Cluster Computing (2022) 25:485–502
https://doi.org/10.1007/s10586-021-03418-w

Efficient cloud service ranking based on uncertain user requirements

Mohammad Hossein Nejat[1] · Homayun Motameni[1] · Hamed Vahdat-Nejad[2] · Behnam Barzegar[3]

Received: 25 December 2020 / Revised: 21 July 2021 / Accepted: 15 September 2021 / Published online: 29 September 2021

Abstract
In a cloud computing environment, there are many providers offering various services of different quality attributes. Selecting a cloud service that meets user requirements from such a large number of cloud services is a complex and time-consuming process. At the same time, user requirements are sometimes described as uncertain (sets or intervals), something which should be taken into account while selecting cloud services. This paper proposes an efficient method for ranking cloud services while accounting for uncertain user requirements. For this purpose, a requirement interval is defined to fulfill uncertain user requirements. Since there are a large number of cloud services, the services falling outside the requirement interval are filtered out. Finally, the analytic hierarchy process is employed for ranking. The results evaluate the proposed method in terms of optimality of ranking, scalability, and sensitivity analyses. According to the test results, the proposed method outperforms the previous methods.

Keywords Cloud service · Quality of service (QoS) · Service selection · Service ranking · Analytic hierarchy process (AHP)

1 Introduction

Cloud computing allows for the use of resources as services on the Internet with no need for any specialized skills or investment costs [4]. Cloud services provide extensive and flexible computing and storage capacities that have come in handy for many companies, especially small and medium-sized enterprises, SMEs [11, 26]. On the other hand, cloud providers offer services with various quality attributes [1]. The growing variety of services has made it difficult for cloud users to select the most proportionate service to their non-functional requirements (i.e. quality attributes) [43].

A cloud layer named the broker is used to collect and manage service information [25]. This layer is mainly responsible for helping users to select services based on the assessment of quality attributes [26]. A cloud broker receives user requirements and searches for the most appropriate service within a mass of services [32]. It should also be able to rank cloud services quickly based on the user requirements.

Selecting specific attributes necessary for the assessment of cloud services is a challenge [23, 28, 42]. Every quality attribute affects ranking; therefore, multi-attribute decision-making (MADM) methods [8] can be an appropriate ranking option [11]. The least time complexity and the highest robustness are the challenges that have attracted the attention of researchers to rank cloud services [13]. Analytic hierarchy process, AHP [31] is one of the most recognized MADM methods that has yielded acceptable ranking results [1, 3, 11, 39]. In a hierarchical structure, AHP draws a pairwise comparison of cloud services based on each quality attribute [17]. Within the service measurement index cloud (SMICloud) framework,

✉ Homayun Motameni
motameni@iausari.ac.ir

Mohammad Hossein Nejat
nejatmohammadhossein@gmail.com

Hamed Vahdat-Nejad
vahdatnejad@birjand.ac.ir

Behnam Barzegar
barzegar@iauns.ac.ir

[1] Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran

[2] Department of Computer Engineering, University of Birjand, Birjand, Iran

[3] Department of Computer Engineering, Babol Branch, Islamic Azad University, Babol, Iran

service ranking is firstly performed based on the AHP and user requirements [11]. However, user requirements are not clear and certain in most cases [36].

Uncertainty in user requirements occurs when the user cannot make an accurate decision about cloud quality attributes. The variety in the cloud environment and the failure in recognizing some of the requirements have made it difficult for the cloud user to accurately express the quality requirements. Parameters such as the exact speed of the processor, the exact number of processors required, the exact bandwidth, and the exact elasticity may not be precisely determined by the users [40]. Therefore, uncertainty in the requirements of the cloud user can be considered from the two aspects of the importance of quality attributes and the amount required by the user for each attribute. To determine the importance of each attribute, the user can enter linguistic variables instead of the exact weight of the attribute. Besides, the user must be able to enter their required values imprecisely. In other words, the required value of each attribute is sometimes vague (e.g. high security), or an interval (for instance, the monthly cost is below $1). For this purpose, researchers have used various methods by integrating AHP with fuzzy theory to estimate uncertain requirements [22, 38, 39, 44]. Nevertheless, a great deal of user requirement information is lost in fuzzification and defuzzification processes. To solve this problem, Abdel-Basset et al. [1] employed the neutrosophic method [35] and performed decision-making with higher accuracy. All of the methods face the same constraint that is the necessity of comparing and ranking all services based on user requirements. Given the fact that fuzzy and neutrosophic computation processes are time-consuming, the response time of ranking systems with a large number of cloud services will increase, dramatically.

This paper proposes the fast cloud service ranking (FCSR) method to reduce the response time of ranking cloud services. Since many of the services are usually quite far from user requirements, this paper proposes the concept of requirement interval based on uncertain user requirements to narrow down the search domain. Therefore, the less relevant services are filtered out. Then, AHP, the performance of which has been proven, is employed for ranking. If no services meet user requirements completely, a service with the highest similarity to user requirements will be selected. This prevents the service deletion process from becoming strict; therefore, the services with the most proportional quality will be selected. The FCSR method has been assessed in different scenarios such as optimality, scalability, and sensitivity. According to the results, the FCSR method shows a shorter execution time and a higher accuracy rate than other methods.

The rest of the paper is structured as follows. Section 2 presents a brief literature review. Section 3 introduces the FCSR method proposed to rank cloud services. A comparison is drawn between FCSR and other existing methods in Sect. 4. Finally, the conclusion and future works are discussed in Sect. 6.

2 Related work

With the increasing number of cloud services, several attempts have been made at assessing and ranking them [37]. Many researchers have analyzed service performance for different purposes such as e-commerce and web applications in a user-independent manner. For instance, a service assessment framework is proposed in [30] for service selection based on the combination of service cost and performance. Likewise, Iosup et al. assess cloud service performance in high workloads by analyzing CPU, I/O, memory, and network resources [15]. However, these metrics are not sufficient to assess services.

Since users are the end consumers of cloud services, selecting the optimal service from a variety of services is a major challenge. In this regard, a recommender system is proposed [14] to create a ranked list of services based on user requirements. In this system, service assessment is limited to their low-level performance metrics including processing, memory, bandwidth, and cost. The comparison metrics of recommender systems performance have been developed within the CloudCmp framework [24]. To this end, cloud service comparison metrics have been proposed including scaling latency, elasticity, response time, and virtual machine cost. These metrics focus on the profit-loss measurement of cloud services and disregard many quality attributes such as user feedback.

Given the importance of cloud service assessment, the Cloud Services Measurement Initiative Consortium, CSMIC [34] proposed the service measurement index (SMI) framework to standardize quality attributes. This framework was proposed based on the International Organization for Standardization (ISO) with a hierarchical structure consisting of 7 attributes and 51 sub-attributes. Since SMI considers all cloud attributes collectively, a major part of it is not utilized by users in practice. To identify the prominent attributes, the quality attributes proposed by SMI were prioritized in 7 attributes and 21 sub-attributes in [47]. Attribute measurement is performed through questionnaires. In fact, each attribute is measured based on its structure and score. However, the measurement might not be accurate because the responses are initialized optimistically and pessimistically. In addition, researchers have been drawn toward the classification of quality attributes as necessary or unnecessary [11]. Dan et al. [26] has introduced a step called refinement to select services that satisfy all essential user requirements. Also,

Jahani et al. [18] added a filtering step to the ranking of services. The aim is to satisfy the user's requirements. In [7], the linear equations used to select appropriate services based on the user requirements. For this purpose, an objective function used as a linear equation. The constraints specified in the user requirements used to maximize the objective function and determining the scope of requirements and eliminating disproportionate services. However, if a service is slightly different from a user requirement, it will be deleted strictly. In this case, the cloud broker may often provide no services meeting user requirements.

Parallel to identifying essential quality attributes, several research studies have been conducted on service ranking [29]. SMICloud framework relies on AHP to compare services with regard to quality attributes and user requirements [11]. In recent years, many efforts have been made to improve the SMICloud framework. The solution that most researchers have considered is the use of integration of MADM methods. In paper [45], an integrated method for ranking cloud services presented in the SMICloud framework. In this method, the best worst method (BWM) used to weigh the quality attributes. Cloud services ranked using the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) method. A case study with eight cloud services and nine quality attributes was considered. Their proposed method was compared with the AHP method in terms of execution time. Kumar et al. [21] introduced the optimal service selection and ranking of cloud computing services (CCS-OSSR) method. They also combined TOPSIS and BWM methods to reduce computational complexity. In a case study, they ranked 11 cloud services based on five quality attributes. Sensitivity analysis was used to evaluate the proposed results. In these methods, the performance of the proposed method in high demand or increasing the number of services has not been evaluated. On the other hand, due to the fact that the SMI structure is hierarchical, the performance of the AHP method (according to the hierarchical structure of the AHP method) is better suited than other MADM methods. According to [13], the main weakness of the TOPSIS method compared to the AHP method is the high sensitivity of TOPSIS. Also, the uncertain requirements of users have not been taken into account.

