A framework of cloud service selection with criteria interactions

Le Sun a,b,*, Hai Dong c, Omar Khadeer Hussain d, Farookh Khadeer Hussain e, Alex X. Liu b,a

a Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAET), Nanjing University of Information Science & Technology, Nanjing, 210044, China
b Department of Computer Science and Engineering, Michigan State University, East Lansing, USA
c School of Science, RMIT University, Australia
d School of Business, UNSW at Australian Defence Force Academy, Australia
e Centre for Artificial Intelligence, School of Software, University of Technology Sydney, Australia

HIGHLIGHTS

- A novel MCDM framework for cloud service selection.
- Service selection based on criteria priority orders and interaction types.
- An executable approach to build knowledge-based ground truth to validate MCDMs.
- Identify disadvantages and limitations of linear MCDMs and manual service ranking.
- Substantial experiments based on real datasets to validate the proposed method.

ABSTRACT

Existing cloud service selection techniques assume that service evaluation criteria are independent. In reality, there are different types of interactions between criteria. These interactions influence the performance of a service selection system in different ways. In addition, a lack of measurement indices to validate the performance of service selection methods has hindered the development of decision making techniques in the service selection area. This paper addresses these critical issues of modeling the interactions between cloud service selection criteria, and designing indices to validate service selection methods. In this paper, we propose a Cloud Service Selection with Criteria Interactions framework (CSSCI) that applies a fuzzy measure and Choquet integral to measure and aggregate non-linear relations between criteria. We employ a non-linear constraint optimization model to estimate the Shapley importance and criteria interaction indices. In addition, we design a priority-based CSSCI (PCSSCI) to solve service selection problems in the situation where there is a lack of historical information to determine criteria relations and weights. Furthermore, we discuss an approximate solution for CSSCI to reduce its computing complexity. Finally, we design three indices to validate the cloud service selection methods. The experimental results preliminarily prove the technical advantage of the proposed models in contrast to several existing models.

1. Introduction

With the development of cloud computing and the proliferation of cloud services on the Internet, cloud service selection has become an area undergoing intense study [1]. Existing service selection techniques assume selection criteria are independent [2,3]. This assumption does not take into account the fact that there are different types of non-linear relations between criteria. In reality, components in a complex system interact with each other in different forms. Those different types of interactions influence the performance of the whole system [4]. This study builds a non-linear modeling mechanism to solve cloud service selection where the selection criteria may have different types of interactions.

There are three types of relations between criteria: positively interacting (or supporting), negatively interacting (or conflicting) and independent [5]. In this paper, we mainly explore how criteria interrelationships (we call them interactions) influence service selection results. If two criteria positively interact with each other, they are similar to each other from a specific perspective (e.g. function), so they influence each other’s utilities positively. However, their interactions influence the overall utility of a service negatively. Here we use utilities to measure the criterion performance or the satisfaction degrees of service users. If two criteria conflict with
each other, the increase of a criterion utility causes the decrease of the utility of the other criterion. If two criteria do not have any relations, they are independent of each other.

We use a real scenario to explain the significance of modeling criteria interactions in service selection. Assume a user Jane wants to select a SaaS from two services with similar functions. He uses three criteria to evaluate services: cost (ct), availability (av) and successibility (su). His preferences are: (1) both av and su are 2 times more important than ct. However (2) if both services have high av, and not very low su, he will prefer the service which has lower ct. Based on the user’s preference (1), we assign weights for criteria: \(w_{ct} = 0.2\), \(w_{av} = 0.4\) and \(w_{su} = 0.4\). Assume the monthly payments and QoS performance of three services are: service1 \(s_1(\text{ct, av, su}) = (8, 99\%, 95\%)\) and service2 \(s_2(\text{ct, av, su}) = (6, 99\%, 80\%)\). We standardize them using this procedure for the type of benefit criteria (i.e. the higher value of a benefit criteria, the better its performance, e.g. av and su), the standardized value of av is: \(sd_{av}(av) = s_1(\text{av})/\max(s_1(\text{av}), s_2(\text{av}), s_3(\text{av}))\); and for the type of cost criteria (i.e. the higher value of a cost criteria, the worse its performance, e.g. ct), the standardized value is: \(sd_{ct}(ct) = 1 - s_1(\text{ct})/\max(s_1(\text{ct}), s_2(\text{ct}), s_3(\text{ct}))\). The standardized values of the three services are: \(sd_{ct}(ct, av, su) = (0.1, 1)\) and \(sd_{av} = (0.25, 1, 0.81)\). The simple weighted addition (SWA) [6] will recommend service1 based on their simple aggregated scores: \(u(s_1) = 0.2 * 0 + 0.4 * 0.4 + 1 * 0.4 * 1 = 0.8 > u(s_2) = 0.2 * 0.25 + 0.4 * 1 + 0.4 * 0.81 = 0.77\). However, the user’s preference (2) indicates that he might prefer service2. This inaccurate recommendation of SWA is because of its inability to aggregate the non-linear relations between preferences (1) and (2), which causes a redundant consideration of the high performance of av and su of service2, and in turn increases the evaluation score of service2. On the other hand, according to preference (2), av and su are both important to the user (he does not want very low su), so it is necessary to consider both of them in a service selection. Therefore, it is important to build a non-linear model to evaluate services based on the complexity of users’ preferences and criteria interactions.

In this paper, we propose a cloud service selection framework that models criteria interactions for service selection. We name this framework Cloud Service Selection with Criteria Interactions (CSSCI). We apply fuzzy measure (FM) [7] and 2-order additive Choquet integral [8] to estimate the overall utility of a service based on the importance of single criteria and the influence of two interactive criteria on the final decision. We use a non-linear constraint optimization model to estimate the importance of single criteria and the interaction indices between criteria. To solve the optimization model, we use a pair-wise comparison method to determine the criterion importance, and define an interaction ratio to assist in estimating the interaction indices. As there may be situations where insufficient historical information exists to support building criteria interactions, we design a priority-based CSSCI (PCSSCI), in which users only need to provide a priority order of criteria and subjectively define types of criteria interactions. PCSSCI uses an interactive interpretive structural modeling (I-ISM) to interactively establish consistent criteria interactions.

To validate the proposed methods, we design three algorithms to evaluate the efficiency of CSSCI, which are: average ratio of a service ranking similar to the service ranking of experts (ar), Kendall Tau distance ratio (ktdr) among rankings of a set of multi-criteria decision making (MCDM) methods, and stability of service ranking (st). The experiment results show that the ar of CSSCI is around 50%, 50% and 10% better than that of SWA [6], ViseKriterijuska Optimizacija I Komoromisno Resenje (VIKOR) [9] and the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [10] respectively; the ktdr of CSSCI is around 2% better than the ktdrs of SWA and TOPSIS, and 3.5% better than the ktdr of VIKOR; and the st of CSSCI performs better than SWA and TOPSIS by around 10%. In particular, SWA, TOPSIS and VIKOR are not capable of modeling non-linear relations between criteria based on users’ interactive preferences. As it is time-consuming for CSSCI to precisely solve an optimization model with multiple constraint conditions, we propose an approximate solution of CSSCI (apprCSSCI), which uses the initial weights of single criteria and tunes a parameter for determining the interaction indices. We compare the service ranking results of apprCSSCI with those of CSSCI. The comparison results show that the approximate solution has a very low Kendall Tau distance (ktd) to the precise solution when the parameter of apprCSSCI is set appropriately. The distinctive contributions of this paper are:

- We propose a novel MCDM framework for cloud service selection, which models users’ nonlinear preferences based on criteria interactions.
- We deal with service selection in situations where there is a lack of historical information, and introduce an objective service selection method that only requires users to provide criteria priority orders and interaction types.
- We design three algorithms to rigorously evaluate the proposed framework. In particular, we propose the first simple and executable approach to establish an expert-knowledge-based ground truth to validate MCDM methods, which is a significant contribution as a lack of expert-knowledge-based labels have hindered the usage of MCDMs in the service selection literature.
- We identify the disadvantages of classical linear MCDM methods for solving service selection problems, and identify the limitations of manual service ranking with respect to (w.r.t.) high criteria numbers.
- We conduct substantial experiments to validate the proposed methods based on the real datasets of SaaS and cloud computing services.

These contributions have not been addressed in the existing literature.

This paper is organized as follows: Section 2 introduces related work; Section 3 introduces the preliminary knowledge; Section 4 defines the problem of cloud service selection with criteria interactions, and overviews the CSSCI; Section 5 describes the CSSCI; Section 6 introduces the PCSSCI; Section 7 presents the experiment processes and results, and discusses the approximate CSSCI; and Section 8 concludes this paper.

2. Related work

In this section, we review the state-of-the-art of Web and cloud service selection approaches over the past five years. We categorize these works into two groups: MCDM-based and non-MCDM-based service selection. After introducing the literature, we analyze the differences between our work and the existing works.

2.1. MCDM-based service selection

Based on our literature review, most service selection problems are solved by MCDMs. MCDMs usually presume that a single decision maker or a group of decision makers selects the most satisfying object from a number of alternatives based on the performance or satisfaction degree of the alternatives in terms of multiple (two or more) criteria. MCDM methods are divided into two categories: multi-objective optimization (MOO) and multi-attributive utility theory (MAUT) [11]. MOOs mostly assume that the objective function values or attributes of the alternatives are known. Problems of MOOs are formulated as interactive multi-objective programming models [12–14]. MAUTs are different from
MOOs mainly based on the way MAUTs process risks and uncertainties [11]. MAUTs focus on the structure of MCDM problems and on the assessment methodologies of individual utilities. A decision-making problem is grouped into MAUTs if the criteria function values are explicit. Popular MAUTs include simple weighted aggregation (SWA) [1], Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [10], Analytic Hierarchy Process (AHP) [15], VIKOR [9] and Best Worst Method (BWM) [16].

