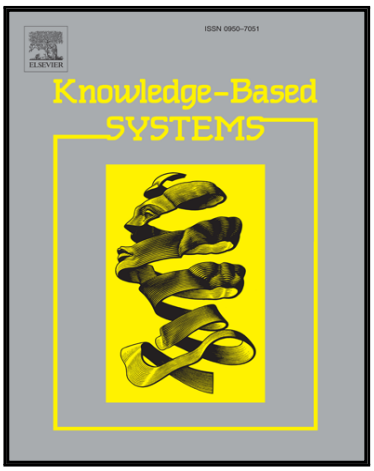

nnCloud Service Selection Using a Markov Chain and the Best-Worst Method

Falak Nawaz , Mehdi Rajabi Asadabadi , Naeem Khalid Janjua , Omar Khadeer Hussain , Elizabeth Chang , Morteza Saberi

PII: S0950-7051(18)30317-4
DOI: 10.1016/j.knosys.2018.06.010
Reference: KNOSYS 4374

To appear in: *Knowledge-Based Systems*

Received date: 29 November 2017
Revised date: 27 April 2018
Accepted date: 12 June 2018

Please cite this article as: Falak Nawaz , Mehdi Rajabi Asadabadi , Naeem Khalid Janjua , Omar Khadeer Hussain , Elizabeth Chang , Morteza Saberi , nnCloud Service Selection Using a Markov Chain and the Best-Worst Method, *Knowledge-Based Systems* (2018), doi: 10.1016/j.knosys.2018.06.010

Highlights

- This paper is the first application of Best Worst Method in could service selection.
- A Markov chain is employed to find a pattern of uncertain customer needs.
- The method results in less consistency in comparison with other similar methods.

Cloud Service Selection Using a Markov Chain and the Best-Worst Method

Falak Nawaz[1], Mehdi Rajabi Asadabadi[1], Naeem Khalid Janjua[2], Omar Khadeer Hussain[1], Elizabeth Chang[1], Morteza Saberi[1]

[1] School of Business, University of New South Wales (UNSW), Canberra, Australia
[2] School of Science, Edith Cowan University (ECU), Perth, Australia
{falak.nawaz, mehdi.asadabadi}@student.adfa.edu.au, {o.hussain, e.chang, m.saberi}@adfa.edu.au, n.janjua@ecu.edu.au

Corresponding Author:

Mehdi Rajabi Asadabadi
rajabi689@yahoo.com
School of Business,
THE UNIVERSITY OF NEW SOUTH WALES
Australia
+61468570489

Cloud Service Selection Using a Markov Chain and the Best-Worst Method

Abstract

Due to the increasing number of cloud services, service selection has become a challenging decision for many organisations. It is even more complicated when cloud users change their preferences based on the requirements and the level of satisfaction of the experienced service. The purpose of this paper is to overcome this drawback and develop a cloud broker architecture for cloud service selection by finding a pattern of the changing priorities of User Preferences (UPs). To do that, a Markov chain is employed to find the pattern. The pattern is then connected to the Quality of Service (QoS) for the available services. A recently proposed Multi Criteria Decision Making (MCDM) method, Best Worst Method (BWM), is used to rank the services. We show that the method outperforms the Analytic Hierarchy Process (AHP). The proposed methodology provides a prioritized list of the services based on the pattern of changing UPs. The methodology is validated through a case study using real QoS performance data of Amazon Elastic Compute (Amazon EC2) cloud services.

Keywords: Cloud Service Selection; MCDM methods; Best Worst Method; Markov chain

1 Introduction

Cloud computing is defined as a model that enables ubiquitous, convenient, and on-demand network access to a shared pool of configurable computing resources. Such resources can be rapidly provisioned through virtualization and released with minimal management effort [1]. Cloud computing has changed the perception of how computational resources can be procured and provisioned with scalability and efficiency [2][3]. As a result, cloud users are now able to focus on their core competencies and leave management of their computational resources to the cloud providers.

As the number of cloud services is constantly growing, a user is exposed to many choices. Having such choices leads to the challenge of selecting the right service from the right cloud provider at the right time [4]. A cloud broker architecture (Figure 1) that takes into account the user requirements gives a ranked list of potential cloud services is proposed in order to reduce the complexity of service selection for the cloud user. Equinix [5] is one such example of a cloud broker architecture which has more than 500 registered cloud providers, each of which provides different types of cloud services. Amazon provides more than 70 [6]. Various cloud service selection algorithms have been employed in cloud broker-based recommendation engines. To the best of our knowledge none of them takes into account the changing needs of the cloud user for best service recommendation.

Traditionally, cloud service selection is considered the process of finding the most suitable cloud service [7] by matching the functional and Quality of Service (QoS) requirements of a user with the description of available cloud services provided by different cloud providers [8]. Existing research [9]–[14] has typically focused on what criteria are to be considered (e.g. performance, price) and the preferences of the user (e.g. which criteria are important to the cloud user). Although different decision-making methods are then applied to the criteria and user requirements to help cloud users to find a suitable cloud service, some uncertainties in the cloud environment hinder the service users from relying on such methods [15]. One such uncertainty is that User Preferences (UPs) are dependent on the user requirements, which dictate the user's level of satisfaction with the service. This level of satisfaction may change over a period of time during service consumption. None of the approaches in current literature captures this information and uses it to improve the service recommendation for future users. For example, when users are utilizing a cloud service, they may change their preferences based on their experience with the service being used. The changed preferences can be collected from the users to understand their satisfaction before and after using the recommended service. Then by grouping the users with similar service requirements, their level of satisfaction can be analyzed to determine any pattern, as shown in a previous study [16]. This would help in the recommendation and selection of the most appropriate cloud service for other users having similar service requirements.

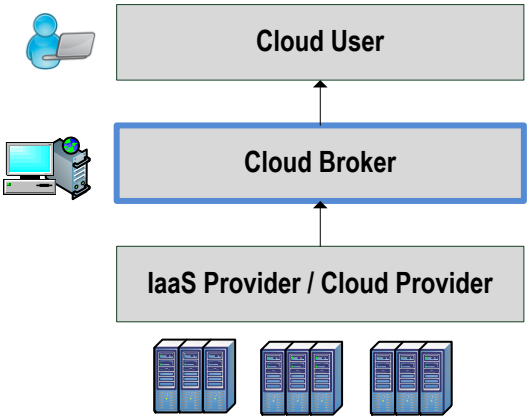

Figure 1. An overview of Cloud Brokerage Architecture

In this paper, we propose a cloud broker architecture for cloud service selection by taking into account the changing UPs over time. We use a Markov chain to capture and track the changes in UPs and find transition patterns in them. The identified patterns are then used for recommendation and selection of the suitable services. Finally, Best Worst Method (BWM) [17] which is a Multi Criteria Decision Making (MCDM) method, is used to rank the services. The main contributions of our work are as follows:

- By using a Markov chain, we capture and model the uncertainty in changes to find the transitioning patterns within them.
- We use a Markov chain in conjunction with BWM to rank the services based on changing user preferences.
- We evaluate our proposed model on real cloud services data. Results show our proposed method results in greater consistency when compared to Analytic Hierarchy Process (AHP)-based approaches that are currently used in cloud service selection.

This paper is organized as follows. In the next section, the related research on cloud service selection is discussed. The framework for cloud service selection is proposed in section 3. In section 4, the implementation of the proposed methodology is presented in algorithmic form. The approach is then validated using a case study in section 5. Section 6 presents a discussion and the performance evaluation of the proposed methodology. Finally, a conclusion is presented in section 7.

2 Related Work

In recent years, a number of research efforts have started solving the problem of cloud service selection. In this section, we present an overview of some of these studies. Existing research is divided into QoS-based and non-QoS-based service selection. QoS-based service selection approaches consider the QoS criteria for decision-making. Table 1 provides the categorization of existing service selection methods used in the literature.

An approach for selecting cloud services was proposed by Rehman, Hussain and Hussain [12] in which they argue that in order to make an informed decision, it is necessary to apply a cloud service selection methodology that utilizes QoS history over different time periods. The proposed approach ranks all services using MCDM in each time period and aggregates the results to determine the overall rank of all available options for cloud service selection.

The variations in the historic QoS are considered by assigning weights using the logistic decay function. A cloud service recommender system is employed by Han, Mehedi Hassan, Yoon, Lee and Huh [18] for the cloud market that matches a specific user's requirements with a suitable cloud service. The method, therefore, selects the best combination of services from different cloud providers. The proposed system maintains a record of all the available resources in the cloud market and uses this information to rank and calculate the QoS values of services. Zeng, Zhao and Zeng [13] designed a cloud service selection algorithm using the maximized-gain and

minimized-cost approach. The algorithm first uses a service discoverer to find all available services and then processes the cloud service user's request. The maximized-gain and minimized-cost service selection algorithm aggregates the gain and cost values by a weighted sum where weights represent the relative importance of involved factors. Yu [19] presented a framework for personalized service recommendation in cloud that implements a user-centric strategy to achieve personalized QoS assessment of cloud services. The proposed framework uses a collaborative filtering technique for a community-based QoS assessment model.

Table 1. Categorization of existing approaches to service selection

	Multi-criteria Decision Making (MCDM)	**Optimization-based Methods**
Focus	MCDM-based methods focus on translating decision-maker's preferences into a set of variables and then finds the best option from a set of finite alternatives [20].	Optimization-based methods seek to find the set of values for decision variables which maximize or minimize the objective function without violation of constraints [21].
Examples	Techniques such as Analytic Hierarchy Process (AHP) [22], Analytic Network Process (ANP) [23], Multi-Attribute Utility Theory (MAUT) [24] and outranking [25] fall into this category.	Techniques such as Dynamic Programming [26], Integer Linear Programming (ILP) [27], Genetic Algorithm (GA) [28] and Stochastic Programming [29] fall into this category.

Many existing approaches for QoS-based cloud service selection use the AHP technique to assign ranks to cloud services. Garg, Versteeg and Buyya [7] proposed SMICloud, which is based on Service Measurement Index (SMI) [30], for comparing and ranking cloud services on the basis of common characteristics defined as SMI criteria. The proposed framework measures all the QoS attributes in SMI and then uses AHP to rank the cloud services. Another approach for selecting cloud services was proposed by Godse and Mulik [11] in which they applied AHP using QoS criteria such as functionality, architecture, usability, vendor reputation, and cost for service selection. Sun, Ma, Zhang, Dong and Hussain [14] presented a fuzzy decision-making framework and MCDM-based approach for cloud service selection. That approach was based on a fuzzy ontology that models uncertain relationships between objects in databases for service matching, calculates the semantic similarity between concepts using AHP, and then uses multi-criteria decision-making to rank cloud services.