Although SMICloud has attracted researchers, it is unable to consider uncertainty in user requirements and attribute values of services. The fuzzy sets theory [47] was employed to solve this issue [33]. This method benefits from AHP in giving weight to quality attributes. Based on expert feedback on cloud services and user requirements, fuzzy AHP (FAHP) was proposed [46] for cloud service ranking. Similarly, Cloud-FuSer was proposed to select top-k services by considering vague user requirements [38]. This method introduces certain parameters of TOP (Top rank match count) and MATCH (match count) for ranking assessment. Moreover, the fuzzy AHP is used by Tajvidi et al. [39] to compare quality attributes. Then the resultant weights of each quality attribute and values of cloud services were employed to perform ranking through the AHP method. However, the problem is that fuzzy sets theory imposes too much computation processing on the system when it is fed uncertain values. At the same time, the ranking accuracy decreases due to fuzzy transforms. In [12], the quality of services is aggregated with user feedback to improve ranking accuracy by applying indeterminacy parameters to fuzzy sets. This is achieved through sensitivity analysis and Spearman's rank correlation analysis, in which the ranking results of the proposed method are compared with MADM basic techniques.

To improve ranking accuracy AHP is integrated with neutrosophic techniques in the NMCDA method [1]. The neutrosophic set added truth, falsity, and indeterminacy to the fuzzy set membership functions in order to improve the accuracy of fuzzy set results to get user requirements with a lower error rate. Tiwari and Kumar [41] compared the efficiency of integrating MADM methods with neutrosophic in terms of accuracy, sensitivity, and execution time in ranking services. The results show that the integration of AHP with the neutrosophic is robust, but in a condition where the number of services is high, the time complexity increases.

All these works draw pairwise comparisons between services based on each attribute for all cloud services. This translates into a time-consuming process when a large number of services are being examined. Since there are too many cloud services nowadays and that the execution time of AHP-based methods depends on the number of services, such processes have long response times. In this study, the FCSR method is proposed to, first, filter inappropriate services by considering uncertain user requirements, and, second, rank services in real-time by reducing ranking computations.

3 Fast cloud service ranking (FCSR)

In this section, the FCSR ranking system describes step by step. In addition, the FCSR time complexity is calculated.

3.1 Ranking cloud services by FCSR

It is a time-consuming task to select a service from numerous cloud services with various quality attributes to meet user requirements. FCSR method first eliminates a considerable number of services which are slightly proportionate to user requirements. Then the ranking operation

is performed to compare a few services rather than all of them. This process reduces the response time.

At first, the user requirement is encoded to the proposed requirement interval, and similarly, cloud service encoding is performed. By comparing the encodings, many services might be filtered out. The remaining services are called candidate services. Consequently, the candidate services are ranked based on user requirements and quality of service. According to Algorithm 1, the FCSR method consists of five steps.

Algorithm 1: FCSR

Input: User requirement, Service set with quality values
Output: Ranked services
1. Encoding quality attributes
2. Converting user requirement into query
3. Creating user requirement interval
4. Creating quality interval of each cloud service
5. Filtering and Ranking services

The FCSR method receives a set of services with quality values and user requirements as inputs. The output includes the ranking of proportional services.

3.1.1 Step 1: encoding quality attributes

Binary encoding is applied to each quality attribute based on its type and value. Generally, quality attributes are classified as numeric, set, and interval categories. Numeric attributes such as a disk capacity can include discrete numbers. For encoding, the range of possible values is divided into n intervals, and n can be regulated by the system. Therefore, n bits are used for encoding the values so that all bits are zero except the bit that indicates the interval. The set type includes those attributes that can have specific values such as portability, which indicates the names of platforms supporting the cloud service. A few bits corresponding to the maximum possible values are employed to encode this type of attribute. The corresponding bits with the values of attribute are considered 1, whereas other bits are considered zero. Some of the quality attributes are of the interval type. For instance, elasticity is regarded as an interval (ranging from 20 to 200 s). To encode, the range of possible values is partitioned into n intervals. Then the quality attribute is encoded with n bits, each of which corresponds to one interval. If the service value interval overlaps with each of these subintervals, that bit is 1, otherwise, it is 0.

Finally, since different quality attributes might have codes of different lengths, the longest code (L) is found. Then the zero bits are added to the right side of other attributes to make their lengths equal to L.

3.1.2 Step 2: converting user requirement into query

For each service, quality factors (e.g. response time), economic factors (e.g. maintenance costs), and technical factors (e.g. availability) are provided according to the application of that service [10]. Therefore, in requirement gathering, the requested service type is selected by the user so that the services of the same type are ranked. Types of services are provided as education, management, accounting, mobile services, etc. In addition, a user should insert the attribute values and the importance of each attribute in requirement gathering.

The user is asked the importance of each quality attribute, which is an integer ranging between 0 and 4 (unimportant, somewhat important, important, very important, and extremely important). If the attribute is unimportant to the user, it will not be used in encoding user requirements and cloud services. Then, the value required by the user for that attribute is entered. It can be a numeric, set, or interval value. Figure 1 shows an example of how to receive user requirements for a few attributes.

Afterward, the user requirement is converted into the query format. Formula 1 shows the query structure of the mth user.

Service Type:

Importance of Accountability: …..	Required value: …
Importance of Agility: …..	Required value: …
Importance of Capacity CPU: …..	Required value: …
Importance of Capacity RAM: …..	Required value: …
Importance of Capacity Disk: …..	Required value: …
Importance of Elasticity: …..	Required value: …
Importance of security: …..	Required value: …
⋮	

Relative importance value:

Unimportant=0

Somewhat important=1

Definitely important=2

Much important=3

Extremely important=4

Fig. 1 Receiving user requirements

$$Q_{m,serviceType} = \{(q_1, value_1, type_1), (q_2, value_2, type_2), \ldots, (q_i, value_i, type_i), \ldots, (q_n, value_n, type_n)\}. \quad (1)$$

In this formula, n indicates the number of quality attributes, and q_i shows the importance of the ith attribute to the user. In addition, $value_i$ shows the value that the user requires for the ith attribute, $type_i$ indicates the type of the ith quality attribute based on user requirement, and *serviceType* indicates The type of service specified by the user. The attributes are sorted from left to right in the order of value. For instance, consider capacity CPU, capacity memory, capacity disk, and elasticity attributes. Figure 2a shows their priorities from the user's perspective and the required code of each attribute; Fig. 2b indicates the user requirement code, which is saved in the array named *sorting*.

In addition, the weight of each attribute (w), the importance of which was determined by the user, is measured. According to the query, each attribute has an importance value. Given the importance of quality attributes, their weights are measured in a normal form. For instance, the weights of attributes shown in Fig. 2 are presented in Table 1.

3.1.3 Step 3: creating user requirement interval

In some cases, the user requirement code might have two or more digits of 1 for a cloud attribute. This occurs when the quality attribute has a set or an interval value, for instance, the user might select middle and advanced levels for the security attribute from the set of the basic, middle, and advanced levels, or the user might select the 20–120 milliseconds (ms) interval for elasticity when the system creates 50 ms length intervals in the possible range between 0 and 200 ms. In this case, if the quality attribute is of the set type, the number of bits will be equal to the number of possible values. If the user requirement is proportionate to a value of the set, the corresponding bit will be 1, otherwise, it is given 0. For instance, if the user requirement includes middle and advanced levels in the three-level security metric, the required code of the security attribute will be 110. Moreover, it is sometimes necessary to determine the importance of each value for the user. The importance of values is then employed to determine the location of the bits. For instance, for the portability attribute if the user selects the importance of platforms as IOS > Android > Linux > Windows, the bits of portability are considered for the user in order of importance from left to right. If the user needs IOS and Android, then the portability attribute requirement code will be 1100.

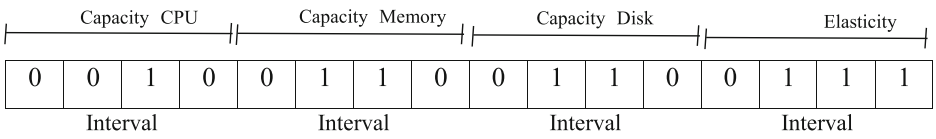
Encoding interval quality attributes follows a different procedure. For each bit of 1, a separate code is considered for the ith attribute. Hence, each simple code of the attribute has a bit of 1, while the other bits of that attribute are 0. For instance, if the values determined for the elasticity attribute vary within [0–50), [50–100), [100–150), and [150–200) and the user requirement value ranges within 20–120, then the binary requirement value will be 0111. As Fig. 3 shows, $attribute_i$ is converted into three simple codes.

This conversion is performed for all of the interval attributes having multiple values of 1 in their code. In this case, it is possible to consider the smallest and largest values for the lower and upper bounds of the interval. In the example demonstrated in Fig. 3, 0001 and 0100 are the smallest and largest codes of the, and there is no need to keep 0010. Similarly, the codes of interval attributes of the *sorting* array which has multiple values of 1 are converted into simple codes. Then the *sorting* array is converted into several separate codes by repeating the single-value or non-interval attributes. The number of codes is as many as the maximum number of created simple codes of attributes. Figure 4 shows the structure of conversion. Accordingly, the *sorting* array has 4 interval attributes, each of which includes 4 bits. Except for the most valuable quality attribute (capacity CPU) in which one bit is 1, other quality attributes have multiple bits of 1. The simple codes are

Fig. 2 Creating user requirement and query

Capacity CPU= (Extremely important, 0010, Interval)
Capacity memory= (Much Important, 0110, Interval)
Capacity disk= (Definitely important, 0110, Interval)
Elasticity = (Somewhat important, 0111, Interval)

(a) A. The query created in accordance with the user requirement



(b) B. User requirement code in the *sorting* array along with the type of each attribute

Table 1 Measuring the weight of each quality attribute

Quality attribute	Importance	Relative importance value	Weight
Capacity CPU	Extremely important	4	0.4
Capacity memory	Much important	3	0.3
Capacity disk	Definitely important	2	0.2
Elasticity	Somewhat important	1	0.1

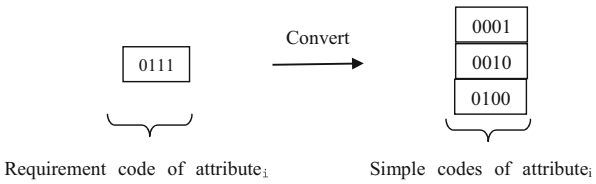


Fig. 3 Converting an interval requirement of an attribute into simple codes

created by considering the smallest and largest values for interval attributes having multiple bits of 1. Then the single-value requirement code is created by repeating the value of the most valuable quality attribute for the lower and upper bounds of the interval.

