the service instances of each service class provided by each CP increases from 10 to 190 by 30. For example, the number of the service instances of the S^1 increases from 40 to 760 by 120, because the service instances of the S^1 are provided by 4 CPs. The QoS value of a service instance is randomly generated and meets the Gauss distribution, the average response time is 500 (ms), the standard deviation (sd) is 100 (ms), the average reliability is 90 (%), the sd is 10 (%), the average cost is 80($), and the sd is 20($). No constraint is set here. The algorithm's parameters are set the same as the experiment on benchmark functions. The single objective genetic algorithm GA uses the method of averagely weighting the normalized QoS objectives to optimize the multi-objective problem. The crossover probability p_c and the mutation probability p_m of the GA are set to 0.8 and 0.2, respectively.

Fig. 9 shows the average non-normalized QoS values of the compromise solutions, i.e. the response time, cost and reliability obtained by different algorithms with an increasing number of the service instances by 30 independent experiments. The solution of the GA becomes worse when the search space becomes larger. Thus, it is difficult to meet the needs of the users. For the multi-objective evolutionary algorithms, the solution of the LS-NSGA-II-DE is better than that of the other algorithms with an increase in the number of service instances, which shows our algorithm has a better scalability.

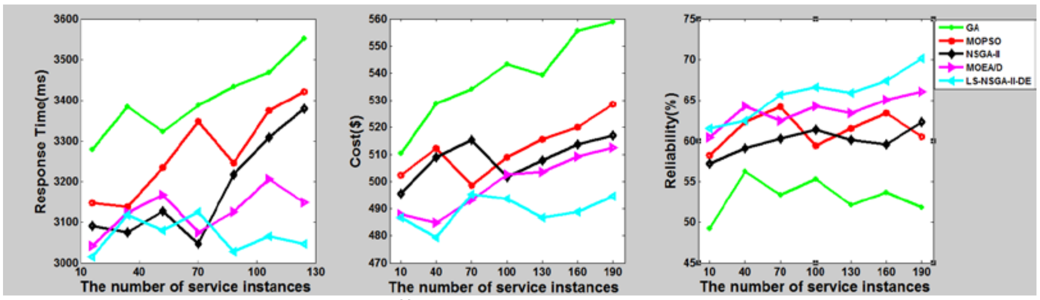


Fig. 9. The QoS obtained by different algorithms with the increasing service instances

7 Conclusion

In this paper, we describe the service composition architecture, the optimization objective and QoS model in Inter-cloud, while most previous works don't consider the Inter-cloud environment. And a new hybrid multi-objective evolutionary algorithm, LS-NSGA-II-DE, is proposed to explore how to select and compose the service instances while considering the goals of reaching a high efficiency on cost and response time, low network latency, and high reliability across multiple CPs, while most previous papers only optimize only one goal. In our proposed algorithm, the DE uses the adaptive mutation operator combined of two mutation strategies and crossover operator to replace the those of the NSGA-II and changes the evolutionary parameters adaptively in the process of generation. So we can search for the optimal solutions more efficient in the solution space and make the algorithm has better convergence and diversity. At the same time, the local search LS is performed for the Non-dominated solution set $F\{1\}$ in each generation, where we set the critical distance changed adaptively according to the number of the individuals in $F\{1\}$ to improve the distribution of the $F\{1\}$ dynamically. The experiments on the multi-objective benchmark functions and service composition analyzes the performance of our proposed algorithm, which shows that the LS-NSGA-II-DE has better optimization ability and scalability and performs well in terms of the distribution and convergence compared with other algorithms.

In our future work, first, we will use more efficient method instead of the simple penalty function to settle the constraints. When the constraints become strict, the penalty function may result in premature. Then, we should consider the preferences of the users, rather than use the compromise solution. Finally, we want to use the multi-objective evolutionary algorithm to solve the problem of the Inter-Cloud resource scheduling on the basis of the above work to meet the user's need well.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No. 61370132, No.61472033), Beijing Natural Science Foundation (No.4152034).