Non-QoS-based approaches define and measure different attributes and metrics for cloud service selection. Li, Yang, Kandula and Zhang [31] discussed the cloud service selection problem and identified the basic attributes for each type of cloud service (such as IaaS, SaaS) that must be taken into consideration when comparing one cloud service with another. Kang and Sim [32] developed a semantic-based cloud service search engine called Cloudle using a cloud ontology. The proposed system maintains a record of all the services in a database. The user's search query is sent to an engine that performs similarity reasoning between the query and the concepts in the database using cloud ontology. The output of the Cloudle search engine is an ordered list of cloud services based on concept similarity, price and cost utility. A similar study for semantic web service ranking was proposed in [33]. Wang, Cao and Xiang [34] proposed a dynamic cloud service selection using an adaptive learning technique in multi-cloud computing. In their proposed approach, each cloud service broker manages some clustered cloud services. The dynamic service selection (DCS) strategy, which consists of a set of dynamic service selection algorithms, uses an adaptive learning mechanism that comprises the incentive, forgetting and degenerate functions. The mechanism is formulated to dynamically optimize the cloud service selection and to return the best service result.

Some of the non-QoS-based approaches also used AHP to rank the attributes and metrics for decision-making. Nie, She and Chen [9] presented an evaluation system of cloud service selection using AHP that calculates the weights of attributes for service evaluation. They also presented a number of qualitative models for decision making in cloud service selection. Filepp, Schwartz, Ward, Kearney, Cheng, Young and Ghosheh [35] proposed virtual machine (VM) image selection service for cloud computing. Their algorithm orders the image based on conformance with specified user requirements and policies by best-fit and least-cost optimization.

Table 2. Comparison of Cloud service selection approaches

Authors	Area	Ranking Method	QoS based
Nie et al. [9]	Cloud service selection	Evaluation index system using AHP	No
Godse and Mulik [11]	Cloud service selection	SaaS selection using AHP	No
Garg et al. [7]	Cloud comparison and ranking	QoS attributes and AHP based ranking	Yes
Z. Rehman et al. [12]	Cloud service selection	Parallel MCDM approach based on QoS history	Yes
Han et al. [18]	Cloud service composition	Selecting best service by matching user requirements and QoS values from multiple services	Yes
Li et al. [31]	Cloud service composition	Highlights the problems and identifies the attributes for cloud comparison	No
Kang and Sim [32]	Cloud search engine	An ontology-based database and uses concept similarity, price and cost utility for ordering	No
Filepp et al. [35]	VM image selection	Image configuration repository and minimum-cost maximum-gain approach	No
Zeng et al. [13]	Cloud service selection	Uses maximum-gain and minimum-cost algorithm	Yes
Chen et al. [36]	Cloud service selection	Conflict detection and constraint programming	-
Sun et al. [14]	Cloud service selection	Fuzzy ontology and MCDM	Yes
Wang et al. [34]	Cloud service selection	Adaptive learning mechanism	No
Ghosh et al. [37]	Cloud service selection	Risk assessment based	
Lin et al. [38]	Cloud service selection	Risk assessment and cloud focus theory	Yes
Gui et al. [10]	Cloud brokering and recommender	Cloud classification model for filtering and categorization for cloud service recommendation	-
Mouratidis et al. [39]	Cloud service selection	Security and privacy requirements-based assessment method	-
Qi Yu [19]	Cloud service recommender	Collaborative filtering-based cloud recommender	Yes
Liu et al. [40]	Cloud service selection	Multi-attribute group decision-making (MAGDM) based	-

There are some other research studies which do not fall in the above two groups. An automatic conflict detection between the user's preferences and enterprise policies was proposed by Chen, Yan, Zhao, Lee and Singhal [36]. The proposed framework checks various conflicts that result from the violation of enterprise policies and inconsistency in cloud service user's requirements. The investigation is followed by the selection of an

appropriate service using the constraint programming that satisfies the user's requirements and also complies with enterprise policies. The proposed system aims to resolve the difficulties of cloud service selection with an emphasis on the involvement of enterprise policies. Gui, Yang, Xia, Huang, Liu, Li, Yu, Sun, Zhou and Jin [10] presented a service brokering and recommendation mechanism for selecting the best public cloud service at the IaaS and PaaS level. The proposed framework consists of a hierarchical information model for integrating heterogeneous cloud information from different providers and a corresponding cloud information collecting mechanism. A cloud service classification model for categorizing and filtering cloud services and an application requirement schema were presented. Liu, Chan and Ran [40] structured a multi-attribute group decision-making (MAGDM) based scientific decision tool to help businesses to determine which cloud computing vendor would be more suitable for their needs. The authors presented a subjective/objective integrated MAGDM technique for decision making in cloud computing services that uses objective attributes such as cost as well as subjective attributes such as TOE factors (Technology, Organization and Environment).

Some studies also focused on cloud service selection by considering factors such as security, privacy and risk assessment for cloud users. Mouratidis, Islam, Kalloniatis and Gritzalis [39] designed a framework to support the selection of cloud providers based on security and privacy requirements. That framework incorporates a modeling language and provides a structured process that supports the elicitation of security and privacy requirements and the selection of a cloud provider based on the satisfiability of the service provider. Ghosh, Member and Ghosh [37] presented a framework to facilitate cloud service selection that calculates the risk estimation based on trustworthiness and competence. Another work in risk assessment based cloud service selection was carried out by Lin, Zeng, Yang, Wang, Lin and Lin [38]. The proposed method is based on cloud theory and generates five property clouds by collecting the risk value and four risk indicators from each virtual machine. The cloud backward generator integrated these five clouds into one based on a weight matrix. Therefore, the risk prediction value is transferred to the risk level quantification, which is used for cloud service selection.

In addition, there are studies that have developed decision-making models based on uncertainty in experts' preferences. Liu, Dong, Chiclana, Cabrerizo, and Herrera-Viedma [41] develop a linear programming approach structured in two stages to minimize the information deviation of the relations between decision makers' preferences based on their confidence levels. Moral and Le [42] studied a group decision making problem. It investigated the problem using a fuzzy approach to obtain experts' preferences focusing on the convergence speed of the consensus. They show that setting a number of rules can control such a speed in the decision making process. Wu, Chiclana, Fujita and Herrera-Viedma [43] proposed a visual interactive framework to facilitate reaching a consensus based on different preferences by various experts. A trust based recommendation mechanism was then submitted to deal with inconsistencies in the expressed preferences. The mechanism finds out whether an unknown expert can be trusted and, hence, the associated preferences should be taken into account. Capuano, Chiclana, Fujita, Herrera-Viedma and Loia [44] proposed a model to consider the real preference of an expert whom is influenced by the opinion of other experts. They assume that the expert is unable to express preferences on some alternatives and employ a user friendly fuzzy ranking model to obtain the preferences. Zhanga, Dong and Herrera-Viedma [45] deal with significant conflicts in experts' preferences that can cause serious issues in the decision making process. They employed a selection process to divide decision makers into different clusters. Individual preference vectors are obtained, and a feedback adjustment process is utilized to help decision makers adjust their preferences. We notice in a review of papers on decision-making models, the insufficient studies undertaken taking into account a frequent and continuous change of preferences. This study performs that through a combined application of a Markovian model and BWM.

There are other versions of Markov chains, the application of each requires its own assumptions. For instance, applying a hidden Markov chain requires the assumption that there exist states of which we are not aware. Considering such assumptions are not within the scope of this research, this paper is the first work introducing Markov chains to cloud service selection. Therefore, we decided to use the general version of Markov chain [46][47] which is applicable to trace changing priorities of users/customers and has recently been examined [16].

To summarize, shown in Table 2 are a variety of approaches proposed in the literature several of which are based on MCDM techniques that assist a user make a service selection decision in the cloud environment. However, the issue of changing UPs has not been addressed in cloud service selection. In this paper, our aim is to assist cloud users to make an informed decision under changing user preferences.

3 Proposed Methodology

The proposed methodology uses a Markov chain in conjunction with BWM in order to find the best service. The Markov chain generates a pattern of the changing priorities of user preferences. This pattern is then used as the input for BWM to find out the priorities of QoS criteria. The QoS priorities are then used to rank the services. The method consists of the steps depicted in Figure 2.

