J Supercomput (2016) 72:1897–1926
DOI 10.1007/s11227-016-1690-2

Cloud service ranking as a multi objective optimization problem

Arezoo Jahani[1] · Leyli Mohammad Khanli[1]

Published online: 9 March 2016

Abstract Cloud computing is a kind of computing model on subscription basis. In cloud computing environment, there are a lot of cloud providers that present variety kind of services with different quality of services. Users have various kinds of applications that should be carried out on suitable cloud services. Consequently the users might encounter problems in choosing the best service. Hence selection of a method to compare services and to choose best service has been regarded as a challenge. In this paper we presented NSGA_SR approach that utilizes both objective and subjective assessments and models ranking problem as a multi objective optimization and then solves it with use of non-dominated sorting genetic algorithm. Numerical experiments, confirmed that the proposed approach outperforms available approaches in terms of flexibility and scalability with increasing number of users and services. Also it converges to optimization of goals and has good stability during the different generations. Also it includes no limitation regarding any additive new quality attribute, service or supplementary function.

Keywords Cloud computing · Quality of service (QoS) · Service ranking · Multi objective optimization (MOO) · Non-dominated sorting genetic algorithm (NSGA-II)

✉ Arezoo Jahani
a.jahani@tabrizu.ac.ir

Leyli Mohammad Khanli
l-khanli@tabrizu.ac.ir

1 Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran

1 Introduction

Cloud computing knowledge performs the eminent role in expanding the systems and distributing the applications throughout the internet. This computing model makes possible access to information and computing resources in a flexible and scalable way at the demanded time [1]. Nowadays, in cloud computing environment, suitable service selection is important, because cloud users always have various kinds of applications with different quality requirements. Moreover, in cloud environment there are various kinds of cloud services with different qualities [2]. They have three types: software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) [3]. Users collide with problem in selecting one service based on their quality of service (QoS) requirements. In service ranking, the accuracy and celerity are vital, so having a suitable method to select and rank the services is a real challenge. Furthermore, it is not possible to select exclusively one service that is the aim of user from countless available ones. Therefore, in order to obtain the highest performance in short time, we need ranking systems. Ranking system can select some services based on user's requirements as candidate services and then it ranks them [4,5].

Ranking systems mostly use QoS values for service ranking. QoS attributes show functional attributes of service from qualitative or quantitative perspectives. Service quality refers to the capability of a network to provide better service with the guarantee of one or a number of commitments [6,7]. Evaluation and comparison of various cloud services is possible only through comparison and evaluation of quality of service. Therefore using the standard QoS values leads to a large acceleration in ranking operations [8]. QoS attributes may have various types such as Boolean, numerical and range [9]. In Boolean type, attributes have two values (one or zero). In numerical type, attributes have a numerical value and in range type, attributes have a value in a range. Such as response time that may be in the range of (20, 40) s [8,9].

Service ranking as a framework can evaluate different cloud services and determine their priorities. Cloud service ranking has four stages: receiving users requirement, measuring QoS attributes by monitoring or benchmarking tools [10], selecting candidate services and service ranking [8,11,12]. This paper has been focused on last two stages: selecting candidate services and service ranking. Candidate services receive rank according to user requirements; it can be through a statistical approach or linear algebra or an evolutionary algorithm. Higher ranking means that service is more suitable to meet the needs of user. Some ranking systems store the feedback of previous users as training information that would be used in ranking process. Therefore, the popularity of services can also effects ranking [13,14].

In ranking process, users may have essential and non-essential requirements. Essential requirements should be satisfied by candidate services. Non-essential requirements have less significance and if they do not be satisfied by candidate or ranked services, there would be no problem [8]. It means that if a service has all essential and non-essential requirements except one essential requirement, it cannot be accepted by user and can not do user's application properly. Accordingly ranking systems try to satisfy essential and non-essential requirements, if it would be possible.

This paper proposes non-dominated sorting genetic algorithm_service rank (NSGA_SR) approach as a ranking approach in order to rank all kinds of cloud services (includes SaaS, PaaS and IaaS [3]). The main aim of present work is cloud service ranking based on QoS attributes which utilizes both essential and non-essential requirements and both objective and subjective assessments in ranking process. Objective assessments are QoS attributes and subjective assessments are previous user's feedback information [8,15]. In fact, this paper covers both data by multi objective optimization (MOO) [16] with multiple functions and considers the range of user's requirements as constraints in the optimization problem. The NSGA_SR approach for service ranking, initially uses a filtering system in order to extract the candidate services that capable to satisfy user's all requirements. Then it finds several optimum pseudo services by non-dominated sorting genetic algorithm (NSGA-II) [17] with the use of two objective functions. At the end, all candidate services would be ranked based on their distance to pseudo services by application of clustering algorithm.

We implemented NSGA_SR approach (presented approach), AHP_SR, SVD_SR and W_SR (our previous approach) as related approaches in MATLAB software. We used QWS [18] real data set that were collected from more than 2500 web services. The experimental results showed that the NSGA_SR is more flexible and scalable with increase in the number of users and services and that NSGA_SR converges to optimization goals. It has also good stability during different generations. Moreover it has no limitations regarding the addition of any new quality or supplementary function.

The rest of the paper is organized as: brief overview of related works is described in Sect. 2. The QoS attributes is described in Sect. 3. The proposed ranking approach is outlined and described in details in Sect. 4. Dataset and Performance comparison are respectively provided in Sects. 5 and 6. At the end, conclusion and future works are discussed in Sect. 7.

2 Related works

With the increasing popularity of cloud computing services, many researchers started to evaluate and compare the services in order to help users to select appropriate services. In this regard, researches have been divided into two groups: solutions to compare and evaluate services and solutions to service ranking.

Solutions to compare and evaluate the services, only compare services based on QoS values without looking at the user's requirements. Monitoring and benchmark tools [10] are major available solutions in this category. These tools are able to compare services with monitoring different QoS features. In general, these tools can calculate and measure the QoS values and most of results are publicly available in the real datasets. Ranking solutions rank the services with assumption of having QoS values. These solutions compare and rank the services on the basis of user's requirements. The results of comparing division can be used in ranking services.

Cloudstone [19] is a benchmark and measurement tool for web 2.0 that was designed in 2008. Cloudstone is a toolkit that uses a set of automation tools to generate load and measure its performance in different deployment environments. This tool has been examined on Elastic Compute Cloud (EC2) and is able to measure the cloud services performance with the use of one question; ”how many concurrent (simulated) users can be supported by a fixed amount of hardware under baseline conditions (no special database tuning, caching)?"

CloudHarmony [20] is an important resource for cloud service performance evaluation. Totally, Cloud Harmony measures all of the attributes that can be supported by any service and is publicly available on internet. Users can select one attribute with several services and see the result of CloudHarmony benchmark in form of a graph, table or figure. CloudCmp [21,22] benchmark is able to estimate performance and cost of an application while it is deployed on a particular cloud provider. These tools are used for a comprehensive measurement on four major cloud providers; the Amazon web service, Microsoft Azure, Google App Engine and Rackspace cloud servers. The results are available on [28]. In general, CloudCmp is a systematic comparing structure of the performance and cost of cloud providers that measures the persistent storage, elastic computing and networking services offered by a cloud along metrics which directly reflects their impact on the performance of customer applications.

The main advantages of CloudCmp facing other tools was that it can be compliance with all aspects (such as computational power, stocks, network and scalability); while other tools could not cover all aspects. In addition, other tools were not extensible to all kinds of providers (for example they were not suitable for PaaS).

Compuware Company presented CloudSleuth [23] monitoring tools that was initially created as an internal resource to help users, measures the reliability and consistency of the most popular public IaaS and PaaS providers. CloudRank-D [24] is benchmark and rank cloud computing systems that are shared to run big data applications. The main focus of this tools is to propose two new simple metric: data processed per second and data processed per Joule as two complementary metrics for evaluating cloud computing systems that is suitable for evaluating whole system level instead of a component or subsystem level, like processor or I/O subsystems. Skoutas et al. ranked web services using multi criteria dominance relationships [25] here it can also be extended to cloud services.

Rehman et al. [26] claimed that benchmark tests are unable to accurately determine or reliably predict the performance of actual cloud applications because they do not produce real workload. Instead, they presented a user feedback based approach that used from a cloud status checker and measured QoS values in the presence of real applications.

Parallel to the increasing popularity of cloud computing, many researches have been carried out about service ranking. Although service ranking was familiar in web services for many years, but some ranking approaches were proposed just on the basis of cloud services and their attributes. Chan and Chieu [11] presented a cloud service provider mapper as a ranking solution that uses from singular value decomposition (SVD) techniques for cloud service ranking. This SVD base approach tried to find the best service upon users requirements. By this way, it would initially find one pseudo service by applying SVD technique on available service and then it finds a suitable service with lowest distance of pseudo service. service ranking system (SRS) [27] is another approach that presented in the form of two modes: static and dynamic. In static mode, SRS ranked all available services in cloud market by not noticing to user's requirements. But in dynamic mode, It ranked services according to user's requirements. Also, service level agreement (SLA) matching approach [28] being as a part of the Cirrocumulus [29] project defined the process of identifying compatible cloud provider for given requirement by matching SLA parameters.

Aggregation approach [15] is the first ranking approach that used both objective and subjective assessments. Instead of utilizing all of the user's requirements it only focused on user's priority (such as normal values for each attribute that showed its significance). CloudRank approach [30,31] benefited from training users feedback to QoS values prediction [32] and so found similar users and gave same ranking result to similar users. Another approach that benefited from Analytical Hierarchical Process (AHP) [8] for service ranking was able to rank them based on both essential and non-essential requirements. This approach proposed a SMICloud (Service Measurement Index) framework. SMICloud ranked services based on SMI attributes that presented by Cloud Service Measurement Index Consortium (CSMIC [33]) and was especially designed for cloud computing services. W_SR (Weight_Service Rank) [34] approach was able to benefit from both essential and non-essential requirements and ranked services by using Min–Max technique as a decision solutions.