2.2.1. Multi-objective optimization

Trummer et al. [17] studied approximate algorithms for solving multi-objective quality-driven service selection (QDSS) problems. The solutions to QDSS problems aim to find services for a workflow that has Pareto-optimal QoS values. Esposito et al. [18] proposed a multi-criteria optimization procedure based on fuzzy inference and Dempster-Shafer theories to optimize the requesters' satisfaction degree for QoS performance and cost. Wang et al. [19] developed a specialized artificial bee colony algorithm to solve QoS-aware Web service selection problems. Cho et al. [20] aimed at solving a problem where the possibility of successfully finding the optimal sets of services decreases when the number or hardness of QoS constraints increases.

2.2.2. Multi-attribute utility theory

MAUT techniques have been widely applied to service selection [21,22]. Viriyasitavat [23] proposed an MCDM approach that includes a Service Workflow Specification language (SWSpec) for service description and matching, and an AHP based approach for decision making. Choi and Jeong [24] developed a system of Web service selection by considering users’ preferences on QoS criteria, and on the basis of AHP techniques. Abdel et al. [25] developed a neutrosophic MCDM model for cloud service selection based on the neutrosophic AHP.

2.2.3. Service selection based on other techniques

We summarize the works on service selection based on techniques other than MCDMs from four perspectives: (1) description logic or language-based; (2) game theory-based; (3) information and probability theory-based; (4) graph theory-based; and (5) Collaborative filtering-based.

2.2.1. Description logic-based

Xiang et al. [26] designed a light weight IoT service description language for fast service searching and matching. Tamani et al. [27] used linguistic quantifiers and fuzzy quantified propositions to select vehicular cloud services based on both service constraints and users’ preferences.

2.2.2. Game theory-based

Do et al. [28] studied price competition mechanisms for cloud providers in cloud service marketplaces. They formulated a two-stage non-cooperative game for price competition models of cloud services. Zhu et al. [29] established a hierarchical dynamic game framework in which service users select services in these two tiers dynamically, based on service performance and cost. Wang et al. [30] used an incentive contract mechanism to connect service providers and requesters.

2.2.3. Information theory and probability theory

Hwang et al. [31] proposed the use of discrete random variables with probability mass functions to represent QoS values. Ahmed et al. [32] proposed a Web service ranking method based on a hidden Markov model (HMM) of response time prediction. Wang et al. [33] applied information and probability theories to filter low-reliability services and select the most appropriate services.

2.2.4. Graph theory-based

Somu et al. [34] proposed a hypergraph-based computational model (HGCM) and a minimum distance-helly property (MDHP) algorithm to assign weights to attributes. In addition, Somu et al. [35] used the rough set-based hypergraph technique to assess the trust of service providers in cloud service selection.

2.2.5. Collaborative filtering-based

The collaborative filtering (CF) is a significant and popular technology in the area of service selection [36–38]. In particular, Ding et al. [39] proposed a CSTrust algorithm based on the CF technology to evaluate the trustworthiness of a cloud service by simultaneously considering the predicted QoS performance and the customers’ satisfaction. Ding et al. [40] then designed a ranking-oriented prediction method based on the CF technology. More recently, Ding et al. [41] further improved the CF-based service selection method by taking into account the time feature of the user similarity.

2.3. Difference between our work and state-of-the-art work

Our work is different from the above methods because our work considers how different types of non-linear relations between criteria influence service selection results based on users’ preferences. Several researchers have considered criteria interrelationships in service composition. For example, Deng et al. [42] took into account QoS correlations in Web service composition. The authors stated that in a service composition process, the QoS performance of a service not only depends on the service itself, but also depends on the QoS of other services. This work is different from our work because it studies how QoS correlations between different atomic services influence the overall QoS of composed services. However, our work focuses on how criteria interactions influence service selection. The criteria in our work can be QoS parameters and other attributes required by service users, such as service costs and types of customer services. In addition, our considered interactions can be interrelationships of criterion performance or user-determined criteria interactions based on users’ preferences or knowledge. Furthermore, we consider the criteria interactions of a service for service selection, but not the criteria interactions between two different services. Xu et al. [43] explored problems of service selections or compositions in E-commerce by accounting for service contextual information, like priori, similarity and correlations of services. The service correlations considered in this work [43] are functional similarities among services, which are different to the criterion interactions discussed in our work.

3. Preliminaries

3.1. MCDM and MAUT methods

Assume an MCDM problem is to select an alternative (or solution) from m alternatives: $a_1, \ldots, a_m$, based on values (or performance, utilities) $i$ of n criteria (or attributes, goals): $c_1, \ldots, c_n$. This decision-making problem can be modeled as a matrix of decision $x_{ij}$, where $x_{ij}, \forall i \in [1,m], j \in [1,n]$ is the value of criterion $c_j$ of alternative $a_i$.

Simple weighted addition (SWA) [45] is the most simple and popular MAUT method. If each criterion has a weight, e.g. $w_j, j \in [1,n]$, the overall SWA value of alternative $a_i$ is $U_i = \sum_{j=1}^{n} w_j \cdot v_{ij}$. TOPSIS [46] ranks alternatives based on the distances from the values of the criterion of an alternative to the positive and negative ideal solutions. The positive and negative ideal solutions can be
the maximum and minimum values of the criteria of alternatives; and they can also be ideal solutions defined by decision makers based on the features of criteria. VIKOR [47] is a compromise ranking method. The ranking results of VIKOR are also based on the distances of an alternative to the ideal solutions. A decision-making problem in VIKOR is formulated by an Lp-metric (Formula (1)).

\[
L_{p,i} = \left\{ \sum_{j=1}^{n} \left( \frac{x_{ij}^* - x_{ij}}{s_{ij} - s_{ij}^*} \right)^p \right\}^{1/p}, 1 \leq p \leq \infty, i = 1, 2, \ldots, m
\]

(1)

where \(x_{ij}^* = \text{max}_k x_{kj}\) and \(x_{ij} = \text{min}_k x_{kj}\).

The group utility of criteria of \(a_i\) is \(S_i = L_{1,i} = \sum_{j=1}^{n} u_j \left( \sum_{j=1}^{n} \left( \frac{x_{ij}^* - x_{ij}}{s_{ij} - s_{ij}^*} \right)^p \right) \). The individual regret is \(R_i = L_{\infty,i} = \max_{1 \leq j \leq m} \left\{ u_j \left( \sum_{j=1}^{n} \left( \frac{x_{ij}^* - x_{ij}}{s_{ij} - s_{ij}^*} \right)^p \right) \right\} \).

A compromise value \(Q_i\) is calculated: \(Q_i = \frac{v(S_i - S^*)}{s - s^*} + \frac{(1-v)(R_i - R^*)}{R - R^*}\), where \(S^* = \max_j S_j\), \(S = \min_j S_j\), \(R^* = \max_j R_j\), \(R = \min_j R_j\), and we set \(v = 0.5\). Compromise ranking results are derived according to the values of \(Q_i\), \(S_i\), and \(R_i\).

3.2. Interpretive structural modeling

Interpretive structural modeling (ISM) [48] is used to establish element relations in a decision system. The process of ISM [49] is based on the mapping of binary matrices and digraphs. Basic components of an ISM include a subordinate relation (represented as R) and a set of elements (represented as \(E = \{ e_1, e_2, \ldots, e_n \}\)) that needs to be partially ordered through the relation. We use \(r_{ij}\) to represent a relation between \(e_i\) and \(e_j\).

The process of ISM is as follows: (1) Identify elements \(E\). Identify the factors related to a complex system based on expert opinions. (2) Identify the contextual relations \(R\) of elements. A contextual relation is a subordinate relation between two elements. If \(e_j \rightarrow e_i\), we say that \(e_j\) has a subordinate relation to \(e_i\), e.g., \(e_j\) influences \(e_i\), \(e_j\) leads to \(e_i\), and \(e_j\) is preferred to \(e_i\). (3) Develop a structural self-interaction matrix (SSIM). SSIM demonstrates the meanings of contextual relations between the elements. Let \(SSIM = \{ r_{ij} \}\), \(e_i \in \{ 1, \ldots, m \}\) and \(j \in \{ 1, \ldots, n \}\). V, A, X, and O represent four different relations between elements, which are defined as: if \(e_j \rightarrow e_i\), \(r_{ij} = V\); if \(e_j \rightarrow e_i\), \(r_{ij} = A\); if \(e_j \rightarrow e_i\), \(r_{ij} = X\); if \(e_j \rightarrow e_i\), \(r_{ij} = O\). (4) Develop a reachability matrix (RM). RM is a binary matrix built from SSIM to represent the reachabilities between elements. First, SSIM is converted to a binary matrix: \(BM = [b_{ij}]\), such that if \(r_{ij} = X, b_{ij} = 1\) and \(b_{ji} = 0\); if \(r_{ij} = A, b_{ij} = 1\) and \(b_{ji} = 0\); if \(r_{ij} = V, b_{ij} = 1\) and \(b_{ji} = 0\); if \(r_{ij} = O, b_{ij} = 0\) and \(b_{ji} = 0\); and \(b_{ij} = 1\) means that an element is always reachable to itself. \(RM = BM^{k+1} * BM = BM^{k+1} * BM = \cdots\). (5) Classify elements based on driver power and dependence. The driver power of an element is the number of elements it can reach; the dependence of an element is the number of its antecedents. (6) Hierarchically partition elements. Rank the elements according to their driver power and dependence, and identify the reachable set and the antecedent set of each element.

3.3. Fuzzy measures and fuzzy integral

A fuzzy integral is a kind of utility aggregation operator that is capable of integrating the importance of single criteria and the importance of criteria coalitions [50]. Decision makers need to define a set of importance values to calculate the FM, which is a set of importance values for all subsets of the criteria. However, the two issues hindered the wide application of FMs [51]: (1) given \(n\) criteria, a FM of these criteria requires the determination of \(2^n - 2\) weight coefficients (see Definition 1); and (2) it is difficult to define these coefficients based on their definitions and decision contexts.

