A Hybrid Approach for Personalized and Optimized IaaS Services Selection

Hamdi Gabsi*, Rim Drira†, Henda Hajjami Ben Ghezala‡
RIADI Laboratory. National School of Computer Sciences
University of Manouba,
la Manouba, Tunisia
*Email: *hamdi.gabsi@ensi-uma.tn, †rim.drira@ensi-uma.tn, ‡henda.benghezala@ensi-uma.tn*

***Abstract*—Cloud computing offers several service models that change the way applications are developed and deployed. In particular, Infrastructures as a Service (IaaS) has changed application deployment as apart from cost savings, it removes the confines of limited resources' physical locations and enables a faster time-to-market. Actually, a huge number of IaaS providers and services is becoming available with different configuration options including pricing policy, storage capacity, and computing performance. This fact makes the selection of the suitable IaaS provider and the appropriate service configuration time consuming and requiring a high level of expertise. For these reasons, we aim to assist beginner cloud users in making educated decisions and optimized selection with regard to their applications' requirements, their preferences, and their previous experiences. To do so, we propose a hybrid approach merging both Multi-Criteria Decision Making Methods and Recommender Systems for IaaS provider selection and services configuration. Moreover, we propose a service consolidation method to optimize the selection results by improving the resources' consumption and decreasing the total deployment cost. Our solution is implemented in a framework called IaaS Selection Assistant (ISA); its effectiveness is demonstrated through evaluation experiments.**

Keywords- IaaS services selection; Services Consolidation; Cost Optimization; Recommender Systems; Multi-Criteria Decision Making.

I. INTRODUCTION

In this research paper, we propose a hybrid approach for personalized and optimized IaaS services selection based on our previous work [1].

The total market value for public cloud infrastructure services, according to a report from the Analytical Research Cognizance [2], is forecast to reach 775 million dollars by 2019, up from 366 million dollars in 2015. One of the greatest benefits of IaaS platforms is the elasticity of a shared pool of configurable computing resources in response to the user's requirements. With the mature of the IaaS landscape, providers vary notably in terms of the services, features, and pricing models they offer. Due to this diversity, selecting the appropriate IaaS provider becomes a challenging task. In fact, each IaaS provider offers a wide range of services, which must be appropriately selected and correctly configured. This fact leaves users in the agony of choice and leads to a steep documentation curve to compare IaaS providers and their services. Thus, it is crucial to assist cloud users during the selection process.

In this context, several works such as [3]-[4] have shown an interest to address IaaS selection issue. However, these works focused mainly on assisting IaaS services selection based on functional application requirements and Quality of Services (QoS), which we call application profile. Few studies have highlighted the importance of involving the user in the selection process by considering his preferences and his previous experiences, which we call user profile. Consequently, there is a need for a selection process centered on both user and application profiles. Moreover, the lack of a standardized framework for the representation of user requirements and selection criteria makes it difficult to compare and evaluate the relevance of IaaS service configurations offered by different providers. Thus, it is important to define clearly relevant selection criteria that should be taken into consideration to evaluate IaaS services and select the most suitable services.

In our work, the selection process is defined as a two-step strategy. The first step consists in detecting automatically suitable IaaS provider meeting user requirements and preferences. The second step consists in retrieving the suitable IaaS service configuration (Virtual Machine (VM) instance) given a specific application requirement. To do so, we propose a hybrid approach based on Recommender Systems (RS) and Multi-Criteria Decision Making Methods (MCDM).

RS are programs, which provide relevant items (e.g., movies, music, books and products in general) to a given user by predicting his/her interest in items based on his/her profile and the ratings given by other similar profiles [5]-[6]. The first step of our approach is based on recommendation techniques.

Once the suitable IaaS provider is chosen regarding the user's profile, the user needs to be assisted to handle the services selection and configuration. For us, the cloud services selection is a MCDM problem [7]-[8]. MCDM can be defined as a process for identifying items that match the goals and constraints of decision makers with a finite number of decision criteria and alternatives [8]. In our work, we consider IaaS Service selection as a MCDM problem since

users have to select a service amongst several candidates' services with respect to different criteria. We study and choose the adequate MCDM technique to assist IaaS services selection.

After identifying suitable IaaS services, we aim to optimize the application deployment cost and improve the IaaS services consumption. To do so, we propose a service consolidation method using the knapsack algorithm [9].

Therefore, this work aims to assist and optimize IaaS services selection by involving the user in the selection process and by combining RS and MCDM techniques.