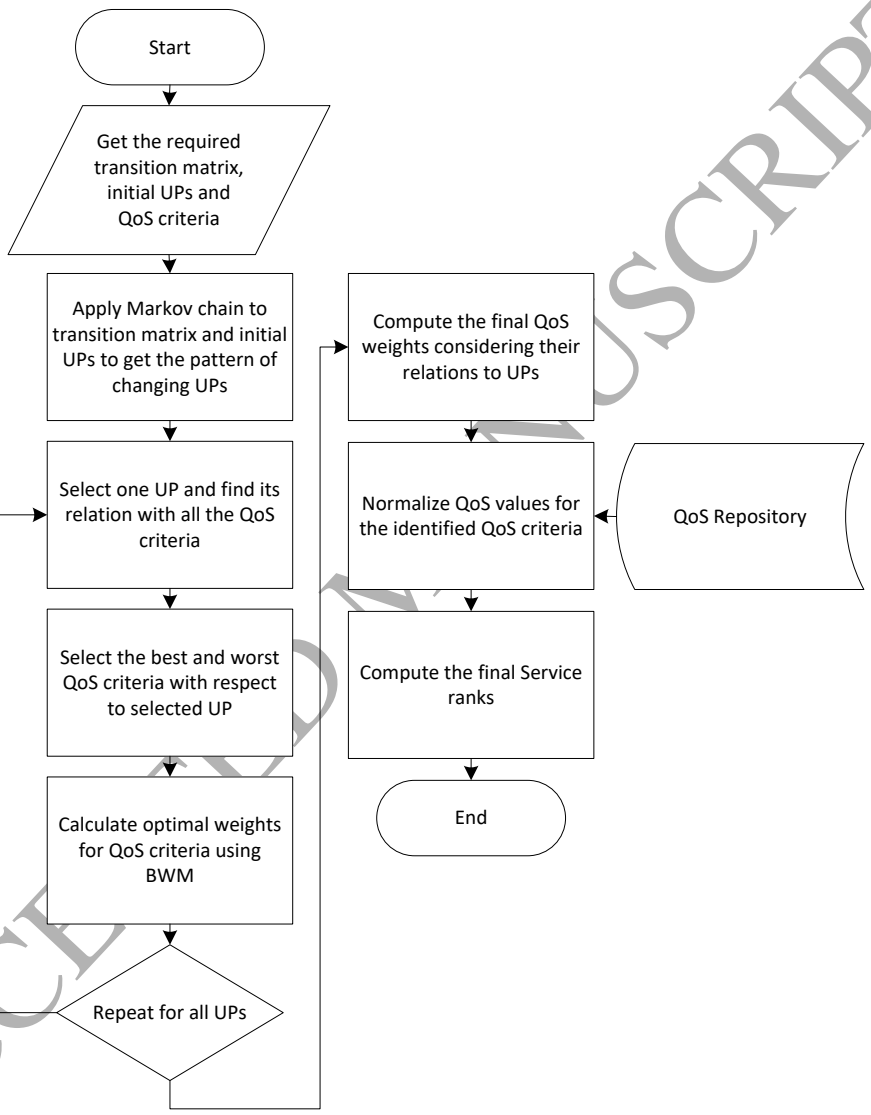

Figure 2. Flowchart showing the sequence of steps in the proposed approach

First, we need to obtain the initial UPs through service queries on the cloud broker [48]. These preferences, however, cannot be considered stable as they may change as the UPs do [49]. The changes can be traced using a Markov chain. Markov chains have previously been applied in other studies to predict a pattern of customer needs [16]. The method is explained based on its previous applications, as follows.

The proposed MCDM method utilizes the Markov decision process. Markov chains are useful in capturing discrete events over a period of time. In this case, a Markov chain finds a pattern of changing UPs. The next step

would be to utilize the captured information for cloud service selection. In cloud service selection, alternatives are ranked against criteria and, therefore, we need to use a MCDM method. A recently proposed MCDM method, BWM, is selected as it requires less pairwise comparisons and leads to greater consistency when compared with similar methods such as AHP [17][50]. We apply a Markov chain, as a discrete time, stochastic process, in combination with BWM to firstly find a pattern of UPs and then, considering the obtained pattern, rank the alternatives. Given this, the Markov chain does not directly make the decision, instead, the power of the Markov chain is leveraged to help BWM to connect the importance of these elements to cloud service specifications and rank the alternatives.

Here, a Markov chain addresses the problem of UPs being discrete events and finds a pattern in them. Although the Markov chain finds the pattern of UPs to be utilized instead of the initial preferences, this does not mean that users may not change priorities. Every user may keep changing their priorities. However, if these changes are traced in terms of the whole system including many other users for similar services, a pattern for the required specifications can be observed. In such a pattern, there will be a certain number of users with each of the preferences. For example, some users may change to low priority for a certain preference while other users may change to high priority for the same preference. Those numbers are computable using the Markov chain as follows.

Let us assume that the initial priority list of UPs is obtained through interviews and normalized in matrix W_{UP} as shown in Matrix (1).

$$W_{UP} = \begin{matrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{matrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \qquad (1)$$

Consider a time set, T = $\{t_1 \dots t_m\}$. There is always a likelihood of changing the preference from one UP to another after a period of time. For example, at time t_1, a_i is greater than a_j, which means more users prefer and select s_i (i^{th}UP) compared with s_j(j^{th}UP) as the most important requirement. At time t_2, UPs may change and s_i may not remain more than s_j. The time interval depends on how regularly users utilize the service and this varies for different service categories. Observing user behavior over time, we can see the proportion of users who have UP_1 as their most important need and so wish to stay with UP_1 or shift to other UPs (e.g. UP_2, UP_3...). Figure 3 is an example showing the transitions where there are four *UPs* (S_1 to S_4).

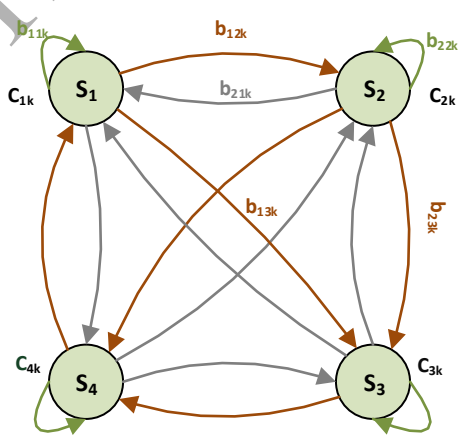

Figure 3. Transitioning between states at kth period

In the above figure, C_{ik} stands for the number of users that are with S_i at period k (S_{ik}), and b_{ijk} represents the number of users that transition from S_i to S_j (changing from one UP to another as their priority). In order to compute the transition matrix, we need to calculate the probability of transitioning from S_i to S_j at k^{th} period for every UP. The probabilities are computed as follows. If there are C_{1k} users who prefer S_1 at time k, and b_{12k} are

those who prefer to transition to S_2, then the probability of this transition is computed as: $\gamma_{12k} = \frac{b_{12k}}{c_{1k}}$. More generally, the probabilities are computed as follows.

$$\gamma_{ijk} = \frac{b_{ijk}}{c_{ik}} \quad (2)$$

If, after a sequence of periods, a p_{ij} can be estimated for which the following condition exists, then the value of p_{ij} in the transition matrix is equal to γ_{ijk} (note that ϵ stands for a small value).

$$\left|\gamma_{ijk} - p_{ij}\right| \leq \epsilon \quad (3)$$

In many cases, finding a p_{ij} that stays within the narrow interval of $(\gamma_{ijk} - \epsilon, \gamma_{ijk} + \epsilon)$ for a large number of successive periods may not be applicable. In such cases, different amounts for p_{ij} can instead be found and used for a reasonable number of periods. To cope with such a situation, a number should be set by decision makers. When the number of successive periods that p_{ij} stays in the interval and goes beyond the number, the transition matrix is computed and will be in use until a new trend appears. Techniques and charts of statistical quality control (SQC) can be utilized for the purpose of recognition of the trends, detection of the points of shifting p_{ij}, and monitoring probabilities in the transition matrix.

In summary, when the sequence γ_{ijk} is relatively close to p_{ij}, the value of p_{ij} is estimated by γ_{ijk}. Since the transition matrix includes n^2 values presented by p_{ij}, we do not expect a considerable effect for minor changes. In the case study presented in the paper, we use a threshold of 0.1. If the absolute variation of the probabilities in the transition matrix divided by its order goes beyond 10 percent, recalculation of the transition matrix is required.

The transition matrix is computed as follows.

$$P = \begin{array}{c} \\ s_1 \\ s_2 \\ \vdots \\ s_n \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & \dots & s_n \end{array} \\ \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1n} \\ p_{21} & \vdots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ p_{n1} & \dots & \dots & p_{nn} \end{bmatrix} \end{array} \quad (4)$$

By frequent multiplication of the transpose of matrix W_{UP} by the transition matrix, P, a set of $W_{UP}^{(i)}$ is obtained, where i is the number of the multiplications.

$$W_{UP}^{(0)^T} = W_{UP}^T I,\ W_{UP}^{(1)^T} = W_{UP}^T P^1, \dots,\ W_{UP}^{(i)^T} = W_{UP}^T P^i \quad (5)$$

A generalized form of the formula is presented in (6):

$$W_{UP}^{(k)^T} = W_{UP}^T P^k\ ,\ \forall\ k = 0,1, \dots, \infty \quad (6)$$

$$W_{UP}^* = W_{UP}^{(k)^T} \text{ when } k \longrightarrow \infty \quad (7)$$

Considering the inherent convergence of the stochastic matrices, we expect that the matrices become the same after three to five times of multiplication. Since the adjusted priorities of UPs are independent of the initial state [51], this method stands independent of the initial priorities of UPs. Thus, rather than the identification of the users' initial (instant) preferences, there should be a focus on forming a transitional matrix (explained above).

The limiting matrix of P can be found by raising P to a large power. In such a matrix, the arrays in each column are consistent. Multiplying any normalized matrix by this limiting matrix leads to the same matrix regardless of

the initial matrix (provided that initial matrix is normalized). The columns of limiting matrix have the same values. So, the values of a row of the matrix can be used as the pattern of UPs which is shown by W_{UP}^{*}.

In practice, each time that the probabilities are computed through interviews and feedback, the transition matrix can be obtained and replaced. If the updated matrix does not change the sequence of alternatives, there is no concern. If a change does occur, then the decision maker has to decide when to shift from the current cloud provider to a new cloud provider, while taking into account other important factors such as obtained trust and effort needed for new negotiations. After finding the pattern of UPs, the relationship between UPs and the QoS criteria need to be computed through BWM.

The relationship between UPs and QoS criteria of available services can be identified through assigning appropriate weights. Each QoS criterion is compared with respect to every UP, resulting in a matrix (as shown in (9)) in which rows represent the UPs and columns represents the QoS criteria. The matrix, namely W_{UP-QoS} is explained in this section.

The relationship of UPs and QoS criteria can be found by asking simple questions such as: 'What is the relative importance of the i^{th} QoS attribute when compared to the j^{th} QoS attribute with respect to the k^{th} UP?' When there are four QoS criteria such as: *CPU*, *Memory*, *Input/Output (I/O)*, and *Cost*, while *Performance* is an UP, then the following question could be asked: "What is the importance of *CPU* when compared to *Cost* considering *Performance*?" With respect to each QoS criteria, they are compared with each UP in separate tables. The calculated importance weights are used to calculate matrix W_{UP-QoS}. We use BWM [17] in the proposed methodology in order to find the relationship between UPs and QoS criteria. The process to compute the weights of the QoS criteria is described as follows:

1. Select an UP from {UP_1, UP_2, …, UPm} to find its relation with all the QoS criteria {Q_1, Q_2,…, Q_n}.

2. Determine the best (e.g. most important) and the worst (e.g. least important) criteria with respect to the selected UP. For example, *CPU* may be the best and *Cost* may be the worst criteria when considering the selected UP: *Performance*.