Afterward, the interleaving process [26] is performed for each code to first put more valuable bits and then less valuable bits of attributes together. Figure 5 shows how to interleave the lower code of the single-value requirement interval shown in Fig. 4.

Each arrow shows where the bit is placed. Since the values are binary, the interleaving method is employed to consider the importance of attributes bit by bit. As a result, the user requirement is saved as two numbers, one of which indicates the lower bound, whereas the other shows the upper bound of the user requirement interval. In this case, the quality interval has been created with respect to user requirements.

3.1.4 Step 4: creating quality interval of each cloud service

Different conditions of cloud environments cause variations in the values of quality attributes of cloud services. Therefore, the quality intervals of services can be created in this step through the same procedure as user requirements (the previous step). For this purpose, the services with the same type required by the user are encoded based on their quality values and in order of quality importance from the user's perspective. According to Step 3.1, the quality interval of services is created based on user requirements.

3.1.5 Step 5: ranking services

Services are ranked and selected in this step. The user requirement is created as an interval in Step 3, and each cloud service has a quality interval in accordance with Step 4. In Step 5, services are classified into three categories, the first of which includes the services that have nothing in common with the requirement interval. The services of this category are disproportionate to user requirements and are thus eliminated. The second category includes the services which either cover user requirements thoroughly or fall exactly within the requirement interval. The services of this category are included in the Candidate set. The third category includes the services that their interval overlaps the user requirement interval. The services of this category are added to the Secondary set.

If the number of services existing in the Candidate set (second category services) exceeds one service, the AHP is employed for service ranking. For this purpose, the importance of each quality attribute is determined by weights (w) in the hierarchical structure of SMICloud in the second step. At the same time, cloud service ranking in the AHP requires pairwise comparisons of cloud services. The method proposed in [42] is employed to quantify the set-valued attributes (e.g. portability), numeric attributes (e.g. disk

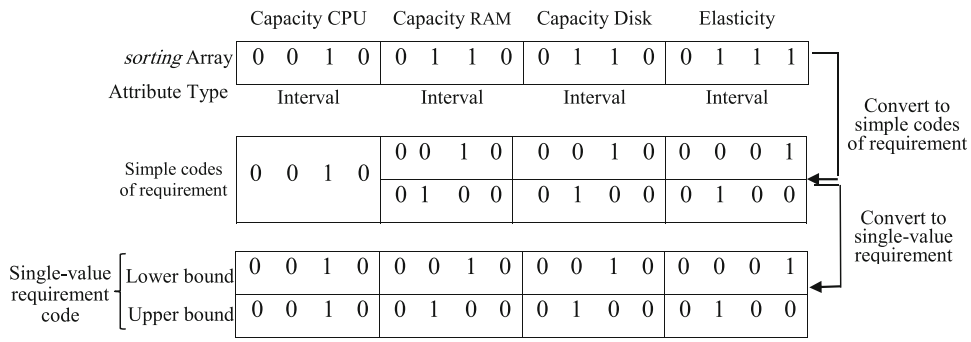


Fig. 4 Converting one row of the *sorting* array to single-value requirement codes

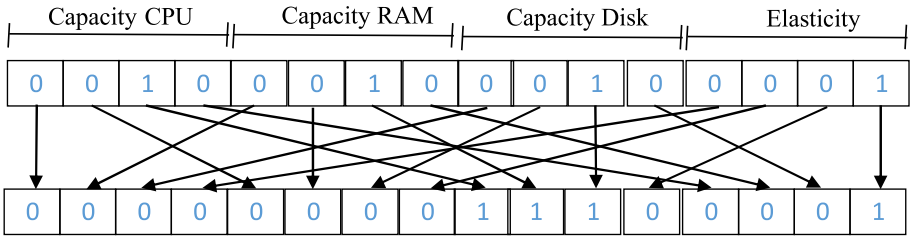


Fig. 5 Interleaving the binary query

capacity), and interval-valued attributes (e.g. elasticity). The comparison of two services s_i and s_j is shown as $\frac{s_i}{s_j}$. The value of the ith service (v_i), the value of the jth service (v_j), and the requirement value of that attribute (v_r) should be available to compare two services s_i and s_j for a specific attribute. If the attribute is numeric, two cases should be considered. If the larger value is better, then $\frac{s_i}{s_j}$ will be equal to the result of $\frac{v_i}{v_j}$; if the smaller value is better, then $\frac{s_i}{s_j}$ will be equal to the result of $\frac{v_i}{v_j}$. If the quality attribute is of the set type, the number of s_i items satisfying user requirements are denoted as same (v_i); therefore, the value of $\frac{v_i}{v_j}$ is shown in Formula 2:

$$\frac{s_i}{s_j} = \begin{cases} \text{same } (v_i), & \text{if same } (v_j) = 0, \\ \dfrac{\text{same } (v_i)}{\text{same } (v_j)}, & \text{otherwise .} \end{cases} \tag{2}$$

If the quality attribute is of the interval type, $len(v_i \bigcap v_r)$ shows the intersection between s_i and user requirement. Formula 3 indicates the value of $\frac{s_i}{s_j}$:

$$\frac{s_i}{s_j} = \begin{cases} \dfrac{\text{len } (v_i \bigcap v_r)}{\text{len } (v_j \bigcap v_r)}, & \text{if } v_j \bigcap v_r \neq \emptyset \text{ and } v_i \bigcap v_r \neq \emptyset, \\ 1, & \text{if } v_j \bigcap v_r = \emptyset \text{ and } v_i \bigcap v_r = \emptyset, \\ w_m, & \text{if } v_j \bigcap v_r \neq \emptyset \text{ and } v_i \bigcap v_r = \emptyset, \\ \dfrac{1}{w_m}, & \text{if } v_j \bigcap v_r = \emptyset \text{ and } v_i \bigcap v_r \neq \emptyset. \end{cases} \tag{3}$$

In this formula, w_m is the weight measured for the mth attribute in Step 2 and ranges within [0, 1]. Given the fact that SMICloud is designed for the hierarchical structure of quality attributes, the computations of each attribute are performed at the lowest level. Finally, service ranking is performed in the Candidate set based on the weights of the highest-level attributes and the values of cloud services. If the Candidate set is empty, the cloud services existing in the Secondary set will be considered (third category). In this case, service ranking is performed with respect to the intersection between service intervals and user requirement intervals. Algorithm 2 shows the service ranking process.

Algorithm 2: Service ranking

Input: *Req*:Requirement Interval,$|s|$:cloud service interval,*Services*:set of cloud services, *Candidate*: set of candidate services, *Secondary*: set of secondary services
Output: Ranking the services
2. **foreach** $s_i \in Services$ **do**
  3. **if** $(Req \ covers \ |s_i|) or (|s_i| \ covers \ Req)$ **then**
    $Candidate = Candidate \bigcup \{s_i\}$
  else
    if $len(|s_i| \bigcap Req) > 0$ **then**
      $Secondary = Secondary \bigcup \{s_i\}$
if $Candidate \neq null$ **then**
  Candidate is sorted with **AHP**
else
  if $Secondary \neq null$ **then**
    secondary is sorted by intersection between service and user requirement interval.

3.2 Time complexity of FCSR

To calculate the time complexity of the FCSR, each step must be considered separately. The steps that are important in terms of time include requirements interval calculation, service interval calculation, attributes weight calculation, candidate services weight calculation, and aggregation the ranking results.

The calculation of the requirements interval is related to the number of quality attributes (m), the time complexity is $O(m)$. The calculation of the services interval is related to the number of cloud services (n) besides the number of attributes, the time complexity of which is $O(m \times n)$. The time taken for calculating the weight for each attribute with the AHP method is related to the number of attributes. According to [11], the time complexity is $O(m^3)$. Calculating the weight of candidate services (c) is related to the number of candidate services and the number of quality attributes. Therefore, the time complexity is $O(m^2 \times c^3)$. Aggregation of the ranking results is related to the number of candidate services and the number of quality attributes. Therefore, the time complexity will be $O(m^2 \times c)$. As a result, the total time complexity of FCSR is $O(m^2 \times c^3 + m^2 \times c + m^3 + m \times n + m)$.

4 Performance comparison

The QWS dataset [2], consisting of more than 2500 real web services and 9 different quality attributes, has been employed to assess the FCSR. However, this dataset does not include some of the cloud service attributes such as VM cost and elasticity. According to the SMICloud framework [11], these excluded cloud attributes have been added to the dataset. In total, 16 quality attributes have been used for cloud services. Amongst them, six attributes exist on the high level of the hierarchical structure, each one depending on different factors. Each factor is a quality attribute existing in the lowest level [11]. The new attributes were modeled on random variables of the normal distribution, and the generated values were utilized to complete the dataset. All of the tests were run on a computer with an Intel Core i7 processor (2.4 GHz) on Windows 10 x64 Enterprise.

