Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
() : –

doi:10.3906/elk-2010-54

An integrated optimal method for cloud service ranking

MOHAMMAD HOSSEIN NEJAT[1], HOMAYUN MOTAMENI[1,*], HAMED VAHDAT-NEJAD[2]

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

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

Received: 12.10.2020 • **Accepted/Published Online:** 21.05.2021 • **Final Version:** ..2021

Abstract: Many cloud providers present various services with different attributes. It is a complex, lengthy process to select a cloud service that meets user requirements from an assortment of services. At the same time, user requirements are sometimes defined with imprecision (sets or intervals), whereas it is also important to consider the quality of user feedback (QoU) and quality of service (QoS) attributes for ranking. Besides, each MADM method has a different procedure, which causes functional contradictions. These contradictions have led to confusion in choosing the best MADM method. It is necessary to provide a method that harmonizes the results. Therefore, choosing a method for ranking cloud services that addresses these issues is currently a challenge. This paper proposes an optimal cloud service ranking (OCSR) method that ranks cloud services efficiently based on imprecise user requirements in both QoS and QoU aspects. OCSR consists of four stages including receiving the requirements, preprocessing, ranking, and integrating the ranking results. At the receiving requirements stage, the query format is created. In the preprocessing stage, a requirement interval is created for considering imprecise user requirements in order to filter inappropriate services. Based on QoS and QoU attributes, cloud services are then ranked through multiple multi-attribute decision-making (multi-MADM) methods such as the prominent MADM techniques. Finally, the ranking outputs of various methods are integrated to obtain the optimal results. The experimental results confirm that the OCSR outperforms the previous methods in terms of optimality of ranking, sensitivity analyses, and scalability.

Key words: Cloud service selection, cloud service ranking, multi-attribute decision-making (MADM), quality of service (QoS), quality of user feedback (QoU)

1. Introduction

Cloud computing plays a prominent role in developing systems and distributing applications on the internet. Cloud provides easy, safe, flexible, and scalable access to information and computational resources (infrastructure, platform, and software) in various services [1]. There are many cloud computing providers that offer similar services with different qualities. At the same time, cloud users always have different requirements for various applications, a fact which makes it complex and time-consuming to compare and select cloud services meeting user requirements [2].

With the development of cloud computing and the availability of abundant services, it is challenging to devise an appropriate method for selecting and ranking services [3]. User requirements are also often met by more than one service; therefore, ranking systems are employed to provide the most appropriate service (i.e. the highest-quality service meeting user requirements) at short notice [4]. Basically, ranking cloud services would

*Correspondence: motameni@iausari.ac.ir

necessitate considering two perspectives: quality of service (QoS) and quality of user feedback (QoU) [2]. There are several contradictory attributes in each perspective [5]; such as achieving high central processing unit (CPU) speed and availability apart from minimizing latency and price in QoS aspect. Also, achieving high security and trust apart from minimizing response time in QoU aspect. In addition to the contradictory quality attributes, the different requirements of users, a large number of cloud services, and user feedback from cloud services are effective in ranking services. Therefore, these factors have made ranking cloud services a complex issue [5]. To address these problems, multi-attribute decision-making (MADM) methods can be considered as the best choice [6]. The MADM methods can help users to simplify and solve the problem of ranking cloud services [1].

Researchers have proposed various methods for ranking cloud services. Many studies address QoS attributes, and extensive efforts have been made to identify the metrics for measuring these attributes in cloud computing environment [7–9]. Quite a few researchers have focused on user requirements and proposed methods for measuring services based on requirements [1, 10, 11]. However, some researchers have evaluated cloud services in terms of trustworthiness and security perspective [3, 12], and many others have proposed solutions to the improvement of existing methods [5, 13, 14]. For instance, Garg et al. proposed a ranking cloud services framework based on user requirements [1]. Because user requirements can be imprecise, solutions to improve this method have been proposed. For this purpose, Kumar et al. proposed a method of combining MADM methods and fuzzy to handle imprecise in the user requirements [13]. The combination of these methods increases the response time. To reduce the response time, a method was designed by applying changes in combination of MADM methods [14]. As fuzzy sets have limitations for considering imprecise user requirements, some parameters were added to the fuzzy set to increase the ranking trustworthiness [5].

Recent cloud service ranking methods have reported similar deficiencies. Nearly 100 quality attributes have been proposed so far, and each study uses only a few of them [5]. Most of the recent methods have overlooked user feedback when ranking services [4], whereas many have used different ranking methods yielding different results [2]. Furthermore, the complex and lengthy comparisons have prevented them from responding when the number of cloud services is high [3]. However, studies have often neglected to rank cloud services within a framework to (1) consider optimality (the highest level of proportion to user requirements) and execution time; (2) perform ranking based on imprecise (sets or intervals) user requirements; (3) provide an approach to merge QoU and QoS aspects. Fixing these deficiencies can effectively improve the ranking of cloud services.

This study presents an integrated method for ranking cloud services to perform ranking of high optimality and reduce execution time. In the OCSR, ranking is based on user requirements to consider QoS and QoU attributes. To reduce execution time, OCSR proposes an approach to eliminate services that do not meet the user requirements. The OCSR method uses multi-MADM method to rank cloud services. Multi-MADM method includes prominent MADM methods and uses their ranking results to achieve optimal rankings. The reason for using multi-MADM methods is that each MADM method has a different procedure that leads to contradictory rankings. Such contradictions in the ranking of cloud services cause confusion in selecting the best service. Therefore, the OCSR has provided a method to integrate contradictory ranks of cloud services obtained from each of the MADM methods. The OCSR method has been assessed in three scenarios such as optimality, sensitivity analysis and scalability. The results show that the OCSR method retained an acceptable execution time by using multi-MADM for ranking. At the same time, it shows a higher optimality and sensitivity rate than other methods.

The main contributions of this research are summarized as follows.

- An integrated method called OCSR based on imprecise user requirements has been proposed. OCSR considers user's QoS and QoU requirements to provide a broader view of service rankings.
- OCSR emphasizes preprocessing to improve the results in the following ways. (1) Appropriate services are selected in the preprocessing stage, and a limited number of services are ranked. (2) Services are not eliminated due to non-compliance with one or multiple user requirement attributes. Instead, service evaluation is based on all QoS and QoU attributes. (3) Preprocessing is automated and needs no expert involvement.
- OCSR employs the multi-MADM to rank cloud services. The ranking results obtained from each MADM method are integrated using a proposed method to produce consensual ranking results.
- A comparison has been applied based on optimality, sensitivity, and scalability metrics between OCSR and other existing methods.

The rest of the paper consists of the following sections. Section 2 presents a literature review. Section 3 discusses the OCSR method proposed for ranking cloud services. Section 4 presents a case study example of the OCSR. Section 5 draws a performance comparison between OCSR and the other existing methods. Finally, Section 6 expresses the conclusion and suggestions for future studies.

2. Related work

The growing number of cloud services has led to great efforts for their evaluation and ranking [15]. Many researchers are interested in solving the service ranking problem based on MADM due to the decision-making problems related to user requirements with various attributes [1]. Common MADM methods, reported in [2, 5] for ranking cloud services, include the analytic hierarchy process (AHP) [16], simple additive weighting (SAW) [17], ViseKriterijumska Optimizacija I Kompromisno Resenje (VIKOR) [18], technique for order of performance by similarity to ideal solution (TOPSIS) [19], weighted product method (WPM) [20], and preference ranking organization METHod for enrichment evaluation (PROMETHEE) [21]. The different procedures for each method make them different in terms of performance [5]. However, choosing the most efficient method for ranking cloud services remains unresolved [22].