To reduce the complexity and the difficulty of determining the \(2^n - 2\) coefficients, a k-order additive FM (or k-additive FM) [51] was proposed, where \(k < n\). A k-additive FM is determined by \(\sum_{i=1}^{n} C_i\) parameters (see Formula (5) and Definition 2). In particular, the 2-additive FM supports the interactions between two criteria, which is widely recognized as the best k-additive FM as it balances the computational complexity and the model richness [52]. A 2-additive FM is determined by \(n(n+1)/2\) parameters, and the Choquet integral (CI) based on a 2-additive FM can then be calculated by using the Shapley importance (see Formula (3)) and the interaction indices (see Formula (4)) of criteria (see Formulas (2) and (6)).

4. Problem definition and framework overview

In this section, we define the cloud service selection problem in CSSCI and overview the framework of CSSCI.

4.1. Problem definition

Given a set of cloud services \(S = \{ s_1, \ldots, s_m \}\), a service is evaluated by \(n\) criteria \(C = \{ c_1, \ldots, c_n \}\). Let the performance of the \(j\)th criterion of the \(i\)th service be \(f_{ij}\), then the utility of \(f_{ij}\) is defined as \(u_{ij} = G(f_{ij})\), where \(G : R \rightarrow [0, 1]\). Let \(\bar{Y}\) represent a user's preferences on different criteria in \(C\), the utility of a cloud service \((s_i)\) is defined as \(u_i = \bar{F}(u_{i1}, \ldots, u_{in}, \bar{Y})\), where \(\bar{F}\) is an aggregation operator. The objective of cloud service selection is to identify \(U = \{ u_1, \ldots, u_m \}\) based on \(\bar{Y}\). \((u_{ij} \forall i \in [1, m], j \in [1, n])\), \(G\), and \(F\). If \(u_i \leq u_{ij}\) then \(s_i \leq s_j\), i.e., \(s_i\) is superior to \(s_j\).

Given \(C = \{ c_1, \ldots, c_n \}\), and its power set \(P(C), Definition 1\) defines the FM of \(C\) [8]. The FM of a criterion subset \(A (A \subseteq C)\) is an assessment of the preferred degree of a decision maker on \(A\).

Definition 1. A FM of \(C\) is a set function \(\mu : P(C) \rightarrow [0, 1]\) satisfying: (1) \(\mu_{\emptyset} = 0\), \(\mu_C = 1\); and (2) \(A, B \subseteq C, A \subseteq B \Rightarrow \mu_A \leq \mu_B\).

The CI utility of a service \((s_i)\) with a criterion utility function \(u : C \rightarrow [0, 1]\) w.r.t a FM \(\mu\) is defined by Formula (2).

\[
ChU_i = \sum_{i=1}^{n} \left( u_{ij}^{(i)} - u_{ij}^{(i-1)} \right) \mu_{\{j\}}^{(i)}
\]

(2)

where \(u_{ij}^{(0)} \leq u_{ij}^{(1)} \leq \cdots \leq u_{ij}^{(n)}\), and \(C_i^{(n)} = \{ c_1^{(n)}, \ldots, c_n^{(n)}\}\). Based on a user's preference for a criterion, we can obtain the partial importance of the criterion to the user in selecting an appropriate cloud service. In addition, the interactions between criteria can also influence the importance of a criterion to the service selection results. Therefore, the FM of a criterion alone cannot reflect its overall importance. Shapley [53] proposed a definition of the importance coefficient, namely Shapley importance, which has been extended to FMs to evaluate the overall importance of a criterion. Given a FM \(\mu\) of \(C\), the Shapley importance of criterion \(c_j\) w.r.t \(\mu\) is defined by Formula (3).

\[
I_j = \sum_{A \subseteq C \backslash j} \frac{(n - |A| - 1)!}{n!} |A|! (\mu_{A \cup j} - \mu_A)
\]

(3)

where \(n\) and \(A\) are the number of criteria in \(C\) and \(|A|\) respectively, and \(C_j\) and \(A \cup \{ c_j \}\) represent \(C \setminus \{ c_j \}\) and \(A \cup \{ c_j \}\) respectively.

Another importance coefficient is the interaction index between criteria, which measures the interaction degree between two criteria from any subset of criteria [5]. Given a FM \(\mu\) of \(C\), the
interaction index between $c_j, c_l \in C$ w.r.t. $\mu$ is defined by Formula (4).

$$I_{jl} = \sum_{A \subseteq C, \lvert A \rvert \geq 2} \left( \frac{n - |A| - 2|A|!}{(n-1)!} \right) \times$$

$$\left( \mu_{A,jl} - \mu_{A,i} - \mu_{A,lj} + \mu_{A} \right)$$

where $jl$ represents set $(c_j, c_l)$.

Formula (5) defines another set function on $C$, which is the inverse of $\mu$ and is called the Möbius transform of $C$ [54].

$$m(A) = \sum_{B \subset A} (-1)^{|A| - |B|} \mu_C, \forall A \subset C$$

(5)

Given the Möbius transform $M$, the FM can be defined by $\mu_A = \sum_{B \subset A} m(B)$ [54].

From Definition 1, if we want to get a FM of $C$, we need to determine $2^k - 2$ coefficients, and the coefficient number increases exponentially with $n$. K-additive FM [54] is proposed to reduce such computational complexity, which is defined in Definition 2 [54].

**Definition 2.** A FM $\mu$ is said to be k-additive if $m(A) = 0$, for all $|A| > k$ ($k < n$), $A \subset C$.

The 2-additive case has been widely explored as it reduces computational complexity while keeping the consideration of criterion interactions. So in this paper, we establish a 2-additive model of service selection.

There are three types of criterion interactions: conflicting, supporting, and independent [55], all of which are symmetric. Each relation has an attached value that indicates the interaction type and degree between the two criteria. These three interactions are defined in Definition 3.

**Definition 3.** Given $c_l$ and $c_j$ ($j, l \in [1, n]$), let $I_l$ and $I_j$ represent the Shapley importance of $c_l$ and $c_j$, and $I_{jl}$ represents the interaction index between them. (1) if $c_j$ and $c_l$ conflict with each other, then $I_{jl} > 0$. Their capacity $\mu_j = I_l + I_j + I_{jl} > I_l + I_j$; (2) if $c_j$ and $c_l$ support each other, then $I_{jl} < 0$ and $\mu_j = I_l + I_j - I_{jl} < I_l + I_j$; and (3) if $c_j$ and $c_l$ are independent, then $I_{jl} = 0$ and $\mu_j = I_l + I_j$.

The FM of $A, A \subset C$, can be defined based on the Shapley importance and interaction indices in Formula (6).

$$\mu_A = \sum_{j \in A} (I_j - \frac{1}{2} \sum_{i \in A \cap (\{j\} \cup \{i\})} I_{ij} + \sum_{j \in A} I_j$$

(6)

We can then calculate the CI utility of a service $s_j$ using Formula (2).

Till now, the main task of multi-criteria cloud service selection is to identify $I_l$ and $I_{jl}$, $\forall j, l \in [1, n]$ and $j \neq l$. We adopt a non-linear constraint optimization model (Formula [7]) [8] to solve this task.

$$\min Z = \sum_{j \in C} \sum_{l \in C} \left( \frac{r_{jl} - \frac{I_l}{I_j + I_l}}{2} + \frac{p_{jl} - \frac{I_l}{I_j + I_l}}{2} \right)$$

s.t.

$$\left\{ \begin{array}{l}
            \sum_{j=1}^{n} I_j = 1 \\
            I_j - \frac{1}{2} \sum_{l \in C \cap \{j\}} I_{jl} + \frac{1}{2} \sum_{l \in C \cap \{j\}} I_{jl} > 0, \forall A \subseteq C, \forall j \in A \\
            I_j = I_l
          \end{array} \right.$$  

(7)

where $r_{jl}$ is the relative importance of $c_j$ to $c_l$ and $p_{jl}$ represents the partial interaction index between $c_j$ and $c_l$. We will define $r_{jl}$ and $p_{jl}$ specifically in CSSCII.

**4.2. Framework overview**

Fig. 1 shows the process of CSSCI, which has two parts: Part I solves service selection problems with enough historical data on the evaluation criteria, so that the criteria interaction indices can be determined objectively based on these data. We call the method in Part I CSSCI. Part II deals with service selection in situations where there is a lack of information to determine the interaction indices and weights of single criteria. We call the method in Part II priority-based CSSCI (PCSSCI).

CSSCI comprises the following steps (see Fig. 1 Part I): Step 1 derives the satisfaction degree of each criterion w.r.t. an alternative. Let $u_i^t$ represent the satisfaction degree of the $j$th criterion of the $ith$ service. For benefit criteria, $u_i^t = f_i^t / \max(f_i^t)$; and for cost criteria, $u_i^t = 1 - f_i^t / \max(f_i^t)$. Step 2 determines the relative importance between criteria based on the cloud users’ preferences by using a pair-wise comparison method. Step 3 identifies the interaction ratio between criteria. Step 4, given the relative importance and interaction ratio, solves the non-linear constraint optimization model to obtain the optimal Shapley importance of criteria and the interaction indices between criteria. Step 5 estimates the overall utility of each service by using the 2-additive CI.

PCSSCI only requires users to provide the priority order of criteria and the types of interactions between criteria, which comprises four main steps: Step 1 is the same as the Step 1 of CSSCI. Step 2, a user determines the criterion priority order based on his preferences, and defines criteria interaction types based on his professional knowledge. Step 3, based on the initial criteria interaction types, an interactive interpretive structural modeling (I-ISM) approach is used to establish consistent interactions between criteria, and to determine the importance weights of criteria by taking into account their consistent interactions. Step 4, given the priority order, importance weights and satisfaction degrees of criteria, estimates the FMs of the criteria and calculates the utilities of services using the CI.