Since this research aims to improve cloud service ranking based on the AHP-related methods, the FCSR method was compared with other well-known AHP-based methods including the AHP [3], FAHP [39], and NAHP [1]. Also, the proposed method is compared with ANP [40] and FANP [9] methods. The quality attributes are weighed in both FAHP and FANP by the fuzzy technique. NAHP uses the neutrosophic method to weigh them. The FCSR performance is then assessed in terms of optimality, sensitivity, and scalability. Optimality measurement is based on the comparison of ranking results with the baseline ranking [26]. Sensitivity measurement is based on the effect of changes in quality attributes on ranking results [20]. Scalability measurement is based on the effects of changes in the number of services, users, and quality attributes on ranking execution time [26].

4.1 Optimality

In ranking, optimality means that the best services are delivered in the best order to the user in order to maximize user satisfaction. Expert Choice [16] software has been utilized to assess the AHP results. Since all of the cloud service comparisons are drawn using precise data, the resultant ranking is considered as a baseline. Expert Choice is responsive to a small number of cloud services [18]. The NDCG parameter [19] and Spearman's correlation analysis [12] were employed to compare the FCSR optimality with that of previous methods. In fact, the NDCG is used extensively as a ranking quality assessment metric. This parameter is determined through Formula 4:

$$NDCG@K = \frac{1}{z_K} \sum_{i=1}^{K} \frac{\mathrm{REL}_i \times (K - i + 1)}{\log_2(max(i, 2))}, \tag{4}$$

where REL_i shows the proportion of ranking to baseline in the ith rank. If the proportion is met on the ith rank, the value of REL_i is 1; otherwise, it is 0. K shows the list length of the best services, and Z_K is a normalization coefficient, the value of which is determined through baseline ranking calculation. The larger the value of NDCG@K, the more optimal the ranking. In the NDCG parameter, different lengths of K (such as the list of the best 1, 5, 10, 15, and 20 ranked services) were used. As Fig. 6 shows, FCSR and the baseline are very similar in ranking. The ranking difference can be due to the fact that the interval creation priority is

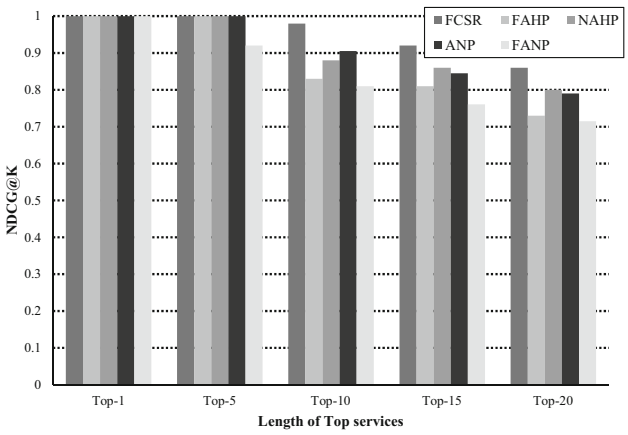


Fig. 6 NDCG@K values of different methods

based on user requirements in all different types of quality attributes in the FCSR method. If the attribute is numeric in the AHP method (baseline), user requirements will be disregarded, and comparison will be based on quality values. Sometimes in the FCSR method, the quality intervals of cloud services and the requirement intervals do not cover each other. In this case, services are ranked by determining their intersection with requirement intervals. This could make some changes in ranking. At the same time, due to considering linguistic values, some information is eliminated in the pairwise comparisons in the FAHP and FANP methods. As a result, there are some variations in the results. The NAHP method has outperformed the FAHP method, which is resulted from using extra information in decision-making. On the other hand, ANP and FANP methods have different ranking procedures that have caused differences in ranking compared to AHP.

As shown in Fig. 6, in TOP 1 and TOP 5 the ranking results are the same between AHP, FAHP, ANP, and NAHP methods. This is because the AHP, FAHP, and NAHP methods, have a similar ranking procedure and have the same results as the AHP. Also, the AHP and ANP ranking methods produced the same ranking results due to the similarity in the ranking procedure. In these cases, similarities in the procedure of ranking overcome differences. Therefore, maximum optimality is obtained. But there are some differences in the ranking from TOP 10, and the reason is the difference in the procedure of weighting the quality attributes in FAHP, FANP, and NAHP. Considering the fuzzy and neutrosophic parameters in weighting the quality attributes causes different weights for each attribute. Therefore, the ranking results are different. As can be seen, there are differences in rankings for $K > 5$ between AHP and ANP. The AHP ranking method is similar to the ANP. However, in ANP, in addition to the AHP procedure, comparisons are made between quality attributes for each service. As can be seen, the ranking result of the FANP method in the TOP 5 is also different from the baseline. The reason is that FANP, in addition to the difference in the ranking procedure with AHP, weights the quality attributes differently. Therefore, the difference in ranking results increases.

Spearman's rank correlation analysis (ρ_r) is a metric employed to assess the correctness of the ranking. This parameter is measured by using Formula 5 through calculating the difference between the ranks of each cloud service in two ranking methods:

$$\rho_r = 1 - \frac{6\Sigma d_i^2}{m(m-1)}, \tag{5}$$

where m indicates the number of services, and d_i shows the difference between the ranks of a service in ranking methods. In this test, 15 cloud services were selected at random. Table 2 shows the results of Spearman's rank correlation analysis. Accordingly, the correlation of AHP and FCSR is 0.882 ($\rho_r(AHP, FCSR) = 0.882$), a value that outweighs the correlations of AHP with FAHP and NAHP. This could be due to the loss of certain information for fuzzy operations. Moreover, if $0.8 < \rho_r \leq 1.0$, then the correlation is interpreted as *very strong*; however, if $0.6 < \rho_r \leq 0.8$, then the correlation is interpreted as *strong* [12]. The correlation of all methods exceeds 0.613.

As can be seen, the correlation between AHP and FAHP is 0.703. The difference of 0.297 in the ranking is due to the use of fuzzy values in FAHP. It caused different weights for each attribute and thus caused a change in ranking results. The ANP method has more ranking differences with AHP, FAHP, NAHP, and FCSR, which is due to the difference in ranking procedures. The FANP method has the most differences from the AHP method in ranking due to the different procedures in ranking the services and weighting the attributes. The FCSR method is more similar to the AHP method in the ranking results. This indicates that the service selection has been done correctly in the preprocessing phase. The difference is related to the cases when there is no cloud service in the Candidate set and the ranking is based on the intersection of the requirement interval with the service interval in the Secondary set.

Table 2 Spearman's rank correlation analysis

	AHP	FAHP	NAHP	ANP	FANP	FCSR (proposed method)
AHP	–	0.703	0.797	0.691	0.633	**0.882**
FAHP	0.703	–	0.903	0.613	0.74	**0.72**
NAHP	0.797	0.903	–	0.663	0.711	**0.754**
ANP	0.691	0.613	0.663	–	0.778	**0.682**
FANP	0.633	0.74	0.711	0.778	–	**0.71**
FCSR (proposed method)	**0.882**	**0.72**	**0.754**	**0.682**	**0.71**	–

FCSR is a proposed method. Therefore, its results are set bold in the table

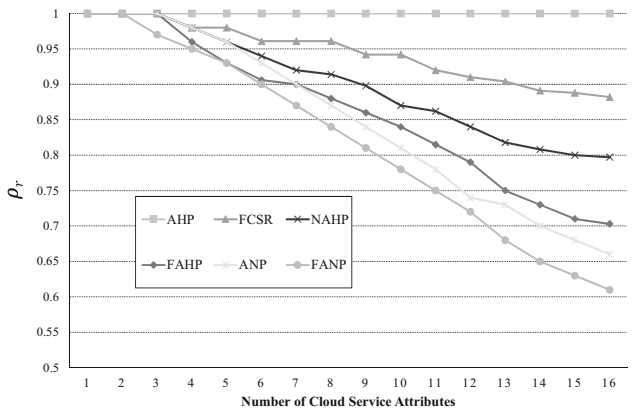


Fig. 7 The effect of number of quality attributes on cloud service ranking

4.2 Sensitivity analysis

Sensitivity analysis is conducted to perceive how many slight changes can affect decisions on a set of equal assumptions [12]. If these changes have negligible effects on ranking, then the ranking method is robust; otherwise, it is sensitive [6]. The sensitivity analysis is useful in cases where there are uncertain and vague values in ranking methods [20]. This test analyzes the effects of changes in the number of quality attributes on ranking results. According to SMICloud [11], the number of quality attributes has been set to range between 1 and 16. These attributes are tested on 15 cloud services selected randomly. Figure 7 draws a comparison of FCSR, FAHP, ANP, FANP, and NAHP with the baseline in the correlation of ranking results. For this purpose, ρ_r is employed by calculating Spearman's correlation analysis metric. Like the optimality test, the AHP result is used as the baseline [18]. Accordingly, increasing the number of quality attributes decreases the ranking optimality in the FCSR method. In fact, increasing the number of quality attributes might increase the mismatch between requirement intervals and service intervals. Therefore, some differences in ranking might emerge with the baseline; however, differences in cloud service ranking are smaller than 0.118. It can also be concluded that fuzzy and neutrosophic initialization can change the ranking results more than FCSR. As the ρ_r in FCSR is larger than 0.8 (i.e. *very strong*) it is robust. Nonetheless, since ρ_r exceeds 0.6 in other methods, it can be interpreted as *strong*.