Imprecise requirements present limitations that researchers have diligently tried to resolve [5]. For instance, Garg et al. [1] proposed the service measurement index cloud (SMICloud) framework for evaluating, measuring, and ranking cloud services through AHP based on requirements with different QoS attributes. Tajvidi et al. [23] designed a fuzzy service selection framework for improving SMICloud. Since the fuzzification of requirements may lead to loss of information, the neutrosophic multi-criteria decision analysis (NMCDA) [5] method, i.e. a hybrid AHP-neutrosophic method, was proposed for ranking cloud services. The neutrosophic set adds the truth, falsity, and indeterminacy to the fuzzy set membership function. It is meant to improve optimality in the fuzzy set results. Since fuzzy and neutrosophic operations are time-consuming, using the aforementioned methods can prolong execution time.

The growing number of cloud services has made service selection a time-consuming task. In fact, it is a challenge that researchers have tried to overcome in recent years. Goraya et al. [22] proposed a method to evaluate MADM methods based on time complexity to solve Cloud Geographical Region (CGR) problem. Lee et al. proposed a hybrid delphi-AHP fuzzy method for reducing the number of quality attributes [24]. The results showed that the reduction in the number of quality attributes failed to significantly affect the execution time

[4]. Jahani et al. [3] divided quality attributes into essential and non-essential categories and tried to reduce execution time by removing the services whose essential attributes failed to meet user requirements. This may, however, prevent a service recommendation to users in many cases due to a slight difference in values of one or multiple attributes. The methodology for optimal service selection (MOSS) [2] uses a prequel preprocess that presents services to experts for exclusion based on experience and consensus. Therefore, the system only ranks services that have no advantages over each other. The main problem with this method is that using expert opinions will not be feasible in conditions where there are numerous services in a real-time processing.

Ranking cloud services using MADM methods individually cannot handle many limitations. Some of these limitations are the lack of a consensus mechanism in ranking results and failure to consider imprecise user requirements. Therefore, researchers have combined MADM methods to improve performance in terms of optimality and trustworthiness of ranking results. For instance, Alhanahnah et al. [25] employed a fuzzy hybrid SAW-AHP method to improve service evaluation in terms of trustworthiness. A hybrid PROMETHEE-AHP method proposed for ranking cloud services based on functional and non-functional requirements [26]. This method considered a large number of attributes to achieve optimal results. Sun et al. [14] proposed the cloud-fuzzy user-oriented SERvice selection (Cloud-FuSer) technique based on fuzzy AHP and fuzzy TOPSIS to overcome data imprecise and decision ambiguities. Jatoth et al. [27] combined AHP with Grey TOPSIS to improve ranking results in terms of optimality. The researchers conducted a case study and designed a model to evaluate the robustness of ranking results. In paper [28], AHP and Fuzzy TOPSIS methods were combined to rank cloud services. The objective was to take account of imprecise user requirements and improve trustworthiness. Gireesha et al. [29] proposed a method to select appropriate services based on considering interval values in service quality values. The researchers combined WPM and weight sum method (WSM) to rank cloud services and conducted a case study to evaluate the ranking results. However, the problem of contradictory results has not been solved and the combination of MADM methods has increased the execution time [30].

According to the existing literature, the main disadvantages of the existing methods are as follows. (1) Most of the existing methods select services based on QoS. (2) Most of the existing methods ignore filtering cloud services before ranking. (3) The results of ranking methods are contradictory. The main aim of this article is to address these issues.

3. Optimal cloud service ranking (OCSR)

As mentioned earlier, ranking cloud services with MADM methods faces many challenges including failure to handle imprecise user requirements, high execution time, and contradictory ranking results. OCSR method produces requirement intervals to consider imprecise user requirements. Services are selected based on the requirement interval, which has the most overlap with user requirements in both QoS and QoU aspects. Therefore, many services are excluded from the ranking process. The remaining services are then ranked through multi-MADM methods, the ranking results of which are merged to yield comprehensive results.

As depicted in Figure 1, OCSR entails receiving the user requirements, preprocessing, ranking through multi-MADM methods, and integrating the results. In the receiving user requirements stage, the user requirements are received based on QoS and QoU attributes and then converted into a query. In the preprocessing stage, QoS and QoU requirement intervals are created according to the query. Cloud services intervals are also created based on their QoS and QoU values. Services that are in the requirement intervals are selected. In the ranking stage, the services remaining from the previous stage are ranked by using multi-MADM method. In

fact, multi-MADM method is used because of contradictory ranking results obtained from each MADM method. In the integration stage, the results of multiple MADM methods are merged to obtain the final rankings. The OCSR method has been presented in Algorithm 1.

Algorithm 1: OCSR

Input: User requirement, the service set with the quality values of each service, and the user feedback regarding each service.
Output: Ranked services
1. Receiving user requirements
2. Preprocessing
3. Ranking through multi-MADM methods
4. Integrating ranking results

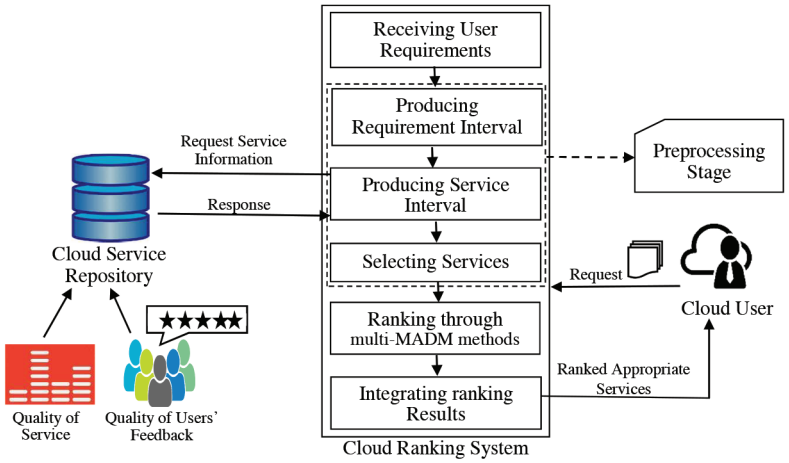

Figure 1. Architecture for OCSR method.

OCSR receives a set of services with quality values, feedback from service users, and user requirements as inputs, whereas ranked appropriate services are the outputs. Each step is discussed in detail here:

3.1. Receiving user requirements

User requirements are received from two different aspects. The first aspect is to consider user requirements in terms of QoS. In the second aspect, user requirements are received based on QoU. According to MOSS [2], 16 quality attributes have been used for cloud services. Amongst them, 8 attributes exist in QoS set which contains CPU speed, portability, memory bandwidth, disk rate, disk capacity, latency, availability, and price. In the second aspect, 8 attributes exist in QoU set, which contains ease of use, trust, security, response time, accessibility, features of serviceability, technical support, and customer service.

The user requirements are received based on the attribute importance and the required value for each attribute. Cloud users can enter the importance value of each attribute, which is an integer between 1 and 9. Table 1 shows these importance values.

Attributes that express as insignificant are not in the user's preferences. Therefore, insignificant attributes are excluded from the decision-making process. Afterward, pairwise comparison matrix (PCM) is created based on the importance values of remaining attributes for each QoS and QoU aspects separately. The PCM matrix

Table 1. Importance value.