In PCSSCI, the priority orders and importance weights of criteria are different. Priority orders indicate the preferences of service users on a set of given criteria, while the importance weights demonstrate the influence of criteria interactions on final decisions. The most prioritized criterion for a user can be different to the most important criterion for decision making. The satisfaction degree of a criterion reflects a user’s satisfaction with the criterion’s performance of a service.

**5. Cloud service selection with criteria interactions**

In this section, we present the steps of CSSCI in detail in a methodological manner and describe a running example of using CSSCII.

**5.1. Definitions of criteria for cloud service selection**

In this work, we discuss 11 popular and important QoS criteria for cloud service ranking, which are availability (av), throughput (th), successability (su), reliability (re), latency (la), response time (res), response time of customer services (rscs), cost (ct), scalability (sc), storage capacity (stc), type of customer services (tcs). The works in [56] and [57] defines criteria (av, th, su, re, la, res) and (rscs, ct, sc, stc, tcs) respectively, which are shown in Table 1, where Type indicates whether a criterion is benefit (B) or cost (C).

Examples in this section are based on both real and simulated QoS datasets (including 2507 services) [58] in terms of the 8 criteria (av, th, su, re, la, res, rscs, and ct in Table 1). In particular, the performance values of criteria av, su, re, la, res, rscs, and ct are collected by Al-Masri and Mahmoud [56]. As to criteria rscs and ct, we simulate their values by taking samples from the unique
Cloud QoS samples collected from service level agreements (SLAs) assume that rscs the upper and lower limits of distributions scale.

**Table 1**

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Type, Unit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>av</td>
<td>B, %</td>
<td>Number of successful invocations/total invocations</td>
</tr>
<tr>
<td>th</td>
<td>B, invokes/s</td>
<td>Number of invocations for a given period of time</td>
</tr>
<tr>
<td>su</td>
<td>B, %</td>
<td>Number of responses/number of request messages</td>
</tr>
<tr>
<td>re</td>
<td>B, %</td>
<td>1-number of error messages/total messages</td>
</tr>
<tr>
<td>la</td>
<td>C, ms</td>
<td>Time taken for the server to process a given request</td>
</tr>
<tr>
<td>res</td>
<td>C, ms</td>
<td>Time taken to send a request and receive a response</td>
</tr>
<tr>
<td>rscs</td>
<td>C, mins</td>
<td>Time taken by a service provider to answer emails and online chat</td>
</tr>
<tr>
<td>ct</td>
<td>C, USD</td>
<td>Dollars/month/50 GB</td>
</tr>
<tr>
<td>sc</td>
<td>B</td>
<td>If service upgrade requires file re-uploading, sc = 0; else sc = 1</td>
</tr>
<tr>
<td>src</td>
<td>B</td>
<td>Maximum storage capacity</td>
</tr>
<tr>
<td>tcs</td>
<td>B</td>
<td>Number of customer services matching user requests * 0.8 + number of the other provided customer services * 0.2</td>
</tr>
</tbody>
</table>

**Table 2**

<table>
<thead>
<tr>
<th>pw</th>
<th>av</th>
<th>su</th>
<th>ct</th>
</tr>
</thead>
<tbody>
<tr>
<td>av</td>
<td>1</td>
<td>1</td>
<td>1.25</td>
</tr>
<tr>
<td>su</td>
<td>1</td>
<td>1</td>
<td>1.25</td>
</tr>
<tr>
<td>ct</td>
<td>0.8</td>
<td>0.8</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 3**

<table>
<thead>
<tr>
<th>$r_{ij}$</th>
<th>av</th>
<th>su</th>
<th>ct</th>
</tr>
</thead>
<tbody>
<tr>
<td>av</td>
<td>0.5</td>
<td>0.5</td>
<td>0.556</td>
</tr>
<tr>
<td>su</td>
<td>0.5</td>
<td>0.5</td>
<td>0.556</td>
</tr>
<tr>
<td>ct</td>
<td>0.444</td>
<td>0.444</td>
<td>0.5</td>
</tr>
</tbody>
</table>

**Table 4**

<table>
<thead>
<tr>
<th>$t_{ij}$</th>
<th>av</th>
<th>th</th>
<th>su</th>
<th>re</th>
<th>la</th>
<th>res</th>
<th>rscs</th>
<th>ct</th>
</tr>
</thead>
<tbody>
<tr>
<td>av</td>
<td>−1</td>
<td>−0.1</td>
<td>−0.98</td>
<td>−0.02</td>
<td>−0.08</td>
<td>−0.06</td>
<td>−0.004</td>
<td>0.999</td>
</tr>
<tr>
<td>th</td>
<td>−0.098</td>
<td>−1</td>
<td>−0.11</td>
<td>−0.26</td>
<td>−0.28</td>
<td>−0.35</td>
<td>−0.02</td>
<td>0.097</td>
</tr>
<tr>
<td>su</td>
<td>−0.98</td>
<td>−0.11</td>
<td>−1</td>
<td>−0.02</td>
<td>−0.11</td>
<td>−0.08</td>
<td>−0.01</td>
<td>0.98</td>
</tr>
<tr>
<td>re</td>
<td>−0.02</td>
<td>−0.26</td>
<td>−0.02</td>
<td>−1</td>
<td>−0.21</td>
<td>−0.056</td>
<td>0.041</td>
<td>0.021</td>
</tr>
<tr>
<td>la</td>
<td>−0.081</td>
<td>−0.28</td>
<td>−0.11</td>
<td>−0.21</td>
<td>−1</td>
<td>−0.39</td>
<td>−0.01</td>
<td>0.08</td>
</tr>
<tr>
<td>res</td>
<td>−0.062</td>
<td>−0.35</td>
<td>−0.079</td>
<td>−0.056</td>
<td>−0.39</td>
<td>−1</td>
<td>−0.021</td>
<td>0.062</td>
</tr>
<tr>
<td>rscs</td>
<td>−0.004</td>
<td>−0.016</td>
<td>−0.014</td>
<td>0.041</td>
<td>−0.0057</td>
<td>−0.02</td>
<td>−0.02</td>
<td>0.004</td>
</tr>
<tr>
<td>ct</td>
<td>0.999</td>
<td>0.097</td>
<td>0.98</td>
<td>0.02</td>
<td>0.08</td>
<td>0.06</td>
<td>0.004</td>
<td>−1</td>
</tr>
</tbody>
</table>

QoS samples can be found in [57]. Criteria sc, stc and tcs are used in examples of priority-based CSSCI, which will be introduced in detail in Section 6. We preprocess the QoS dataset [56] by replacing the outliers using the interquartile range (IQR) method [59]. That is, for a set of performance data of a QoS criterion (e.g. availability), we replace the value below $Q_i = 1.5 \times IQR$ using Q1 and replace the value above $Q_3 + 1.5 \times IQR$ using Q3, where Q1 and Q3 are respectively the first and the third quartiles of the criterion data, and $IQR = Q_3 - Q_1$. Throughout this section, we describe a running example of estimating the utility of two SaaSs ($s_1$ and $s_2$) by using CSSCI based on three criteria: {av, su, ct}.

5.2. Determine relative importance

CSSCI allows decision makers to apply a pair-wise comparison to determine the partial importance of each criterion (represented by $w_i$, $\forall i \in [1, n]$) based on the users’ preferences, and the relative importance of criterion $c_i$ to criterion $c_j$ is $r_{ij} = w_i/w_j$, $\forall i, j \in [1,n], i \neq j$. Inspired by the idea of guaranteeing the consistency of the pair-wise comparison matrix in [60] and [61], CSSCI requires cloud service users to first determine the most important criterion (suppose criterion $c_i$). Then users conduct $n - 1$ times pair-wise comparison between $c_i$ and $c_j$, $\forall j \in [1,n], i \neq j$. CSSCI can then determine the pair-wise relations between $c_i$ and $c_j$ ($i \neq j \neq k$) by Formula (8), where $c_i/c_j$ represents the importance ratio of $c_i$ to $c_j$. Compared with the traditional pair-wise comparison method [62] which requires $(n^2 - n)/2$ times pair-wise comparisons, our method only requires decision makers to make $n$ times decisions: choose the most important criterion and then make $n - 1$ decisions in scales [0 min, 208 min] and [0.0, 18], based on an assumption that av and ct positively influence each other (i.e. the higher the availability of a cloud service, the more expensive the service), while rscs and ct negatively influence each other, where the upper and lower limits of rscs and ct are determined by 10 Cloud QoS samples collected from service level agreements (SLAs) and the Websites of service providers, where details of the 10 Cloud
Table 5: Optimized interaction indices among three criteria.

<table>
<thead>
<tr>
<th>$I_i^*$</th>
<th>av</th>
<th>su</th>
<th>ct</th>
</tr>
</thead>
<tbody>
<tr>
<td>av</td>
<td>−1</td>
<td>−0.29</td>
<td>0.27</td>
</tr>
<tr>
<td>su</td>
<td>−0.29</td>
<td>−1</td>
<td>0.26</td>
</tr>
<tr>
<td>ct</td>
<td>0.27</td>
<td>0.26</td>
<td>−1</td>
</tr>
</tbody>
</table>

times pair-wise comparisons. In particular, this method can always keep the consistency of the comparison matrix.

$$c_j^* = \frac{c_j}{c_k^*}$$  \hspace{1cm} (8)

**Example 1.** A decision maker believes av is the most important criterion for selecting a SaaS, su has similar importance to av, and av is 5/4 times more important than ct. Their pair-wise comparison matrix is shown in Table 2. The consistency ratio of this matrix is 0. The partial importance of these three criteria is equal to the values of the eigen vector in Table 2: $(w_{av}, w_{su}, w_{ct}) = (0.357, 0.357, 0.286)$. CSSCI can then determine the relative importance among av, su, and ct (see Table 3).