Reference

[1] C. Qu, R. N. Calheiros, R. Buyya, "A reliable and cost-efficient auto-scaling system for web applications using heterogeneous spot instances," *Journal of Network & Computer Applications*, vol.65, pp.167-180, 2016. Article (CrossRef Link).

[2] N. Grozev, R. Buyya., “Inter-Cloud architectures and application brokering: taxonomy and survey,” *Software: Practice and Experience*, vol. 44, pp. 369–390, March 2014. Article (CrossRef Link)

[3] J. Amin., S. Elankovan and O. Zalinda, “Cloud computing service composition: A systematic literature review,” *Expert Systems with Applications: An International Journal*, vol.41, no.8, pp.3809-3824, 2014. Article (CrossRef Link).

[4] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, “A fast and elitist multi objective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Compute.*, vol. 6, no. 2, pp. 182–197, Apr. 2002. Article (CrossRef Link).

[5] R. Storn, K. Price, “Differential evolutional-A simple and efficient heuristic for global optimization over continuous spaces,” *Journal of Global Optimization*, vol. 11, no. 4, pp.341-359, 1997. Article (CrossRef Link).

[6] Z. Ye, X. Zhou and A. Bouguettaya, “Genetic Algorithm Based QoS-Aware Service Compositions in Cloud Computing,” in *Proc. of 16th International Conference on DASFAA*, pp. 321-334, 2011. Article (CrossRef Link).

[7] M. Zhang, L. Liu, S. Liu, “Genetic Algorithm Based QoS-aware Service Composition in Multi-Cloud,” in *Proc. IEEE Conference on Collaboration & Internet Computting*, pp.113-118, 2015. Article (CrossRef Link).

[8] Q. Yu, L. Chen, and B. Li, “Ant colony optimization applied to web service compositions in Cloud computing,” *Computers & Electrical Engineering*, vol. 41, pp. 18-27, 2015. Article (CrossRef Link).

[9] M. Shojafar, N. Cordeschi, D. Amendola and et al, “Energy-saving adaptive computing and traffic engineering for real-time-service data centers,” in *Proc. of IEEE International Conference on Communication Workshop. IEEE*, pp.1800-1806, 2015. Article(CrossRef Link)

[10] M. Shojafar, N. Cordeschi and et al., “Energy-efficient Adaptive Resource Management for Real-time Vehicular Cloud Service,” *IEEE Transactions on Cloud Computing*, pp99:1-1, 2016. Article (CrossRef Link)

[11] M. Shojafar, S. Javanmardi, S. Abolfazli, et al, “FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method,” *Cluster Computing*, 2015, vol. 18, no. 2, pp.829-844. Ariticle (CrossRef Link)

[12] Y. Yao, “A Rule-Based Web Service Composition Approach,” in *Proc. of International Conference on Autonomic and Autonomous Systems (ICAS)*, pp.150-155, 2010. Article (CrossRef Link).

[13] J. Cao , X. Sun, and et al, “Efficient Multi-objective Services Selection Algorithm Based on Particle Swarm Optimization,” in *Proc. of IEEE Asia-pacific Services Computing Conference*, pp:603-608, 2010. Article (CrossRef Link).

[14] H. Wada, J. Suzuki, and et al, “E3: A Multi objective Optimization Framework for SLA-Aware Service Composition,” *IEEE Transactions on Services Computing*, vol. 5, no. 3, pp.358-371, 2012. Article (CrossRef Link).

[15] J. Feng, L. Kong, “A Fuzzy Multi-objective Genetic Algorithm for QoS-based Cloud Service Composition,” in *Proc. of International Conference on Semantics*, pp. 202-206, 2015. Article (CrossRef Link).

[16] L. Liu, M. Zhang, “Multi-objective Optimization Model with AHP Decision-making for Cloud Service Composition,” *KSII Transactions on Internet & Information Systems*, vol. 9, no. 9, pp. 3293-3311, 2015. Article (CrossRef Link)

[17] S. K. Garg, A. N. Toosi and et al “SLA-based Virtual Machine Management for Heterogeneous Workloads in a Cloud Datacenter,” *Journal of Network and Computer Applications*, vol. 45, no. 10, pp. 108-120, 2014. Article (CrossRef Link).