Description	Insignificant	Somewhat important	Definitely important	Much important	Extremely important	Intermediate values
Importance	1	3	5	7	9	2, 4, 6, 8

is shown in Equation (1).

$$\mathrm{PCM}_{aspect} = \begin{bmatrix} 1 & pcm_{12} & \dots & pcm_{1m} \\ pcm_{12} & 1 & \dots & pcm_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ pcm_{m1} & pcm_{m2} & \dots & 1 \end{bmatrix} \tag{1}$$

In Equation (1), $\mathrm{PCM}_{aspect} = [pcm_{ij}]_{m \times m}$ where m is the number of remaining attributes in each aspect and pcm_{ij} is computed using (2).

$$pcm_{ij} = \frac{impotance_i}{importance_j} \tag{2}$$

In this equation, $importance_i$ and $importance_j$ denote the importance of i_{th} and j_{th} attributes. The weight of each attribute is calculated through AHP [16].

The required value for each attribute can be entered by the user based on the type of attribute (numeric, set, and interval). The numeric type, e.g. CPU speed, takes discrete values. The set type includes attributes that can take multiple values e.g., portability, representing platforms supporting cloud services. Some quality attributes are described as intervals e.g., latency (ranging from 20 to 200 seconds). After user requirements are received, the format becomes a query. Equation (3) shows the m_{th} query structure.

$$\begin{aligned} \mathrm{Q}_m = \{&(w_1, a_1, value_1, type_1, aspect_1), (w_2, a_2, value_2, type_2, aspect_2), ..., \\ &(w_i, a_i, value_i, type_i, aspect_i), ..., (w_n, a_n, value_n, type_n, aspect_n)\} \end{aligned} \tag{3}$$

In this equation, n is the number of remaining attributes, whereas w_i indicates the weight of the i_{th} attribute. a_i indicates the i_{th} attribute. The value of the i_{th} attribute required by the user is shown as $value_i$. The system also determines the i_{th} quality attribute type in $type_i$. $aspect_i$ is used to separate attributes from QoS and QoU aspects. If $aspect_i$ is 1, it belongs to QoS; otherwise, it belongs to QoU. For example, the importance value, required value, and type of four attributes (price, availability, portability, and latency) are shown in Table 2.

Table 2. User requirements for 4 QoS attributes.

Quality attribute	Importance value	Required value	Type
Price	9	(0-1] dollar/hour	Interval
Availability	7	(99-100]%	Interval
Portability	5	{IOS,Android,Linux}	Set
Latency	3	[25-125]millisecond	Interval

According to Table 2, the weight of each attribute is computed after receiving the user requirements. the query structure for received requirement is as follows.

$$Q_m = \{(0.375, price, (0-1], Interval, 1), (0.2917, availability, (0.99-1], Interval, 1),$$
$$(0.208, portability, \{IOS, Android, Linux\}, Set, 1), (0.125, latency, [25-125], Interval, 1)\}$$

For instance, $(0.125, latency, [25-125], Interval, 1)$ shows the query for the latency attribute. The calculated weight for this attribute is 0.125, and the required value is [25–125]. The latency attribute is of interval type and belongs to QoS.

3.2. Preprocessing

The services that are appropriate to user requirements are selected in the preprocessing stage, for a few number of services are ranked instead of all. This process reduces the execution time. Preprocessing involves creating the user requirement interval, creating the quality interval for cloud services, and selecting appropriate services. The preprocessing is conducted as follows:

Step 1: Creating the user requirement interval. After creating the query, the attributes are separated in two sets of QoS and QoU attributes by *aspect*. In order to encode interval and numerical types, the values are partitioned into n intervals as n can be adjusted by system. The attribute is then encoded in n bits, each bit of which corresponds to an interval. If the required value (*value*) overlaps with each subinterval, the bit is 1, otherwise 0. A number of bits corresponding to the maximum possible number of values are used for encoding set type. Bits corresponding to requirement value are 1, while other bits are 0. The longest code (L) is found at the end since various quality attributes may have different lengths of code. Then, zero bits are added to the right of the values of other attributes to bring the length of all to L. Attributes are arranged from left to right based on their weight (w). The generated user requirement code is saved in the *sorting* array.

In some cases, if attribute type is interval or set, several digits of 1 maybe generated for each attribute code. For instance, the required value for latency attribute is considered as [25–125]. Assuming that all services cover values [0–200] ms and system has specified four subintervals as [0–50), [50–100), [100–150), and [150–200). The binary code for latency will be 0111. As observed, there are several digits of 1 in the attribute code. Therefore, the attribute code is converted to simple codes to create user requirement interval. For instance, the latency's code is converted to 0001, 0010, and 0100. Since only the minimum and maximum values are important in the interval, there is no need to 0010. After converting the codes of all interval type attributes into simple code, the multiple simple codes of requirement are created from *sorting* array. Then by repeating the simple code or non-interval attribute type, the single value requirement is created. The interleaving process [4] is then performed on lower bound and upper bound of single value requirement. Afterward, interleaved lower bound and interleaved upper bound are created. Finally, the requirement interval is obtained from them. This step is carried out separately for building the QoS and QoU intervals based one user requirements. Figure 2 shows the formation of the QoS requirement interval with 4 attributes.

Step 2: Creating the quality interval for cloud services. Every cloud service has unique quality attribute values. At the same time, users provide different feedbacks for services. Hence, the QoS and QoU intervals of cloud services are created for user requirement with a similar procedure. In this regard, services are encoded based on QoS and QoU values and also importance values of attributes. According to user requirements, the QoS and QoU intervals are created for services.

Step 3: Selecting appropriate services. Appropriate services can be selected after the user requirement and services are created as intervals. The selected service has a complete overlap of QoS and QoU intervals

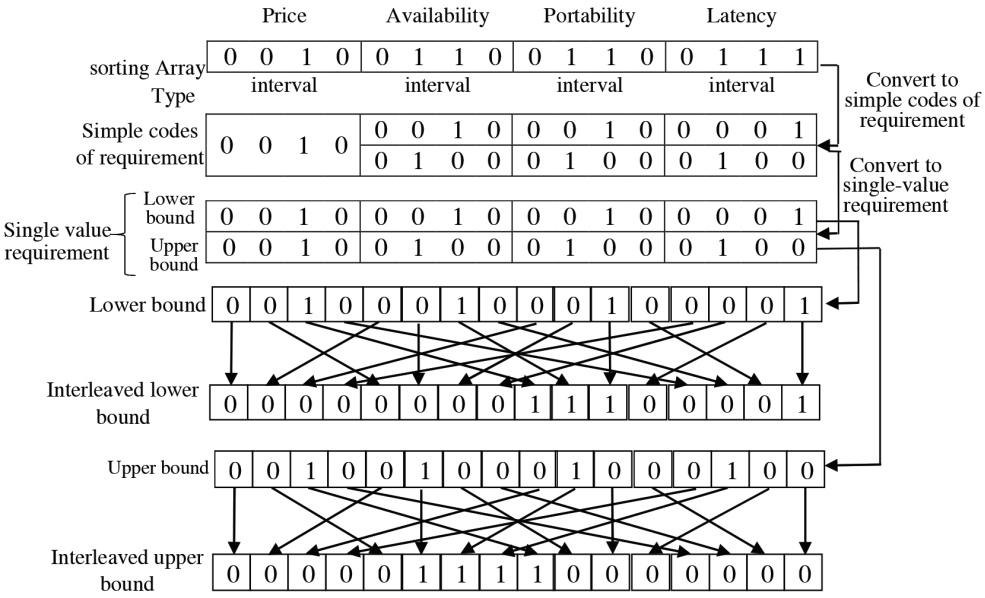

Figure 2. The creation of the user requirement interval with 4 QoS attributes.

with the QoS and QoU of requirement intervals. If no services achieve full overlap with user requirements, the service with the highest interval overlap ($IntervalOverlap$) will be selected. Equation (4) demonstrates the calculation of the overlap between the requirement interval and the i_{th} service interval.

$$IntervalOverlap = \frac{|QoS_r \cap QoS_{s_i}| + |QoU_r \cap QoU_{s_i}|}{2} \tag{4}$$

In this equation, QoS_r and QoU_r represent the requirement intervals of QoS and QoU. Likewise, QoS_{s_i} and QoU_{s_i} refer to the interval sizes of i_{th} services in QoS and QoU.