5.3. Determine interaction indices

We define an interaction ratio $t_{ij} = -\rho_{ij}$, where $\rho_{ij}$ is the Spearman co-efficient [63] between $c_i$ and $c_j$, $\rho_{ij} \in [-1, 1]$, and $t_{ij} = t_{ji}$; and the interaction index $p_{ij} = t_{ij} * b$, where $b \in [0, +\infty]$ is a constant coefficient. The optimization model of Formula (7) is then represented by Formula (9). Based on the service dataset of [58], we obtain the interaction ratio $t$ (see Table 4) of the eight criteria (see Table 1). We can see the interaction of av and su has a high-negative influence on the evaluation utility of a service, that is, av and su positively interact with each other, and ct negatively interacts with all the other criteria.

$$\min Z = \sum_{i \in C} \sum_{j \in C \setminus \{i\}} \left[ \left( t_{ij} - \frac{l_i}{l_i + l_j} \right)^2 + \left( t_{ij} * b - \frac{l_j}{l_i + l_j} \right)^2 \right]$$

s.t.

$$\sum_{i=1}^{n} l_i = 1$$

$$l_i = \frac{1}{2} \sum_{j \in C \setminus A} l_j + \frac{1}{2} \sum_{j \in A \setminus i} l_j > 0, \forall A \subseteq C, \forall i \in A$$

$$l_j = l_j$$

$$b \in [0, +\infty]$$

By solving Formula (9), we get the optimal importance weights (i.e. Shapley importance) of criteria ($I_i^*$) and their interaction indices ($I_i^*$). We use the augmented Lagrangian algorithm [64] to solve this model.

**Example 2.** Given criteria av, su, and ct ($n = 3$), we know their relative importance (see Table 3) and their interaction ratios (see Table 4); and we set the initial values of importance weights as: $I_i^0 = w_i$, and the initial values of the interactive indices as: $I_i^0 = t_{ij} * b$, where $c_i, c_j \in \{av, su, ct\}$ and $b = 1$. We then get the optimized $I_i^* = 0.3571, I_j^* = 0.3571, I_k^* = 0.2858, b^* = 0.4162$ and the optimized $I_i^*$ in Table 5.

5.4. Cloud service ranking based on 2-order additive Choquet integral utility

Based on the $I_i^* \forall i \in \{1, n\}$ and $I_j^* \forall j \in \{1, n\}, i \neq j$, we use Formula (6) to calculate the capacity ($\mu$) of a set of criteria A. The CI utility of a service is calculated by Formula (2).

Table 6: Utilities of two cloud services based on three criteria.

<table>
<thead>
<tr>
<th>Cloud services</th>
<th>av</th>
<th>su</th>
<th>ct</th>
<th>Utilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>0.95</td>
<td>0.97</td>
<td>0.72</td>
<td>0.83</td>
</tr>
<tr>
<td>s2</td>
<td>0.96</td>
<td>0.96</td>
<td>0.86</td>
<td>0.91</td>
</tr>
</tbody>
</table>

**Example 3.** Table 6 shows an example of the utility calculations of s1 and s2. Columns 2–4 of Table 6 show the satisfaction degrees of two SaaSs. We calculate their utilities based on criteria $C = \{av, su, ct\}$. The column Utilities shows the CI utilities of services. We can see that though the Shapley importance and partial utilities of av and su of s1 are higher than those of av and su of s2, the Choquet utility of s2 is higher than that of s1. As the interactions of (av, ct) and (su, ct) have a positive contribution to the overall performance of a service, and the interaction of (av, su) has a negative contribution, so an increase of the ct utility would increase the service utility even though the importance of ct is less than that of av and su.

6. Priority-based CSSCI

Section 5 introduces service rankings based on CSSCI in cases where decision makers have enough datasets to determine the interaction indices of criteria. However, in some cases, we do not have enough historical data, so we have to select services based on experts’ or users’ preferences. In this section, we propose a method for service selection in cases where there is a lack of objective coefficients between criteria. This method requires users to provide priority orders of criteria and the types of interactions between criteria, so we call it priority-based CSSCI (PCCSSI). We overviewed PCCSSI in Section 4.

6.1. Interactive interpretive structural modeling

We first introduce the I-ISM that assists in determining consistent criteria relations by interacting with decision makers in relation building processes, which is an extension of our previous work [57].

We let $r_{ij}$ represent an interrelation between $c_i$ and $c_j$ (i.e. av, su, ct). We set the initial values of each $r_{ij}$ to random numbers in [0, 1]. We use Formula (6) to calculate the capacity ($\mu$) of a set of criteria A. The CI utility of a service is calculated by Formula (2).
a reachability matrix (RM) is stable \((RM = RM^k = RM^{k+1})\), consistent and symmetric \((r_{ij} = r_{ji} \text{ for all } i \text{ and } j)\). Given a relation \(R\) of a set of criteria, if one of the following two cases exists, \(R\) is inconsistent and we need to adjust RSSIM of \(R\) (1) if both \(r_{ij}\) and \(r_{ji}\) exist for any \(c_i, c_j \in C\); (2) if \(\exists p_i, p_j, p_{i,j} \in P_0\) and \(p_{i,j} \neq p_i \circ p_j\). Fig. 2 shows an example (Example 4) of using relation operations to interactively check inconsistencies and build consistent criteria relations.

**Example 4.** Given criteria \([c_1, c_2, c_3, c_4]\). Fig. 2(a) shows an initial RSSIM that is built by a decision maker, which indicates initial relations \(R_0\) between criteria: (1) \(p_{141} = r_{14,1} = 1\); (2) \(p_{131} = r_{13,1} = 1\); (3) \(p_{122} = r_{12,2} = 1\); (4) \(p_{123} = r_{12,3} = 1\). The relations between \(c_1\) and \(c_4\) are transitively conflicts with \(c_3\), which \(c_3\) on path \(p_{13,2}\) that connects \(c_1\) and \(c_4\), and \(p_{13,2} = 2\); (1) \(p_{12,1} = r_{12,3} = -1 \otimes 1 = -1\); (2) \(p_{12,2} = r_{12,3} = 1 \otimes 1 = 1\); and \(p_{12,2} = r_{12,3} = 1\). The powering matrix is \(\text{RSSIM}_1 = \cdots \text{RSSIM}_3\). The RM is shown in Fig. 2c. Relations \(r_{42,1}\) is inconsistent and \(r_{12,3} = -1\) are the identified new transitive relations, and \(r_{12,3} = -1\) is an identified direct relation.

6.2. Determine weights of criteria based on I-ISM

After establishing the relations between criteria, we introduce a method to determine the criteria weights based on the influence of criteria relations on the final decision. As discussed previously, given a relation \(r_{ij}\) between criteria \(c_i\) and \(c_j\) \((i \neq j)\), if \(c_i\) and \(c_j\) are supportive of each other, \(r_{ij} = -1\); and \(u(i \cup j) = u_i + u_j + r_{ij} \ast u(i \cap j) < u_i + u_j\); if \(c_i\) and \(c_j\) are conflicting with each other, \(r_{ij} = -1\); and \(u(i \cup j) = u_i + u_j + r_{ij} \ast u(i \cap j) > u_i + u_j\); if \(c_i\) and \(c_j\) are independent of each other, \(r_{ij} = 0\) and \(u(i \cup j) = u_i + u_j\). Therefore, ideally the most important criterion in a set \(C\) is conflicting with all the other criteria; and the ideal least important criterion is supportive of all the other criteria. Given a set of criteria \(C = \{c_1, \ldots, c_n\}\), let \(c_m \in C\) and \(c_k \in C\) be the most and least important criteria respectively. The interactions between \(c_m\) and other criteria are \((r_{m1}, \ldots, r_{mn}) = (1, \ldots, 1)\). The interactions between \(c_k\) are \((r_{1k}, \ldots, r_{nk}) = (-1, \ldots, -1)\).

Assume the interactions between \(c_i\) and all the other criteria are \(r_{ij}\) for all \(c_j \in C\), the distance between \(c_i\) and \(c_m\) is \(d_{im} = \sqrt{\sum_{j=1}^{n} (r_{ij} - r_{mj})^2}\); and the distance between \(c_i\) and \(c_k\) is \(d_{ik} = \sqrt{\sum_{j=1}^{n} (r_{ij} - r_{kj})^2}\). Then we calculate the importance weight of \(c_i\) by \(g_i = d_{il}/(d_{l} + d_{m})\), and normalize it by \(u_i = g_i/\sum_{l=1}^{n} g_l\).
are calculated as
\[ w = [0.0986, 0.0986, 0.0986, 0.1148, 0.1148, 0.1225, 0.1225, 0.1148] \]

For example, the weight of \( w_2 \) is calculated as: \( dl_1 = 3.1623; dm_1 = 4.6904; g_1 = dl_1/(dl_1 + dm_1) = 0.4027; \) and \( g_1 \) is normalized as \( w_1 = 0.0986. \) We can see criteria 7–8 have the highest weights, and criteria 1–3 have the lowest weights, because criteria 1–3 support criteria 4–6 and 9, so criteria 1–3 have the most supporting criteria in C.

6.3. Priority-based utility calculation for services

Assume the satisfaction degrees (or the performance) of criteria \( C = \{c_1, c_2, ..., c_9\} \) of service s are \( x_1, ..., x_9; \) a service user has given a priority order of criteria: \( c_{1}\geq \cdots \geq c_{9}; \) and the importance weights of criteria \( (w_1, ..., w_9) \) have been determined by the method introduced in Section 6.1. We calculate utilities by aggregating the performance of criteria \( s \) by taking the following steps [65].