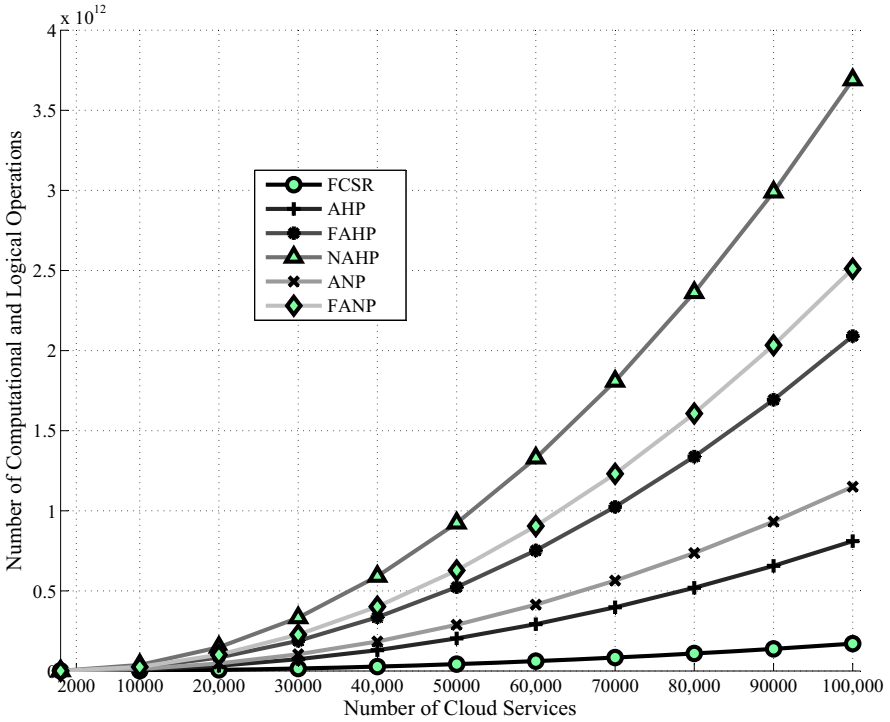
4.3 Scalability

The scalability test analyzes the reactions to three parameters, which are the number of services, the number of users, and the number of quality attributes. In all of the tests, user requirements were considered constant to assess the execution times and number of computational and logical operations of different methods while only increasing one parameter.

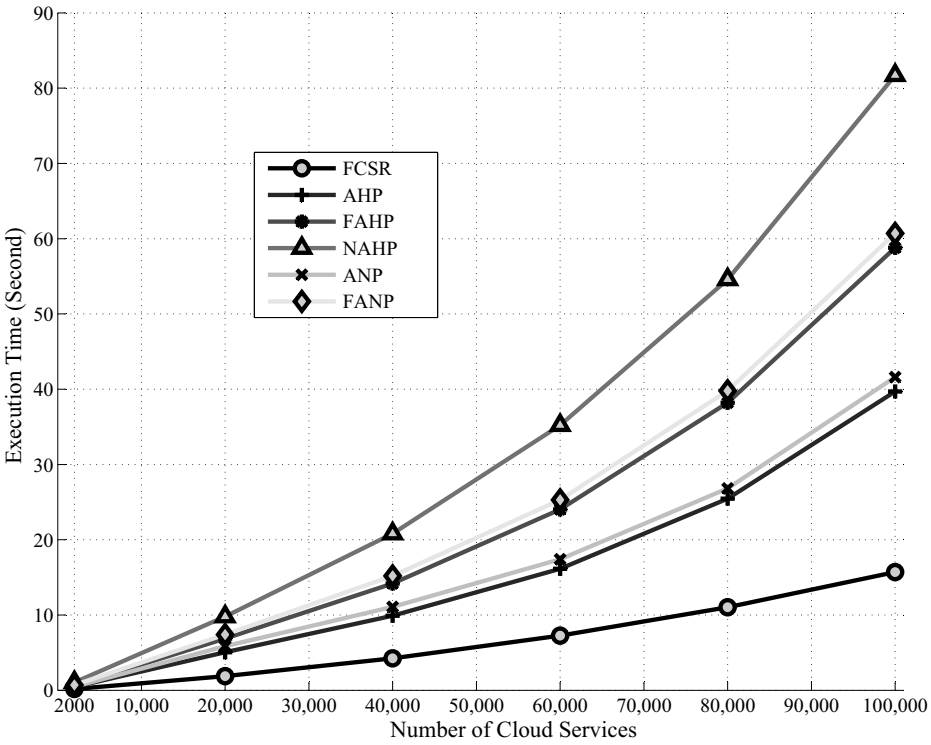
Increasing the Number of Services here, the tests were conducted with 2000, 10000, 20000, 40000, 60000, 80000, and 100000 services. The computational complexity of the methods can be evaluated based on the number of multiplications and logical operations in the algorithms according to [5, 27]. Computations performed in the AHP method include pairwise comparisons of services based on each attribute, weight calculations on each attribute, weight calculations on each service, and a final ranking of services [11]. Computations performed in the FAHP method include pairwise comparisons of services, mean value calculation for each service, fuzzy weight calculation of each attribute, de-fuzzification, weight calculation of each attribute, calculation of the final weight of each service, and the final ranking [5]. In the NAHP method, computations include pairwise comparisons of services, calculation of the mean value of each service, calculation of neutrosophic weight, conversion of neutrosophic numbers to real numbers, calculation of the final weight of each service, and ranking [41]. The ranking procedure of the ANP method includes pairwise comparisons of services based on each attribute, pairwise comparisons of attributes based on each service, weight calculations on each attribute, weight calculations on each service, and aggregation of results in the final matrix [40]. In the FANP method besides all the steps of the FAHP method, pairwise comparisons of quality attributes per service, and aggregation of results are also considered in the ranking procedure [9]. The FCSR method involves the preprocessing and ranking stages. The preprocessing stage, includes sorting attributes, constructing the requirement interval, constructing the service interval, and comparing the service interval with the requirement interval. The ranking stage is done with the number of candidate services similar to the AHP method. Figure 8a shows the effect of the number of services on computational and logical operations.

As shown in Fig. 8a, the number of computations increases as the number of cloud services increases. The FCSR method has the lowest number of computations due to the reduction in the number of services in the preprocessing stage. Also, the AHP method has less computation than ANP. The reason is pairwise comparisons between quality attributes in each service and aggregation computations in ANP. On the other hand, AHP and ANP methods have the lowest number of computations after FCSR. Therefore, it can be seen that fuzzification and de-fuzzification operations require more computations. On the other hand, the NAHP method has the highest number of

Fig. 8 Effect of the number of services on execution time



(a) Effect of the number of services on computational and logical operations



(b) Effect of the number of quality attributes on response time

Table 3 Details of execution time for FCSR

Number of services	Number of services selected for ranking	Service selection process (s)	Ranking process (s)
2000	461	0.056	0.11
10,000	1937	0.31	0.79
20,000	7876	0.62	1.26
40,000	13,823	1.71	2.49
60,000	21,735	2.44	4.857
80,000	32,391	3.81	7.29
100,000	41,634	5.3	10.4

computations. The main reason is the consideration of more operations in the neutrosophic than the fuzzy method.

Figure 8b shows the mean result of 30 executions for each case. Accordingly, FCSR shows lower execution time and higher scalability in comparison with other methods. This is due to the fact that in the FCSR method the disproportional services are eliminated before ranking. The execution times of AHP, FAHP, ANP, FANP, and NAHP increase dramatically when the number of services has exceeded 10,000 cloud services. At the same time, FAHP, FANP, and NAHP methods show longer execution times than the AHP and ANP methods due to fuzzy and neutrosophic calculations.

The FCSR computations process mainly includes selecting proportionate services and ranking cloud services. For the selection of proportional services, both user requirement intervals and quality intervals of all services are measured. In the ranking process, the residual services are compared. Table 3 shows the details on the FCSR execution time. It includes the number of services selected for ranking, the time spent for selecting proportional services (service selection process), and ranking execution time (ranking process). For instance, 1937 proportional services were selected for ranking from 10,000 cloud services. It took the system 0.3140 seconds (s) to select this number of services. Moreover, service ranking takes 0.7871 s; thus, the whole execution time is 1.1011 s.