3.3. Ranking through multi-MADM methods

The appropriate services remaining from the previous stage are ranked in this stage. This stage consists of two steps: Step 1, known as separate ranking, ranks the cloud services remaining from the previous stage using MADM methods. With every method, ranking is done separately and in accordance with QoS and QoU attributes. Step 2, known as merging ranks, combines the ranks obtained from QoS and QoU in each MADM method.

Step 1: Separate ranking. In this step, the appropriate services are ranked based on QoS and QoU attributes. As stated earlier in Section 2, contradictions in the ranking results of MADM methods cause ambiguity in determining the best ranking. Therefore, OCSR uses multi-MADM method instead of one. According to [22], the main methods for ranking cloud services include AHP, PROMETHEE II, TOPSIS and VIKOR. According to results in [31], WPM has high reliability and low time complexity. In recent studies, WPM considers for ranking cloud services [2, 29]. Therefore, in this study, the multi-MADM method includes PROMETHEE II [32], AHP [16], TOPSIS [33], WPM [34], and VIKOR [18]. In all methods, the weight of the attributes and the structure for storing the values of each service must be specified. Attribute weights are

calculated in Section 3.1, and service attribute values are considered in the matrix structure (one matrix for QoS and another for QoU attributes). Each row represents a service, and each column indicates a QoS or QoU attribute value. Service attribute values should be normalized before the ranking process (all values should be between 0 and 1). The higher the value of some attributes (e.g. availability), the better (positive attributes); and at the same time, the lower the other attributes (e.g. price or response time), the better (negative attributes). Hence, normalized values of positive and negative attributes are calculated according to Equation (5).

$$x_{ij}^{norm} = \begin{cases} \frac{x_{ij}}{\max(x_{ij})}, & \textit{if } x \textit{ is positive} \\ \frac{x_{ij}}{\min(x_{ij})}, & \textit{if } x \textit{ is negative} \end{cases} \tag{5}$$

In this equation, x_{ij} is the j_{th} attribute of the i_{th} service, and the normalized value of each attribute is placed in x_{ij}^{norm}. Ranking occurs after preparing weights and attributes. The ranking methods used in this study are briefly introduced as follows:

1. PROMETHEE II

PROMETHEE II [32] carries out ranking using the weight of each attribute (QoU or QoS) and normalized the value of each attributes in Equation (5). Then, calculate the differences of i_{th} service with respect to other services (i.e. $d(v_{ij}, v_{kj})$ shows the difference between values of services s_i and s_k according to j_{th} attribute). Afterward, preference function is calculated for all pairs of services according to Equation (6).

$$p_j(s_i, s_k) = \begin{cases} 0, & d(v_{ij}, v_{kj}) \leq 0 \\ v_{ij} - v_{kj}, & d(v_{ij}, v_{kj}) > 0 \end{cases} \tag{6}$$

In this equation, $p_j(s_i, s_k)$ is preference function for j_{th} attribute between services s_i and s_k. the values of the j_{th} attribute of s_i and s_k are v_{ij} and v_{kj} respectively. Then, the aggregated preference functions $\pi(s_i, s_k)$ are calculated according to Equation (7):

$$\pi(s_i, s_k) = \frac{[\sum_{j=1}^{n} w_j p_j(s_i, s_k)]}{\sum_{j=1}^{n} w_j} \tag{7}$$

By determining the leaving and the entering outranking flows, leaving (positive) and entering (negative) flows are calculated for i_{th} service according to Equations (8) and (9).

$$\varphi^{+}(i) = \frac{1}{m-1} \sum_{k=1}^{m} \pi(s_i, s_k) \quad (i \neq k) \tag{8}$$

$$\varphi^{-}(i) = \frac{1}{m-1} \sum_{k=1}^{m} \pi(s_k, s_i) \quad (i \neq k) \tag{9}$$

where m is number of services. The net outranking flow of each service is calculated using (10). Finally, by sorting the descending values of φ, the rank of each service is obtained.

$$\varphi(i) = \varphi^{+}(i) - \varphi^{-}(i) \tag{10}$$

2. AHP

AHP [16] simplifies multi-attribute problems in the hierarchical structure. The hierarchical structure includes three levels, namely goal, attributes (QoS or QoU), and services. In this structure goal is ranked services at the highest level, attributes are in the middle, and services are at the lowest level. After preparing weights and normalizing attribute values based on Equation (5), the final value of each service is calculated using Equation (11) in which s_i indicates the final value of the i_{th} service and m is number of attributes. It ranks services in descending order.

$$s_i = (\sum_{j=1}^{m} w_j x_{ij}^{norm})/m \tag{11}$$

3. TOPSIS

TOPSIS [33] prepares weights in Section 3.1 and normalizes attribute values according to Equation (5), and then calculates the maximum (x_j^*) and minimum (x_j^-) values of each attribute. Then, the Euclidean distance of each service is calculated from the maximum and minimum attribute values using Equations (12) and (13).

$$D_i^* = (\sum_{j=1}^{n} d(x_{ij}^{norm}, x_j^*) \tag{12}$$

$$D_i^- = (\sum_{j=1}^{n} d(x_{ij}^{norm}, x_j^-) \tag{13}$$

In these equations, $i = 1, 2, 3, ..., n$ shows the number of services, whereas $j = 1, 2, 3, ..., m$ indicates the number of attributes. $d(x_{ij}^{norm}, x_j^*)$ is the distance between the maximum j_{th} attribute and the normalized i_{th} service. $d(x_{ij}^{norm}, x_j^-)$ is the distance between the minimum j_{th} attribute and the normalized j_{th} service. D_i^* and D_i^- are the distance between the i_{th} service and the maximum and minimum values, respectively. Then, the closeness coefficient (CC) is calculated according to Equation (14).

$$CC_i = \frac{D_i^-}{D_i^- + D_i^*} \tag{14}$$

In this equation, CC_i is the closeness coefficient for the i_{th} service that is calculated for all services, which are ultimately ranked in a descending order based on the values of CC.

4. WPM

WPM [34] uses Equation (5) to calculate normalized values through negative or positive quality attributes. Then, the values of each service are calculated using Equation (15) while considering attribute weights (w_j), in which m is the number of attributes and A_i^* is the value of i_{th} service. Then, service values are ranked in a descending order.

$$A_i^* = \prod_{j=1}^{m} (x_{ij}^{norm})^{w_j} \tag{15}$$

5. VIKOR

VIKOR [18] is based on the concept the best rank should be the closest to the ideal solution. After weights are created for each attribute, the comparison matrices are formed through the normalized values by using Equation (5). It is then necessary to calculate positive ideal (f_i^*) which represents the maximum value of the i_{th} positive attributes, and the negative ideal (f_i^-), which represents the minimum value in the i_{th} negative attributes. Then, the S_j and R_j values, which respectively represent utility measure and regret measure, are calculated for the j_{th} service according to Equations (16) and (17), and added to the utility (U) and regret (R) sets.

$$U_j = \sum_{i=1}^{m} w_i \frac{f_i^* - f_{ij}}{f_i^* - f_i^-} \tag{16}$$

$$R_j = \max_{i=1}[w_i \frac{f_i^* - f_{ij}}{f_i^* - f_i^-}] \tag{17}$$

In these equations, m is the number of attributes, and w is the weight of each. Next, the average values of S and R are calculated as the value of the j_{th} service using Equation (18). Finally, services are ranked with respect to Q values in descending order.

$$Q_j = \frac{\frac{U_j - U^*}{U^- - U^*} + \frac{R_j - R^-}{R_j - R^-}}{2} \tag{18}$$

In this equation, U^* and U^- represent the best and worst values in the utility set, and R^* and R^- represent the best and worst values in the regret set.