3. Determine the preference of the best criterion over all other criteria using a number between 1 and 9 where 1 is the best preference and 9 is the least preference. The resulting Best-to-Others vector would be $A_B = (a_{B1}, a_{B2}\ldots, a_{Bn})$ where a_{Bj} indicates the preference of the *best criterion B* over criterion j such that $a_{BB} = 1$. For the above example, the vector A_B shows the preference of *CPU* over all the other criteria considering UP *Performance*.

4. Determine the preference of all the criteria over the worst criterion using a number between 1 and 9. The resulting Others-to-Worst vector would be $A_W = (a_{1W}, a_{2W}\ldots, anW)^T$ where a_{jW} indicates the preference of the criterion j over the *worst criterion W* such that a_{WW} =1. For this example, the vector shows the preference of all the criteria over *Cost* considering *Performance*.

5. Finally, the optimal weights (w_1, w_2, …, w_n) are found. The optimal weight for the criteria is the one where for each pair of w_B/w_j and w_j/w_W, we have $w_B/w_j = a_{Bj}$ and $w_j/w_W = a_{jW}$. To satisfy these conditions for all *j*, we should find a solution where the maximum absolute differences $\left|\frac{w_B}{w_j} - a_{Bj}\right|$ and $\left|\frac{w_j}{w_w} - a_{jw}\right|$ for all *j* is minimized. Considering the non-negativity and sum condition for the weights, the following problem is formulated.

$$\left.\begin{array}{c} \left|\frac{w_B}{w_j} - a_{Bj}\right| \leq \xi, for\ all\ j \\ \left|\frac{w_j}{w_w} - a_{jw}\right| \leq \xi, for\ all\ j \\ \sum_j w_j = 1 \\ \mathrm{W}_j \geq 0, \text{for all } j \end{array}\right\} \quad (8)$$

Here ξ represents a measure of the consistency of comparison. The optimal weights (w_1, w_2, …, w_n) are obtained by solving the problem in (8) for the QoS criteria {q_1, q_2…, q_n} with respect to the selected UP.

An algorithm for computing optimal weights is described in the next section. A more elaborate example of the calculation of optimal weights for three criteria by solving the above equations has previously been given by Rezaei [17]. In the above, QoS criteria {Q_1…Q_n} are compared with respect to each UP. Each time, a set of weights for {Q_1…Q_n} is computed, those weights build the rows of the W_{UP-QoS} (9) presented below.

$$W_{UP-QoS} = \begin{array}{c} \\ UP_1 \\ UP_2 \\ \vdots \\ UP_m \end{array} \begin{array}{c} \begin{array}{cccc} Q_1 & Q_2 & \dots. & Q_n \end{array} \\ \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & \vdots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ w_{m1} & \dots & \dots & w_{mn} \end{bmatrix} \end{array} \quad (9)$$

Now, a matrix of QoS priorities is found that reflects the final weights of the QoS criteria considering their relationship with UPs. This matrix is determined by the product of the matrix containing the pattern of the priorities of the UPs and relation of UPs with the QoS criteria.

$$\mathrm{W_{FQoS}} = W^*_{UP} \times W_{\mathrm{UP-QoS}} \quad (10)$$

After that, the QoS criteria of the available services (options) are normalized. QoS values of all the services form an evaluation matrix D, which has the following form, where *Q* represents the QoS criterion and *OP* represents the service (option).

$$D = \begin{array}{c} \\ OP_1 \\ OP_2 \\ \vdots \\ OP_m \end{array} \begin{array}{c} \begin{array}{cccc} Q_1 & Q_2 & \dots. & Q_n \end{array} \\ \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & \vdots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ r_{m1} & \dots & \dots & r_{mn} \end{bmatrix} \end{array}$$

(11): The evaluation matrix

Since each criterion has its own units and range of values, the matrix D is normalized using (12) to make the QoS values of different criterion comparable.

$$u_{ij} = \frac{rij}{\sum_{i=1}^{m}(rij)} \quad (12)$$

The normalized matrix W_{OP-QoS} is given by:

$$W_{OP-QoS} = \begin{array}{c} \\ OP_1 \\ OP_2 \\ \vdots \\ OP_m \end{array} \begin{array}{c} \begin{array}{cccc} Q_1 & Q_2 & \ldots. & Q_n \end{array} \\ \begin{bmatrix} u_{11} & u_{12} & \ldots & u_{1n} \\ u_{21} & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & \ldots & \ldots & u_{mn} \end{bmatrix} \end{array} \quad (13)$$

The final ranking of the services is calculated using the product of the transposed matrix W_{FQoS} and the normalized evaluation matrix $W_{OP\text{-}QOS}$.

$$W_{FOP} = W_{OP\text{-}QoS} \times (W_{FQoS})^T \quad (14)$$

The matrix W_{FOP} contains the corresponding ranking for all the services with the highest value as the most suitable one. In the next section, the applicability of the proposed method is applied to select the best cloud service from a set of services.

4 Implementation

The proposed method of cloud service selection requires the transition matrix of changing UPs and the QoS criteria of available services as input and then, after performing the relevant computations, returns the final ranking of the services. As shown in Algorithm 1, the input and output parameters are defined first. The algorithm then computes the limiting matrix of *P* (denoted as $W_{UP}^{(k)T}$) by repeatedly multiplying the transition matrix with itself for *n* number of times (line 1-5). The value of *n* can be set by the user to adjust the number of iterations. Generally, after 4 or 5 iterations, the values in the limiting matrix $W_{UP}^{(k)T}$ settle down and do not change much after that. So, the values of the row of the matrix can be used as the pattern of UPs which is shown by W_{UP}^{*} (line 6).

After this, the relations between UPs and QoS criteria (W_{UP-QoS}) are found (line 7-9). Taking one UP at a time and calling Algorithm 2 (BWM) to find the priorities of the UP with respect to all the QoS criteria does this. A matrix of final QoS priorities W_{FQoS} is then computed by the product of the matrix containing the pattern of the UPs' priorities W_{UP}^{*} and relationship of UPs with the QoS criteria W_{UP-QoS} (line 10). Next, the QoS criteria of the available options (matrix D) are normalized to make the QoS values of different criteria comparable as every criterion has its own units and range of values (line 11-17). The normalized matrix is denoted as W_{OP-QoS}. Finally, the ranking of the services (W_{FOP}) is calculated using the product of a transposed matrix W_{FQoS} and the normalized evaluation matrix W_{OP-QoS} (line 18).

Algorithm 1. Main algorithm for cloud service selection with changing user preferences

Input: D = Averaged QoS values
P = Transition matrix
$W^{(0)}_{UP}$ = Initial user preferences
UP = set of user preferences
Q = set of QoS attributes

Output:
W_{FOP} = Final list of available service (options) rankings

1. initialize *n*;
2. initialize *k=0;*
3. while (*k*++ <= *n*)
4. $W^{(k)T}_{UP} = W^{(k-1)T}_{UP} \times P;$
5. end while

6. $W^{*}_{UP} = W^{(k)T}_{UP}$;
7. **foreach** *up* ϵ UP do
8. $W_{UP\text{-}QoS}$ = BWM(*up*,Q);
9. end for
10. $W_{FQoS} = W^{*}_{UP} \times W_{UP\text{-}QoS}$;
11. **for** $i < n$ cols in D
12. **for** $j < m$ rows in D
13. select r_{ij} from D;
14. $u_{ij} = r_{ij} / \sum_{i=1}^{m} (r_{ij})$
15. set u_{ij} in W_{OP-QoS};
16. end for
17. end for
18. $W_{FOP} = W_{OP\text{-}QoS} \times W^{T}_{FQoS}$;
19. **return** W_{FOP};

In Algorithm 1, for every UP and a set of weights for the QoS criteria $\{Q_1 \ldots Q_n\}$ is computed. This is used to build the rows of the W_{UP-QoS} matrix. First of all, the consistency index table (Table 3) is loaded, which shows the maximum value of ξ (maxξ) for any comparison a_{Bj} and a_{jW} (where a_{Bj} is the preference of the best criterion over the criterion *j* and a_{jW} is the preference of criterion *j* over the worst criterion). The process of finding the maxξ has previously been described by Rezaei [17]. This gives an indication of how consistent the comparison is. There are three possibilities:

1. A comparison is *fully consistent* when $a_{Bj} \times a_{jW} = a_{BW}$, for all *j*, where a_{BW} is the preference of the best criterion over the worst criterion. In this case, the value of ξ is 0 for all *j*.
2. A comparison is *partially consistent* when $a_{Bj} \times a_{jW}$ is lower or higher than a_{BW} for any *j*. In this case, the value of ξ for the comparison is anything between 0 and maxξ.
3. A comparison is considered *inconsistent* when a_{Bj} and a_{jW} have the maximum value (close to a_{BW}) for any *j* which will result in maxξ.

Table 3. Consistency Index (CI) [41]

a_{BW}	1	2	3	4	5	6	7	8	9
maxξ	0.00	0.44	1.00	1.63	2.30	3.00	3.73	4.47	5.23

Algorithm 2 allows decision maker to compute the weights when the comparisons are either *fully consistent* or *partially consistent*. If the comparisons are *inconsistent*, it allows decision maker to re-adjust the comparisons. For *partially consistent*, we propose using a threshold value between 0 and maxξ that allows decision maker to adjust the consistency according to the problem.

Algorithm 2. Best-Worst Method

BWM (*up*, Q)

1. CI = Consistency Index table;
2. choose best criterion a_{BB} where a_{BB} ϵ Q
3. choose worst criterion a_{WW} ϵ Q
4. j = 1;
5. set $a_{BW} = x$ where x ϵ [1,9]
6. **while (**j ≤ count(Q))
7. **do**
8. set $a_{Bj} = x$ where x ϵ [1,9];
9. set $a_{jW} = x$ where x ϵ [1,9];
10. while($a_{BW} = a_{Bj}$ *and* $a_{BW} = a_{jW}$);
11. A_B.add(a_{Bj});
12. A_W.add(a_{jW});

13. **end while**
14. **for** all *j* in Q
15. Form equation of the form $|w_B - w_j * a_{Bj}| = \xi$ where $a_{Bj} \epsilon$ A_B;
16. Form equation of the form $|w_j - w_W * a_{jW}| = \xi$ where $a_{jW} \epsilon$ A_W;
17. **end for**
18. Form equation $\sum_j w_j = 1$;
19. Compute w_j for *up* (row) in matrix $W_{UP\text{-}QoS}$ by solving system of linear equations (e.g. using Gaussian Elimination);
20. return $W_{UP\text{-}QoS}$;

After initialization, algorithm allows decision maker to set a_{BW} that defines the preference of the best criterion over worst. Ideally from set of {1…,9}, highest possible value for a_{BW} is 9 (line 5). All the comparisons with respect to best criterion and the worst criterion are perform one by one in a while loop (line 6-13). If any comparison is inconsistent, this loop asks the decision maker to re-adjust the comparison. Next, the comparisons are used to form equations of the form as given (8) where weights become the variables and the comparison becomes the co-efficient (line 14-18). For solving these equations, we propose using Gaussian Elimination (Gauss-Jordan Elimination) method to compute the optimal weights (line 19). The result of this operation is the optimal weights, which are set in matrix $W_{UP\text{-}QoS}$ for the corresponding row of *up* and returned to Algorithm 1.