The contributions of this paper can be summarized as follows:

- Defining a classification for relevant criteria that should be used during the selection process. These criteria consider both applications profiles including functional and non-functional requirements and user's profile including personal preferences, previous experiences and even lessons learned from experiences of other users.
- Presenting a new hybrid approach based on MCDM and RS techniques for IaaS provider and services selection.
- Proposing a consolidation method to increase the selected services consumption and optimize the application deployment cost.
- Implementing this approach in a framework, which we term ISA for IaaS providers and services selection.

The present work is a comprehensive extension to our previous work [1]. We present three extensions to our initial approach and demonstrate the improved framework ISA that encompasses the enhancements of our IaaS service selection process.

First, we generalize our approach to cover medium and large application profiles, which cannot be satisfied by a single IaaS service. We consider, in this context, a cloud application as a set of deployment entities, each deployment entity presents a particular functional requirement characterized by a specific configuration in terms of CPU, storage, memory and networking capacity and defines several non-functional requirements. In that respect, the user's application can be deployed on several VMs with different configurations. Each VM will be assigned to a particular deployment entity.

Second, handling medium and large applications requires improvements of our proposed selection process in order to reduce the search space and improve the overall response time of our approach while maintaining high precision. For this purpose, we propose a mapping strategy based on the workload type of each deployment entity composing the user's application and the VM configuration families proposed by IaaS providers.

Third, we optimize our selection approach to take into account the scenario where the proposed services (VMs) may be not entirely used. We need to increase the IaaS service consumption while maintaining or decreasing the total application deployment cost. Therefore, we propose a method for service consolidation using the knapsack algorithm to reach this purpose.

The remainder of this paper is organized as follows: Section II presents a motivating scenario. Section III summarizes existing IaaS service selection techniques. Section IV illustrates the proposed cloud services selection criteria. Section V details our hybrid selection approach. Section VI presents and evaluates the framework ISA. Section VII provides concluding remarks and outlines our ongoing works.

II. Motivating scenario

Let us suppose the following scenario where a recently launched company named "A" is planning to develop flexible and innovative customer-centric services to attract new customers and improve its efficiency. In order to provide these services with high efficiency and low maintenance cost, "A" plans to use IaaS services, considering the following reasons:

- Cost reduction: The maintenance cost of dedicated hardware, software, and related manpower in "A" will be highly reduced by using cloud services.
- Improvement in flexibility and scalability: IaaS services enable "A" to respond faster to changing market conditions by dynamically scaling up and down on demand.
- Faster time to market: IaaS services enable "A" to expeditiously dispose its developed services to the market.

To deploy its services, "A" looks for IaaS services. However, most of A's engineers lack expertise in cloud services field to be able to select easily and efficiently appropriate IaaS provider and services. In today's market, there are many IaaS providers. Each provider offers several services varying in QoS attributes with possibly different functional configuration such as numbers of virtual cores and memory size. In order to select appropriate IaaS services among a growing number of available services, "A" tries to compare its applications profiles (functional & non functional requirements) to IaaS providers offers. To do so, the company needs to peruse the content of each provider website and compare service offerings to decide the most suitable IaaS service with regard to its needs. This type of selection process can be more complicated as the company's requirements evolve and diversify.

Therefore, automatic IaaS services selection becomes a highly required necessity in order to entirely take advantages of cloud computing services and improve the efficiency of many companies.

III. Related Work

Several studies have addressed the selection of IaaS services. We present a classification of the recent research approaches.

A. Recommender systems

RS can be defined as programs, which attempt to recommend suitable items to particular users by predicting a user's interest in items based on related information about the users, the items and the interactions between them [5]. Generally, RS use data mining techniques to generate meaningful suggestions taking into account user's preferences. Many different approaches using RS have been developed to deal with the problem of cloud services selection.

Zhang et al. [10] have offered a cloud recommender system for selecting IaaS services. Based on the user's technical requirements, the system recommends suitable cloud services. The matching between technical requirements and cloud services features is based on a cloud ontology. The proposed system uses a visual programming language (widgets) to enable cloud service selection.

Zain et al. [6] propose an unsupervised machine learning technique in order to discover cloud services. The authors classify cloud services into different clusters based on their QoS. The main focus of this study is to offer users the option of choosing a cloud service based on their QoS requirements.

B. MCDM-based approaches for cloud service selection

The MCDM approach is defined as a process for specifying items that best fit the goals and constraints of decision makers with a finite number of decision criteria and alternatives [11]. Several MCDM methods are used for cloud service selection such as the analytic hierarchy process/analytic network process (AHP/ANP) [12], Multi-Attribute Utility Theory (MAUT) [13], and Simple Additive Weighting (SAW) [11].