The main disadvantages of ranking systems is that none of them did not use objective and subjective assessments and essential and non-essential requirements. The related works had low level of flexibility and scalability. Also they were limited in terms of adding new qualitative attribute or service. The aim of present article is to address these problems and propose a new ranking system.

3 Quality of services

Comparing and evaluation of services can be possible by QoS values [35,36]. Therefore it is important that QoS attributes would be recognized and measured [37–39]. QoS attributes have many different types such as response time and availability. Therefore, we should have a criterion that would cover wide range of qualitative attributes and make it possible to compare service providers and their services. In this regard CSMI Consortium specified attributes which were named SMI attributes to evaluate and compare cloud services [40]. Specified attributes are based on International Organization for Standardization (ISO [41]) and they include seven attributes and some sub attributes. The main attributes described are as follows [8,33,42]:

1. *Accountability* contains attributes used to measure the properties related to the service provider organization. These properties may be independent of the service being provided. If a service does not have accountability, no one tend to use and deploy his/her data on it. Because it is possible that the service was not able to respond during needed time due to the unavailability. Some of sub attributes of accountability are: auditability, compliance, data ownership, ease of doing business, governance, ownership, provider business stability.
2. *Agility* indicates the impact of a service upon a client's ability to change direction, strategy, or tactics quickly and with minimal disruption. In the presence of this attribute, users can change or expand their services without paying any extra cost. Adaptability, capacity, elasticity, extensibility, flexibility, portability and scalability are sub attributes of agility.

3. *Cost* the amount of money spent on the service by client. Acquisition and transition cost, ongoing cost and profit cost or cost sharing are sub attributes.
4. *Performance* because of many kinds of cloud providers, users need to understand about the performance of each service faced with their application. Efficiency means less time or energy for most work is done right. The purpose of the performance of a system, determine the maximum time that a system can not resist or the accomplishment of a given task measured against preset known standards of accuracy, completeness, cost, and speed. In a contract, performance is deemed to be the fulfillment of an obligation, in a manner that releases the performer from all liabilities under the contract. Features and functions of provided services like accuracy, functionality, suitability interoperability and service response time are regarded as its sub attributes.
5. *Assurance* the probability of system success or unsuccess in doing the duties without any failures called assurance. In fact service assurance is a procedure or set of procedures intended to optimize performance and provide management guidance in end-user applications. Service assurance is an all-encompassing paradigm that revolves around the idea that maximizing customer satisfaction inevitably maximizes the long-term profitability of an enterprise. Sub attributes are availability, maintainability, recoverability and reliability.
6. *Security and privacy* indicate the effectiveness of a service provider's controls on access to services, service data and the physical facilities form which services are provided. Sub attributes are access control and privilege management and data geographic.
7. *Usability* by this feature, users can easily learn using from service and switch to that. Accessibility, client personnel requirements and instability are some of sub attributes.

4 Assumptions of NSGA_SR

The assumptions of proposed NSGA_SR approach are presented here. NSGA_SR can ranks the services based on both objective and subjective assessments. Also it can consider both essential and non-essential requirements in ranking process. Users can enter the required values in three ways of Boolean, numerical and range. NSGA_SR has two objective functions: one is to find suitable services that satisfies all essential requirements and most of non-essential requirements. In fact it maximizes user's objectives optimization and minimizes the deviations from the soft constraints (Objective assessment). Another feature is to use training user's feedback in order to show the satisfaction of training users. In fact it maximizes service satisfaction by training users (Subjective assessment). These objective functions and other assumptions such as modeling of service and users and also QoS vector are described in Sects. 4.1–4.5. The NSGA_SR approach and its steps will have described in detail in Sect. 5.

Here we modeled ranking problem as a MOO problem. We also modeled essential and non-essential requirement respectively with hard and soft constraint. In following sections we called essential and non-essential requirement with these names.

4.1 Modeling cloud services

All cloud services that are available in system's catalogue are called cloud services. Modeling of services is depicted in Eq. (1).

$$\begin{aligned} S &= \{s_n | 1 \leq n \leq N\} \\ s_n &= \langle q_{n1}, q_{n2}, \ldots, q_{nj}, \ldots, q_{nQ} \rangle \end{aligned} \tag{1}$$

S is a set that includes services like s_n. if we have N services in total, then n will be between 1 and N. Each s_n is known by its QoS attributes. So s_n is an ordered pair that has Q element. q_{nj} shows j as the qualitative attribute of service n. If we have Q qualitative attributes, then each service would be shown by Q value.

4.2 Modeling ranking system's users

All users that have a ranking request from the system at same time are generally called users. Modeling process of users is depicted in Eq. (2).

$$\begin{aligned} U &= \{u_m | 1 \leq m \leq M\} \\ u_m &= \{R_m, C_m, SC_m, HC_m\} \\ R_m &= \langle R_{m1}, R_{m2}, \ldots, R_{mj}, \ldots, R_{mQ} \rangle \\ C_m &= \langle C_{m1}, C_{m2}, \ldots, C_{mj}, \ldots, C_{mQ} \rangle \\ SC_m &= \langle SC_{m1}, \ldots, SC_{mj}, \ldots, SC_{mQ} \rangle, \quad SC_{mj} \in \{0, 1\} \\ HC_m &= \overline{SC_m} = \langle HC_{m1}, \ldots, HC_{mj}, \ldots, HC_{mQ} \rangle, \ HC_{mj} \in \{0, 1\} \end{aligned} \tag{2}$$

U is a set of services that includes users like u_m. m is between 1 and M, if we have M total users that use the system at same time. Each u_m is identified by four values that are as follows:

- R_m is an ordered pair and shows user's requirements and includes R_{mj} which shows j th QoS requirement of user m. Each R_{mj} can be a single value, a Boolean or an arrange. If it has arange as value it will be like $[R_{mj1}, R_{mj2}]$ in a way that the first value is (min R_{mj}) and the next is (max R_{mj}). If it has a Boolean or single value will be like $[R_{mj}, R_{mj}]$ that has two same values.
- C_m is an ordered pair too and shows user's coefficient (or user's priority) and includes C_{mj} that shows j th QoS coefficient of user's m. The user enters coefficient between 0 and 1 and it is not needed to the sum of coefficient to be 1 (it is regarded to be one of the advantages of proposed approach).
- SC_m is an ordered pair and shows user's soft constraint and includes SC_{mj} that shows j th QoS soft constraint of user's m. Each SC_{mj} is a Boolean value that can be zero or one.
- HC_m is an ordered pairs and shows user's hard constraint and includes HC_{mj} that shows j th QoS hard constraint of user's m. Each HC_{mj} is a Boolean value

Table 1 Main variables in NSGA_SR

N	Number of services	s_n	n th service
M	Number of users	u_m	m th user
R_m	QoS requirement of user m	C_m	QoS coefficient of user m
SC_m	QoS soft constraint of user m	HC_m	QoS hard constraint of user m
T	Number of training users	tu_t	t th training user
K	Number of candidate services	CS	Set of candidate services
Q	Number of quality attributes		

that can be zero or one. Hard constraint includes quality attributes that are not soft constraint. So $HC_m = \overline{SC_m}$.

4.3 Modeling of training users

All the previous users that have ranking request from system are called training users. These users can give feedback after finishing request between zero and one which is stored as training information and used in ranking. However users are able to have a ranking request either by using or not using training information (this is again one of the advantages of present approach). Modeling of training users is depicted in Eq. (3).

$$\begin{aligned} TU &= \{tu_t | 1 \le t \le T\} \\ tu_t &= \langle C_{t1}, C_{t2}, \ldots, C_{tj}, \ldots, C_{tQ} \rangle \end{aligned} \tag{3}$$

TU is a set of training users like tu_t that give feedback to the same service. t would be between 1 and T, if we have T training users as it has been figured out in Table 1. Each tu_t is known by training user's coefficients. C_{tj} is an ordered pair and shows j th QoS coefficient of training user's t. The main variables of NSGA_SR are shown in Table 1. As indicated in Table 1, K is the number of candidate services and CS is a set of candidate services that will be used in Sect. 5.

4.4 QoS vector modeling

All of the users, services and training users are modeled by QoS attribute. QoS attributes has been described in Sect. 3. The description, measurement methods and unit of measurement of seven main QoS attributes are depicted in Table 2. We will evaluate proposed approach with the use of these seven main attributes (Q = 7). Here, we can confirm that our approach is more flexible whit adding any new attribute and we will show that in Sect. 6.

4.5 Modeling objective functions

In MOO [16] we have more than one objective function in which we want to minimize or maximize its components at the same time. In fact, because of availability of many

Table 2 Quality of service attributes

	Quality of service	Description	Measurement method	Unit of measurement
1	Accountability	Interval time for access to user data in demand time	Time required to send requests and receive replies	ms
2	Agility	Ability to change or expand cloud services without spending	Ability to change services without spending (rate of meet the changing needs of the user)	%
3	Cost	Performance of cloud services based on SLA	Relative error messages to the total number of messages	%
4	Performance	Needed costs to run the service	The cost of allocated resources	$
5	Assurance	Number of answered service requests at a time	The total number of invokes at a time	Invoke/s
6	Security and privacy	Data protection	The data protection	%
7	Usability	Learn easy and switch to cloud services easily	The number of successful invokes/total number of invokes	%

different objectives, we cannot improve all of the objectives simultaneously and we have to set a tradeoff among them. As it was mentioned aimed use both objective and subjective assessments. Objective assessments obtained from QoS values and subjective assessments obtained from the feedback of previous users as training information. In NSGA_SR we have two main objective functions, one for objective assessment and the other for subjective. They are described as follow:

(1) Objective assessment (maximizes user's objectives optimization and minimizes deviations from the soft constraints) this object is defined to receive user's requirements and increase system ability in selection of suitable services according to user's requirements. In this objective, the services that can satisfy all hard constraints and most of the soft constraints, have more chance to select.