[18] G. Canfora M. Di Penta, and et al, “An approach for QoS-aware service composition based on genetic algorithms,” in *Proc. of Conference on Genetic and Evolutionary Computation*, pp. 1069–1075, 2005. Article (CrossRef Link).

[19] R. Storn, “On the usage of differential evolution for function optimization,” *Fuzzy Information Processing Society*, pp. 519–523, 1996. Article (CrossRef Link).

[20] Y. Zhou, C. Zhang, et al, “Multi-objective service compositon optimization using differential evolution,” in *Proc. of 11t International Conference on Natural Computation*, pp. 233-238, 2015. Article (CrossRef Link).
[21] S. Das, A. Abraham, et al, A. Konar, “Differential evolution using a neighborhood based mutation operator,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp.526-553, 2009. Article (CrossRef Link).
[22] S. Das, A. Konar, and et al , “Two Improved Differential Evolution Schemes for Faster Global Search,” in *Proc. of Genetic & Evolutionary Computation Conference*, pp. 991– 998, 2015. Article (CrossRef Link).
[23] N. Noman, H. Lba, “Enhancing differential evolution performance with local search for high dimensional function optimization” in *Proc. of Genetic & Evolutionary Computation Conference*, pp. 967-974, 2015. Article (CrossRef Link).
[24] K. Deb, A. Sinha, S. Kukkonen, “Multi-Objective Test Problems, Linkages, and Evolutionary Methodologies,” in *Proc. of Genetic & Evolutionary Computation Conference*, pp. 1141-1148, 2016. Article (CrossRef Link).
[25] Q. Zhang, et al, “MOEA/D: A multi objective evolutionary algorithm based on decomposition,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712-731, 2008. Article (CrossRef Link).
[26] C. A. C. Coello, G.T. Pulido, and M.S Lechuga, “Handling multiple objectives with particle swarm optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256-279, 2004. Article (CrossRef Link).
[27] W. Dong, L. Kang, W. Zhang, “Opposition-based particle swarm optimization with adaptive mutation strategy,” *Soft Computing*, pp. 1-10, 2016. Article(CrossRef Link).
[28] E. Zitzler, L. Thiele, “Multi objective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach,” *IEEE Transactions on Evolutionary Computation*, vol. 3, no.4, pp. 257-271, 2000. Article (CrossRef Link).
[29] K. Deb, “Multi-objective Optimization Using Evolutionary Algorithms: An Introduction,” *John Wiley & Sons*, vol. 2, no. 3, pp. 509, 2011. Article(CrossRef Link)
[30] http://www.uoguelph.ca/~qmahmoud/qws/.
[31] M.A. Abido, “Multi objective Evolutionary Algorithms for Electric Power Dispatch Problem,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp.315-329. Article (CrossRef Link).

Li Liu received her Ph.D. degree from University of Science and Technology Beijing (USTB), 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 in the areas of Cloud Computing, service composition and optimization.

Shuxian Gu received the B.E. degree in automation from the North China University of Science and Technology, China, in the year 2015. She is currently working toward the M.S. degree in the School of Automation and Electrical Engineering, University of Science and Technology Beijing (USTB). Her research interests include service composition and resource scheduling in Inter-Cloud

Dongmei Fu is a Professor and supervisor of PHD student in School of Automation and Electrical Engineering, University of Science and Technology Beijing (USTB). Her research interests are in the areas of infrared image technology, intelligent data analysis and modeling.

Miao Zhang received the M.S. degree in automation from School of Automation and Electrical Engineering, University of Science and Technology Beijing(USTB), China, in the year 2015. He is currently as a PHD student in School of Information and Electronics, Beijing Institute of Technology, Beijing, China. His research interest includes service composition and resource scheduling in Cloud computing.

Dr. Rajkumar Buyya is a Fellow of IEEE, Professor of Computer Science and Software Engineering and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is one of the highly cited authors in computer science and software engineering worldwide for over 500 publications.