Chung et al. [14] used the ANP for service selection. They suggest a set of high level criteria for cloud service selection and use a survey of CIO, CEO, and ICT experts to determine the importance of each criterion.

Lee et al. [15] proposed a hybrid MCDM model focused on IaaS service selection for firms' users that are based on balanced scorecard (BSC), fuzzy Delphi method (FDM) and fuzzy AHP. BSC is used to prepare a list of decision making factors. FDM is used to select the list of an important decision-making factors based on the decision makers' opinion (using a questionnaire) and FAHP is used to rank and select the best cloud service. This work's focus is on the migration of the whole company ICT to cloud based on a set of general cloud service features.

Zia et al. [8] propose a methodology for multi-criteria cloud service selection based on cost and performance criteria. The authors present this selection problem in a generalized and abstract mathematical form. Table I illustrates the mathematical form. The service selection process is fundamentally a comparison between the vector service descriptor D against all rows of the decision matrix followed by the selection of the services whose description vector best matches with the user's requirement vector.

Table I. Problem formalization [8]

Mathematical form	Description
Services set	S_1, S_2, ..., S_n A set of services contains all the service offerings from, which the user (decision maker) will select the suitable service with regard to his requirements. a service is to be selected by the user (decision maker).
Performance criteria set	C_1, C_2,..., C_n A set of values where C_i represents a criterion that may be a useful parameter for service selection.
Performance measurement functions set	To each criteria C_i there corresponds a unique function f_i, which when applied to a particular service, returns a value p_i that is an assessment of its performance on a predefined scale.
Service descriptor (vector)	A row vector D_i that describes a service S_i, where each element d_j of D_i represents the performance or assessment of service S_i under criteria C_j. Performance criteria must be normalized to eliminate computational problems resulting from dissimilarity in measurement units. The normalization procedure is used to obtain dimensionless units that are comparable.
Decision matrix	The service descriptor vectors D_i can be combined to form the decision matrix where each value is the evaluation of the service s_i against the criteria c_j.
User requirement criteria vector	A vector R where each value r_i is the user's minimal requirement against a criteria c_j. These values must be normalized as the vector service descriptor.
User priority weights vector	A vector W where each value w_i is the weight assigned by a user to criteria. c_i

Table II summarizes the most used approaches by identifying the approach's input, the approach's output and the application areas.

The above-mentioned research studies did not fail to take into consideration the application's functional requirements. However, they present two main shortcomings; (i) they do not accommodate the user' preferences in the decision making and (ii) they handle every application deployment as a new case without taking into account the results of similar previous experiences.

TABLE II. SELECTION APPROACHES

Domain	Method	Input	Output	Application	Literature
Multi-criteria decision-making (MCDM)	SAW	Subjective assessment of relative importance of criteria.	Evaluation value of alternatives.	Applied when requiring low decision accuracy.	[11][8][16]
Multi-criteria optimization	Matrix factorization	Different types of data of interest to users and represented by matrix .	QoS estimation and a set of recommended services.	Applied to a problem that involves different types of data and has missing entries.	[17][18]
Logic based matching approach	First-order logic	Service description and user requirements.	Matched services	Applied to filter out unmatched services to reduce computation complexity.	[11][19]
Recommender System	Collaborative filtering	User's profile	Recommended items	Applied to find personalized recommendations according to user's profile.	[4][10][20]

To the best of our knowledge, no specific research study has taken into account both the user's profile and the application's requirements. Consequently, there is a need for a structured selection process where clearly both selection criteria are defined and used.

IV. CLOUD SERVICES SELECTION CRITERIA

Specifying clear selection criteria presents crucial importance in order to recommend the relevant IaaS services. Our purpose is to clearly identify these criteria and take them into account to personalize the selection process according to the user's profile and respond to his application requirements. Thus, we classify selection criteria into three categories. The first category is the application's profile, which includes functional and non- functional requirements. The second category is the user's profile, which represents user's personal preferences and previous experiences. The third category is the previous experiences of other users with their ratings. Figure 1 illustrates our proposed selection criteria.