The purpose of scalability measurement is to quantify the behavior of a cloud-based system when the number of cloud services increases or demand increases. Here, system behavior refers to the average system response time. We also interpolated the diagrams in Fig. 8b to examine the scalability of the proposed method. The interpolation result is given in Formulas 6–11.

$$FCSR : f(x) = 0.1374x^3 + 1.194x^2 + 5.56x + 5.715, \quad (6)$$

$$AHP : f(x) = 1.828x^3 + 3.888x^2 + 11.4x + 12.83, \quad (7)$$

$$ANP : f(x) = 2.154x^3 + 3.72x^2 + 11.54x + 14.05, \quad (8)$$

$$FAHP : f(x) = 2.075x^3 + 5.872x^2 + 18.01x + 18.84, \quad (9)$$

$$FANP : f(x) = 2.14x^3 + 5.754x^2 + 18.6x + 20, \quad (10)$$

$$NAHP : f(x) = 2.161x^3 + 7.371x^2 + 26.27x + 27.66. \quad (11)$$

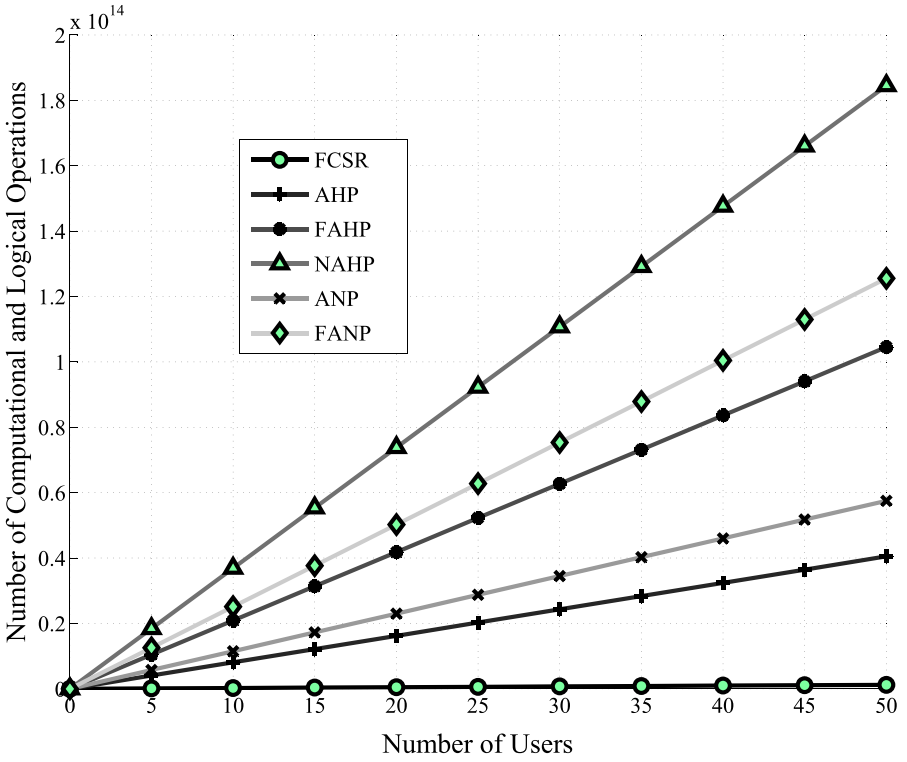
As can be seen, with the increasing number of cloud services, the FCSR method increases with a smoother slope than previous methods.

Increasing the number of users in this test, a large number (100,000) of cloud services were considered to perform ranking for a variable number of users (1–50). The values of user requirements were considered randomly. It was assumed that users receive their responses sequentially (in a queue). As Fig. 9 shows, the FCSR method is more scalable than other methods as the number of users increases. Since all methods are based on matrix comparisons, increasing the number of users affects the number of computational and comparative operations. The FCSR method tolerates the time to select appropriate services. As a result, FCSR ranks candidate services instead of all services.

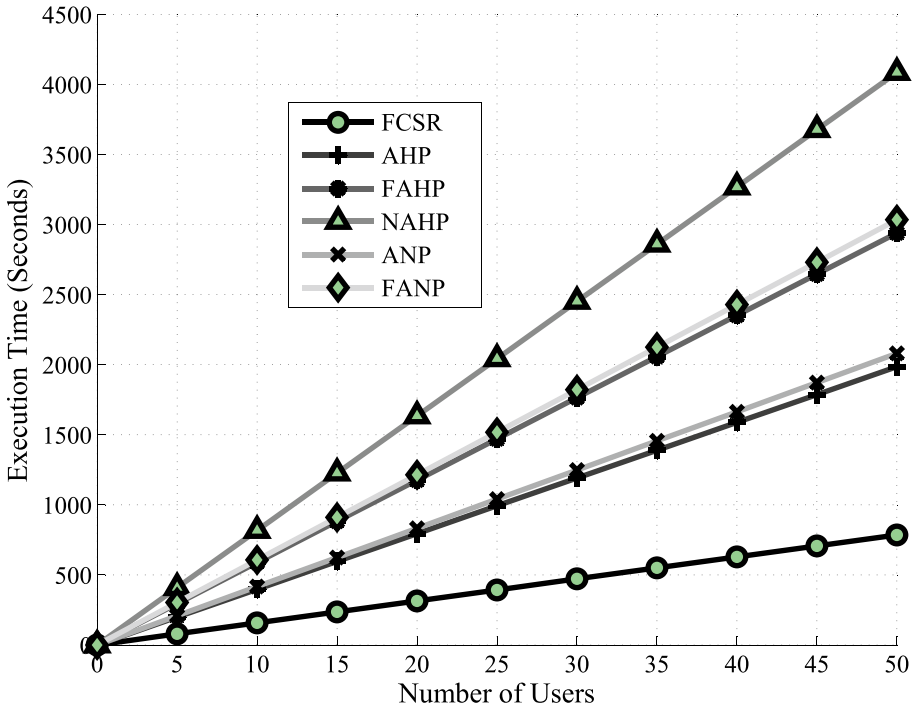
According to Fig. 9a, filtering the services reduces computational and logical operations. Therefore, increasing the number of users has less effect on execution time. AHP and ANP methods have less execution time than FAHP, FANP, and NAHP methods. The reason is the additional parameters for fuzzy and neutrosophic methods that have a large impact on computational and logical operations as the number of users increases.

As shown in Fig. 9b, the FCSR method outperforms other methods, by significantly decreasing the execution time. The difference in execution time of AHP and ANP methods (also FAHP and FANP methods) is due to the difference in the number of calculations and comparisons

Fig. 9 Effect of number of users on scalability

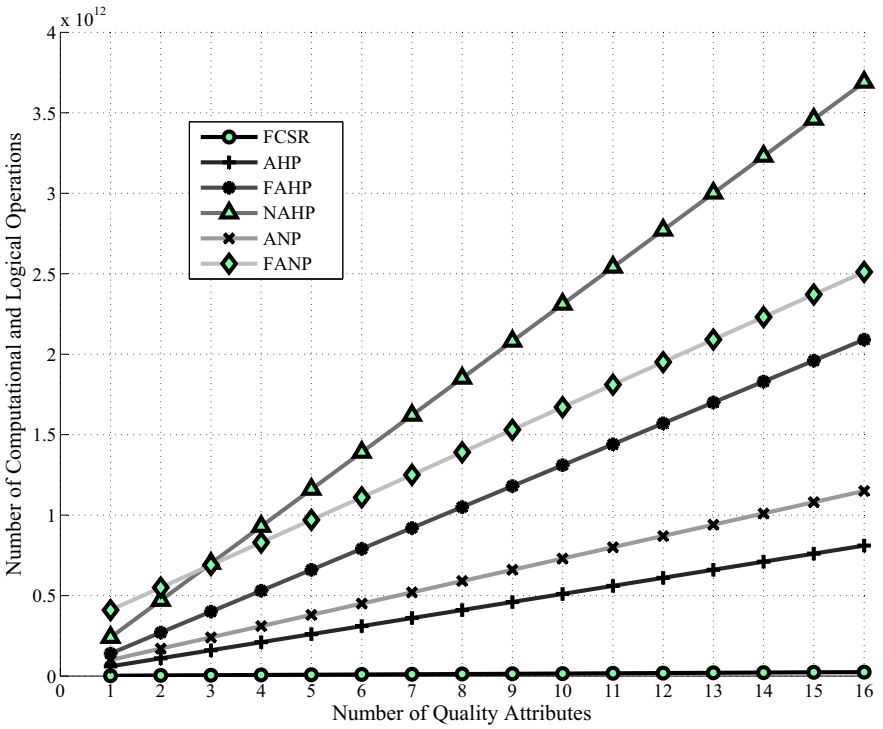


(a) Effect of the number of users on computational and logical operations

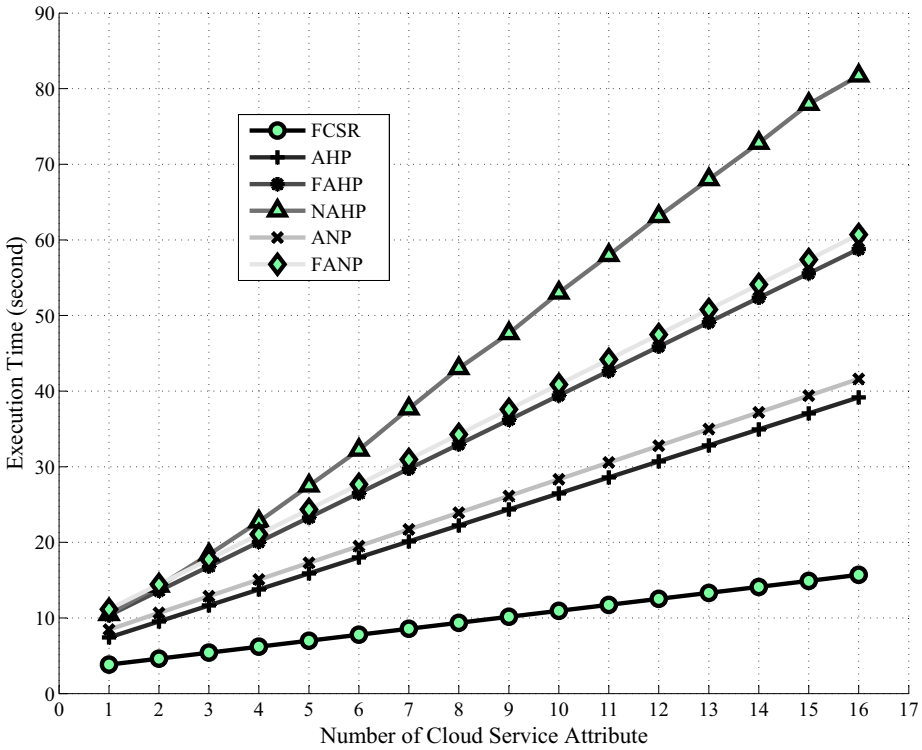


(b) Effect of the number of users on execution time

Fig. 10 Effect of number of quality attributes on scalability



(a) Effect of the number of quality attributes on computational and logical operations



(b) Effect of the number of quality attributes on response time

of the ANP method with the AHP method. Since the NAHP method uses the neutrosophic method, because the neutrosophic method uses more parameters than the fuzzy method, it has more computations than the FAHP and FANP methods. As a result, NAHP has more execution time.