Here it should be mentioned that in this objective, we have some attributes that must be maximized such as availability or security. On the contrary, we have some attributes that should be minimized such as cost or error rate. Suppose that we have p attributes which should be maximized and q attributes that should be minimized ($p + q = Q$) and each of these can be either hard or soft. The first objective function is depicted in Eq. (4).

$$\begin{aligned} minF_1(objective) = &-\sum_{j=1}^{p}(C_{mj} \times q_{nj} \times HC_{mj} + C_{mj} \times |q_{nj} - R_{mj}| \times SC_{mj}) \\ &+\sum_{j=1}^{q}(C_{mj} \times q_{nj} \times HC_{mj} + C_{mj} \times |q_{nj} - R_{mj}| \times SC_{mj}) \end{aligned} \quad (4)$$

C_{mj} shows jth QoS coefficients of user u. q_{nj} shows j that is the qualitative attribute of service n. HC_{mj} shows j that is the QoS hard constraint of user u. R_{mj} shows j that is the QoS requirement of user u. SC_{mj} is j that is the QoS soft constraint of user u and the deviations from the soft constraints calculate in objective function.

p is the number of attributes with maximization objective and q is the number of attributes with minimization objective. Here we change first objective function as a minimum function (min $F_1(x)$) and calculate reverse total of p attributes with total of q attributes. Hard constraints are directly multiplied in the amount of QoS constraint. On the other hand soft constraints are multiplied in the subtracting of the amount of user QoS requirements.

(2) Subjective assessment (to maximize service satisfaction by means of training users of the system) this function typically is a guarantee of QoS. Because training user produces his/her feedback of services after finishing application and such function tries to select services that receive better feedback from training (previous) users. So it can be a QoS guarantee. Second objective function is depicted in Eq. (5).

$$\begin{aligned}\min F_2(\text{subjective}) &= -\sum_{j=1}^{p}\left(\sum_{t=1}^{T}\left(C_{tj}\times q_{nj}\times HC_{mj}+C_{tj}\times|q_{nj}-R_{mj}|\times SC_{mj}\right)/T\right)\\ &\quad+\sum_{j=1}^{q}\left(\sum_{t=1}^{T}\left(C_{tj}\times q_{nj}\times HC_{mj}+C_{tj}\times|q_{nj}-R_{mj}|\times SC_{mj}\right)/T\right)\end{aligned} \tag{5}$$

C_{tj} shows j that is the QoS coefficients of training user t. Other variables are described in first objective function. T shows the number of training users that give feedback to the same service such as s_n. Second function is calculated as below; for all available services ($\sum_{j=1}^{P}$) and for all available training users ($\sum_{t=1}^{T}$), user's feedback (coefficients of training user) has been multiplied in value of the attribute. At the end the total amount of obtained values has been divided in to the number of training users (notice that, as we saw in first objective function, the process is different in hard or soft constraints). Any MOO problem has some constraints. The constraints of our proposed approach are depicted in Eq. (6).

$$\min R_{mj} \le q_{nj} \le \max R_{mj},\ j : 1 \le j \le Q \tag{6}$$

R_{mj} shows the value of j that is the QoS requirement of user u. If user insert requirements in range shape, the lower limit of range, would be set in min R_{mj} and the upper limit of rage, would be set in max R_{mj}. If user inserts only one value as requirement, the lower and upper limit, would be set to the same number. To sum up constraints expresses the fact that "all attribute values must be in the range of user's requirements." By having these two objective functions, the services can satisfy more user's requirements and receive better score from training users who are luckier to be selected.

5 NSGA_SR approach

The proposed NSGA_SR approach is presented in this section. NSGA_SR approach includes three steps that are shown in Algorithm 1.

1. *Filtering step* initially, in filtering step some candidate service that can satisfy users requirements would be selected. In this paper we showed the candidate services with K. this variable can be initialized by ranking system designer.
2. *Finding pseudo services with NSGA-II step* After that some pseudo services would be extracted by NSGA-II that are not available in fact and do not have any priority to each others.
3. *Clustering step* At the end candidate services would be ranked by clustering algorithm.

NSGA_SR inputs are sets of services, training users, requirements and coefficients of requirements. The output is a service ranking of candidate service. The NSGA_SR flowchart has been shown in Algorithm 1. All of the steps are depicted in Fig. 1.

Algorithm 1 NSGA_SR Algorithm: Nondominated Sorting Genetic Algorithm_Service Rank // Cloud Service Ranking as MOO problem with NSGA-II

```
Require: a full service set $S$, an users Requirement set $R_m$, an users coefficient set $C_m$, an soft constraint
    set $SC_m$, a Training users set $TU$, a preference multi object function
Ensure: a service ranking $R$
1: Filtering_Algorithm // Select candidate services from service set $S$
2: NSGA-II Algorithm // Select the pseudo services
3: Clustering Algorithm // Ranking candidate services
```

5.1 Filtering step

This step evaluates services according to users requirements and extracts the candidate services. Filtering step has two main aims: to maximize the user's requirements (or not to exceed the constraints of the user) and to minimize violations of soft constraints. Generally, the number of candidate services is initialized by designer of ranking system and are shown by K. filtering step has been depicted in Algorithm 2.

As it is depicted in Algorithm 2, by the time the number of candidate services gets larger than the total number of available services, all of them would be considered as candidate services and then sent to second step. Otherwise the algorithm extracts services that can satisfy all hard and soft constraints. After that, if the size of extracted candidate services (CS) would be smaller than k and the size of soft constraint (SC_m) be larger than zero, it selects one soft constraint with fewer coefficient and sets it with ($[-\infty, +\infty]$) in user requirement (R_m) then it remove soft constraint (SC_m). This process reoccurs until K or more than K candidate services would be founded.

If the size of candidate service would be smaller than K and gets equal with zero and the size of soft constraint grows not bigger than zero, in this condition there would be not any suitable services and the algorithm must be exited.

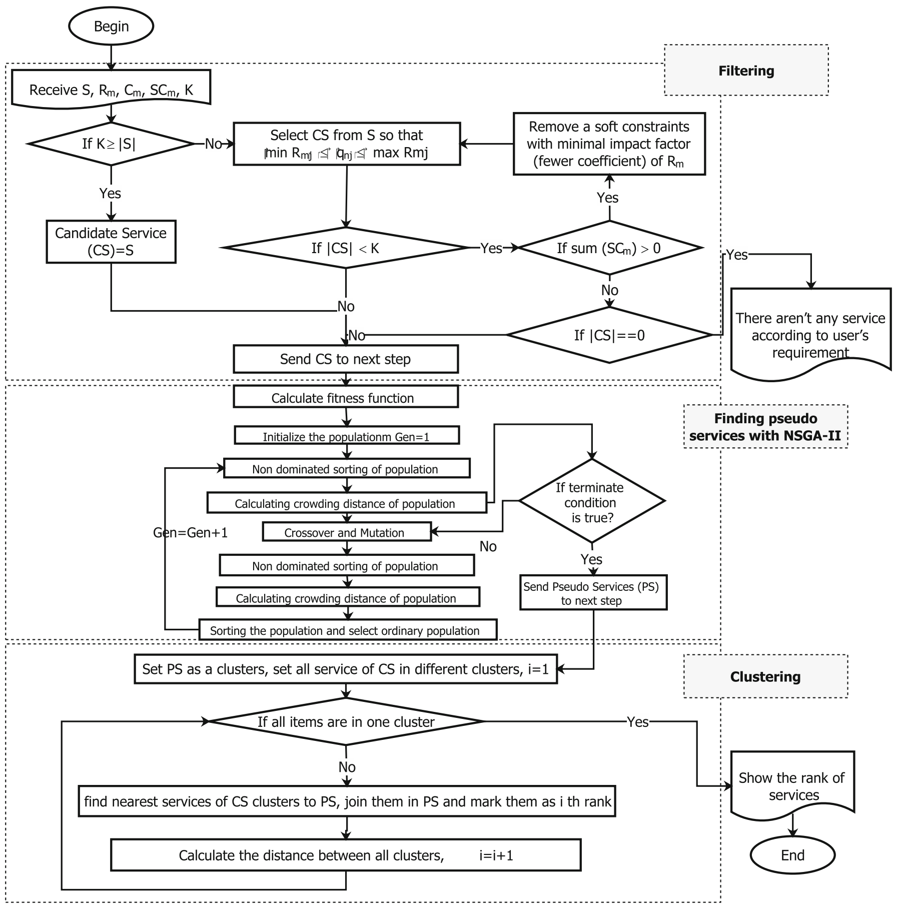
Begin
Filtering
Receive S, Rm, Cm, SCm, K
If K≥ |S|
No
Select CS from S so that
min Rmj ≤ qnj ≤ max Rmj
Remove a soft constraints with minimal impact factor (fewer coefficient) of Rm
Yes
Candidate Service (CS)=S
If |CS| < K
Yes
If sum (SCm) > 0
Yes
No
There aren't any service according to user's requirement
No
If |CS|==0
No
Send CS to next step
Calculate fitness function
Finding pseudo services with NSGA-II
Initialize the populationm Gen=1
Non dominated sorting of population
Calculating crowding distance of population
If terminate condition is true?
Gen=Gen+1
Crossover and Mutation
No
Non dominated sorting of population
Yes
Send Pseudo Services (PS) to next step
Calculating crowding distance of population
Sorting the population and select ordinary population
Set PS as a clusters, set all service of CS in different clusters, i=1
Clustering
If all items are in one cluster
Yes
No
Show the rank of services
find nearest services of CS clusters to PS, join them in PS and mark them as i th rank
Calculate the distance between all clusters, i=i+1
End

Fig. 1 NSGA_SR flowchart

5.2 Finding pseudo services with NSGA-II step

This step finds pseudo services through NSGA-II that are not really available but they satisfy user's requirements. This step maximizes the selecting chance of services that can satisfy more requirements of user. This level has been shown in Algorithm 3. In the following sections coding method, calculating fitness function and the NSGA-II evolution would be described.