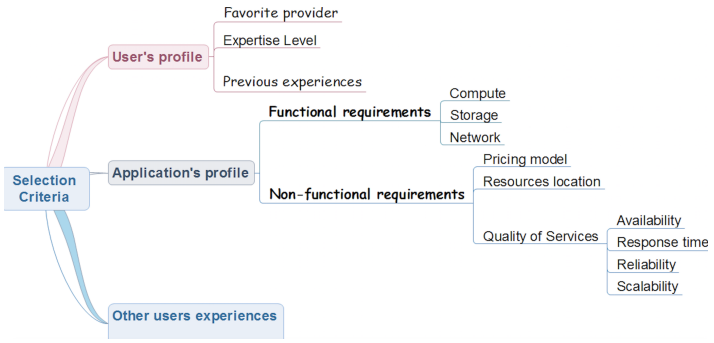

Figure 1. Selection Criteria

As shown in Figure 1, the selection criteria are classified as the following:

- **Application's profile:** the application's profile defines the functional and non-functional application requirements.
  In our context, we consider that a cloud application is a set of deployment entities each deployment entity has specific functional and non-functional requirements. We define the application profile as a set of all deployment entities' requirements. The functional requirements contain the following specifications:
  - Storage: represents storage needs in terms of memory space.
  - Network: represents connection needs and network usage.
  - Compute: gathers calculation needs and the virtual machine's capacity.

  Non-functional requirements include pricing models, the quality of services (QoS) and the resources location.
  - The pricing model: depends on the user's estimated budget. The pricing model can be evaluated per hour or per month. Also, it can be on demand, reserved or bidding.
  - QoS: we focus on the response time and availability. The availability is the time ratio when the service is functional to the total time it is required or expected to function in.
  - Resources location: The user can precise his nearest resources location because it is important to take into account the proximity when selecting the cloud infrastructure services. According to [19], during the interaction between the users and servers, there is a strong inverse correlation between network distance and bandwidth. Thus, factoring the proximity into the selection of IaaS services can significantly reduce the client's response time and increase the network bandwidth.
- **User's profile:** it includes user's favorite providers, expertise level in cloud and previous experiences. A favorite provider can be chosen based on previous successful experiences using this provider. We take this choice into consideration while identifying the appropriate cloud provider meeting user's requirements. In our case, the user can specify one or multiple favorite providers. The user's expertise level can be: beginner, intermediate or expert. The weight of a user's previous

experience in our knowledge base increase with his level of expertise and experience in order to enhance our recommendations relevance. A previous experience contains the selected IaaS provider, the deployed application profile and a rating out of 5 presenting feedback and an evaluation of this experience. We suppose that evaluating ratings are trustworthy and objective.

- **Previous users experiences:** The more the knowledge base of our recommender system is rich, the more recommendations will be relevant. Therefore, previous users experiences, which include the deployed application's profile, the selected IaaS provider and the evaluating rating will improve the accuracy of our recommendations.

Based on the selection criteria, more precisely the application profile, we propose to optimize the search space. Indeed, we suppose that each deployment entity is characterized by a specific workload type. According to Singh et al. [21], cloud workloads can be defined based on four main low-level measurable metrics that when adjusted can affect the workloads' performance. These metrics are the CPU cores, the memory size (RAM), the networking capacity and the storage size, which present respectively the compute, the network and the storage requirements.

We propose a mapping between the workload type of the application deployment entities and the VM configuration families proposed by the IaaS providers. The above-mentioned metrics will be used as a high-level interface that maps the workload type onto a set of candidate IaaS services. Indeed, the workload type can be automatically extracted based on the weight assigned to each metric, for instance, computation-intensive workload is characterized by a higher weight for the CPU metric. In the case that the weights given by the users are equal or insignificantly different, the workload type is defined as general. Thus, workload types are easily identifiable. If we can manage to map these workloads type onto specific categories of IaaS services, then the service selection will become more efficient by decreasing the search space and improving the overall response time. For this purpose, our mapping strategy is based on identifying IaaS service categories disposed by cloud provider, then, establishing the relation between the service categories and the workload type.

Cloud providers dispose IaaS services in different categories with various configurations in terms of CPU, storage, memory and networking capacity. We conduct that most cloud providers classify their services into the following categories based on VM configurations: compute optimized, memory optimized, storage optimized and general purpose.

These categories are identified to offer better performance with respect to a specific workload types (such as computation-intensive or memory-intensive). Thus, It is obvious that the relation between the workload type and the IaaS services configurations are based on the service category. More precisely, the computation-intensive workload type is mapped to IaaS services of the compute optimized category, the memory-intensive workload type is mapped to the memory optimized category, the storage-intensive workload type is mapped to the storage optimized category, and general workload type is mapped to general purpose category.