1. Let \( L_i(C) \) represent a sequence of ordered criteria where the \( i \)th criterion has the highest priority in C. \( L_i = [c_k | k \in [1, i]] \), for all \( i \in [1, n]; \) and \( L_0 = \emptyset. \) For example, we use nine QoS criteria for evaluating service s, which are represented by \( C = [c_1, c_2, ..., c_9] = [av, re, sc, stc, la, th, tcs, rscs, ct]. \) The criterion priority given by a service user is: \( C = [c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9]; \) then \( L_4 \) of C is \( L_4(C) = [c_1, c_2, c_3, c_4]); \) \( C = \{c_1, c_2, c_3, c_4\} \) and \( L_0(C) = C. \) For subsets \( C = \{c_2, c_3\} \subset C \) and \( C' = \{c_1, c_2\} \subset C, \) let \( L_{ks}(C) \) represent the longest \( L \) in \( C, \) then \( L_{ks}(C') = L_0 = \emptyset \) as there is no \( c_{1} \) in \( C', \) and \( L_{ks}(C') = L_2. \)

2. Rank satisfaction degrees of criteria: \( x_{ind}(1), ..., x_{ind}(j) \), where \( ind(i) \) refers to the \( i \)th largest satisfaction degree. For example, given the performance and satisfaction degrees of the nine criteria of \( s \) are shown in rows \( s_1 \) and \( x(s_1) \) in Table 7 respectively, the criterion order in terms of satisfaction degrees is \( C = c_{1}\geq c_{4}\geq c_{3}\geq c_{2}\geq c_{1}\geq c_{5}\geq c_{6}\geq c_{7}\geq c_{8}\geq c_{9}, \) and \( x_{ind}(1) = x_4. \)

3. Let \( H_j \) represent the collection of \( j \) criteria having the largest satisfaction degrees, \( H_j = [c_{ind}(k) | k \in [1, j]]. \) For example, \( H_4 \) of \( C \) is \( C. \)

4. Determine FMFs of \( H_j \) of C. For any subset \( A \subseteq C, \) the FM of A has the property \( m(A) = m_{ks} = \sum_{j=1}^{n} w_j, \) where \( w_j \) is the weight of criterion \( c_j. \) For example, for \( H_2 = \{c_4, c_3\}, L_{ks} = L_0 = \emptyset \) and \( m(H_2) = 0; \) and for \( H_4 = \{c_4, c_3, c_2, c_1\}, L_{ks} = L_4 = \{c_1, c_2, c_3, c_4\} \) and \( m_{ks} = w_1 + w_2 + w_3 + w_4. \)

5. Determine Choquet weights of the ordered criteria \( q_j = m(H_j) - m(H_{j-1}), \forall j \in [1, n]. \)

6. Calculate Choquet utilities of service s: \( U(s) = Choq(c_1, ..., c_9) = \sum_{i=1}^{n} q_i x_{ind}(i). \)

We use Example 6 to show the process and efficiency of priority-based utility calculation for service selection.

Example 6. Given nine QoS criteria C, the users have given the priority order \( C_0 \) (in step (1)); importance weights: \( w_1 = 0.0986, w_2 = 0.0986, w_3 = 0.1148, w_4 = 0.1148, w_5 = 0.1148, w_6 = 0.1148, w_7 = 0.1225, w_8 = 0.1225, \) and \( w_9 = 0.0986; \) and satisfaction order \( C_0 \) (in step (2)). We can derive the \( H_j \) and \( L_{ks} \) in Table 7. The Choquet Utility of SI is: \( U_1 = \sum_{j=1}^{n} q_j x_{ind}(i) = \sum_{i=1}^{n} (m(H_j) - m(H_{j-1}))x_{ind}(j) = (m(lo_0) - 0)x_4 + (m(lo_0) - m(lo_1))x_3 + (m(lo_0) - m(lo_2))x_2 + (m(lo_0) - m(lo_3))x_1 + (m(lo_0) - m(lo_4))x_0 \)

7. Experiments and analysis

We conduct experiments in Matlab and R on a 64-bit Windows 10 platform with an Intel Core i7-6700HQ CPU and 16 GB RAM. We use both real [56] and simulated criteria datasets of SaaSs for CSSCI validation, which include eight criteria (av, th, su, re, la, res, rscs, and ct) in Table 1) to evaluate 2507 services. Section 5 gives a detailed description of this dataset.
In addition, we use examples to analyze the performance of priority-based CSSCI based on a QoS dataset of cloud computing services. We collected the QoS performance of 80 cloud computing services (called a computing dataset) from the Network-Testing Website of CloudHarmony (https://cloudharmony.com/speedtest). We tested the performance of the following QoS parameters: Throughput 1 [1–128 kB/4 threads] (MB/s, abbreviated as th1), Throughput2 [256 kB–10 MB/2 threads] (MB/s, th2) and Latency (ms, la1) at 4:40 pm, 20:20 pm, 21:40 pm of 03/26/2018, and 8:05 am, 9:55 am, 1:47 pm, 4:40 pm, and 7:30 pm of 03/27/2018. We averaged the throughputs and latencies at these time points. We also calculated the average availability (% per month, av1) of these 80 services from the Service-Status Website of CloudHarmony (https://cloudharmony.com/status). We preprocessed the computing dataset using the IQR method, which has been explained in Section 5.1. We share experimental datasets, codes, results and their descriptions in [58].

We compare the proposed CSSCI with three other decision making methods for ranking cloud services, which are SWA [66], VIKOR [67], and TOPSIS [68]. We introduced these three methods in Section 3. In particular, for VIKOR, we set \( v = 0.5 \) and choose the optimal service \( s^{*} \) such that \( Q_{s^{*}} = \min \{Q_{1}, \ldots, Q_{m}\} \), where \( m \) is the number of alternatives.

One way to evaluate the performance of an MCDM method is its conformity to other MCDM methods or the intuitive opinions of decision-makers [61]. Therefore, we design two algorithms average ratio of matching to expert opinions (ar) and Kendall Tau distance ratio (ktdr) to validate the conformity of our method to expert opinions and other MCDM methods respectively. Sensitivity analysis [69] is another important validation method for MCDMs, so we design a stability (st) algorithm to test the sensitivity of service ranking results influenced by the satisfaction degrees, importance weights and interaction indices of criteria.

We validate each service ranking method w.r.t. a different number of criteria (\( cn = \{3, 4, 6, 8\} \)) and a different number of services (\( sn = \{2, 5, 10, 15, \ldots, 250\} \)) each time. The evaluation of \( ar \) depends on the ground truth (GT) which is manually marked by service experts (see Part I of Algorithm 1). During the survey process, the experts pointed out that the manual ranking of services by humans based on five criteria is much more difficult than the manual ranking of services based on four criteria, so we did not validate \( ar \) w.r.t. five or more criteria.

The optimized importance weights and interaction indices are calculated based on decision makers' preferences and criteria coefficients in Table 4 by using the optimization model in Formula (9). The columns in Table 9 show the importance weights of criteria in criteria sets \( C_{k} = \{av, th, su, ct\}, |C_{1}| = 3; C_{2} = \{av, th, su, ct\}, |C_{2}| = 4; C_{3} = \{av, th, su, re, rsc, ct\}, |C_{3}| = 6; \) and \( C_{4} = \{av, th, su, re, la, res, rsc, ct\}, |C_{4}| = 8 \) correspondingly.

### 7.1. Average ratio of matching to expert opinions

We use GT to represent the order given by the experts. The higher the \( ar \) of a method, the better the method. Algorithm 1 shows how we collect and form the GT of rankings of a number of service groups (Part I), and how to calculate \( ar \) (Part II).

#### Algorithm 1 Test average ratio of matching to expert opinions

1. Given \( m \) cloud service groups \( \{g_1, \ldots, g_m\} \), each group contains two cloud services \( g_k = \{s_l, s_j\}, k \in [1, m], i \neq j \).
2. **Part I:** determine the GT of the order of the services in each group:
   3. A group of \( h \) cloud experts \( \{e_1, \ldots, e_h\} \) rank two services in each group, which will be counted to determine the ground truth \( \{gt_{1}, \ldots, gt_{m}\} \) of service ranking of the \( m \) groups. An expert has three options when ranking: \( s_j \geq s_i \) (superior to \( s_i \)), \( s_j \geq s_i \) and \( s_i > s_j \) (an expert cannot determine the order of \( s_i \) and \( s_j \) based on the given knowledge).
4. For each \( g_k \), \( k \in [1, m] \) do
   5. For each \( e_l \), \( l \in [1, h] \) do
     6. If \( e_l \) chooses \( s_i \geq s_j \), \( CNT_{s_j} = CNT_{s_i} + 1; // CNT_{s_j} \) counts the number of experts who rank \( s_j \) as superior to \( s_i \) in the \( k \)th group.
     7. If \( e_l \) chooses \( NA \), then continue to rank \( g_{k+1} \); // do nothing for \( g_k \)
8. end for
9. \( n_{gt_k} = \frac{CNT_{s_j}}{n}; // \) the ratio of the experts' ranking \( s_j \geq s_i \) of the \( k \)th group.
10. If \( n_{gt_k} > 0.5 \), then \( gt_k = (s_1 \geq s_2) \);
11. If \( n_{gt_k} \geq 0.5 \), then \( gt_k = (s_2 \geq s_1) \);
12. Else \( gt_k == NA; \)
13. end for
14. **Part II:** calculate the average ratio of matching to expert opinions:
15. For \( vk \in [1, m] \) do
16. If \( (gt_v == NA) \), then remove \( gt_v \) and \( rc = rc + 1; // \) if the experts cannot determine the order of the services in the \( k \)th group, then we do not use this group for \( ar \) calculation, and \( rc \) counts the number of removed groups
17. If \( (R_k == gt_v) \), \( mc = mc + 1; // R_k \) represents a ranking of two services in the \( k \)th group using a ranking method, and \( mc \) counts the number of \( R_k \) matching to \( gt_v \)
18. end for
19. return \( ar = \frac{mc}{m-rc} \).