5 Case Study

To validate our approach, a case study was undertaken on the dataset of Amazon EC2 IaaS cloud services. We used the QoS monitoring data of four EC2 IaaS services that were collected by PRTG monitoring service (https://prtg.paessler.com). The data consisted of the average of the hourly measurements of *response time* for 300-time periods of the four EC2 instances that includes *CPU*, *Memory* and *I/O* performance of the monitored services. In addition to these, the fourth criterion *price per hour* (denoted as *Cost*) is also included for each service as quoted by Amazon (www.amazon.com). In this case study, UPs are denoted *Performance*, *Availability*, *Reliability* and *Price*. Table 4 shows the brief description and the criteria of the four available cloud services.

Table 4. Available cloud services and average QoS criteria

Service	Detail	Instance Type	CPU (*ms*)	Memory (*ms*)	I/O (*ms*)	Cost (*$/hr*)
OP1	EC2 EU	Small	2056.19	1455.72	1035.82	0.0885
OP2	EC2 EU	Micro	80.77	81.94	260.42	0.0200
OP3	EC2 SA	Micro	860.15	126.66	722.40	0.0270
OP4	EC2 US East	Small	2200.70	532.28	4187.19	0.0650

As mentioned previously, UPs are likely to change over time. The transition matrix *P* is computed as given below. From this, W_{UP} is easy to compute.

$$P = \begin{array}{c} \\ Performance \\ Availability \\ Reliability \\ Price \end{array} \begin{array}{c} \begin{array}{cccc} Perf. & Avail. & Relia. & Price \end{array} \\ \begin{bmatrix} 0.29 & 0.20 & 0.38 & 0.13 \\ 0.26 & 0.25 & 0.30 & 0.19 \\ 0.29 & 0.26 & 0.29 & 0.16 \\ 0.26 & 0.22 & 0.23 & 0.29 \end{bmatrix} \end{array} \quad (15)$$

Although we do not need to use the initial UPs, as explained in the methodology, to show that W^*_{UP} is independent of the initial UPs, assume that the initial UPs' priorities are obtained as below:

$$W_{UP}^{*} = \begin{matrix} Performance \\ Availability \\ Reliability \\ Price \end{matrix} \begin{bmatrix} 0.38 \\ 0.16 \\ 0.26 \\ 0.20 \end{bmatrix} \quad (16)$$

Based on the above two matrices, the following matrices are calculated as $W_{UP}^{(k)^T} = W_{UP}^{(k)^T} * P$

$W_{UP}^{(0)^T} = [0.279 \quad 0.228 \quad 0.313 \quad 0.179]$ (17)

$W_{UP}^{(1)^T} = [0.278 \quad 0.234 \quad 0.306 \quad 0.181]$ (18)

$W_{UP}^{(2)^T} = [0.277 \quad 0.234 \quad 0.306 \quad 0.182]$ (19)

$W_{UP}^{(i)^T} = [0.277 \quad 0.234 \quad 0.306 \quad 0.182] \; i \geq 3$ (20)

Now if the limiting transition matrix ($P_{limiting}$) is computed, as presented below, it is a matrix with equal values in each column. Given that, any normalized matrix multiplied by it will be equal to W_{UP}^{*}, which is a representation of the column values of $P_{limiting}$.

$$P_{limiting} = \begin{bmatrix} 0.277 & 0.234 & 0.306 & 0.182 \\ 0.277 & 0.234 & 0.306 & 0.182 \\ 0.277 & 0.234 & 0.306 & 0.182 \\ 0.277 & 0.234 & 0.306 & 0.182 \end{bmatrix} \quad (21)$$

Any normalized matrix multiplied by this matrix would result in matrix $W_{UP}{}^{T}$, presented in Matrix (22). The changing UPs through the use of the transition matrix will settle into the following $W_{UP}{}^{T}$,, regardless of the initial UPs. This final matrix obtained from the application of a Markov chain is employed as the inputs to the second part of the method.

$$W_{UP}{}^{T} = \begin{matrix} Perf. & Avail. & Relia. & Price \end{matrix} \\ [0.277 \quad 0.234 \quad 0.306 \quad 0.182] \quad (22)$$

Now, with respect to each UP, QoS criteria are compared to each other. The calculated weights are used to calculate matrix W_{UP-QoS}. The comparison is performed using the BWM approach. According to this approach, for each UP, the best criterion is selected. Then, the weight is assigned to every criterion in comparison with the best criteria. A similar process is performed for the worst criterion. The optimal weight is computed by solving the (8) such that the ratios $\left|\frac{w_B}{w_j} - a_{Bj}\right|$ and $\left|\frac{w_j}{w_w} - a_{jw}\right|$ are minimized for all criteria and should not exceed $\max \xi$ given in Table 3 for a_{BW} where a_{BW} the preference of the best criterion over the worst criterion. For instance, the optimal weight for *Memory* in Table 5 satisfies these constraints. In this case, the ratios $|0.599/0.212 - 3| = 0.037$ and $|0.212/0.062 - 4| = 0.58$ are clearly less than the $\max \xi$ for a_{BW} which is $a_{BW} = 9$ (see Table 3). Moreover, algorithm 2 takes the best and worst preferences as input and returns the computed optimal weights as given in Table 5. It represents the QoS comparison with respect to *Performance* UP.

Table 5. QoS comparisons with respect to *Performance*

Performance	***CPU***	***Memory***	***I/O***	***Cost***
Best (*CPU*)	1	3	5	9
Worst (*Cost*)	9	4	2	1

Optimal Weights	0.599	0.212	0.127	0.062
CI (ξ) = 0.040	CR = 0.008			

Similarly, QoS criteria are compared with respect to *Availability*, *Reliability* and *Price* in Table 6, Table 7 and Table 8 respectively and the optimal weights are presented in Matrix W_{UP-QoS}.

Table 6. QoS comparisons with respect to *Availability*

Availability	***CPU***	***Memory***	***I/O***	***Cost***
Best (*Memory*)	3	1	5	9
Worst (*Cost*)	3	9	2	1
Optimal Weights	0.202	0.609	0.127	0.062
CI (ξ) = 0.041	CR = 0.008			

Table 7. QoS comparisons with respect to *Reliability*

Reliability	***CPU***	***Memory***	***I/O***	***Cost***
Best (*I/O*)	4	3	1	9
Worst (*Cost*)	2	3	8	1
Optimal Weights	0.142	0.209	0.577	0.072
CI (ξ) = 0.047	CR = 0.009			

Table 8. QoS comparisons with respect to *Price*

Price	***CPU***	***Memory***	***I/O***	***Cost***
Best (*Cost*)	4	8	5	1
Worst (*Memory*)	3	1	2	9
Optimal Weights	0.067	0.129	0.202	0.602
CI (ξ) = 0.062	CR = 0.011			

$$W_{UP-QoS} = \begin{bmatrix} 0.599 & 0.212 & 0.127 & 0.062 \\ 0.202 & 0.609 & 0.127 & 0.062 \\ 0.142 & 0.209 & 0.577 & 0.072 \\ 0.067 & 0.129 & 0.202 & 0.602 \end{bmatrix} \quad (23)$$

Now, W_{FQoS} is computed by multiplying the transpose of W_{UP} ((22)) by $W_{UP\text{-}QoS}$ ((23)) as follows.

$$W_{FQoS} = [0.277 \quad 0.234 \quad 0.306 \quad 0.182] \begin{bmatrix} 0.599 & 0.212 & 0.127 & 0.062 \\ 0.202 & 0.609 & 0.127 & 0.062 \\ 0.142 & 0.209 & 0.577 & 0.072 \\ 0.067 & 0.129 & 0.202 & 0.602 \end{bmatrix}$$

$$W_{FQoS} = [0.269 \quad 0.289 \quad 0.278 \quad 0.163] \qquad (24)$$

This matrix, W_{FQoS}, consists of the final QoS weights, which will be used to compute the final ranking of the services.