V. Hybrid approach for IaaS services selection based on RS & MCDM

The selection of IaaS provider and services configuration is a complex issue. To tackle this issue, we propose a two steps selection process. The first step focuses on selecting the IaaS provider based on RS approach, which is the collaborative filtering. The purpose of this step is to reduce the number of inappropriate IaaS provider, which may not interest the user. The second step concerns the configuration of services within the selected provider from the first step. It's based on the SAW algorithm, which is a MCDM method. Our proposed approach shows how MCDM techniques and RS are complementary in order to involve both technical and personal aspects in the selection process.

Figure 2 illustrates our proposed approach.

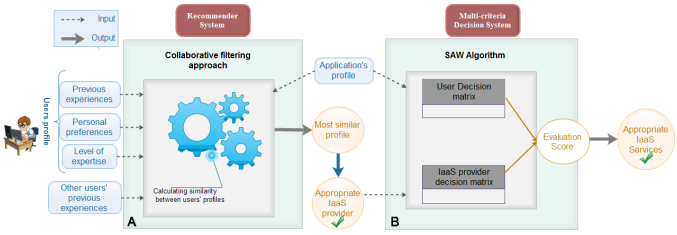

Figure 2. Hybrid approach for IaaS services selection

A. Recommender System

The first step aims to take into consideration the user's preferences, previous experiences and expertise level during the selection process. In our approach, we use the collaborative filtering algorithm also known as k-NN collaborative filtering. This recommendation algorithm bases its predictions on previous users experiences and their profiles. The main assumption behind this method is that other users ratings can be selected and aggregated so that a reasonable prediction of the active user's preferences is deduced.

To recommend the IaaS provider meeting the user's profile, first, we select the users profiles, which have the same or higher expertise level than the active user "A". For instance, if "A" has the expertise level intermediate, then, from our knowledge base, we select a first list named "list 1" of users profiles, which are intermediate or expert and their rated experiences.

Second, among the high rated previous experiences of "list 1", we select those, which are based on the favorite providers of "A" in order to create a second list named "list 2".

Third, among these experiences, "A" can refine "list 2" by identifying experiences that have similar workload types to his application's profile workload. We obtain "list 3". Indeed, we aim by these three steps verifying if "A" favorite providers can be suitable for "A" application profile. Otherwise, we skip the second step to apply the third step on "list 1".

Then, a rating $R_{(A,f_i)}$ is calculated for each one of candidate providers f_i of list3. $R_{(A,f_i)}$ is calculated as below:

$$R_{(A,f_i)} = \frac{\sum_{j=1}^{n} w_{(A,j)}(v_{j,f_i} - \overline{v_j})}{\sum_{j=1}^{n} |w_{(A,j)}|}$$

where n is the number of identified users' profiles of "list 3", $w_{(A,j)}$ is the similarity between the profile of "A" and the identified users profiles j of "list 3", v_{j,f_i} is the rate given by the user j to the provider f_i, $\overline{v_j}$ is the rating's average given by the user j to the favorites providers of "A". We calculate similarity between "A" and the identified users using cosine similarity.

$$w_{(A,j)} = \frac{\sum_{k=1}^{n} v_{A,k} * v_{j,k}}{\sqrt{\sum_{k=1}^{n} v_{A,k}^2 \sum_{k=1}^{n} v_{j,k}^2}},$$

where the sum on k is the set of providers for which "A" and the selected users in" list 3" both assigned a rating, $v_{j,k}$ is the rate given by the user j to the provider k.

Finally, we propose to "A", the set of providers sorted according to the rate calculated, thus the active user can select one provider.

B. Multi-Criteria Decision Making Selecting the Cloud Instances

Once the IaaS provider is selected, the second step consists in determining the suitable IaaS service for each deployment entity.

Several and conflicting criteria have to be taken into account when making a service selection decision. No single service exceeds all other services in all criteria but each service may be better in terms of some of the criteria. Since users have to decide which service to select amongst several candidates services with respect to different criteria, we consider IaaS Service selection as a MCDM problem.

Among MCDM methods, we use the SAW method also known as weighted linear combination or scoring methods. It is based on the weighted average of different criteria.

In our case, the number of service configuration components such as CPU cores and memory size scale linearly in most services configurations. Hence, a linear model is suitable for this kind of problem. The basic assumption being that there is a correlation of identity between real-world cloud instance performance and the underlying low-level specification of the hardware, which is specified on the cloud providers websites. Hence, we want to map the performance of the IaaS service to the right deployment entity using a simple linear model. The purpose of using SAW method in our approach is to respond exactly to the application's profile.

To do so, first, the user introduces functional requirements for each deployment entity; compute requirements (e.g., virtual Central Processing Unit (vCPU)), memory requirements (e.g., RAM size) storage requirements (e.g., hard drive's size), network requirements (e.g., throughput and bandwidth).