5.2.1 Coding method

To search the answer in MOO problems, at first we must code the problem with proper chromosome structure. Each chromosome of the NSGA_SR problem is indicative through a sequence of services that respectively intended to the users. Therefore each service is also indicative by its QoS attributes. We proposed the method for NSGA_SR

Algorithm 2 Filtering Algorithm

```
Require: a full service set $S$, an users Requirement set $R_m$, an soft constraint set $SC_m$ , a cofficient set
    $C_m$, the number of candidate service $K$
Ensure: a candidate service set $CS$
1: if ($k \geq |S|$) then
2:   // if the number of candidate services is bigger than the number of all services, filtering step is not
     need
3:   $CS = S$;
4: else
5:   Select services from $S$ such as $s_n$ so that ($minR_{mj} \leq q_{nj} \leq maxR_{mj}$) and put them in $CS$ set;
6:   if ($|CS| < K$) then
7:     if (sum($SC_m > 0$) then
8:       // if there is still some soft constraint
9:       Select the Index of min$\{C_m \times SC_m^{transpose} | C_{mj} \times SC_{mj} \neq 0, 1 \leq j \leq Q\}$
10:      $C_{m(Index)} = \infty$;
11:      $SC_{m(Index)} = 0$;
12:      $R_{m(Index)} = [-\infty, +\infty]$;
13:      Go to 4;
14:    else
15:      if (CS == 0) then
16:        // if algorithm can not find any candidate service
17:        return "there isn't any service according to user's Requirement" as a output ;
18:        Exit;
19:      end if
20:    end if
21:  end if
22: end if
23: return $CS$ (candidate service set) as a output
```

Algorithm 3 NSGA-II Algorithm

```
Require: public input: an users Requirement set $R_m$, an users coefficient set $C_m$, an users soft coefficient
    set $SC_m$, an users hard coefficient set $HC_m$, a preference multi object function
    Private input: $n_{pop}$, problem$_{size}$=$M \times Q$, $p_c$,$p_m$, $n_c$, $n_m$
Ensure: a pseudo service set $PS$
1: Population=Initialize Population($n_{pop}$, problem$_{size}$);
2: Nondominated sorting of Population;
3: Calculating crowding distance between Population items;
4: while  stop condition ()  do
5:   Parents=select parents(Population, $n_c$);
6:   for $parent_1$, $parent_2 \in$ parents do
7:     $Child_1$, $Child_2$= crossover($parent_1$, $parent_2$, $p_c$);
8:     add $Child_1$ and $Child_2$ to Population;
9:   end for
10:  solution=select solution(Population, $n_m$);
11:  for each solution $\in$ solution do
12:    mutate solution= mutate(solution, $p_m$);
13:    add mutate solutiom to Population;
14:  end for
15:  Nondominated sorting of Population;
16:  Calculating crowding distance between Population items;
17:  $PS$=Get best solution(Population, $n_{pop}$);
18: end while
19: Return $PS$
```

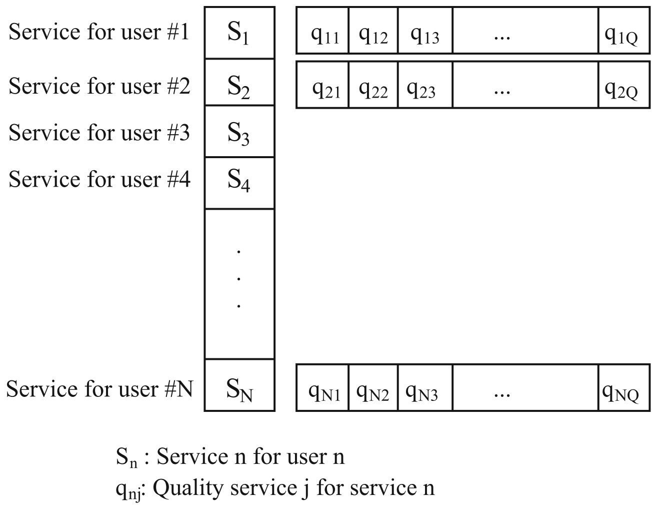

Fig. 2 Coding for a chromosome (for one user such as *m*)

coding in Fig. 2. Our coding method is a two–dimensional integer array of length $M \times Q$. (M is number of users who use ranking system at the present time and Q is the number of qualitative attributes of each service) an element at index (i, j) of the array represents the value of the QoS that is based on the j qualitative attribute of user i.

5.2.2 Calculating fitness function

As it was mentioned, NSGA_SR is an approach that tries to rank services as a MOO problem according to objective and subjective assessments. Here two objective functions (4) and (5) have been produced that can be utilized as fitness function in second step of NSGA_SR.

5.2.3 NSGA-II evolution

Evolution of NSGA-II in NSGA_SR second step and according to Algorithm 3, has been performed in following steps (In order to understand required variables in Algorithm 3 take a look at Table 3):

1. *Generating initial population* Initial population is a set of chromosomes in which each chromosome includes its QoS for every user. Another method to generate initial population is the use of candidate services produced in filtering step and set them as initial population of the algorithm. By this work, the initial population would be selected intelligently and may accelerate the convergence of algorithm. But it does not have any capability to produce final answer. The effect of product generation (randomly or intelligently) will be examined in details in Sect. 7.
2. *Calculating the fitness* According to the objective of optimization, chromosomes' fitness is calculated by application of (4) and (5).

Table 3 Evolution parameters of NSGA_SR

Name of parameter	Parameter	Value
Number of candidate services	K	100
Population size	n_{pop}	50
Maximum number of generations	$\text{Max}_{\text{iteration}}$	100
Rate of crossover	p_c	0.08
Number of generated solutions with crossover	n_c	Round $(p_c \times n_{\text{pop}}/2) \times 2$
Rate of mutation	p_m	0.03
Number of generated solutions with mutation	n_m	Round $(p_m \times n_{\text{pop}})$

3. *Nondominated sorting of population* Non-dominated sorting of population is the most important part of NSGA-II. It sorts chromosome based on dominated or dominant when facing other chromosomes. In fact this part sorts the population (includes services) based on all objective functions and finds some Pareto front that each includes some services. For the reason that, more than one objective function exist, it is difficult to sort the population in order to select the better solutions of population and also move them to next generation. Non-dominated sorting compares population based on all objective functions and then sorts them. Services with lowest Pareto front are considered to be better than the others while services that have same Pareto front, are same and do not have any priority to each others.
   In order to identify solutions (chromosome or service) that are on the first Pareto front, each solution must be compared with every other solutions in the population. The solutions that are not dominated by any other solution are marked as the first Pareto front. Now we have all solutions on first Pareto front. To find the members of other Pareto fronts, we should omit the solutions on first Pareto front. Now again, we should find solutions that are not dominated by other solutions and mark them as second Pareto front. This process continues until the number of Pareto front of all solutions would be identified.
   Formally, the procedure happens in following levels: at first we should define two variables for each solution; domination count is the number of solutions which dominate a solution. Dominant set is a set of solutions in which a solution dominates over them. After that, all pairs of solutions would be compared. In every comparison, if solution A dominates over solution B, B would be added to the dominant set of A and if solution B dominate over A, domination count of B gets incremented. Now the solutions that have domination count equal to zero acquire first Pareto mark and form the current Pareto front. To find the other pareto, we should omit the solutions with first Pareto front and should increase the domination count from solutions that first Pareto solutions are dominated over. Therefore by repeating this process we are able to find all Pareto fronts. Such procedure will be continued until we would not have any solution without Pareto front or could not find any solution with domination count equal with zero.

Method of domination
In competition between the solutions, we use objective functions. First we calculate all objective functions for every solution and therefore we use Eq. (7).

$$\begin{aligned}&(A\ dom\ B) \Rightarrow \forall j : f_j(A) \le f_j(B)\\&\exists j_0 : f_j0(A) < f_j0(B)\\&1 \le j \le Q\end{aligned} \tag{7}$$

As it has been depicted in (7), solution A have dominance over solution B, if A does not worse than B in any way and does better than B at least in one way.