The QoS monitoring data of available cloud services (given in Table 4) is used to calculate matrix W_{OP-QoS}. This data consists of the QoS criteria and their values averaged over 300 time periods for four different cloud services. The evaluation matrix D corresponding QoS data of the available services and their criteria is given below:

$$D = \begin{matrix} \\ OP_1 \\ OP_2 \\ OP_3 \\ OP_4 \end{matrix} \begin{matrix} CPU \\ \begin{bmatrix} 1/2056.19 \\ 1/80.77 \\ 1/860.15 \\ 1/56.41 \end{bmatrix} \end{matrix} \begin{matrix} Memory \\ \begin{bmatrix} 1/1455.72 \\ 1/81.94 \\ 1/126.66 \\ 1/73.93 \end{bmatrix} \end{matrix} \begin{matrix} I/O \\ \begin{bmatrix} 1/1035.82 \\ 1/260.42 \\ 1/722.40 \\ 1/122.34 \end{bmatrix} \end{matrix} \begin{matrix} Cost \\ \begin{bmatrix} 0.0885 \\ 0.0200 \\ 0.0270 \\ 0.0250 \end{bmatrix} \end{matrix} \qquad (25)$$

$$D = \begin{matrix} \\ OP_1 \\ OP_2 \\ OP_3 \\ OP_4 \end{matrix} \begin{matrix} CPU \\ \begin{bmatrix} 0.000486 \\ 0.012380 \\ 0.001163 \\ 0.017727 \end{bmatrix} \end{matrix} \begin{matrix} Memory \\ \begin{bmatrix} 0.000687 \\ 0.012204 \\ 0.007895 \\ 0.013526 \end{bmatrix} \end{matrix} \begin{matrix} I/O \\ \begin{bmatrix} 0.000965 \\ 0.003839 \\ 0.001384 \\ 0.008174 \end{bmatrix} \end{matrix} \begin{matrix} Cost \\ \begin{bmatrix} 0.0885 \\ 0.0200 \\ 0.0270 \\ 0.0250 \end{bmatrix} \end{matrix} \qquad (26)$$

The evaluation matrix D is normalized to make the QoS values of different criteria comparable using the formula: $u_{ij} = \frac{rij}{\sum_{i=1}^{m}(rij)}$. The normalized evaluation matrix W_{OP-QoS} is given by:

$$W_{OP-QoS} = \begin{bmatrix} 0.015 & 0.020 & 0.067 & 0.551 \\ 0.390 & 0.356 & 0.267 & 0.125 \\ 0.037 & 0.230 & 0.097 & 0.168 \\ 0.558 & 0.394 & 0.569 & 0.156 \end{bmatrix} \qquad (27)$$

After computing the above matrix, W_{FOP} is computed by multiplying $W_{OP\text{-}QoS}$ ((27)) by the transpose of W_{FQoS} ((24)) as follows.

$$W_{FOP} = \begin{bmatrix} 0.015 & 0.020 & 0.067 & 0.551 \\ 0.390 & 0.356 & 0.267 & 0.125 \\ 0.037 & 0.230 & 0.097 & 0.168 \\ 0.558 & 0.394 & 0.569 & 0.156 \end{bmatrix} \begin{bmatrix} 0.269 \\ 0.289 \\ 0.278 \\ 0.163 \end{bmatrix}$$

$$W_{FOP} = \begin{matrix} OP_1 \\ OP_2 \\ OP_3 \\ OP_4 \end{matrix} \begin{bmatrix} 0.118 \\ 0.302 \\ 0.131 \\ 0.448 \end{bmatrix} \quad (28)$$

Thus, based on QoS criteria of the available services and the relevant computations, it turns out that, taking the feedback of previous users into account for this service category, the most appropriate service is OP_4 for the user as given in (28). The transition matrix in this case study transforms the user behaviour for most preferred criteria from *Performance* to *Reliability* overtime. That means the UPs change from *Performance* to *Reliability* overtime for services in this category.

6 Evaluation

In this section we assess the suitability of our proposed approach from two perspectives: pairwise comparisons of the method and convergence of the method. This is performed through evaluating its performance against existing approaches.

6.1 Pairwise comparison perspective

The proposed method in this study outperforms the AHP-based approaches, frequently used in cloud service selection. Here, a comparison of the results of the applications of both BWM and AHP is submitted.

The number of comparisons: To compare the proposed approach against the existing approaches, we eliminate the Markov chain. This is because the methods, previously applied, in the area of cloud service selection do not consider the possibility of changes in UPs. Therefore, the problem set to an ordinary MCDM problem and the consideration of changes in UPs handled through the Markov chain has been disregarded. Through this, the method becomes comparable with the existing approaches. In other words, most existing research in cloud service selection used AHP for pairwise comparison [7][9][11] while in our study, we employed BWM for pairwise comparison, which is more efficient than AHP [17]. The main reason for this efficiency is that BWM is a vector-based method in which only two vectors (for best and worst comparison) are required to make the entire matrix. Hence it requires fewer comparisons compared with matrix-based MCDM methods such as AHP. For BWM, only 2n-3 comparisons are needed while, for instance, AHP needed n(n-1)/2 comparisons (see Figure 4).

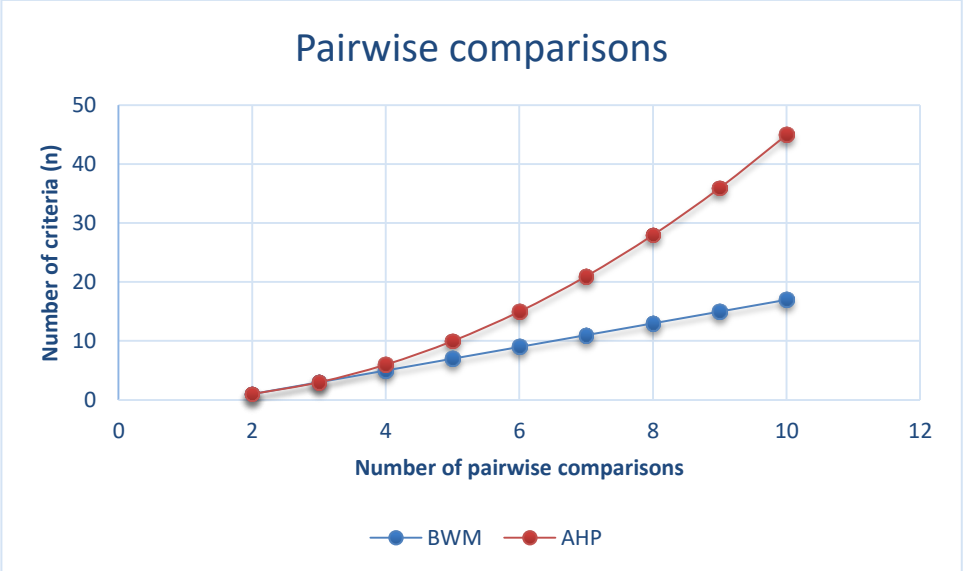

Figure 4. Pairwise comparisons in BWM and AHP

The inconsistency issue: The inconsistency of MCDM methods is commonly measured using a ratio, namely Consistency Ratio (CR). CR provides us with a measure of the reliability of the produced results. For the purpose of evaluation with the existing MCDM methods (i.e. AHP), we conducted an experiment to calculate and evaluate CR from both BWM and AHP. For this purpose, we performed 20 different comparisons to determine weights in BWM as well as AHP for similar cases including the ones presented in case study. We observed that 80% of the comparisons in BWM are consistent (i.e. CR< 0.1) compared with 60% in AHP as shown in Figure 5. Moreover, for the remaining 20% of the comparisons in BWM, the CR was in the range of 0.1 and 0.2. This shows that the final weights obtained from BWM are considerably more reliable than AHP. This is due to more consistent comparisons than AHP. In section 4, three different situations of comparison consistency in BWM algorithm are described, namely fully consistent, partially consistent, and inconsistent. The final weights derived using that algorithm are always consistent (including fully and partially consistent) and given that, we can claim that BWM is more reliable than AHP and so provides greater consistency.

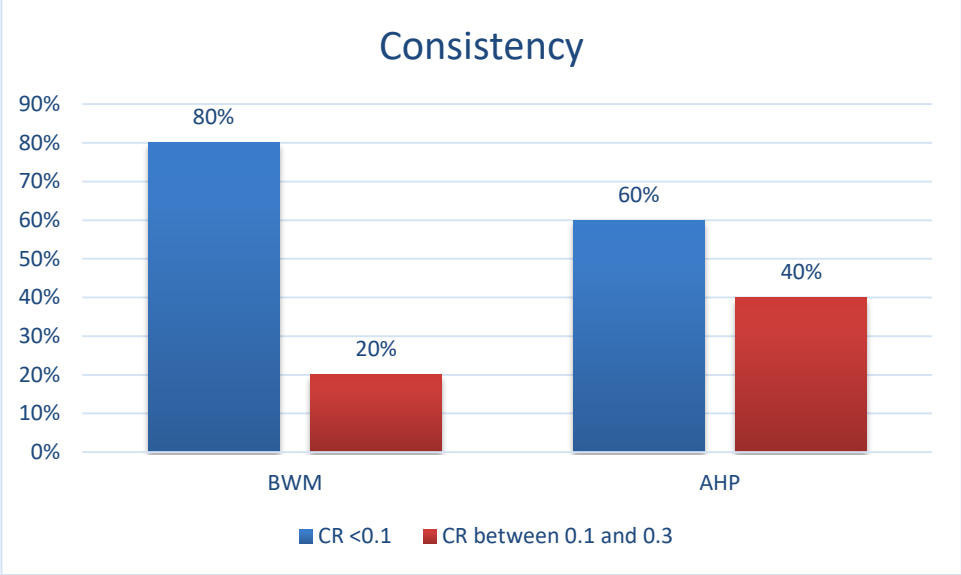

Figure 5. Consistency ratio in BWM and AHP

6.2 Convergence Perspective

Computing Convergence Speed of the Markov Chain: The convergence speed of the Markov chain contributes to the efficiency level of the approach, which happens by reducing the number of matrix multiplications. In this section probability state redistributions and the number of iterations from the initial state to the stabilizing state of the Markov chain are visualised. Figure 3 shows the Markov chain that we used for our case study and utilises the convergence properties of the model such as *periodicity* (which means that the chain should not get trapped in cycles), and *irreducibility* (which means there is positive probability of visiting all other states). Figure 6 shows the evolution of the state distributions over time from an initial distribution. It is clear from this figure that state probabilities for this Markov chain stabilize at step eight. These stabilized probabilities are used as the adjusted priorities of UPs. As explained previously, considering the inherent convergence of transition matrices no matter what initial probability distributions we use, the adjusted UPs obtained using the transition matrix are independent of the initial state [51].