Second, for each specified requirement the user assignes a particular weight presenting its importance.

Third, based on the weight assigned to each requirement the workload type of the deployment entity is deducted and a set of candidate services is identified. The user inserts the QoS required (e.g., response time and availability) and the pricing model.

To be able to apply the SAW algorithm, we need to formalize our decision problem. For that, we define a decision matrix related to the user. In parallel an analogous decision matrix is defined for the IaaS provider selected in the first step. The decision matrix is a combination of service descriptor vectors. Each service descriptor vector represents the performance of a service under a particular criterion. These criteria represent functional and non-functional requirements for the user. Table III demonstrates an extract form of the decision matrix related to Azure Microsoft [22].

TABLE III. EXTRACT OF DECISION MATRIX FOR MICROSOFT AZURE (VIRTUAL MACHINE)

Service	VCPU	RAM	Hard Drive's size	Cost
A0	1	0.75 GB	19 GB	$0.02/h
A1	1	1.75 GB	224 GB	$0.08/h
A2	2	3.5 GB	489 GB	$0.16/h
A3	4	7 GB	999 GB	$0.32/h
A4	8	14 GB	2039 GB	$0.64/h
A5	2	14 GB	489 GB	$0.35/h
A6	4	28 GB	999 GB	$0.71/h

The SAW algorithm is based on the calculation of one score to each alternative (an alternative in our case is an IaaS service offered by the selected IaaS provider). According to the following SAW formula, the alternative score is calculated as (A_i)=$\sum w_j v_{ij}$, where w_j is the alternative's weight i according to criterion j and v_{ij} its performance. The alternative with the highest score will be suggested. By applying this formula, the recommended IaaS service will automatically be the most performing service, because

it has the highest performing values in the decision matrix (highest number of vCPU, largest hard drive's size, highest cost, etc.). However, this does not entirely meet the user's requirements, because, he/she must not necessarily select the most performing IaaS service, which will evidently have the highest cost. Whereas, he/she should select the service, which meets exactly his/her requirements in order to pay the minimum possible cost. To solve this, we proceed as follows:

- First, we create a decision matrix representing each deployment entity's functional and non-functional requirements. Then, we determine for each service descriptor vector, the absolute value of the difference between its criteria performance and those of the service descriptor vector related to the IaaS provider. In this way, we will have significant values. In fact, low criteria values mean that they accurately match the user's requirements.
- Second, we calculate the score for each alternative using SAW algorithm. Yet, to be able to do so, we need to modify each criterion's weight to get significant results. Indeed, we have previously mentioned that a low criterion's value means that it may interest the user, if this criterion has a high weight, the multiplication of its weight by its value gives a low score. Therefore, this alternative will be considered as unimportant, yet this is not the case. To solve this problem we take the dual of each weight, meaning that, the subtraction of 1 by the weight's value given by the user. Then we normalize each weigh by dividing on the sum of the weights. Thus, we ensure that the weight values are between 0 and 1 and the sum is always equal to 1. Consequently, one low weight value indicates major importance of a given criterion. Therefore, we can calculate the score for each alternative using the SAW algorithm. The most relevant alternative (IaaS service) will incontrovertibly have the lowest score.

To illustrate this, we propose our personalized SAW algorithm 1. We suppose that the user has introduced his/her decision matrix $UserMat[i][j]$ as well as the weights of each criterion $Weight[j]$. In addition, we suppose that we have the decision matrix $ProvMat[i][j]$ containing IaaS services offered by the IaaS provider. In the decision matrix $UserMat$, $UserMat[i][j]$ represents the IaaS service i under the criterion j.

$$UserMat = \begin{bmatrix} u_{00} & \dots & u_{0n} \\ \vdots & \ddots & \vdots \\ u_{n0} & \dots & u_{nm} \end{bmatrix}$$

The personalized SAW algorithm gives as output, the index i representing the adequate cloud service i in the decision matrix.