Step 2: Merging ranks. Rankings that are achieved based on QoS and QoU are merged for each MADM method. Equation (19) calculates the merged rank ($MergedRank$).

$$MergedRank^m_{S_i=1\ldots n} = \frac{V_{QoS} + V_{QoU}}{2} \tag{19}$$

In this equation, m represents the MADM method, and S_i represents the i_{th} service. V_{QoS} and V_{QoU} respectively represent the value for the i_{th} service that is obtained in ranking and based on QoS and QoU. After calculating $MergedRank$ for all services, values are ranked in descending order and the merged rank of every service is obtained using each MADM method.

3.4. Integrating ranking results

This stage integrates the ranks obtained from each method. In this regard, the average rank of each service among different ranking methods is calculated using Equation (20).

$$AR_i = \frac{\sum_{m=1}^{M} R_{im}}{M} \tag{20}$$

In this equation, AR_i is the average rank of the i_{th} service, M is the number of MADM methods, and R_{im} is the rank of the i_{th} service with the m_{th} method. If services have the same AR, the number of times services superior over each other is calculated in each MADM method. The service that is preferred by more number of MADMs ranks better. Finally, services are ranked based on their AR values.

4. Case study

The following is an example of ranking cloud services with OCSR. As mentioned in Section 3, the OCSR has four stages in which services are ranked according to user requirements and values of quality attributes (QoS and QoU) of each service. We want to rank the 37 cloud services provided in [2] according to the requirements of a cloud user.

4.1. Receiving user requirements

The user requirements are received in the requirement form in which the importance and required values of the attributes are specified. Figure 3 shows the requirements received from a cloud user.

QoS requirements	
Importance of CPU speed: **1**	Required value: **-**
Importance of Core concurrency: **1**	Required value: **-**
Importance of Memory bandwidth: **7**	Required value: **[4-8]** GB/sec
Importance of Disk rate: **5**	Required value: **[30-550]** KB/sec
Importance of Disk seeks: **1**	Required value: **-**
Importance of Latency: **8**	Required value: **[40-55]** millisecond (ms)
Importance of Availability: **6**	Required value: **[99.95-100] %**
Importance of Price: **9**	Required value: **[25-145]** cent/hour (₡/h)
QoU requirements	
Importance of Security: **9**	Required value: **(2-4]**
Importance of Response time: **6**	Required value: **(3-5]**
Importance of Accessibility: **5**	Required value: **(3-4]**
Importance of Features of serviceability (Features): **3**	Required value: **(2-4]**
Importance of Ease of use: **1**	Required value: **-**
Importance of Technical support: **1**	Required value: **-**
Importance of Customer service: **1**	Required value: **-**
Importance of Trust: **8**	Required value: **(2-4]**

Figure 3. Requirements received from a cloud user.

As shown in Figure 3, the importance values and the required values for QoS and QoU attributes are entered by the user. The importance values are specified by numbers (1 to 9) according to Table 1. Attributes with an importance value of 1 are insignificant to this user. Therefore, they are not considered in the decision-making process. The weights of the remaining attributes are then calculated. For this purpose, the PCM_{QoS} matrix is constructed based on the importance of QoS attributes relative to each other. The PCM_{QoU} matrix is also constructed based on the importance of QoU attributes relative to each other. The PCM_{QoS} and PCM_{QoU} matrices are as follows.

$$\mathrm{PCM_{QoS}} = \begin{array}{c} \\ \text{Latency} \\ \text{Memory bandwidth} \\ \text{Disk rate} \\ \text{Availability} \\ \text{Price} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{Latency} & \text{Memory bandwidth} & \text{Disk rate} & \text{Availability} & \text{Price} \end{array} \\ \begin{bmatrix} 1 & 8/7 & 8/5 & 8/6 & 8/9 \\ 7/8 & 1 & 7/5 & 7/6 & 7/9 \\ 5/8 & 5/7 & 1 & 5/6 & 5/9 \\ 6/8 & 6/7 & 6/5 & 1 & 6/9 \\ 9/8 & 9/7 & 9/5 & 9/6 & 1 \end{bmatrix} \end{array}$$

$$\mathrm{PCM_{QoU}} = \begin{array}{c} \\ \text{Security} \\ \text{Response time} \\ \text{Accessibility} \\ \text{Features} \\ \text{Trust} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{Security} & \text{Response time} & \text{Accessibility} & \text{Features} & \text{Trust} \end{array} \\ \begin{bmatrix} 1 & 9/6 & 9/5 & 9/3 & 9/8 \\ 6/9 & 1 & 6/5 & 6/3 & 6/8 \\ 5/9 & 5/6 & 1 & 5/3 & 5/8 \\ 3/9 & 3/6 & 3/5 & 1 & 3/8 \\ 8/9 & 8/6 & 8/5 & 8/3 & 1 \end{bmatrix} \end{array}$$

In each PCM matrix, the weights of the attributes are calculated according to the AHP method. Finally, the query structure (Q) is created according to Equation (3).

4.2. Preprocessing

In this stage, the requirement intervals are created in both QoS and QoU aspects. In each aspect, each attribute must be encoded to create the user requirement interval. Encoding is based on user requirements and subintervals specified by the system (system subintervals). Table 3 shows the codes corresponding to each attribute of the QoS aspect.

Table 3. Encoding for each QoS attribute.

QoS Attribute	User required value	System subintervals	Attribute code
Latency	[40–55] ms	[0–50), [50–100), [100–150), [150–200)	0011
Memory bandwidth	[4–8] GB/sec	[0–5), [5–10), [10–15), [15–20)	0011
Disk rate	[30-550] KB/sec	[0–300), [300–600), [600–900), [900–1200)	0011
Availability	[99.95–100] %	[99.7–99.8), [99.8–99.9), [99.9–99.95), [99.95–100]	1000
Price	[25–145] cent/h	[0–50), [50–100), [100–150), [150–200)	0111

As shown in Table 3, there are 4 system subintervals for each attribute, so 4 bits are assigned to each attribute with the default value of 0. If the user required value matches any system subinterval, that bit will be 1. Table 4 shows the corresponding code for each QoU attribute according to the user requirements and the system subintervals. The system subintervals in QoU are based on the level of user satisfaction. These levels are considered in the range between 1 to 5 ([1–5]). In this case, 1 means the lowest level of user satisfaction and 5 means the highest level of user satisfaction.

The calculations for generating the required interval are described in Section 3.2. Therefore, we have skipped the details. The requirement intervals for QoS and QoU aspects are as follows:

$RequirementInterval_{QoS} = [(1000000010011001)_2, (10100001110100000)_2] = [65689, 82848]$

$RequirementInterval_{QoU} = [(11011001)_2, (10011011000000000)_2] = [217, 158720]$

After this step, 9 cloud services are selected as appropriate services out of 37 cloud services. The values of services in the QoS and QoU aspects are given in Tables 5 and 6.

Table 4. Encoding for each QoU attribute.