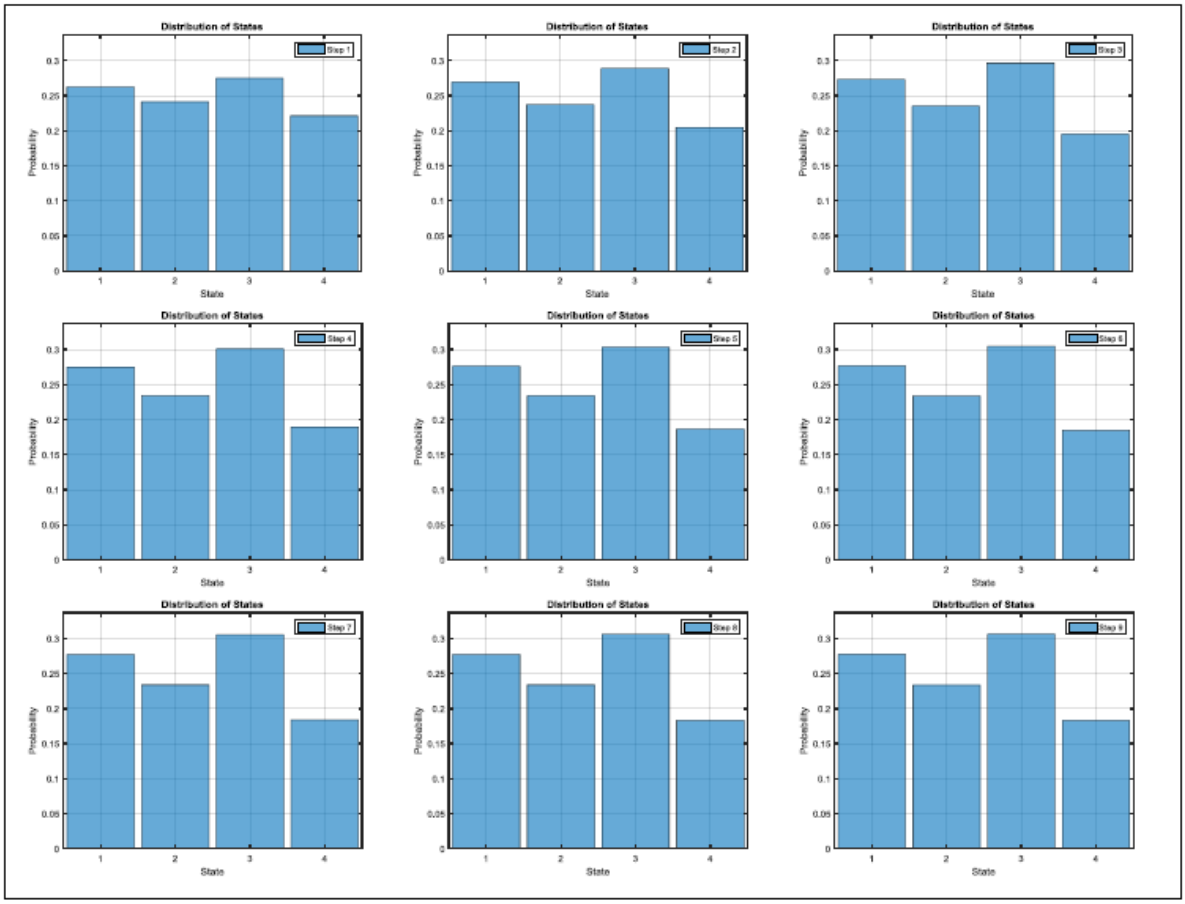

Figure 6. Markov Chain Convergence Steps

6.2 Discussion

There are many examples of applying Markov chains as statistical models to address real world problems. Markov processes are the basis for Markov Chain Monte Carlo (MCMC) methods [57], such as Metropolis-Hastings (MH) [58] and Gibbs sampler [59] algorithms. MCMC includes algorithms for sampling from probability distributions. Monte Carlo methods [60] are a broad class of computational algorithms that rely on repeated random sampling to obtain optimization, integration and generating draws for probability distributions. Their essential idea is the use of randomness to solve problems. Markov chain enhances these algorithms for generating random samples while exploring the state space in a smart way. MCMC algorithms are commonly used for simulating random objects with specific probability distributions. In other words, these algorithms use different strategies for generating samples (say $x^{(i)}$) while exploring the state space (say X) using a Markov chain mechanism. This mechanism is constructed so that the chain spends more time in the most important regions. The convergence property of Markov chains plays a fundamental role in MCMC simulation. For any starting probability distribution, the chain will converge to an invariant distribution, as long as the transition matrix obeys the following properties: *(i) irreducibility:* for any state of the Markov chain, there is a positive probability of visiting all other states, so that the transition graph is connected; *(ii) aperiodicity:* the chain should not get trapped in cycles. MCMC samplers are irreducible and aperiodic Markov chains that have the target distribution as the invariant distribution. In our proposed framework, we use the convergence property of Markov chain to obtain adjusted and stabilized UPs to find suitable cloud services instead of using the initial UPs, as the adjusted UPs are independent of the initial state.

Due to the significance of the cloud service selection problem, there have been numerous studies proposing solutions [52][53]. This paper has focused on linking the service selection process to the user preferences so that the best service is always selected based on the desires of the users. Given that, a user-oriented service selection

process has been developed. Applying such an approach motivates service providers to concentrate on satisfying the cloud users at different time intervals, which is mutually beneficial. One of the challenges in cloud service selection is that UPs are frequently changing, and such frequent changes prevent the establishment of a method to select a particular service.

In comparison with previously published papers in the area of cloud service selection, this paper proposes a selection approach in which a cloud user can find the best service under changing user preferences. The method utilizes a Markov chain. A Markovian transition matrix is built based on historical data of changes in user preferences. The transition matrix is capable of computing a pattern of UPs that is independent of instant UPs and, hence, builds a robust model. Such a model does not fluctuate based on minor changes in a user's desires. The efficiency of a Markov chain in finding the solution to dynamic and real world problems is well established [54]–[56]. Markov chains have previously been applied to the supplier selection problem [51], intrusion detection systems [54] and activity recognition in smart homes [56]. Although Markov chains have been well examined in a variety of research areas to address real word problems [46], the application of the method in combination with MCDM methods has only recently been proposed [51]. In addition, Markov chains have not previously been employed to address the cloud service selection problem and this study proposes its first application.

There is a previous study on the supplier selection process that proposes a Markov chain in combination with the Analytic Network Process (ANP) to trace changing customer needs for the supplier selection process [16]. In comparison with that study, this work combines a Markov chain with a recent MCDM method for cloud service selection. An approach to handle changing QoS data over a period of time to select a cloud service was previously presented [12] but it did not discuss the service selection under changing user preferences.

BWM has advantages when compared with methods such as ANP. ANP is a well examined MCDM method that is capable of the consideration of internal relations between the elements. ANP, however, requires too many pairwise comparisons that can become confusing and time consuming. Therefore, in this study for the first time, BWM is integrated with a Markov chain in order to address the service selection problem.

7 Conclusion

In summary, although different decision-making methods have been previously applied to help cloud users find a suitable cloud service, some uncertainties such as unstable UPs in the cloud environment encourage further studies. In this paper, we discussed the cloud service selection problem in an environment where the priorities of users keep changing. We proposed a framework that finds a pattern of changing UPs using a Markov chain independent of the initial user preferences. The pattern is then linked to QoS criteria of all available services to find weights using the BWM method. The weights of criteria of all services are then used to determine the overall rank of options for cloud service selection, along with the pattern of UPs. We validated the proposed methodology employing a case study using the performance data EC2 cloud service. The results show that utilizing the previous users' experience and feedback produces more suitable service recommendation and selection for future cloud use. The proposed approach is also more efficient than traditional MCDM approaches such as ANP and AHP due to a lower number of comparisons determining the weights of the service criteria. We recommend future studies examine the applicability of other methods in combination with BWM to address the concerns of uncertainty in the decision process in cloud service selection.

References

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *N. I. o. S. a. Technol. U.S. Dep. Commer.*, 2011.

[2] B. Varghese and R. Buyya, "Next Generation Cloud Computing: New Trends and Research Directions," *Futur. Gener. Comput. Syst.*, pp. 1–25, 2017.

[3] L. Wu and R. Buyya, "Service level agreement (SLA) in utility computing systems," *arXiv Prepr. arXiv1010.2881*, vol. abs/1010.2, p. 27, 2010.

[4] F. Nawaz, N. K. Janjua, O. K. Hussain, F. K. Hussain, E. Chang, and M. Saberi, "Event-driven

approach for predictive and proactive management of SLA violations in the Cloud of Things," *Futur. Gener. Comput. Syst.*, vol. 84, pp. 78–97, Jul. 2018.

[5] "Equinix." [Online]. Available: http://www.equinix.com/industries/cloud-providers/.

[6] D. Lin, A. C. Squicciarini, V. N. Dondapati, and S. Sundareswaran, "A Cloud Brokerage Architecture for Efficient Cloud Service Selection," *IEEE Trans. Serv. Comput.*, vol. 1374, no. c, pp. 1–1, 2016.

[7] S. K. Garg, S. Versteeg, and R. Buyya, "SMICloud: A framework for comparing and ranking cloud services," *Proc. - 2011 4th IEEE Int. Conf. Util. Cloud Comput. UCC 2011*, no. Vm, pp. 210–218, 2011.

[8] S. Ding, Z. Wang, D. Wu, and D. L. Olson, "Utilizing customer satisfaction in ranking prediction for personalized cloud service selection," *Decis. Support Syst.*, vol. 93, pp. 1–10, 2017.

[9] G. Nie, Q. She, and D. Chen, "Evaluation Index System of Cloud Service and the Purchase Decision-Making Process Based on AHP," in *Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011) November 19--20, 2011, Melbourne, Australia: Volume 3: Computer Networks and Electronic Engineering*, L. Jiang, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 345–352.

[10] Z. Gui, C. Yang, J. Xia, Q. Huang, K. Liu, Z. Li, M. Yu, M. Sun, N. Zhou, and B. Jin, "A Service Brokering and Recommendation Mechanism for Better Selecting Cloud Services," *PLoS ONE 9(8) e105297*, vol. 9, no. 8, 2014.

[11] M. Godse and S. Mulik, "An Approach for Selecting Software-as-a-Service (SaaS) Product," in *Proceedings of the 2009 IEEE International Conference on Cloud Computing*, 2009, pp. 155–158.

[12] Z. U. Rehman, O. K. Hussain, and F. K. Hussain, "Parallel cloud service selection and ranking based on QoS history," *Int. J. Parallel Program.*, vol. 42, no. 5, pp. 820–852, 2014.

[13] W. Zeng, Y. Zhao, and J. Zeng, "Cloud Service and Service Selection Algorithm Research," in *Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, 2009, pp. 1045–1048.

[14] L. Sun, J. Ma, Y. Zhang, H. Dong, and F. K. Hussain, "Cloud-FuSeR: Fuzzy ontology and MCDM based cloud service selection," *Futur. Gener. Comput. Syst.*, vol. 57, pp. 42–55, 2016.

[15] Y. Chen, L. Jiang, J. Zhang, and X. Dong, "A Robust Service Selection Method Based on Uncertain QoS," vol. 2016, no. 2, 2016.