Algorithm 1 Personalized SAW Algorithm

Require: $Weight[i] \neq 0$
  $Min = 0$
  for $int\ i\ from\ 0\ to\ n$ **do**
    for $int\ j\ from\ 0\ to\ n$ **do**
      $Sub[i][j] = abs(ProvMat[i][j] - UserMat[i][j])$
    end for
  end for
  for $int\ j\ from\ 0\ to\ m$ **do**
    $DualWeight[j] = 1 - Weight[j]$
    $Normalize(DualWeight[j])$
  end for
  for $int\ i\ from\ 0\ to\ n$ **do**
    $Score[i] = 0$
    for $int\ j\ from\ 0\ to\ m$ **do**
      $Score[i] = Score[i] + Sub[i][j] * DualWeight[j]$
    end for
  end for
  for $int\ i\ from\ 0\ to\ n$ **do**
    if $Score[i] < Min$ **then**
      $Min \leftarrow Score[i]$
      $Index \leftarrow i$
    end if
  end for
  return i

C. Service consolidation

Identifying suitable IaaS services (i.e., VMs in our case) for each deployment entity does not ensure that the VM will be entirely used. In a typical scenario, the selected VM is underutilized [23]. To increase the resource utilization, we aim to integrate as many deployment entities as possible to be assigned to each selected service, thus decreasing the number of required services for application deployment. The final configuration must support all the requirements of the application and the preferences of the user with respect to the service performance and price.

To do so, we proceed as follows; first, we start with the largest service S_L, which has the highest performance in the list of proposed services. We use the price as an indicator of service capacity. Second, we accommodate as many deployment entities as possible in this service with respect to its performance (i.e., the service performance can respond to the added deployment entities). Third, we upgrade the service by choosing the next higher performance of the VM instance of the same family as the service S_L, then we consolidate more deployment entities in the service. If the new service's configuration (i.e., the upgraded service) has an equal or lower price than the earlier configuration of all consolidated services, the upgrade is positive and acceptable. We continue the same process for the remaining deployment entities of the application.

To consolidate deployment entities in a service, we cast consolidation into the optimization knapsack problem [9]. Indeed, the knapsack problem is a combinatorial optimization problem. Given a set of items, each item has a weight and a value, the knapsack problem consists in identifying the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

First, let us formalize the knapsack's problem in our context:

- The knapsack is the largest service, which is not entirely used
- The items are the deployment entities
- The weight is the cost of each single service assigned to a deployment entity

Second, to solve the knapsack problem and handle the challenge of consolidating multiple deployment entities into services, a greedy approximation algorithm [9] is used. The greedy algorithm is an algorithmic paradigm that follows the problem solving heuristic of making the locally optimal choice at each stage with the intent of finding a global optimum. It iteratively makes one greedy choice after another, which reduce each given problem into a smaller one and approximate a globally optimal solution in a reasonable amount of time. In our case, the greedy choice consiste in selecting in each iteration the largest deployment entity among the non-integrated entities.

We detailed our consolidation approach in Algorithm 2. Service consolidation has advantages and disadvantages. Consolidating deployment entities can reduce the network overhead and increases the application's performance. However, service consolidation can cause several challenges related to fault tolerance.

VI. ISA: A Framework for IaaS Selection Assistant

We conduct a set of experiments to evaluate the efficiency of our proposed approach. To do so, we develop the framework ISA by extending our previous framework. In our previous work [1], we suppose that the application profile can be satisfied by just one VM. In this evaluation, we assume that an application profile may require more than one VM. The main purpose through this evaluation is, firstly, to demonstrate that the idea of merging RS and MCDM techniques in a structured approach based on two well defined steps as explained in Section V, provides satisfactory results for several application types (i.e., medium and large applications). Secondly, we aim to validate that our approach proves to be efficient rather than using RS and MCDM techniques each independently.

The framework ISA has been designed to support different IaaS providers such as Amazon, Google and Azure

Algorithm 2 Services Consolidation Algorithm

Input: DT Set of application deployment entities
SD Set of single services assigned to each deployment entity
Initial_Application_Price
Output: Updated_deployment_entities (After consolidation)
Updated_services (After consolidation)
Begin
$Consolidation\_cost \leftarrow$ Initial_Deployment _cost
$i, j \leftarrow 0$
$S_L \leftarrow S_k$, where S_k is the largest in SD)
Update (SD) : $SD \leftarrow SD$ - $\{S_L\}$
Update (DT) : $DT \leftarrow DT$ - $\{Entity_i\}$, where $Entity_i$ is the deployment entity performed by the service S_L
while $(\neg Empty(SD)) \vee (i \leq nb\_services)$ **do**
  while $(\neg Empty(DT)) \vee (j \leq nb\_entities)$ **do**
    Select the largest entity $Entity_L$
    if S_L performance respond to $Entity_L$ **then**
      $Consolidate$ ($Entity_L$, S_L)
      $Update$ (SD)
      $Update$ (DT)
    else
      $S'_L \leftarrow$ Upgrade (S_L)
      Calculate_New_cost
      if $New\_cost \leq Consolidation\_cost$ **then**
        $Consolidation\_cost \leftarrow New\_cost$
        $Consolidate$ ($Entity_L$, S'_L)
        $Update$ (SD)
        $Update$ (DT)
      end if
    end if
    $Entity_L \leftarrow Entity_{L+1}$ {Next_ Largest_entity $\in DT$}
    $j \leftarrow j + 1$
  end while
  $S_L \leftarrow S_{L+1}$ {Next_ Largest_service $\in SD$}
  $Update$ (SD)
  $Update$ (DT)
  $i \leftarrow i + 1$
end while
return SD, DT
End

Microsoft. It aims to guide users step by step in the selection process and propose relevant services.