4. *Calculating crowding distance* Always, after non-dominated sorting of population, we must calculate crowding distance. Later to finding all Pareto fronts, it should be noticed that the solutions which have less Pareto fronts will be better than the others and if some solutions have same Pareto fronts, they are same and do not have any priority to each other. But "how we can select some of solutions on a Pareto front?" Crowding distance can do this by calculating crowding distance for each Pareto front.
In fact, crowding distance is estimating the density of solutions surrounding a particular solution in a population. About a particular solution, we calculate the average distance of two solutions on either side of this solution along with each objective and in each Pareto front. For each Pareto front, crowding distance of both the initial and final solution would be set to be infinite.
Presently, every solution of one Pareto front with high crowding distance is considered to be better than the others and is luckier to remain. As an instance, we showed non-dominated sorting and crowding distance for a MOO problem with two objective functions in Fig. 3 (both of the objective functions are minimizing, so the ideal spot is (0, 0)).
As depicted in Fig. 3, if we have two objective functions $F(1)$ and $F(2)$ and we showed all solution with points in shape (a), with determination Pareto fronts, we can reach to shape (b) and with calculating crowding distance, we can reach to shape (c) of Fig. 3. Here, the solutions with lowest Pareto front and highest crowding distance were better than the others.
5. *Crossover* After generating median population, solutions selected pairs and then the crossover was carried according to the rate of crossover (P_c). We used two–point crossover method in which both two solutions replaced the number of their gens with each other's.
6. *Mutation* In order to protect the algorithm from stopping in local optimum, we must use mutation in NSGA-II. Some solutions with rate of mutation (P_m) were selected and a gene was randomly selected from each and then it was replaced with a random number.
7. *Sorting and replacing* After performing above actions, to select the number of median solution and then to transfer into the next generation, we sorted population. For sorting procedure we used solution's priority (with lowest Pareto front and highest crowding distance). Here,we should notice that if we want to sort the population, first it should be based on Pareto fronts and next on crowding distance.

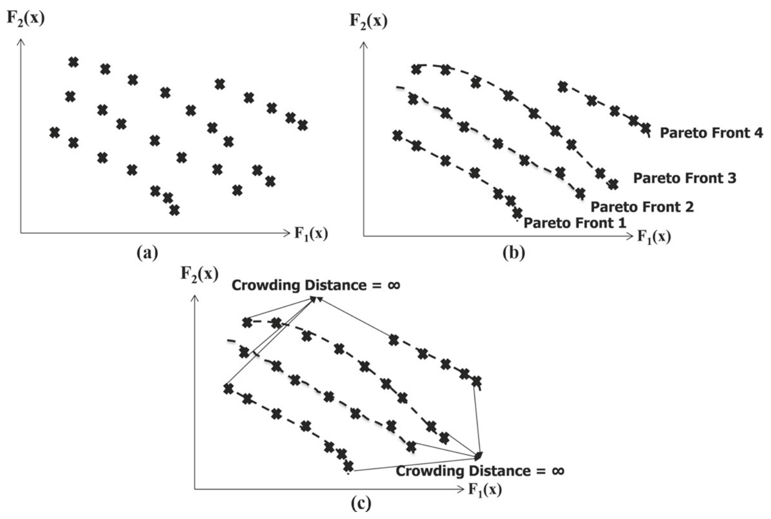

Fig. 3 Methods of nondominated sorting and calculating crowding distance

In fact we should firstly sort based on crowding distance and next based on Pareto fronts.

8. *Termination condition* To obtain objective optimization, the NSGA-II evolution ends with following condition:
   - To reach the maximum number of generations,
   - To converge to the final answer and have no need to repeat the algorithm.

5.3 Clustering step

Clustering step has been shown in Algorithm 4. This algorithm is similar to hierarchical clustering algorithm by having little changes. In hierarchical clustering algorithm, the aim is to find clusters that have lowest distance from each other. In every iteration, two clusters with lowest distance join together. The iteration would be continued until there remain only one cluster.

We have used similar clustering algorithm. Our clustering algorithm receives two inputs: candidate services and pseudo services. Here we inserted pseudo services in cluster as a base cluster. Now, in every iteration it is tried to find nearest candidate services to pseudo services cluster and mark them as first rank and join them in cluster of pseudo services and this act would be repeated until all services receive rank and join to the cluster of pseudo services.

6 Dataset description

To evaluate the NSGA_SR approach, we used QWS (Quality of Web Service) dataset [17]. QWS is a real world dataset which includes nine main QoS that are collections

Algorithm 4 Clustering Algorithm

```
Require: a candidate service set $CS$, a pseudo service set $PS$
Ensure: a service ranking $R$
1: Put all services of $CS$ in separable clusters and put $PS$ in one cluster;
2: i=1;
3: while (all $CS$ and $PS$ isn't in one cluster) do
4:   Calculate distance between $PS$ and all $CS_k$; // $CS_k$ is k th candidate service
5:   Mark all nearest services to $PS$ cluster's as $i$ th Rank and add them in $PS$ cluster;
6:   $i = i + 1$
7: end while
8: $R$=all rank of CS set
9: return $R$
```

of more than 2500 web service. The information of QWS dataset has been collected from multiple heterogeneous sources (i.e. UDDI, service portals, or search engines). Therefore it adds another level of complexity with respect to service providers managing and their administration. To develop examinations that the desired service number is more than 2500, more quality information services have been produced according to QWS. NSGA_SR and compared solutions have been implemented in MATLAB software (Version R2015a 7.14.0) in an operating system with an Intel Corei5 Duo 2.53 processor, 4 GB Ram and Windows 7 x86 Enterprise. QWS data set lacked cost and security attributes, so we randomly generated these attributes and added to the data set.

Constraints of users were produced randomly with normal distribution between the highest and lowest values for each quality of service. Training users information can be produced randomly or can be started with empty values and recorded by receiving every feedback from training users. We produced training information randomly.

Moreover we assumed that all feedback is a filter and no user can enters untrue feedback. As it was mentioned, NSGA_SR uses NSGA-II, we showed NSGA-II parameters in Table 3. We obtained these parameters with examination, trial and error procedure. They are best parameters to development algorithms suitably.

7 Performance comparison

In order to evaluate the NSGA_SR approach, we had seven experiments: five experiments included: flexibility, scalability, optimality, convergence and stability. Other two experiments included: how to create an initial population and number of candidate services. Experiments results were compared with results of three related work which were: AHP base service rank, SVD base servicer rank and W_SR. For ease of nomenclature we called them respectively with AHP_SR, SVD_SR and W_SR (SR for Service Rank).

7.1 Flexibility

In flexibility experiment, the rate of flexibility and feasibility of adding new services or features were investigated in proposed approach, AHP_SR and SVD_SR approaches.

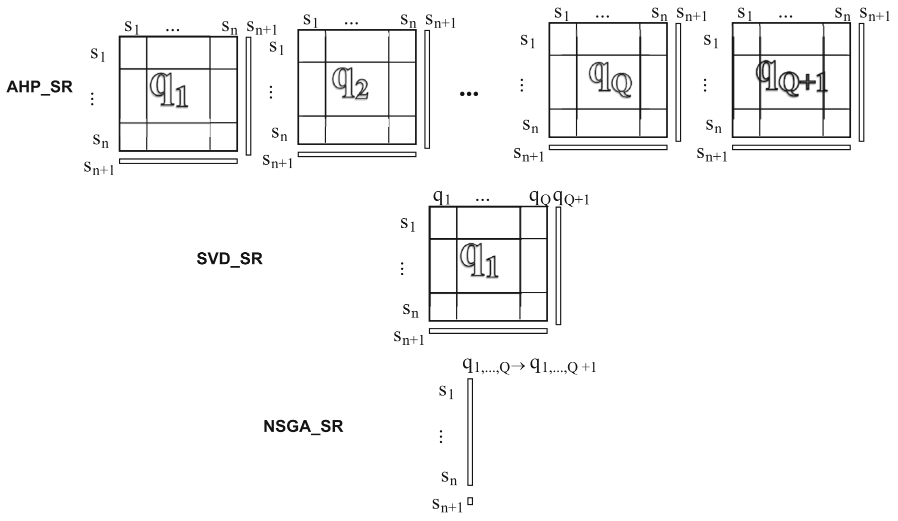

Fig. 4 Flexibility comparisons of NSGA_SR and previous approaches

This test tried to reveal the main concept of flexibility. More accurate results would be investigated in scalability test. In this section we assumed that, NSGA_SR has N services and Q qualitative attributes. Here, the aim was to find the amount of work that was required by adding a new service or an attribute. All ranking approaches at first required comparing the services.

As depicted in Fig. 4, AHP_SR compared services by using a matrix. It means that we put the QoS value in a matrix and to compare each two services we compared their QoS. In other hand, for each QoS, AHP_SR like q_i, made a $n \times n$ matrix (n is number of service) to compare all services according to q_i. Therefore if we want to compare Q QoS, we should have Q matrix. As it was mentioned, to add a new attribute, AHP_SR like (q_{Q+1}), added a $n \times n$ matrix and a new service, and added a row and a column to each matrix that consume time and space.

SVD_SR uses a matrix to compare the services that included: services in row and QoS attributes in column. Therefore in order to add a new attribute, a column must be added to base matrix and to add a new service, a row must be added to base matrix.

Opposite to the previous works, in NSGA_SR every service is displayed by a digit which is used to compare services. Therefore in NSGA_SR, and to add a new service, only a digit is added and to add a new attribute, only the way of calculating digits for services would be changed. It is understandable that why NSGA_SR is more flexible than other approaches.

Additionally, the other reason for high flexibility of NSGA_SR is that it uses both objective and subjective assessments and for the sake of application of MOO, adding a new objective function is easily possible and need no more time. For example any new feature (such as adding distance between users and services in the rankings) can be easily done as a third or fourth objective function.

7.2 Scalability

Scalability test is considered in two parts: scalability with increasing the number of users and scalability with increasing the number of services.

7.2.1 Scalability with increasing the number of users

In this test we had 10,000 fixed services, 1000 candidate services (K), 5 training users. The number of users were varied between 1 to 50 with step 10. For each scenario we had seven QoS that we reproduced randomly. The average results of 30 runs for all of the tasks have been reported. The results are shown in Fig. 5.

As depicted in Fig. 5, with increasing the number of users, NSGA_SR depicted more scalability than the other approaches. The heaviest computational part of the proposed approach is to find candidate services (first step or filtering) and calculate the fitness of chromosomes.