[16] M. R. Asadabadi, "A customer based supplier selection process that combines quality function deployment, the analytic network process and a Markov chain," *Eur. J. Oper. Res.*, vol. 263, no. 3, pp. 1049–1062, 2017.

[17] J. Rezaei, "Best-worst multi-criteria decision-making method," *Omega*, vol. 53, pp. 49–57, 2015.

[18] S.-M. Han, M. Mehedi Hassan, C.-W. Yoon, H.-W. Lee, and E.-N. Huh, "Efficient Service Recommendation System for Cloud Computing Market," in *Grid and Distributed Computing: International Conference, GDC 2009, Held as Part of the Future Generation Information Technology Conferences, FGIT 2009, Jeju Island, Korea, December 10-12, 2009. Proceedings*, D. Śl\kezak, T. Kim, S. S. Yau, O. Gervasi, and B.-H. Kang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 117–124.

[19] Q. Yu, "CloudRec: a framework for personalized service Recommendation in the Cloud," *Knowl. Inf. Syst.*, vol. 43, no. 2, pp. 417–443, May 2015.

[20] L. Sun, H. Dong, F. K. Hussain, O. K. Hussain, and E. Chang, "Cloud service selection: State-of-the-art and future research directions," *J. Netw. Comput. Appl.*, vol. 45, pp. 134–150, 2014.

[21] M. Moghaddam and J. Davis, "Service selection in web service composition: A comparative review of existing approaches," *Web Serv. Found.*, 2014.

[22] L. F. de Oliveira Moura Santos, L. Osiro, and R. H. P. Lima, "A model based on 2-tuple fuzzy linguistic

representation and Analytic Hierarchy Process for supplier segmentation using qualitative and quantitative criteria," *Expert Syst. Appl.*, vol. 79, pp. 53–64, 2017.

[23] S. Jharkharia and R. Shankar, "Selection of logistics service provider: An analytic network process (ANP) approach," *Omega*, vol. 35, no. 3, pp. 274–289, 2007.

[24] P. Ishizaka, A. and Nemery, "Multi-attribute utility theory, in Multi-Criteria Decision Analysis," *Methods Software, John Wiley Sons Ltd, Chichester, UK.*, 2013.

[25] J. Wang, J. Wang, H. Zhang, and X. Chen, "Multi-criteria decision-making based on hesitant fuzzy linguistic term sets: An outranking approach," *Knowledge-Based Syst.*, vol. 86, pp. 224–236, 2015.

[26] S. E. Bellman, R. E., & Dreyfus, *Applied dynamic programming*. Princeton university press, 2015.

[27] M. Bartlett and J. Cussens, "Integer Linear Programming for the Bayesian network structure learning problem," *Artif. Intell.*, vol. 244, pp. 258–271, 2017.

[28] A. K. Das, S. Das, and A. Ghosh, "Ensemble feature selection using bi-objective genetic algorithm," *Knowledge-Based Syst.*, vol. 123, pp. 116–127, 2017.

[29] K.-C. Huang, M.-J. Tsai, S.-J. Lu, and C.-H. Hung, "SLA-constrained service selection for minimizing costs of providing composite cloud services under stochastic runtime performance," *Springerplus*, vol. 5, no. 1, p. 294, Mar. 2016.

[30] The Cloud Service Measurement Initiative Consortium (CSMIC), "Service Measurement Index Introducing the Service Measurement Index (SMI)," *Http://Www.Cloudcommons.Com/About-Smi; Accessed 2013-04-12*, no. September, pp. 1–8, 2011.

[31] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: Comparing Public Cloud Providers," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, 2010, pp. 1–14.

[32] J. Kang and K. M. Sim, "Cloudle: An Ontology-Enhanced Cloud Service Search Engine," in *Web Information Systems Engineering -- WISE 2010 Workshops: WISE 2010 International Symposium WISS, and International Workshops CISE, MBC, Hong Kong, China, December 12-14, 2010, Revised Selected Papers*, D. K. W. Chiu, L. Bellatreche, H. Sasaki, H. Leung, S.-C. Cheung, H. Hu, and J. Shao, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 416–427.

[33] F. Nawaz, K. Qadir, and H. F. Ahmad, "SEMREG-Pro: A semantic based registry for proactive web service discovery using publish-subscribe model," in *Proceedings of the 4th International Conference on Semantics, Knowledge, and Grid, SKG 2008*, 2008, pp. 301–308.

[34] X. Wang, J. Cao, and Y. Xiang, "Dynamic cloud service selection using an adaptive learning mechanism in multi-cloud computing," *J. Syst. Softw.*, vol. 100, pp. 195–210, 2015.

[35] R. Filepp, L. Shwartz, C. Ward, R. D. Kearney, K. Cheng, C. C. Young, and Y. Ghosheh, "Image selection as a service for cloud computing environments," in *2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, 2010, pp. 1–8.

[36] C. Chen, S. Yan, G. Zhao, B. S. Lee, and S. Singhal, "A Systematic Framework Enabling Automatic Conflict Detection and Explanation in Cloud Service Selection for Enterprises," in *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, 2012, pp. 883–890.

[37] N. Ghosh, S. Member, and S. K. Ghosh, "SelCSP : A Framework to Facilitate Selection of Cloud Service Providers," *IEEE Trans. CLOUD Comput.*, vol. 3, no. 1, pp. 66–79, 2015.

[38] F. Lin, W. Zeng, L. Yang, Y. Wang, S. Lin, and F. Lin, "Cloud computing system risk estimation and service selection approach based on cloud focus theory," *Neural Comput. Appl.*, vol. 28, no. 7, pp. 1863–1876, 2017.

[39] H. Mouratidis, S. Islam, C. Kalloniatis, and S. Gritzalis, "A framework to support selection of cloud providers based on security and privacy requirements," *J. Syst. Softw.*, vol. 86, no. 9, pp. 2276–2293, 2013.

[40] S. Liu, F. T. S. Chan, and W. Ran, "Decision making for the selection of cloud vendor: An improved approach under group decision-making with integrated weights and objective/subjective attributes," *Expert Syst. Appl.*, vol. 55, no. 2016, pp. 37–47, 2016.

[41] W. Liu, Y. Dong, F. Chiclana, F. J. Cabrerizo, and E. Herrera-Viedma, "Group decision-making based on heterogeneous preference relations with self-confidence," *Fuzzy Optim. Decis. Mak.*, vol. 16, no. 4, pp. 429–447, Dec. 2017.

[42] M. J. Moral and L. Le, "A Comparative Study on Consensus Measures in Group Decision Making," *Int. J. Intell. Syst. Press. 2017., Press*, no. 1, pp. 283–287, 2017.

[43] J. Wu, F. Chiclana, H. Fujita, and E. Herrera-Viedma, "A visual interaction consensus model for social network group decision making with trust propagation," *Knowledge-Based Syst.*, vol. 122, no. Supplement C, pp. 39–50, 2017.

[44] N. Capuano, F. Chiclana, H. Fujita, E. Herrera-Viedma, and V. Loia, "Fuzzy Group Decision Making with Incomplete Information Guided by Social Influence," *IEEE Trans. Fuzzy Syst.*, vol. 6706, no. c, 2017.

[45] H. Zhanga, Y. Dong, and E. Herrera-Viedma, "Consensus building for the heterogeneous large-scale GDM with the individual concerns and satisfactions," *IEEE Trans. Fuzzy Syst.*, vol. PP, no. 99, p. 1, 2017.

[46] Karlin, *A first course in stochastic processes*. Academic Press.

[47] S. P. Meyn and R. L. Tweedie, "Markov Chains and Stochastic Stability," *Springer-Verlag*, p. 792, 1993.

[48] P. Zheng, X. Xu, and S. Q. Xie, "A weighted interval rough number based method to determine relative importance ratings of customer requirements in QFD product planning," *J. Intell. Manuf.*, May 2016.

[49] J. O. Gutierrez-Garcia and K. M. Sim, "Agent-based cloud service composition," *Appl. Intell.*, vol. 38, no. 3, pp. 436–464, 2013.

[50] J. Rezaei, "Best-worst multi-criteria decision-making method: Some properties and a linear model," *Omega*, vol. 64, no. Supplement C, pp. 126–130, 2016.

[51] M. R. Asadabadi, "A Markovian-QFD approach in addressing the changing priorities of the customer needs," *Int. J. Qual. Reliab. Manag.*, vol. 33, no. 8, pp. 1062–1075, 2016.

[52] S. Deng, H. Wu, D. Hu, and J. Leon Zhao, "Service Selection for Composition with QoS Correlations," *IEEE Trans. Serv. Comput.*, vol. 9, no. 2, pp. 291–303, Apr. 2016.

[53] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, "Smart Cloud Storage Service Selection Based on Fuzzy Logic, Theory of Evidence and Game Theory," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2348–2362, Aug. 2016.

[54] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-Based Syst.*, vol. 78, pp. 13–21, 2015.

[55] R. Pourmoayed, L. R. Nielsen, and A. R. Kristensen, "A hierarchical Markov decision process modeling feeding and marketing decisions of growing pigs," *Eur. J. Oper. Res.*, vol. 250, no. 3, pp. 925–938, 2016.

[56] K. S. Gayathri, K. S. Easwarakumar, and S. Elias, "Probabilistic ontology based activity recognition in smart homes using Markov Logic Network," *Knowledge-Based Syst.*, vol. 121, pp. 173–184, 2017.

[57] U. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, D. Goldberg, J. Holland, F. Ionescu, V. Pupezescu, U. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, O. M. Rokach, C. Hong, C. Functions, E. D. Michie, D. J. Spiegelhalter, C. C. Taylor, M. Kubat, R. C. Holte, S. Matwin, D. Freitag, C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An Introduction to MCMC for Machine Learning," *Mach. Learn.*, vol. 50, no. 1, pp. 5–43, Jan. 2003.

[58] S. Chib and E. Greenberg, “Understanding the Metropolis-Hastings Algorithm,” *Am. Stat.*, vol. 49, no. 4, pp. 327–335, 1995.

[59] W. R. Gilks and P. Wild, “Adaptive Rejection Sampling for Gibbs Sampling,” *J. R. Stat. Soc. Ser. C (Applied Stat.*, vol. 41, no. 2, pp. 337–348, 1992.

[60] S. Raychaudhuri, “Introduction to Monte Carlo simulation,” *2008 Winter Simul. Conf.*, pp. 91–100, 2008.