The number of quality attributes in this test, the number of quality attributes was changed from 1 to 16. These attributes were tested on a dataset of a large number (100,000) cloud services.

Figure 10a shows the number of computational and logical operations in each method by increasing the number of quality attributes. As mentioned earlier, FCSR has fewer computational and logical operations than other ranking methods because other methods perform rankings on all cloud services. The FCSR performs rankings on selected appropriate services, which reduces the number of computational and logical operations. In addition, with increasing the number of quality attributes, calculations in fuzzy and neutrosophic methods increase due to the consideration of fuzzy and neutrosophic parameters and variables. The neutrosophic method has more parameters than the fuzzy method, so it has more calculations. The ANP method has more calculations than the AHP because the pairwise comparison of quality attributes is performed for each service in the ANP, while the AHP method does not have this step and this factor increases the number of ANP calculations compared to the AHP. As can be seen, the number of calculations of the FANP method is up to three quality attributes more than the NAHP method. This is because fuzzy operations and calculations of the ANP method are more than neutrosophic operations in less than three quality attributes. By increasing the number of services to over three attributes, the number of computational and logical operations of NAHP increases compared to FANP. This is because increasing the number of quality attributes has a greater impact on the number of neutrosophic operations.

Figure 10b compares the mean response time of 30 executions on each method. In the FCSR method, increasing the number of quality attributes has less effect than other methods because of less computational and logical operations. Since the operations of AHP-based methods are matrix-based, fewer services are effective in computational and logical operations. With increasing the number of quality attributes, execution time in FAHP, FANP, NAHP methods has more increase than AHP and ANP. The reason is the use of fuzzy and neutrosophic methods to weigh quality attributes. As can be seen, the execution time in NAHP has increased by five quality attributes compared to other methods. The reason is the effect of the number of quality attributes on the neutrosophic operation.

5 Discussion

In this study, a solution is proposed to address the limitation of uncertainty in user requirements as an opportunity to rank cloud services efficiently. For this purpose, the uncertain requirements are converted into requirement intervals. Services that are not at requirement intervals are then filtered. The FCSR method was evaluated in terms of optimality, sensitivity analysis, and scalability.

To evaluate the optimality of the FCSR method, two parameters of NDCG and Spearman's correlation analysis are used. The results in Fig. 6 show that the FCSR optimality at K = 1 and 5 is 100%. At K = 10, the optimality is 98%. At K = 15 and 20, it is 92% and 86%, respectively. However, the previous methods have less than 80% optimization at K = 20. Spearman's correlation analysis parameter is used to calculate the similarity of the results of the ranking methods in pairs. Table 2 shows that the ranking similarity of the FCSR method to the AHP method is 0.882, which is the highest similarity compared to other methods. The extent of this similarity is interpreted as *very strong*. The results show that the operations of fuzzy and neutrosophic methods reduce the similarity of the ranking results. For example, the similarity of AHP and FAHP is 0.703 and the similarity of AHP with NAHP is 0.797. The similarity of AHP with ANP and FANP is 0.691 and 0.633, respectively. Comparison of the ranking results of different methods with baseline shows that the FCSR method is more optimal than other methods. It can be concluded that the preprocessing stage selects the appropriate services with high accuracy.

Sensitivity analysis was used to evaluate the effect of small changes in the ranking results. For this purpose, Spearman's correlation analysis parameter was used to calculate the similarity of ranking results. As shown in Fig. 7, the sensitivity of the FCSR method is finally 0.118. The reason for this difference is that increasing the number of quality attributes causes a mismatch between the requirements interval and service quality interval. However, the sensitivity of the NAHP, FAHP, ANP, and FANP methods is finally 0.203, 0.297, 0.34, and 0.39, respectively. This shows that the FCSR is more robustness than other methods.

To evaluate scalability, two parameters of execution time and the number of computational and logical operations are used. As the number of cloud services increases, Figs. 8, 9, and 10 show that the number of computational and logical operations in the FCSR is significantly lower than in previous methods, which reduces response time. For example, with the increase in the number of cloud services, Fig. 8 shows that the number of computational and logical operations with 10^5 cloud services for FCSR

and AHP methods are 1.7×10^{11} and 8.1×10^{11}, respectively. This shows that the FCSR method has an 81% reduction in the number of computational and logical operations compared to the AHP method. In addition, scalability experiments show that fuzzy and neutrosophic operations have a large effect on the number of computational and logical operations in numerous cloud services, which increases execution time. The number of computational and operational operations of NAHP and FAHP methods are 3.96×10^{12} and 2.9×10^{12}, respectively. Therefore, the number of computational and logical operations of NAHP and FAHP methods is 4.89 and 2.58 times the AHP method, respectively. Similarly, the effect of preprocessing, fuzzy and neutrosophic operations with an increasing number of users, and the number of quality attributes can be seen in Figs. 9 and 10.

6 Conclusion

This paper has proposed the FCSR method to rank cloud services efficiently, by eliminating the services that are disproportionate to user requirements. It is based on the creation of user requirement intervals. Correspondingly, the interval of cloud services is created on the basis of user requirements and service quality. Accordingly, the services that are more proportionate to user requirements are selected for ranking. The evaluations have considered scalability, optimality, and robustness. According to the scalability test results, the FCSR runtime has been much shorter than those of the previous methods when performed for a large number of cloud services. At the same time, the ranking optimality analysis has indicated that the FCSR method is highly correlated with the baseline ranking (*very strong*). Finally, FCSR has a considerable rate of robustness to changes in the number of quality attributes.

As user feedbacks affect service selection in addition to the quality attributes of cloud services, we intend to expand our method by considering users feedbacks on the services that have been used. User feedback is taken into account to show the level of user satisfaction with the service. It could be considered as a multi-objective problem apart from quality values by integrating multi-objective algorithms and the FCSR method.

References

1. Abdel-Basset, M., Mohamed, M., Chang, V.: NMCDA: a framework for evaluating cloud computing services. Future Gener. Comput. Syst. **86**, 12–29 (2018)
2. Al-Masri, E., Mahmoud, Q.H.: Discovering the best web service. In: Proceedings of the 16th International Conference on World Wide Web, pp. 1257–1258 (2007)
3. Alelaiwi, A.: Evaluating distributed IoT databases for edge/cloud platforms using the analytic hierarchy process. J. Parallel Distrib. Comput. **124**, 41–46 (2019)
4. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: 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 (2009)
5. Chang, D.Y.: Applications of the extent analysis method on fuzzy AHP. Eur. J. Oper. Res. **95**(3), 649–655 (1996)
6. Christopher Frey, H., Patil, S.R.: Identification and review of sensitivity analysis methods. Risk Anal. **22**(3), 553–578 (2002)
7. Devi, R., Shanmugalakshmi, R.: Cloud providers ranking and selection using quantitative and qualitative approach. Comput. Commun. **154**, 370–379 (2020)
8. Dyer, J.S., Fishburn, P.C., Steuer, R.E., Wallenius, J., Zionts, S.: Multiple criteria decision making, multiattribute utility theory: the next ten years. Manag. Sci. **38**(5), 645–654 (1992)
9. Fan, J., Yu, S., Yu, M., Chu, J., Tian, B., Li, W., Wang, H., Hu, Y., Chen, C.: Optimal selection of design scheme in cloud environment: a novel hybrid approach of multi-criteria decision-making based on F-ANP and F-QFD. J. Intell. Fuzzy Syst. **38**(3), 3371–3388 (2020)
10. Garg, R.: MCDM-based parametric selection of cloud deployment models for an academic organization. IEEE Trans. Cloud Comput. (2020). https://doi.org/10.1109/TCC.2020.2980534
11. Garg, S.K., Versteeg, S., Buyya, R.: A framework for ranking of cloud computing services. Future Gener. Comput. Syst. **29**(4), 1012–1023 (2013)
12. Gireesha, O., Somu, N., Krithivasan, K., Shankar Sriram, V.S.: IIVIFS-WASPAS: an integrated multi-criteria decision-making perspective for cloud service provider selection. Future Gener. Comput. Syst. **103**, 91–110 (2020)
13. Goraya, M.S., Singh, D., et al.: A comparative analysis of prominently used MCDM methods in cloud environment. J. Supercomput. **77**(4), 3422–3449 (2021)
14. Han, S.M., Hassan, M.M., Yoon, C.W., Huh, E.N.: Efficient service recommendation system for cloud computing market. In: Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, pp. 839–845 (2009)
15. Iosup, A., Ostermann, S., Yigitbasi, M.N., Prodan, R., Fahringer, T., Epema, D.: Performance analysis of cloud computing services for many-tasks scientific computing. IEEE Trans. Parallel Distrib. Syst. **22**(6), 931–945 (2011)
16. Ishizaka, A., Labib, A.: Analytic hierarchy process and expert choice: benefits and limitations. OR Insights **22**(4), 201–220 (2009)
17. Izadpanah, S., Vahdat-Nejad, H., Saadatfar, H.: A framework for ranking ubiquitous computing services by AHP analysis. Int. J. Model. Simul. Sci. Comput. **9**(04), 1850023 (2018)
18. Jahani, A., Khanli, L.M.: Cloud service ranking as a multi objective optimization problem. J. Supercomput. **72**(5), 1897–1926 (2016)
19. Jiang, Y., Tao, D., Liu, Y., Sun, J., Ling, H.: Cloud service recommendation based on unstructured textual information. Future Gener. Comput. Syst. **97**, 387–396 (2019)
20. Kumar, R.R., Mishra, S., Kumar, C.: Prioritizing the solution of cloud service selection using integrated MCDM methods under fuzzy environment. J. Supercomput. **73**(11), 4652–4682 (2017)
21. Kumar, R.R., Kumari, B., Kumar, C.: CCS-OSSR: a framework based on hybrid MCDM for optimal service selection and ranking of cloud computing services. Clust. Comput. **24**, 1–17 (2020)