About candidate services, we can say NSGA_SR endures required time throughout filtering step to decrease required time in second and third steps. In first step NSGA_SR can find some candidate services and rank them instead of ranking all services and so decreases require time. About calculating the fitness; we can say that NSGA_SR uses a suitable coding method that causes to have linear increase with increasing the number of users. As mentioned in flexibility test, AHP_SR uses a matrix to compare services and it needs more time and have a nonlinear increase time with increase in users. Therefore it is less scalable than NSGA_SR. SVD_SR could not answer to more than 5 users in the same conditions that we had with NSGA_SR. W_SR performed well. However, because it compared services based on all QoS, compared to NSGA_SR, it

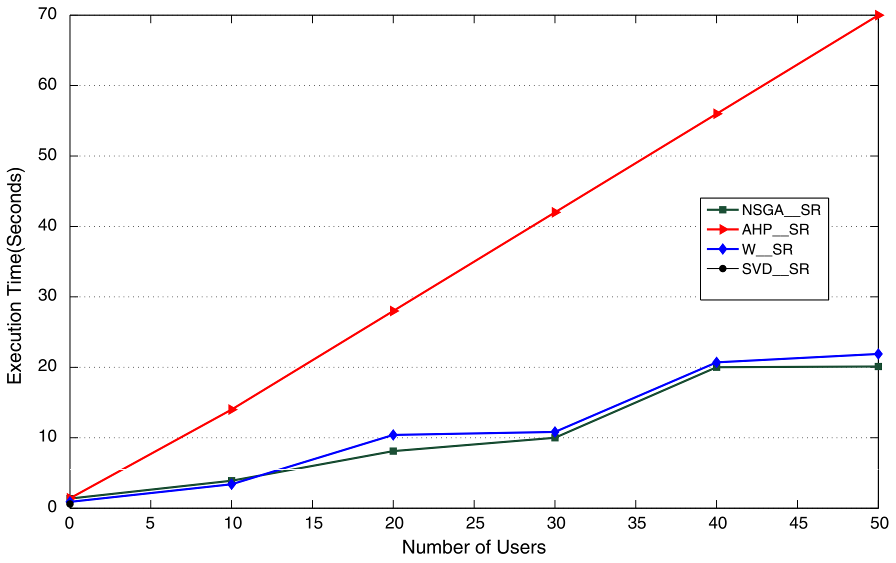

Fig. 5 Scalability with increasing number of users

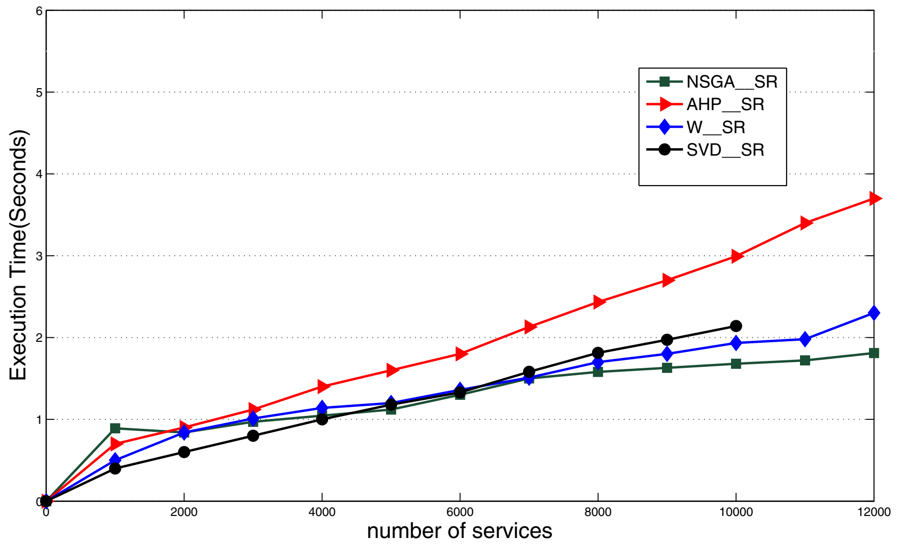

Fig. 6 Scalability with increasing number of services

needed more time. As mentioned NSGA_SR uses only one digit for each service and is more scalable than other approaches.

7.2.2 Scalability with increasing number of services

In this test we had 1000 fixed candidate services, 1 user and 5 training users and the number of services were varied from 1000 to 12,000 with 1000 step. The results are shown in Fig. 6.

As it was came in Fig. 6, execution time of AHP_SR, SVD_SR and W_SR initially were less than NSGA_SR, but with increasing in the number of services, it became more than NSGA_SR because of the flexibility issue. SVD_SR uses SVD technique and could not answer in more than 10,000 services. W_SR compare services based on all QoS and is less scalable than NSGA_SR. The main focus of Fig. 6 is when number of service is 1000. In this point, the number services are same with number of candidate services. When the number of services is 1000, filtering step do not run in NSGA_SR and W_SR. Therefore all available services rankings cause NSGA_SR to consume more time. However in more than 1000 services, filtering step can decrease execution time of NSGA_SR.

7.3 Optimality

NSGA_SR proposed approach tries to rank services in an optimum way. It means that services with highest ability to satisfy user's requirements receive best rank. We used Expert Choice [43] software to know optimum solution and compare ranking approaches. Expert Choice is ranking software that can be used in every solution. It

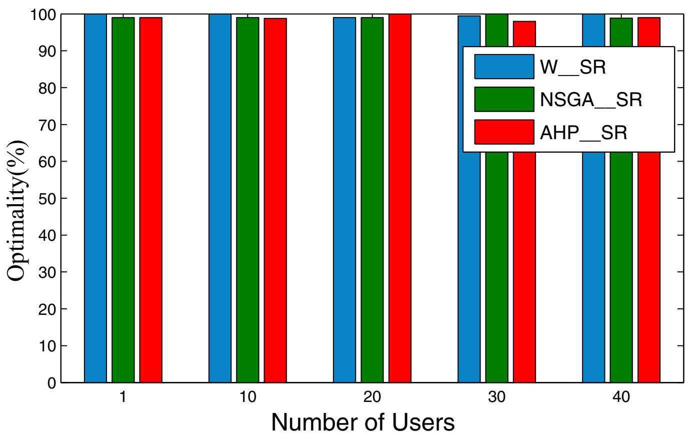

Fig. 7 Scalability with increasing number of services

is based on AHP technique and is responsive in low number of service. In this test we had 20 fixed candidate services, 100 services and 5 training users and the number of users were varied from 1 to 40 with step 10. The results are shown in Fig. 7. As depicted in Fig. 7, NSGA_SR achieved to more than 98 % optimality. W_SR achieve to about 100 % and AHP_SR achieved to about 98 %. The results show that all three approaches had acceptable rate of optimality.

As depicted in Fig. 7, NSGA_SR achieved to more than 98 % optimality. W_SR achieve to about 100 % and AHP_SR achieved to about 98 %. The results show that all three approaches had acceptable rate of optimality.

7.4 Convergence

NSGA_SR uses NSGA-II that is a MOO and evolution algorithm. Such algorithm has a main feature by which it can converge to best answer after some iteration. NSGA-II that we used must be converged to the best Pareto front after some iteration. If this convergence would not happen, it means that evolution of population has not been done properly. To ensure acceptable convergence of NSGA-II that used in NSGA_SR, we investigated values of best and average fitness in different generations.

In this test we used two main objective functions that were mentioned in Sect. 4 and the number of generation was from 1 to 500 with step 1. In each generation we calculated value of best fitness and average fitness. The results are showed in Fig. 8 and numerical values in Table 4.

As depicted in Fig. 8, two diagrams respectively show the values of best fitness and average fitness. The diagram of best fitness that goes always above average fitness has step changes. This diagram in higher generations gradually convergence to numbers and reach the first Pareto front. The cause of being step in this diagram is that always improved chromosome is replaced with worst chromosome and the best chromosomes remains for next generations. With these explanations the diagram can have a swing

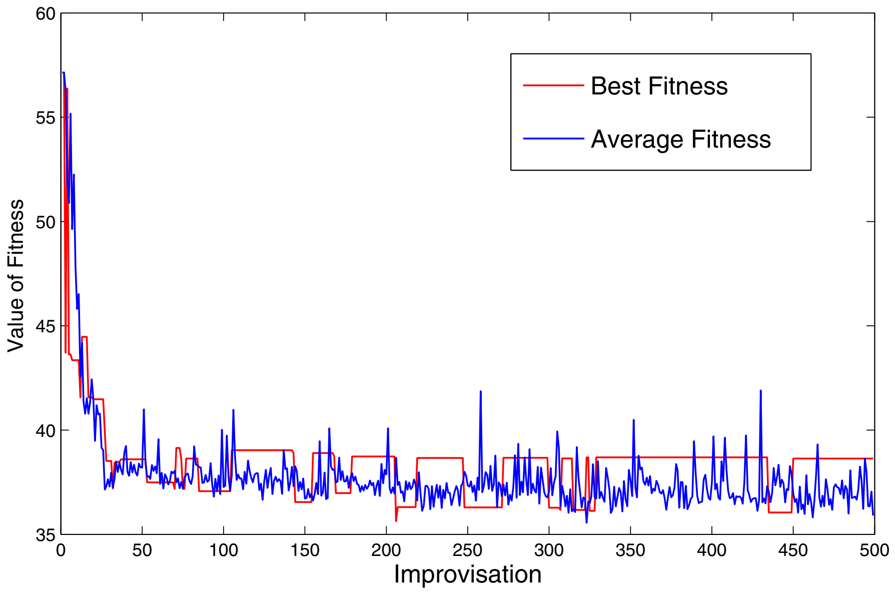

Fig. 8 Investigate the convergence of used NSGA-II to optimization goals

toward lower values (minimizing the objective functions). This stream in average diagram is similar a swing diagram because in each generation average chromosome value fitness has been calculated. The general trend of diagram shows evolution of population and convergence to best Pareto front. This convergence shows success in population evolution.