QoU Attribute	User Required value	System subintervals	Attribute Code
Security	(2-4]	[1-2],(2-3],(3-4],(4-5]	0110
Response time	(3-5]	[1-2],(2-3],(3-4],(4-5]	1100
Accessibility	(3-4]	[1-2],(2-3],(3-4],(4-5]	0100
Features	(2-4]	[1-2],(2-3],(3-4],(4-5]	0110
Trust	(2-4]	[1-2],(2-3],(3-4],(4-5]	0110

Table 5. QoS values for appropriate services.

Services	Latency	Memory bandwidth	Disk rate	Availability	Price
S_1	44	66	512	99.96	96
S_2	47	68	512	99.95	144
S_3	42	58	538	99.97	238
S_4	45	63	36	99.95	62
S_5	47	59	36	99.98	312
S_6	47	74	321	99.99	136
S_7	46	76	321	99.97	68
S_8	53	96	99	99.95	56
S_9	53	92	99	99.95	28

Table 6. QoU values for appropriate services.

Services	Security	Response time	Accessibility	Features	Trust
S_1	2.2567	3.0522	3.2002	2.5893	2.6056
S_2	3.1625	3.6966	3.1125	3.100	3.5341
S_3	3.9976	3.7425	3.0551	2.5989	3.4441
S_4	4.1664	3.6058	3.2384	2.8200	3.4601
S_5	4.2720	4.6743	4.0194	2.5088	4.5703
S_6	4.3029	4.5822	4.1775	2.2977	4.5640
S_7	3.7657	3.9521	3.4372	2.5939	3.8398
S_8	2.8254	3.6443	3.1296	2.5061	3.3164
S_9	3.5344	4.0760	3.5273	3.3682	2.7732

4.3. Ranking through multi-MADM methods

At this stage, the appropriate services are evaluated. The multi-MADM procedure is used for ranking as described in Section 3.3. For this purpose, five MADM methods i.e. PROMETHEE II, AHP, WPM, TOPSIS, and VIKOR rank the appropriate services based on QoU and QoS separately. Due to space constraints, the ranking details of each MADM method are not provided and only the ranking results are presented. Table 7 shows the QoS and QoU ranking results for each method.

According to Table 7, there is no consensus between rankings on QoS and QoU aspects. For example, the AHP method in the QoS aspect ranks S_5 at number 1. While in the QoU aspect, it ranks S_5 at number 3.

Table 7. QoS and QoU ranking results for MADM methods.

Services	AHP		VIKOR		TOPSIS		PROMETHEE II		WPM	
	QoS	QoU	QoS	QoU	QoS	QoU	QoS	QoU	QoS	QoU
S_1	3	5	9	5	3	5	3	5	3	5
S_2	9	6	8	6	9	6	9	6	9	6
S_3	5	7	7	7	7	7	8	7	8	9
S_4	8	4	5	4	8	4	7	4	7	7
S_5	1	3	1	3	5	3	5	3	1	4
S_6	7	9	3	2	1	9	1	9	5	2
S_7	2	2	4	9	4	2	2	2	4	3
S_8	6	8	6	8	2	8	6	8	2	8
S_9	4	1	2	1	6	1	4	1	6	1

Therefore, the rankings generated in QoS and QoU aspects must be merged in each MADM method. OCSR uses Equation (19) to merge the service rankings. Table 8 shows the results of $MergedRank$, where $>$ means having a higher rank.

Table 8. Results of merged ranks

MADM methods	Merged ranks
AHP	S_5>S_3>S_9>S_7>S_6> S_2>S_8>S_4>S_1
VIKOR	S_5>S_7>S_9>S_6> S_8>S_4>S_3>S_1>S_2
TOPSIS	S_3>S_5>S_7>S_9>S_6> S_4>S_8>S_2>S_1
PROMETHEE II	S_5>S_7>S_3>S_9>S_6 >S_8>S_4>S_2>S_1
WPM	S_9>S_5>S_3>S_7>S_6> S_8>S_4>S_2>S_1

As can be observed, the ranking results of MADM methods are different from each other. AHP, VIKOR, and PROMETHEE II methods rank S_5 at number 1. The WPM method selects the S_9 service as the best service. In TOPSIS method, S_3 ranks at number 1. Therefore, the ranking results of MADM methods are not consistent with each other and should be integrated.

4.4. Integrating ranking results

In this stage, the ranking results of all the methods are integrated to create the final ranking. For this purpose, Equation (20) is used to calculate the AR for appropriate services. Table 9 shows the values obtained for AR in each service.

Table 9. Integrated ranks of appropriate services.

Services	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
AR	8.8	7.8	3.2	6.8	1.4	4.8	3	6.2	3

As shown in Table 9, the lowest value of the AR is related to S_5. Therefore, S_5 ranks at number 1. If the AR for two services is equal, the number of times they were superior to each other is calculated. For example,

according to Table 9, the AR for S_9 and S_7 is equal. According to Table 8, the number of times S_9 is ranked higher than S_7 is 2 and the number of times S_7 is ranked higher than S_9 is 3. So it becomes $S_7 > S_9$. Finally, the services are ranked as follows.

$S_5 > S_7 > S_9 > S_3 > S_6 > S_8 > S_4 > S_2 > S_1$.

5. Performance comparison

In order to evaluate OCSR approach, the cloud services have been presented in [2] were analyzed in order to simulate real information for use in the ranking system. In total, 16 quality attributes have been used for cloud services. Amongst them, 8 attributes exist on QoS, and the other attributes are QoU attributes. In this study, we used random values to increase the number of services. The minimum and maximum values recorded for each attribute has been considered an interval. The random values in the interval were then generated with a normal distribution. User requirements were also generated with a normal distribution between the maximum and minimum QoS or QoU values. All of the tests were conducted on a computer with an Intel Core i7 Duo 2.4 processor on Windows 10 Enterprise x64. Since this study aims to improve cloud service ranking based on MADM methods, the OCSR method was compared with MOSS [2] and the well-known MADM methods including PROMETHEE II, AHP, WPM, TOPSIS, and VIKOR. MOSS is considered as the baseline method as their results are the aggregate of other methods. All methods were evaluated in the aspect of optimality, sensitivity analysis, and scalability. Optimality compares the ranking results of various methods with the best ranking results [4]. Sensitivity analysis measures the effectiveness of changes in ranking results [13]. Scalability has been used to assess the impact of execution time changes by increasing the number of services and users [4].

5.1. Optimality

Optimality in cloud service ranking means that the best services with the best ranking are delivered to users in order to perfectly meet their requirement. MOSS performs MADM calculations on all services. Therefore, its ranking results are used as baseline to evaluate the results of other methods. The average of the evaluation results ($Optimality_{average}$) calculated according to the Spearman's correlation analysis (ρ_r) [29] and the Normalized Discount Cumulative Gain (NDCG) [28] parameters. It used to compare optimality between various ranking methods. Spearman's Correlation Analysis is calculated based on the ranking difference between the two methods using Equation (21).

$$\rho_r = 1 - \left(\frac{6\sum d_i^2}{n(n-1)}\right) \tag{21}$$

In this equation, n is the number of services and d_i is the service ranking difference with each ranking method. NDCG is also widely used as a metric for evaluating ranking quality. The NDCG parameter is calculated using Equation (22).

$$NDCG@K = \frac{1}{z_K}\sum_{i=1}^{K}\frac{\mathrm{REL}_i \times (K-i+1)}{\log_2(max(i,2))} \tag{22}$$

In this equation, REL_i is ranking fitness compared to the baseline in the i_{th} ranking. If the correlation is met on the i_{th} ranking, its value would be 1, otherwise it would be 0. K is the length of services, and Z_K is the normalizing constant that is obtained by calculating the baseline ranking. $NDCG@K$ stands for K services in which the NDCG parameter compares the ranking results of each method with the baseline. The higher the $NDCG@K$ value, the closer the ranking indicator is to the optimum.