#### 7.1.1. Comparison of average ratio of matching to expert opinions

We analyze \( ar \) in terms of different criteria sets: \( C_1 = \{av, su, ct\} \) and \( C_2 = \{av, th, su, ct\} \). The optimized importance weights of \( C_1 \) are \( l_{c1} = \{0.357, 0.357, 0.286\} \), and the optimized interaction indices of \( C_1 \) are shown in Table 5. For \( C_2 \), the optimized importance is \( l_{c2} = \{0.294, 0.144, 0.294, 0.267\} \), and the interaction indices are shown in Table 10. For both \( C_1 \) and \( C_2 \), we rank the 2507 services using the four methods: SWA, VIKOR, TOPSIS and CSSCI; and then we randomly choose 330 pairs of services, and each group contains two services to be compared by the decision makers. Cloud experts rank services in each group based on Algorithm 1. The \( ar \) values of CSSCI, SWA, VIKOR, and TOPSIS are shown in Fig. 3. We can see the accuracy of the ranking of CSSCI in terms of both 3 and 4 criteria which is higher than the accuracy of the other three methods. The \( ar \) values of CSSCI are 0.99 and 0.927 in cases of 3 and 4 criteria respectively. In addition, in the case of service rankings based on four criteria, the \( ar \) is lower than the \( ar \) in the case of three criteria. One of the reasons for this may be that the degree of human confusion increases when the number of criteria used for service ranking is increased.

### Table 9
Optimized Shapley importance of C with different \( |C| \).

| \( |C| \) | av | th | su | re | la | res | rscs | ct |
|---|---|---|---|---|---|---|---|---|
| 3 | 0.3571 | – | 0.3571 | – | – | – | – | 0.2857 |
| 4 | 0.2943 | 0.1441 | 0.2943 | – | – | – | – | 0.2672 |
| 6 | 0.24 | 0.0957 | 0.2344 | 0.1181 | – | – | 0.0995 | 0.25 |
| 8 | 0.216 | 0.0994 | 0.1312 | 0.0732 | 0.1021 | 0.0941 | 0.0732 | 0.216 |

### Table 10
Optimized interaction indices between four criteria.

<table>
<thead>
<tr>
<th>( l_{ij} )</th>
<th>av</th>
<th>th</th>
<th>su</th>
<th>ct</th>
</tr>
</thead>
<tbody>
<tr>
<td>av</td>
<td>–</td>
<td>–0.0165</td>
<td>–0.2202</td>
<td>0.2147</td>
</tr>
<tr>
<td>th</td>
<td>–0.0165</td>
<td>–1</td>
<td>–0.0183</td>
<td>0.0153</td>
</tr>
<tr>
<td>su</td>
<td>–0.2202</td>
<td>–0.0183</td>
<td>–1</td>
<td>0.2102</td>
</tr>
<tr>
<td>ct</td>
<td>0.2147</td>
<td>0.0153</td>
<td>0.2102</td>
<td>–1</td>
</tr>
</tbody>
</table>
7.1.2. Analysis of the average ratio of matching to expert opinions

We use special examples to analyze the ar of rankings based on three criteria. Table 11 shows the service ranking results of three groups, where SID is the Service ID; the columns of av, su, and ct are the normalized partial utilities of the three criteria; the columns of GT, SWA, TOPSIS, and CSSCI refer to the service orders ranked by these methods. Group1 is a case where the service ranking of SWA is different from the rankings of GT and CSSCI. We can see that the utilities of av and su of service 98 are higher than those of service 98, while the ct utility of service 97 is lower than that of service 98. As discussed earlier, ct negatively interacts with both av and su, while av and su positively interact with each other (see Table 5). Even though the Shapley importance and the performance of av and su are higher than those of ct (see IC1), the cloud experts believe service 97 (ordered as 1) is superior to service 98 (ordered as 2). In this case, the service ranking by SWA does not reflect the interactive relations between the criteria. Group2 is the case where the ranking by TOPSIS is different from the rankings by GT and CSSCI. From Table 11, the utility of ct of service 1274 is much lower than that of service 1275. However, the utilities of av and su of service 1274 are much higher than the performance of service 1275, and the Shapley importance of av and su is higher than that of ct. Therefore, the cloud experts believe service 1274 is superior to service 1275. TOPSIS calculates the utility of a service based on the distance of each criterion performance to the optimal performance of that criterion. The closer the criterion performance of a service to the optimal criterion performance, the better the service. In group2, as the performance of the ct of service 1275 has a much shorter distance to the optimal ct utility compared with service 1274, TOPSIS over-weighs the utility of ct. So it ranks service 1275 as superior to service 1274. Group3 shows the case where the service ranking of CSSCI is different to the ranking of GT. In particular, cloud experts believe service 77 is better than service 78, which is different to the rankings by the other methods. Human limitations may cause this result. Readers can refer to [58] for more groups of two-service rankings by the four MCDM methods and experts.

7.2. Kendall tau distance ratio among rankings

The lower the ktdr of a method, the better the performance of this method. We test the average ktdr w.r.t. different cn and sn. Fig. 4(a) shows the ktdrs of the SWA, CSSCI, VIKOR, and TOPSIS w.r.t. 3, 4, 6, and 8 criteria respectively. Each ktdr of method w.r.t. a criteria number is an average ktdr of a set of groups where each group contains different numbers (10, 30, 55, 75, and 100) of cloud services. For example, if a group contains 10 services, we separate 2500 services into 250 groups, rank the services in each group and calculate the average ktdr of these groups. We can see from Fig. 4(a) that the CSSCI has the lowest ktdr, which is around 2% lower than those of SWA and TOPSIS, and 3.5% lower than the ktdr of VIKOR. Fig. 4(b) shows the ktdrs w.r.t. different sn. We can see that the ktdr of CSSCI is around 10% lower than that of VIKOR and 5% lower than the ktdrs of VIKOR and TOPSIS w.r.t. sn = 10. However, as sn increases (≥ 50), the ktdrs of the four methods increase to a similar level, which reflects the common sense that the increase of sn can increase the difficulty of deriving an informed ranking.

Algorithm 2 Test stability of rankings

1: Given a criterion set C, |C|= n, I^0 = \{I^0_1, I^0_2, \ldots, I^0_n\} are the initial values of importance weights of single criteria and interaction indices between criteria, S = \{s_1, \ldots, s_n\} is a set of services, and N is the times of iteration of generating random numbers.
2: Test the stability of ranking w.r.t. changes of all items in I^0, t = \{t_1, \ldots, t_N\} is a set of changing ratios.
3: for k = 1 to l do
4: for r = 1 to N do
5: // For all \(i, j \in [0, 1], i, j \in [1, n], i < j\);
6: \(I^t_{ij} = I^0_i + I^0_j + t(k) \cdot I^t_{ij} \cdot rd; // change the Shapley importance all criteria, where rd is a random number in [-1, 1]\)
7: \(I^t_{ij} = I^0_i + I^0_j + t(k) \cdot rd; // change the interaction indices between criteria
8: Solve the optimization model min2 to get \(I^t;\)
9: Calculate the CI of each service based on \(I^t\) and rank services according to their CIs: orderk_r = order(S);
10: end for
11: Calculate average ktdr among the N orders: mktdr = \(\frac{\sum_{r=1}^{N} ktdr(order_{k_r}, order_{k_q})}{N-1}\);
12: end for
13: return mktdr // mktdr is a vector of length l, where each value represents the average ktdr produced by a ranking method w.r.t. a t value

7.3. Stability of rankings

We use the 'mean of ktdr' (represented by mktdr) of the service rankings of a method w.r.t. randomly changed importance weights and interaction indices to measure the stability of this method. The lower the mktdr, the higher the stability of the method. We test stabilities of service rankings based on Algorithm 2. A set of randomly changing ratios (r) are tested (lines 2 and 3 in Algorithm 2). We repeat each step \(N = 20\) times (line 4). After randomly changing the initial importance weights and interaction indices (lines 6 and 7), we calculate the optimized weights and indices (line 8) and derive the CI utility of each service (line 9).
Fig. 5(a) compares the mktdrs which are the averages of the cases considering different t and cn w.r.t. different sn. Fig. 5(b) compares the mktdrs which are averages of the cases considering different t and sn w.r.t. different cn. Fig. 5(c) compares the mktdrs which are averages of the cases considering different t and cn w.r.t. different t. From Fig. 5(a), the stabilities of SWA, TOPSIS, VIKOR and CSSCI are similar and very high when the number of services is equal to or less than 5, which means the minor changes of criterion importance and interaction indices do not have significant influence on the rankings of a small set of services. When the number of ranked services are 20, the stabilities of both of SWA and TOPSIS are 10% and 80% worse than the stabilities of CSSCI and VIKOR respectively. Fig. 5(b) shows that when the number of criteria increases, the mktd rs of the four methods increases, which means the more criteria used to rank the services, the more stable the ranking results. Fig. 5(c) shows that when t increases over 0.08, the ranking stabilities of SWA, TOPSIS and CSSCI decrease dramatically compared to the stability of VIKOR.

7.4. An approximate solution of CSSCI

In the previous sections, we described the service ranking process based on CSSCI, which needs to solve an optimization model to derive the optimal individual importance and interaction indices for criteria. The experiment results show that CSSCI is an efficient service ranking framework in terms of ar, ktdr and st. However, for cases with n criteria, it requires the optimization model to satisfy \( n(2^n + 2) + C_2^2 + 2 \) constraints (see Section 5.3), which is the most time consuming component of CSSCI. So the time complexity of CSSCI is \( O(2^n) \).

To reduce such complexity, we use approximate weights to avoid solving the optimization model. We use initial indices of singlet criterion derived from the pair-wise comparison to approximately represent the optimized individual index, and determine a fixed value of b (see Formula (9)) based on experiments to determine the approximate interaction indices. As we have proved the efficiency of CSSCI in previous experiments, we design this experiment as follows: first, we fix \( b = \{0, 0.1, 0.2, \ldots, 1\} \), and test these values w.r.t. different cn = \{3, 4, 6, 8\} and sn = \{10, 15, 20, 25, \ldots, 50\} by measuring the dissimilarity between this approximate setting (we call it apprCSSCI) and the optimized CSSCI. In this case, we use rate of ktd to represent the dissimilarity between apprCSSCI and CSSCI. Given a set of groups of services \( \{gp_1, \ldots, gp_m\} \) and each group contains n services \( |gp_1| = \cdots = |gp_m| = n \), the rate of ktd is defined as mean(ktd_{1, \ldots, ktd_m})/n, where ktd_{i}(yi \in \{1, \ldots, m\}) is the ktd of the ith group of services. Based on the definition of ktd, the rate of ktd demonstrates the rate of the average number of pairwise disagreements between two ranking lists for a fixed number of services being ranked.