7.5 Stability

Stability is one of the important features of an evolution algorithm. In fact stability verifies that algorithm does not follow any certain condition and has not been achieved by accident. To investigate stability of NSGA-II that we used in second step of NSGA_SR, we run convergence test in some iteration and extract execution time during each iteration. If the difference between the values of fitness in different generations will not evident, it indicates good stability of the algorithm. This averment can satisfy by calculate variance and standard deviation because variance and standard deviation has an inverse relationship with stability. Therefore if variance and standard deviation have less value, the stability will be more. The results showed in Fig. 9 and numerical values are shown in Table 5.

As depicted in Fig. 9, the value of best fitness in different executions shows the same number, variance and standard deviation to be close to zero. Therefore the NSGA-II that was used in second step of NSGA_SR has stability. We do not compare NSGA_SR with other approaches in stability test, because only the approaches which are based on evolution algorithm can be investigated from the perspective of algorithm stability.

Table 4 Summary of results of used NSGA-II convergence test

Number of generation	1	50	100	150	200	250	300	350	400	450	500
Average fitness	51.5864	40.3050	37.9038	36.2312	36.1155	35.8507	38.4279	36.8326	37.3168	36.4628	35.6990
Best fitness	51.5864	39.5261	36.0946	38.7242	38.7242	35.6804	39.0748	35.5014	38.8724	38.8387	35.2856

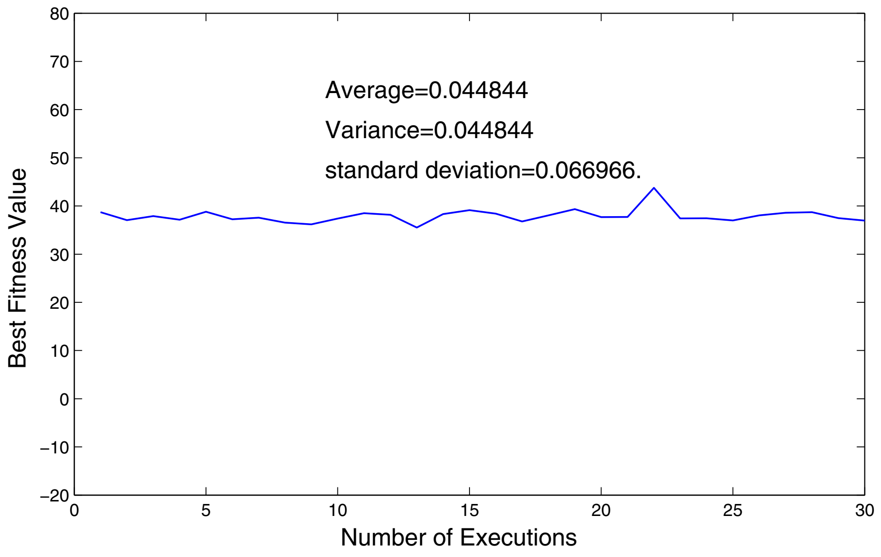

Fig. 9 Investigate the stability of used NSGA-II to optimization goals

Table 5 Summary of results of used NSGA-II stability test

Number of execution	1	5	10	15	20	25	30
Best fitness value	39.4484	41.0479	39.6258	38.1913	38.4916	37.1960	37.0277
Average = 0.0448440365							
Variance = 0.044844934							
Standard deviation = 0.066966							

7.6 How to create an initial population (random or intelligent)

As it was said in Sect. 5.2, we can initialize the population in two ways: randomly or intelligently. This subsection investigates the effect of creating initial population on speed of convergence. In intelligent method, first step result or candidate services were used as initial population. In random method, population initialized with random variables. The results are shown in Table 6.

As figured out in Table 6, we had 10,000 services and 100 candidate services and initial population were various from 1 to 300 with step 50. In each scenario we calculated the number of iteration to converge into best answers. To increase the accuracy of algorithm we run algorithm 30 times for each iteration and calculated the average level of results.

As shown in Table 6, because of little difference between the results, there was not any significant difference between the randomly initial populations or intelligent ones. We understand that there is no effect in initial population. So we decided to generate population in a combined method. 80 % generated randomly and 20 % generated intelligently.

Table 6 Effect of initial population in convergence

Number of initial population	1	50	100	150	200	250	300
Random	1	3	7	10	13	17	21
Intelligent	1	3	6	9	14	16	22

7.7 Number of candidate service (K)

This section investigates execution time with different number of candidate services (K). It includes two tests: effect of K with increasing in the number of users and effect of K with increasing in the number of services.

7.7.1 Effect of K with increasing in number of user

In this test we had 20,000 fixed services and the number of users was various from 1 to 50 with step 10. We collected the results for 4 different value of K (100, 1000, 12,000 and 20,000). For each scenario we had seven QoS. The results are shown in Fig. 10.

As shown in Fig. 10, this test is extension of scalability test with increasing in the number of users. AHP_SR and SVD_SR ranked all services and had not filtering step. So variables K do not have any effect on their performance. Therefore execution time in these approaches do not change with different values of K. SVD_SR cannot answer

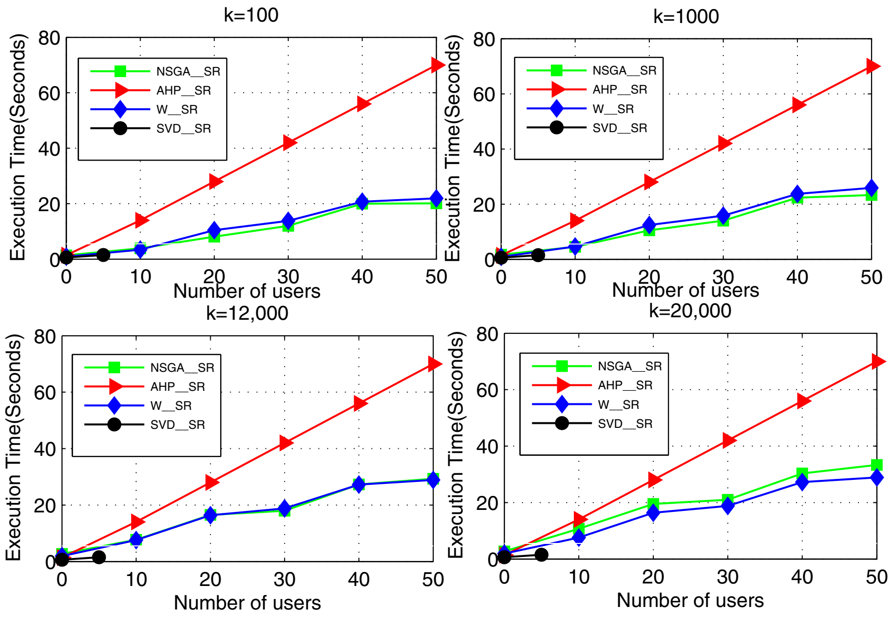

Fig. 10 Effect of K with increasing in number of users

to more than five users. On the contrary, NSGA_SR and W_SR need more time to execute by increase in K. But the required time with any K is less than the time of other approaches. NSGA_SR and W_SR are same. But NSGA_SR can have less execution time in lower variable of K and oppositely; W_SR can have less execution time in higher value of K.

W_SR ranks services based on only objective assessment. But NSGA_SR ranks services based on both objective and subjective assessments. Thus, in cases that user has a ranking request based on both objective and subjective assessments, application of NSGA_SR is better and in cases that user has a ranking request based on objective assessment, W_SR is better.

7.7.2 Effect of K with increasing in number of services

In this test we had 20 fixed users and the number of services was ranged from 1 to 24,000 with step 5000. We collected the results for four different value of K (100, 1000, 12,000 and 20,000). For each scenario we had seven QoS. The results are presented in Fig. 11.

As figured out in Fig. 11, AHP_SR and SVD_SR rank all services and have not filtering step. Thus, variable K does not have any effect on their performance. Therefore execution time in these approaches do not change with different values of K. SVD_SR cannot answer to more than 10,000 services. On the contrary, NSGA_SR and W_SR with increase in K need more time to execute. But the required time with any K is less than other approaches time. NSGA_SR and W_SR are same. But NSGA_SR can have less execution time in lower variable of K. However W_SR can have less execution time in higher value of K.

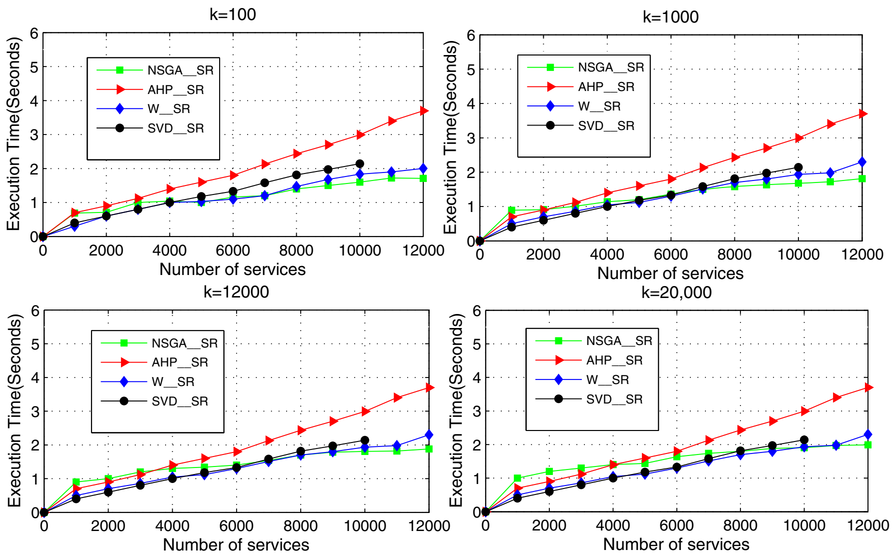

Fig. 11 Effect of K with increasing in number of services

W_SR ranks services based on only objective assessment. But NSGA_SR ranks services based on both objective and subjective assessments. So in cases that user has a ranking request based on both objective and subjective assessments, using the NSGA_SR is usable.