In order to calculate the optimality of the OCSR method, 100 cloud services were randomly selected. Appropriate services were selected after a random query, and the ranking was carried out using multiple MADM methods. The results were then separately compared with the MOSS method. In this regard, the results of each individual method was compared based on the $NDCG@K$ and Spearman's correlation analysis metrics. Finally, the average of these two metrics was calculated in the $Optimality_{average}$. These operations were repeated 30 times. Table 10 shows the average optimality comparison of various methods. Results demonstrate that the proposed OCSR method is ideal for ranking with a 100% optimality. It is followed by the AHP and TOPSIS methods that perform the ranking with a nearly identical optimality (75%).

Table 10. Optimality comparison of different methods.

	AHP	PROMETHEE II	TOPSIS	VIKOR	WPM	OCSR
$Optimality_{\rho_r}$	0.8776	0.5788	0.9345	0.717	0.7524	**1**
$Optimality_{NDCG}$	0.6299	0.4055	0.5809	0.4055	0.4055	**1**
$Optimality_{average}$	0.7538	0.49215	0.7577	0.56125	0.57895	**1**

5.2. Sensitivity analysis

Sensitivity analysis determines how small changes affect the results [29]. If these changes do not affect the results, it will be the robust ranking [35]. In this test, the number of attributes was varied from 1 to 16. We tested them in the dataset containing 20 cloud services which were selected randomly. In each step, the ranking results of the OCSR, AHP, VIKOR, PROMETHEE II, WPM, and TOPSIS were compared with baseline. To compare the ranking results, ρ_r has been used as Spearman's correlation analysis. In addition, if $0.8 < \rho_r \leq 1.0$ is obtained for a method, then the correlation is interpreted as *very strong* and the method is robust [29]. Figure 4 shows the results of the sensitivity analysis.

As depicted in Figure 4, with increasing the number of quality attributes, the OCSR ranking results are the same as the baseline ($\rho_r = 1$). This is because the OCSR can select appropriate services well and ranking is done with the same procedure in the smaller search space. Therefore, according to the results, the OCSR is robust and is not sensitive to increasing the number of attributes. In other methods, the correlation of the results with baseline decreases with increasing the number of attributes. This is because different procedures of MADM methods cause differences in the ranking results. As depicted, these differences are increased with increasing the number of attributes. As the ρ_r in AHP and TOPSIS is larger than 0.8, they can be interpreted as *very strong* and they are also robust.

5.3. Scalability

In order to evaluate performance in terms of scalability, this section evaluates the effect of a number of cloud services and the number of users on the response time. In both tests, 30 random queries were produced. Therefore, each ranking method was executed 30 times. Then the average execution time for each method was calculated.

Increasing the Number of Services. Here, the tests were conducted with 10,000 to 100,000 services with 10,000 steps in between. Figure 5 shows the execution time of the seven different methods.

As depicted in Figure 5, the MOSS method is shown to have the highest execution time, for it needs to evaluate all cloud services for each query. Furthermore, this method includes WPM, PROMETHEE II, TOPSIS,

QoS requirements	
Importance of CPU speed: **1**	Required value: **-**
Importance of Core concurrency: **1**	Required value: **-**
Importance of Memory bandwidth: **7**	Required value: **[4-8]** GB/sec
Importance of Disk rate: **5**	Required value: **[30-550]** KB/sec
Importance of Disk seeks: **1**	Required value: **-**
Importance of Latency: **8**	Required value: **[40-55]** millisecond (ms)
Importance of Availability: **6**	Required value: **[99.95-100] %**
Importance of Price: **9**	Required value: **[25-145]** cent/hour (₡/h)
QoU requirements	
Importance of Security: **9**	Required value: **(2-4]**
Importance of Response time: **6**	Required value: **(3-5]**
Importance of Accessibility: **5**	Required value: **(3-4]**
Importance of Features of serviceability (Features): **3**	Required value: **(2-4]**
Importance of Ease of use: **1**	Required value: **-**
Importance of Technical support: **1**	Required value: **-**
Importance of Customer service: **1**	Required value: **-**
Importance of Trust: **8**	Required value: **(2-4]**

Figure 4. The effect of number of quality attributes on cloud service ranking.

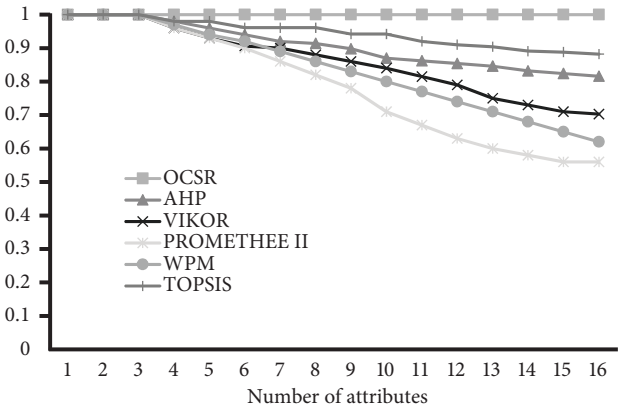

Figure 5. The effect of the number of cloud services on the execution Time of Various methods

AHP, and VIKOR techniques. The execution time of the OCSR method is less than the MOSS method because of preprocessing and reduction in ranking space. Due to the matrix structure of MADM methods, the number of cloud services is effective in the execution time. Therefore, although OCSR is the combination of five methods, it has a lower execution time than PROMETHEE II, TOPSIS, and VIKOR. For example, for 60,000 cloud services, an average of 11,620 cloud services is selected as the appropriate services in the preprocessing stage of

OCSR. Preprocessing takes 0.0059 seconds and ranking of selected services takes 0.1409 seconds. As a result, the total execution time is 0.1468. While VIKOR, PROMETHEE II, and TOPSIS methods need 0.223, 0.24, and 0.1546 seconds to rank 60,000 cloud services, respectively.

Increasing the Number of Users. In this test, 100,000 cloud services and 50 users with different requirements are considered to perform ranking. The responses are created sequentially. Figure 6 shows the effect of number of users on execution time.

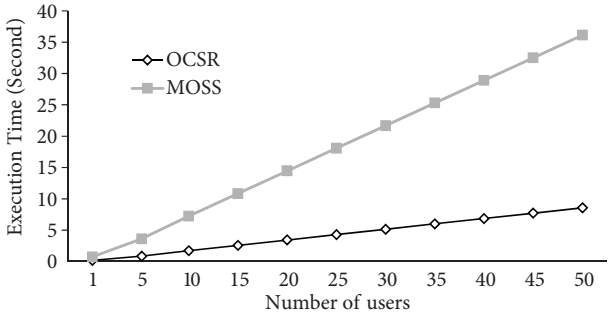

Figure 6. Effect of the number of users on execution time.

As shown in Figure 6, with increasing the number of users, OCSR depicted less execution time than the MOSS method. This is because the preprocessing stage in the OCSR method eliminates inappropriate services automatically and reduces the ranking space computations. For example, to respond to 20 cloud users, the average execution time is as follows. For the OCSR method, the preprocessing takes 0.0084 seconds per user. The average number of services selected per user is 19,370. The ranking of selected services takes 0.1628 seconds. Therefore, for 20 users, the preprocessing and ranking cloud services should be repeated 20 times and the ranking takes 3.424 s. On the other hand, the MOSS method takes 14.4518 s. This is because MOSS needs to compare all of the services and it must be repeated for each user. Given that responses are assumed to be received sequentially (in a queue), as users increase, the execution time increases in MOSS significantly.