22. Kwon, H.K., Seo, K.K.: A decision-making model to choose a cloud service using fuzzy AHP. Adv. Sci. Technol. Lett. **35**(1), 93–96 (2013)
23. Lee, S., Seo, K.K.: A hybrid multi-criteria decision-making model for a cloud service selection problem using BSC, fuzzy Delphi method and fuzzy AHP. Wirel. Pers. Commun. **86**(1), 57–75 (2016)
24. Li, A., Yang, X., Kandula, S., Zhang, M.: CloudCmp: comparing public cloud providers. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, pp. 1–14 (2010)
25. Li, J., Squicciarini, A.C., Lin, D., Sundareswaran, S., Jia, C.: MMB cloud-tree: authenticated index for verifiable cloud service selection. IEEE Trans. Dependable Secure Comput. **14**(2), 185–198 (2015)
26. Lin, D., Squicciarini, A.C., Dondapati, V.N., Sundareswaran, S.: A cloud brokerage architecture for efficient cloud service selection. IEEE Trans. Serv. Comput. **12**(1), 144–157 (2016)
27. Martin, A., Lakshmi, T.M., Venkatesan, V.P.: A study on evaluation metrics for multi criteria decision making (MCDM) methods—TOPSIS, COPRAS and GRA. Int. J. Comput. Algorithm **7**(01), 29–37 (2018)
28. Oriol, M., Marco, J., Franch, X.: Quality models for web services: a systematic mapping. Inf. Softw. Technol. **56**(10), 1167–1182 (2014)
29. Repschlaeger, J., Wind, S., Zarnekow, R., Turowski, K.: Decision model for selecting a cloud provider: a study of service model decision priorities. In: Proceedings of the Nineteenth Americas Conference on Information Systems, Chicago, Illinois, 15–17 August 2013 (2013)
30. Ribas, M., Furtado, C., de Souza, J.N., Barroso, G.C., Moura, A., Lima, A.S., Sousa, F.R.: A Petri net-based decision-making framework for assessing cloud services adoption: the use of spot instances for cost reduction. J. Netw. Comput. Appl. **57**, 102–118 (2015)
31. Saaty, T.L.: Decision making with the analytic hierarchy process. Int. J. Serv. Sci. **1**(1), 83–98 (2008)
32. Saravanan, M., Aramudhan, M., Pandiyan, S.S., Avudaiappan, T.: Priority based prediction mechanism for ranking providers in federated cloud architecture. Clust. Comput. **22**(4), 9815–9823 (2019)
33. Shivakumar, U., Ravi, V., Gangadharan, G.: Ranking cloud services using fuzzy multi-attribute decision making. In: 2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–8. IEEE (2013)
34. Siegel, J., Perdue, J.: Cloud services measures for global use: the service measurement index (SMI). In: 2012 Annual SRII Global Conference, pp. 411–415. IEEE (2012)
35. Smarandache, F.: Neutrosophy: Neutrosophic Probability, Set, and Logic: Analytic Synthesis and Synthetic Analysis. American Research Press (1998)
36. Somohano-Murrieta, J.C.B., Ocharán-Hernández, J.O., Sánchez-García, A.J., de los Ángeles Arenas-Valdés, M.: Requirements prioritization techniques in the last decade: a systematic literature review. In: 2020 8th International Conference in Software Engineering Research and Innovation (CONISOFT), pp. 11–20. IEEE (2020)
37. Sun, L., Dong, H., Hussain, F.K., Hussain, O.K., Chang, E.: Cloud service selection: state-of-the-art and future research directions. J. Netw. Comput. Appl. **45**, 134–150 (2014)
38. Sun, L., Ma, J., Zhang, Y., Dong, H., Hussain, F.K.: Cloud-fuser: fuzzy ontology and MCDM based cloud service selection. Future Gener. Comput. Syst. **57**, 42–55 (2016)
39. Tajvidi, M., Ranjan, R., Kolodziej, J., Wang, L.: Fuzzy cloud service selection framework. In: 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), pp. 443–448. IEEE (2014)
40. Tchernykh, A., Schwiegelsohn, U., Alexandrov, V., Talbi, E.: Towards understanding uncertainty in cloud computing resource provisioning. Procedia Comput. Sci. **51**, 1772–1781 (2015)
41. Tiwari, R.K., Kumar, R.: A framework for prioritizing cloud services in neutrosophic environment. J. King Saud Univ. Comput. Inf. Sci. (2020). https://doi.org/10.1016/j.jksuci.2020.05.009
42. Tran, V.X., Tsuji, H., Masuda, R.: A new QoS ontology and its QoS-based ranking algorithm for web services. Simul. Model. Pract. Theory **17**(8), 1378–1398 (2009)
43. Wibowo, S., Deng, H.: Multi-criteria group decision making for evaluating the performance of e-waste recycling programs under uncertainty. Waste Manag. **40**, 127–135 (2015)
44. Wibowo, S., Deng, H., Xu, W.: Evaluation of cloud services: a fuzzy multi-criteria group decision making method. Algorithms **9**(4), 84 (2016)
45. Youssef, A.E.: An integrated MCDM approach for cloud service selection based on TOPSIS and BWM. IEEE Access **8**, 71851–71865 (2020)
46. Yu, P.L.: Multiple-Criteria Decision Making: Concepts, Techniques, and Extensions, vol. 30. Springer, New York (2013)
47. Zadeh, L.A.: Fuzzy sets. Inf. Control **8**(3), 338–353 (1965)

Mohammad Hossein Nejat is a Software Engineering Ph.D. Student at Islamic Azad University, Sari, Iran. He received M.S. Degree in Computer Engineering-Software Engineering in Islamic Azad University, Zanjan, Iran. His current research interests include software quality, cloud computing, and evolution algorithms.

Homayun Motameni received B.S. Degree in Computer Engineering-Software Engineering in Shahid Beheshti of Tehran University and M.S. Degree in Computer Engineering-Machine Intelligence from Islamic Azad University-Science and Research Branch in 1995 and 1998, respectively. He received Ph.D. Degree in Computer Engineering-Software Engineering from Islamic Azad University-Science and Research Branch in 2007. His current research interests include evolution algorithms, Petri Net, software systems modeling and evaluation using Petri Net, and machine learning.

Hamed Vahdat-Nejad is currently an Assistant Professor at the Computer Engineering Department of the University of Birjand. He was a Visiting Professor at the Superior University in Lahore in Summer of 2018 and at the Daffodil International University in Dhaka in Summer of 2017. He received his PhD from Computer Engineering Department of University of Isfahan in 2012, his Master's Degree from Ferdowsi University of Mashhad in 2007, and his Bachelor's Degree from Sharif University of Technology in 2004. He was a Research Scholar at the Middleware laboratory of Sapienza University of Rome in 2011–2012 period. Currently, his research is focused on pervasive computing, cloud computing, and context-awareness. He has (co)published about 50 papers in conferences and journals, and leads the Pervasive and Cloud Computing Lab at the University of Birjand. He has served as the Chairman of the 1st AND 2nd International Workshop on Context-aware Middleware for Ubiquitous Computing Environments, "3rd, 4th, and 5th International workshop on Pervasive and Context-aware middleware" as well as 1st Conference on Healthcare Computing Systems and Technologies. He has served as TPC Member for many conferences. Previously, he has served as Associate and Guest Editor for Elsevier Computers and Electrical Engineering and Journal of Computing and Security, among others.

Behnam Barzegar received B.S. Degree in Computer Engineering in Islamic Azad University, Sari, Iran and M.S. Degree in Computer Engineering-Software Engineering from Islamic Azad University Najaf Abad Branch in 2006 and 2009, respectively. He received Ph.D. Degree in Computer Engineering-Software Engineering from Islamic Azad University-Sari Branch in 2018. He has (co) published about 30 papers in conferences and journals, His current research interests include cloud computing, evolution algorithms, software systems modeling and evaluation using Petri Nets.