Having to say, the proposed NSGA_SR ranking approach has some advantages:

- Application of both objective and subjective assessments
- Service ranking based on both essential and non-essential requirements
- Receiving users requirements in Boolean, numerical or range,
- No need to normalize the input values,
- Accountability in lack of feedback information of training users,
- Flexibility by increasing in services or users,
- Scalability by increasing in services or users,
- Having convergence and stability.

8 Conclusion and future work

Changing needs of users parallel to the development of Internet–based technologies causes a challenge to fulfill the ever changing demands of the users. One of the new technologies to provide the services on the Internet is cloud computing services. Cloud computing, by providing an improved business model based on subscription, enables service providers to have various services with different quality attributes. But in such a vast space, finding a suitable service according to the changing needs of users has become a major challenge. Ranking systems, by focusing on user's needs and values of quality attributes of various cloud services, enable the users to select most appropriate service. Building a high qualitative ranking system requires a method to compare the services with the ability to choose the ones that meet the user's quality constraints.

The aim of ranking cloud services based on both objective and subjective assessments on the one hand and both essential and non-essential requirements on the other hand let us to consider service ranking in cloud computing. In this regard we proposed NSGA_SR as a ranking approach. In fact NSGA_SR is a MOO problem with two objective functions that can be solved by NSGA-II. This proposed approach has three steps. NSGA_SR approach for service ranking, at first uses a filtering system in order to extract the candidate services that have ability to satisfy user's all requirements. Afterward it finds several optimum pseudo services by NSGA-II. At the end all candidate services would be ranked based on their distance with pseudo services by application of clustering algorithm.

In the future with the aim of selecting and composing best services, we will extend our approach for service composition that need great attention to focus on user's requirements. We can also extend our approach by other evolution algorithms and then compare the results. We can also apply users' similarity as subjective assessment in service ranking. Also we can use machine learning in cloud service ranking based on QoS. Aside from all of these we can propose a new monitoring method to achieve the values of quality attributes and so rank services that help to ensure to accuracy of QoS values.

References

1. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 25(6):599–616
2. Ding S, Yang S, Zhang Y, Liang C, Xia C (2014) Combining QoS prediction and customer satisfaction estimation to solve cloud service trustworthiness evaluation problems. Knowled Based Syst 56:216–225
3. Buyya R, Vecchiola C, Selvi C (2013) Cloud computing architecture 4:111–140
4. Chen CT, Lin KH (2010) A decision making method based on interval valued fuzzy sets for cloud service evaluation. In: 4th international conference on new trends in information science and service science (NISS), Gyeongju
5. Hao Y, Zhang Y, Cao J (2010) Web services discovery and rank: an information retrieval approach. Future Gener Comput Syst 26(8):1053–1062
6. Stojanovic MD, Bostjancic Rakas SV, Acimovic Raspopovic VS (2010) End-to-end quality of service specification and mapping: the third party approach. Comput Commun 33(11):1354–1368
7. Chen L, Feng Y, Jian W, Zheng Z (2011) An enhanced QoS prediction approach for service selection. In: IEEE international conference on services computing, Washington, DC
8. Garg SK, Versteeg S, Buyya R (2013) A framework for ranking of cloud computing services. Future Gener Comput Syst 29(4):1012–1023
9. Itani W, Ghali C, Kayssi AI, Chehab A (2011) Accountable reputation ranking schemes for service providers in cloud computing. In: International conference on cloud computing and services science, Frank Leymann
10. Katsaros G, Subirats J, Fito JO, Guitart J, Gilet P, Espling D (2013) A service framework for energy aware monitoring and VM management in clouds. Future Gener Comput Syst 29(8):2077–2091
11. Chan J, Chieu T (2010) Ranking and mapping of applications to cloud computing services by SVD. In: IEEE/IFIP network operations and management symposium workshops (NOMS Wksps), Osaka
12. Katchabaw MJ, Lutfiyya HL, Bauer MA (2005) Usage based service differentiation for end-to-end quality of service management. Comput Commun 28(18):2146–2159
13. Zhang R, Zettsu K, Kidawara Y, Kiyoki Y (2012) Web service ranking based on context. In: Second international conference on cloud and green computing, Xiangtan
14. Segev A, Toch E (2009) Context based matching and ranking of web services for composition. IEEE Trans Serv Comput 2(3):210–222
15. Qu L, Wang Y, Orgun MA (2013) Cloud service selection based on the aggregation of user feedback and quantitative performance assessment. In: IEEE 10th international conference on services computing, Santa Clara
16. Alarcon-Rodriguez (2009) A multi objective planning framework for analysing the integration of distributed energy resources. In: A thesis presented in fulfilment of the requirements for the degree of Doctor of Philosophy, Institute of Energy and Environment, Department of Electronic and Electrical Engineering, University of Strathclyde
17. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
18. Almasri E, Mahmoud QH (2008) Investigating web services on the world wide web, refereed track: web engineering—web service deployment, New York
19. Sobel W, Subramanyam Sh, Sucharitakul A, Nguyen J, Wong H, Klepchukov A, Patil Sh, Fox O, Patterson D (2008) Cloudstone: multi platform, multi-language benchmark and measurement tools for web 2.0. In: Proceedings of cloud computing and its applications, CCA-08. http://www.cca08.orgpapers.php
20. Miller P (2009) The importance of benchmarking clouds. CloudHarmony. http://www.cloudharmony.com
21. Li A, Yang X, Kandula S, Zhang M (2010) CloudCmp: comparing public cloud providers. In: IMC '10 proceedings of the 10th ACM SIGCOMM conference on internet measurement, New York
22. Li A, Yang X, Kandula S, Zhang M (2011) CloudCmp: shopping for a cloudmade easy. HotCloud'10 proceedings of the 2nd USENIX conference on Hot topics in cloud computing, USENIX Association Berkeley, CA
23. Abubakr T (2011) Tools for benchmarking the cloud: Cloud Sleuth. https://www.cloudsleuth.net

24. Luo C, Zhan J, Jia Z, Wang L, Lu G, Zhang L, Xu C, Sun N (2012) CloudRank-D: benchmarking and ranking cloud computing systems for data processing applications. Front Comput Sci 6(4):347–362
25. Skoutas D, Sacharidis D, Simitsis A, Sellis T (2010) Ranking and clustering web services using multicriteria dominance relationships. IEEE Trans Serv Comput 3(3):163–177
26. Rehman U, Hussain OK, Parvin S, Hussain FK (2012) A framework for user feedback based cloud service monitoring. sixth international conference on complex, intelligent and software intensive systems, Palermo
27. Choudhury P, Sharma M, Vikas K, Pranshu T, Satyanarayana V (2012) Service ranking systems for cloud vendors. Adv Mater Res 433:3949–3953
28. Lejeune J, Arantes L, Sopena J, Sens P (2012) Service level agreement for distributed mutual exclusion in cloud computing. In: 12th IEEE/ACM international symposium on cluster, cloud and grid computing, Ottawa
29. Wright State University (2010) Cirrocumulus: a semantic framework for application and core services portability across heterogeneous clouds, project at Kno–e–sis Center at Wright State University. http://knoesis.org/node/70
30. Zheng Z, Wu X, Zhang Y, Lyu MR, Wang J (2013) QoS ranking prediction for cloud services. IEEE Trans Parall Distrib Syst 24(6):1213–1222
31. Zheng Z, Wu X, Zhang Y, Lyu MR (2010) CloudRank: a QoS-driven component ranking framework for cloud computing. In: 29th IEEE international symposium on reliable distributed systems
32. Dikaiakos MD, Zeinalipour Yazti D (2004) A distributed middleware infrastructure for personalized services. Comput Commun 27(15):1464–1480
33. CSMIC (2011) Service measurement index version 1.0. Carnegie Mellon University Silicon Valley, Moffett Field
34. Jahani A, Mohammadkhanli L, Razavi SN (2014) W_SR A QoS based ranking approach for cloud computing. Comput Eng Syst 3(2):55–62
35. Durao F, Carvalho J, Fonseka A, Garcia V (2014) A systematic review on cloud computing. J Supercomput 68(3):1321–1346
36. Sun L (2016) Cloud-FuSeR: fuzzy ontology and MCDM based cloud service selection. Future Gener Comput Syst 57:42–55
37. Almulla M, Yahyaoui H, Al-Matori K (2015) A new fuzzy hybrid technique for ranking real world web services. Knowl Based Syst 77:1–15
38. Singh S, Chana I (2014) QRSF: QoS-aware resource scheduling framework in cloud computing. J Supercomput 71(1):241–292
39. Chen JH (2015) A hybrid model for cloud providers and consumers to agree on QoS of cloud services. Future Gener Comput Syst 50:38–48
40. CSMIC (2011) CSMIC SMI overview diagram TwoPointOne. Carnegie Mellon University Silicon Valley, Moffett Field
41. Raisanen V (2004) Service quality support-an overview. Comput Commun 27(15):1539–1546
42. Yau SS, Yin Y (2011) QoS-based service ranking and selection for service based systems. In: IEEE international conference on services computing, Washington, DC
43. Ishizaka A, Labib A (2009) Analytic hierarchy process and expert choice: benefits and limitations. OR Insight 22(4):201–220