Fig. 6(a) shows the average rate of ktd w.r.t. a series of b values, which averages the ktds in terms of different cn and sn. We can find the optimal \( b = 0.22 \) corresponding to the lowest rate of ktd. We then set \( b = 0.22 \) and calculate the average ktd between CSSCI and apprCSSCI w.r.t. a series of sn values, and compare this ktd with the average ktds between CSSCI and SWA, CSSCI and TOPSIS, and CSSCI and VIKOR (see Fig. 6(b)). We can see that if sn \( \leq 20 \), the ktd between CSSCI and apprCSSCI is less than 2, which means there are no more than two pairwise disagreements between two ranking lists given by CSSCI and apprCSSCI respectively. When sn \( \geq 25 \), the ktd increases dramatically. In addition, we can see that the ktds between CSSCI and SWA, CSSCI and TOPSIS, and CSSCI and VIKOR are larger than the ktd between CSSCI and apprCSSCI, which means compared to the other three methods, apprCSSCI can produce more similar results to CSSCI. We can derive from Fig. 6 that when \( b = 0.22 \) and sn \( \leq 20 \), the performance of apprCSSCI is close to that of CSSCI. As the main step of apprCSSCI is the calculation of the initial coefficients between criteria, its time complexity is \( O(n^r) \) where n is the number of criteria.

In summary, in cases requiring a high ranking accuracy but not requiring high time efficiency, we can use CSSCI. However, in cases ranking small number of services and requiring high time efficiency, we can use apprCSSCI.

7.5. Performance analysis of priority-based CSSCI

In this section, we analyze the performance of PCSSCI. As PCSSCI derives interaction indices of criteria based on the users’ preferences rather than objective datasets, and users only need to provide priority orders of criteria rather than specific time values of pair-wise comparisons, its decision-making procedure is very different with data-based CSSCI, apprCSSCI, SWA, TOPSIS and VIKOR. Therefore, we cannot use ar and ktdr to evaluate PCSSCI. We use two methods to analyze its results: (1) use an example to demonstrate its distinguishing features and advantages, and (2) use Algorithms 2 to estimate its stability of service ranking.

Example 7. A user needs to rank two cloud computing services \( s_1 \) and \( s_2 \) that are evaluated by criteria \( C_{c4} = \{c_1, c_2, c_3, c_4\} = \{av1, th2, th1, la1\} \). He provides a priority order: \( c_1 \geq c_2 \geq c_3 \geq c_4 \) and defines criteria interaction relations based on his knowledge: \( r_{12} = 0, r_{13} = 0, r_{14} = 0, r_{23} = -1, r_{24} = -1, \) and \( r_{34} = -1 \). The reachability matrix of these relations is \( RM_{c4} = \)

---

Fig. 4. Average ktdr w.r.t. different numbers of (a) criteria and (b) services per group.

---

Fig. 6(a) shows the average rate of ktd w.r.t. a series of b values, which averages the ktds in terms of different cn and sn. We can find the optimal \( b = 0.22 \) corresponding to the lowest rate of ktd. We then set \( b = 0.22 \) and calculate the average ktd between CSSCI and apprCSSCI w.r.t. a series of sn values, and compare this ktd with the average ktds between CSSCI and SWA, CSSCI and TOPSIS, and CSSCI and VIKOR (see Fig. 6(b)). We can see that if sn \( \leq 20 \), the ktd between CSSCI and apprCSSCI is less than 2, which means there are no more than two pairwise disagreements between two ranking lists given by CSSCI and apprCSSCI respectively. When sn \( \geq 25 \), the ktd increases dramatically. In addition, we can see that the ktds between CSSCI and SWA, CSSCI and TOPSIS, and CSSCI and VIKOR are larger than the ktd between CSSCI and apprCSSCI, which means compared to the other three methods, apprCSSCI can produce more similar results to CSSCI. We can derive from Fig. 6 that when \( b = 0.22 \) and sn \( \leq 20 \), the performance of apprCSSCI is close to that of CSSCI. As the main step of apprCSSCI is the calculation of the initial coefficients between criteria, its time complexity is \( O(n^r) \) where n is the number of criteria.

In summary, in cases requiring a high ranking accuracy but not requiring high time efficiency, we can use CSSCI. However, in cases ranking small number of services and requiring high time efficiency, we can use apprCSSCI.
We estimate the stability of PCSSCI by comparing it to the stability of CSSCI based on Algorithm 2 in the following two situations:

1. For each ranking of a group of services, we fix the satisfaction degrees and the priority order, and change the interaction-based weights by randomly adding and subtracting $t$ from themselves. We test the means of $ktdr$ of CSSCI and PCSSCI w.r.t. (a) $sn = \{2, 5, 10, 15, 20\}$, (b) $cn = \{3, 4, 6, 8\}$ and (c) $t = \{0.03, 0.05, 0.08, 0.1\}$ (see Fig. 7). For Fig. 7(a), the means of $ktdr$ of CSSCI and PCSSCI are derived by averaging the $ktdr$s of the situations considering different $cn$ and $t$.

2. For each ranking of a group of services, we fix the interaction-based weights, and change the satisfaction degrees by randomly adding and subtracting $t$ proportions from themselves. The testing results show that for all $t$ values, the average $ktdr$s of the ranking results of PCSSCI are 0, which means the service ranking results of PCSSCI are not sensitive to small changes in the satisfaction degrees of criteria as long as the priority order and interaction-based weights are fixed.

### 7.6. Significance of CSSCI

We summarize the significance of CSSCI through performance testing as follows:

- CSSCI takes into account both the single importance of criteria and non-linear interactions between criteria. The service ranking results of CSSCI fulfill the interactive preferences of service users.
- CSSCI has excellent stability w.r.t. small changes in criterion importance and interaction indices.
• Neither SWA nor TOPSIS is capable of measuring non-linear interactions between criteria in a service ranking process. SWA tends to over-weigh criteria with high importance, while TOPSIS tends to over-weigh criteria which is close to the service with optimal criterion performance.

• Though the performance of TOSIS in terms of $ar$ and $ktdr$ is closing to the performance of CSSCI, TOPSIS cannot measure and aggregate non-linear criterion interactions in service selection, so it cannot fulfil users’ interactive preference, which limits the application of TOPSIS in user-preference-based service selection problems.

• When the number of criteria increases, the difficulty of manually ranking services by cloud experts increases. It is much more difficult for experts to manually rank services by considering the non-linear relations among five or more criteria than four or less criteria, so it is necessary to develop CSSCI to rank services automatically.

8. Conclusion

In this paper, we proposed a CSSCI framework to evaluate the influence of different types of criteria interactions on cloud service selection results. We applied the FM, CI and non-linear constraint optimization techniques to identify the Shapley importance and interaction indices of criteria. In addition, we proposed a priority-based CSSCI to deal with the case where there is a lack of historical information to determine the weights of single criteria and interaction indices. In the experiments, we designed three algorithms to validate the MCDM methods. The experiment results show the service ranking results of CSSCI are the most efficient compared with the other popular MDCM methods. In the future, we will address two main issues of CSSCI: (1) CSSCI only supports users’ preferences and the QoS performance of crisp data. However, by surveying cloud experts and cloud users in our experiments, a mechanism supporting fuzzy expression is much more desirable; (2) we choose criteria in a criterion set $C$ purposely in this work to show the advantages of CSSCI. However, a more strenuous evaluation of different criteria in $C$ will improve the generality of CSSCI.

Funding

This work is supported by the National Natural Science Foundation of China (Grants No 61702274) and the Natural Science Foundation of Jiangsu Province (Grants No BK20170958), PAPD and partially supported by the China-USA Computer Science Researcher Center.

References


Hai Dong is a Lecturer and Vice-Chancellor’s Research Fellow at the School of Science in RMIT University, Melbourne, Australia. He received a Ph.D. and a Master’s degree from Curtin University of Technology, Australia, in 2010 and 2006. His research interests include service-oriented computing, semantic search, ontology, and cloud computing. He has published over 50 research publications in leading international journals and conferences.

Omar Khadeer Hussain is currently a Senior Lecturer at the School of Business, UNSW Canberra. Prior to joining the School in February 2014 as a Lecturer, he worked as a Senior Research Fellow at Curtin University. Omar’s research areas of interests are Logistics and Supply Chain Management, Distributed and Grid Systems, Decision Support, Group Support Systems and their applications to Logistics areas. Omar’s research has been published in international journals such as The Computer Journal, Journal of Intelligent Manufacturing etc. He has won university and faculty level awards from his research and as the main supervisor has supervised 7 Ph.D. students to completion. In 2011, he was awarded with an APDI Fellowship from the ARC on a Linkage project with Prof Elizabeth Chang as the lead CI.

Farookh Khadeer Hussain is an Associate Professor at the School of Software, University of Technology Sydney. He is an Associate Member of the Advanced Analytics Institute and a Core Member of the Centre for Quantum Computation and Intelligent Systems. He is a member of the Institute of Electrical and Electronics Engineers.

Alex X. Liu is a Professor in Department of Computer Science and Engineering, Michigan State University. He received his Ph.D. degree in Computer Science from The University of Texas at Austin in 2006. He received the IEEE & IFIP William C. Carter Award in 2004, the National Science Foundation CAREER Award in 2009, and the Michigan State University Withrow Distinguished Scholar Award in 2011. His research focuses on networking and security.