6. Conclusion

This paper proposed the OCSR method for optimal and fast ranking of cloud services. For this purpose, the imprecise user requirements are received, and the requirement and cloud service intervals are created based on the QoU and QoS attributes. The services meeting user requirements are then selected accordingly. Selecting cloud services while considering QoS and QoU reduces the execution time as well as the risk of selecting low-quality services. In this study, AHP is used to calculate the weight of attributes. Since MADM yielded contradictory results, multiple MADM methods (including AHP, PROMETHEE II, TOPSIS, WPM, and VIKOR) are used for ranking. The final ranking was obtained by merging the results of each method. The proposed method has evaluated cloud services by comparing response time and optimality. The experimental results show that OCSR is 100% optimal and robust. Also, execution time has been reduced compared to MOSS. At the same time, the ranking optimality analysis has indicated that the OCSR method is highly correlated with the baseline ranking.

Future research intends to rank cloud services in parallel by using MADM methods while considering parallel processing structures that minimize the ranking execution time.

References

[1] Garg SK, Versteeg S, Buyya R. A framework for ranking of cloud computing services. Future Generation Computer Systems 2013; 29 (4):1012-1023.

[2] Hussain A, Chun J, Khan M. A novel customer-centric Methodology for Optimal Service Selection (MOSS) in a cloud environment. Future Generation Computer Systems 2020; 105:562-580.

[3] Jahani A, Khanli LM. Cloud service ranking as a multi objective optimization problem. The Journal of Supercomputing 2016; 72 (5):1897-1926.

[4] Lin D, Squicciarini AC, Dondapati VN, Sundareswaran S. A cloud brokerage architecture for efficient cloud service selection. IEEE Transactions on Services Computing 2016; 12 (1):144-157.

[5] Abdel-Basset M, Mohamed M, Chang V. NMCDA: A framework for evaluating cloud computing services. Future Generation Computer Systems 2018; 86:12-29.

[6] Yu P-L. Multiple-criteria decision making: concepts, techniques, and extensions. Springer Science & Business Media; 2013.

[7] Siegel J, Perdue J. Cloud services measures for global use: the service measurement index (SMI). In: 2012 Annual SRII global conference: IEEE; 2012. pp. 411-415.

[8] Iosup A, Ostermann S, Yigitbasi MN, Prodan R, Fahringer T et al. Performance analysis of cloud computing services for many-tasks scientific computing. IEEE Transactions on Parallel and Distributed systems 2011; 22 (6): 931-945.

[9] Ribas M, Furtado C, de Souza JN, Barroso GC, Moura A et al. A Petri net-based decision-making framework for assessing cloud services adoption: The use of spot instances for cost reduction. Journal of Network and Computer Applications 2015; 57:102-118.

[10] Han S-M, Hassan MM, 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; 2009. pp. 839-845.

[11] Li A, Yang X, Kandula S, Zhang M. CloudCmp: comparing public cloud providers. In: Proceedings of the 10th ACM SIGCOMM conference on Internet measurement; 2010. pp. 1-14.

[12] Li J, Squicciarini AC, Lin D, Sundareswaran S, Jia C. MMB ^{cloud}-Tree: Authenticated Index for Verifiable Cloud Service Selection. IEEE Transactions on Dependable and Secure computing 2015; 14 (2):185-198.

[13] Kumar RR, Mishra S, Kumar C. Prioritizing the solution of cloud service selection using integrated MCDM methods under Fuzzy environment. The Journal of Supercomputing 2017; 73 (11): 4652-4682.

[14] Sun L, Ma J, Zhang Y, Dong H, Hussain FK. Cloud-FuSeR: Fuzzy ontology and MCDM based cloud service selection. Future Generation Computer Systems 2016; 57: 42-55.

[15] Sun L, Dong H, Hussain FK, Hussain OK, Chang E. Cloud service selection: State-of-the-art and future research directions. Journal of Network and Computer Applications 2014; 45:134-150.

[16] Saaty TL. Decision making with the analytic hierarchy process. International Journal of Services Sciences 2008; 1 (1): 83-98.

[17] Karim R, Ding C, Miri A. An end-to-end QoS mapping approach for cloud service selection. In: 2013 IEEE ninth world congress on services: IEEE; 2013. pp. 341-348.

[18] Opricovic S, Tzeng G-H. Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS. European journal of operational research 2004; 156 (2):445-455.

[19] Triantaphyllou E. Multi-criteria decision making methods. In: Multi-criteria decision making methods: A comparative study: Springer; 2000. pp. 5-21.

[20] Rezaei J, Nispeling T, Sarkis J, Tavasszy L. A supplier selection life cycle approach integrating traditional and environmental criteria using the best worst method. Journal of Cleaner Production 2016; 135: 577-588.

[21] Toinard C, Ravier T, Cérin C, Ngoko Y. The promethee method for cloud brokering with trust and assurance criteria. In: 2015 IEEE International Parallel and Distributed Processing Symposium Workshop: IEEE; 2015. pp. 1109-1118.

[22] Goraya MS, Singh D. A comparative analysis of prominently used MCDM methods in cloud environment. The Journal of Supercomputing 2021; 77 (4): 3422-3449.

[23] Tajvidi M, Ranjan R, Kolodziej J, Wang L. Fuzzy cloud service selection framework. In: 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet): IEEE; 2014. pp. 443-448.

[24] 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. Wireless Personal Communications 2016; 86 (1): 57-75.

[25] Alhanahnah M, Bertok P, Tari Z, Alouneh S. Context-aware multifaceted trust framework for evaluating trustworthiness of cloud providers. Future Generation Computer Systems 2018; 79:488-499.

[26] Bhushan SB, Pradeep RC. A network QoS aware service ranking using hybrid AHP-PROMETHEE method in multi-cloud domain. In: International Journal of Engineering Research in Africa: Trans Tech Publ; 2016. pp. 153-164.

[27] Jatoth C, Gangadharan G, Fiore U, Buyya R. SELCLOUD: a hybrid multi-criteria decision-making model for selection of cloud services. Soft Computing 2019; 23 (13): 4701-4715.

[28] Jiang Y, Tao D, Liu Y, Sun J, Ling H. Cloud service recommendation based on unstructured textual information. Future Generation Computer Systems 2019; 97: 387-396.

[29] Gireesha O, Somu N, Krithivasan K, VS SS. IIVIFS-WASPAS: an integrated multi-criteria decision-making perspective for cloud service provider selection. Future Generation Computer Systems 2020; 103: 91-110.

[30] Alabool H, Kamil A, Arshad N, Alarabiat D. Cloud service evaluation method-based Multi-Criteria Decision-Making: A systematic literature review. Journal of Systems and Software 2018; 139:161-188.

[31] Kumar U, Ahmadi A, Verma AK, Varde P. Current trends in reliability, availability, maintainability and safety: an industry perspective. Springer; 2015.

[32] Brans J-P, Vincke P. Note—A preference ranking organisation method: (The PROMETHEE method for multiple criteria decision-making). Management Science 1985; 31 (6): 647-656.

[33] Yoon K. A reconciliation among discrete compromise solutions. Journal of the Operational Research Society 1987; 38 (3): 277-286.

[34] Triantaphyllou E, Mann SH. An examination of the effectiveness of multi-dimensional decision-making methods: A decision-making paradox. Decision Support Systems 1989; 5 (3): 303-312.

[35] Christopher Frey H, Patil SR. Identification and review of sensitivity analysis methods. Risk Analysis 2002; 22 (3): 553-578.