For this evaluation, we have used Eclipse Modeling Framework, Java Platform Enterprise Edition (JEE) and Mahout eclipse framework [24]. We conduct experiments on 20 real users (PhD students).

We define the experiments' conditions as follows:

- Supported IaaS provider: Amazon, Google, Microsoft Azure
- Number of users: 20
- Number of items (IaaS services): 45
- Active user's profile:
  - Favorite provider: Amazon
  - Expertise level: Beginner
  - Previous experiences: 0
- Active user's application profile: It is defined in Table IV
- The non-functional requirements are defined as follows:
  - QoS: QoS is defined in Table IV
  - Pricing model: Per hour
  - Resource Location: US regions (e.g., US-West, US-East, etc.)

According to the weights given by the user, we assign for each deployment entity the appropriate workload type. Table V illustrates the assigned workload types.

TABLE V. WORKLOAD MAPPING

Deployment Entities	Workload Type
E1	General Purpose
E2	Storage Optimized
E3	General Purpose
E4	Compute Optimized

We define in Table VI the decision matrix "$ProvMat[][]$" used by the personalized SAW algorithm of our approach. For the sake of brevity, we present in Table VI six configuration models of Virtual Machines instances provided by Amazon [25]. Each value in Table VI is verified and identified from cloud provider's official web site. We carry out simulations and evaluations from two steps.

The first step consists on evaluating the effectiveness of ISA using the recall (R), the precision (P), the Top-k precision (P_k) and the R- precision (P_r) metrics. In this context, the precision evaluates the capability of the our framework to retrieve top-ranked IaaS services that are most relevant to the user need, and it is defined to be the percentage of the retrieved IaaS services that are truly relevant to the users requirements. The recall evaluates capability of the system to get all the relevant services. It is defined as the percentage of the services that are relevant to the user requirements.

Formally, we have;

$$P = \frac{|S_{Rel}|}{|S_{Ret}|} \qquad R = \frac{|S_{Rel}|}{|Rel|}$$

$$P_k = \frac{|S_{Rel,k}|}{k} \qquad P_r = P_{|Rel|} = \frac{|S_{Rel,Rel}|}{|Rel|}$$

where Rel denotes the set of relevant IaaS services, S_{Ret} is the set of retuned services, S_{Rel} is the set of of returned relevant services and $S_{Rel,k}$ is the set of relevant services in the top k returned services. Among the above metrics, P_r is considered to most precisely capture the precision and ranking quality of the framework. We also plotted the recall/precision curve (R-P curve). An ideal selection framework has a horizontal curve with a high precision value; an inappropriate framework has a horizontal curve with a low precision value. The R-P curve is considered by the (Information Retrieval) IR community as the most informative graph showing the effectiveness of a selection framework [26].

We evaluated the precision of the retrieved services for each deployment entity, and report the average Top-2 and Top-5 precision. To ensure the top-5 precision is meaningful, we ensure that ISA returns a total of 20 services per application profile. The Figure 3 illustrates the results. The top-2 and top-5 of ISA for the deployment entities E1, E2, E3 and E4 are respectively 98% for the Top-2 retrieved services and 80%, 60%, 80%, 80% for the Top-5 retrieved services. In order to interpret our results and illustrate the overall performance of ISA, we plot the average R-P curves for different applications profiles. As mentioned previously, a good selection framework has a horizontal curve with a high precision value. Typically, precision and recall are inversely related, ie. as precision increases, recall falls and vice-versa. A balance between these two needs to be achieved by a selection framework. As illustrated by the Figure 4, for a recall average equals to 0.68 we have 0.87 as precision average value. In fact, as an example, for the active user's application profile defined in Table IV, ISA returns a total of 20 services i.e., $|S_{Ret}| = 20$, for each deployment entity, we have the following precision values; $\frac{18}{20}, \frac{17}{20}, \frac{17}{20}, \frac{18}{20}$. We obtain a precision average $P = 0.87$. As a recall value, we have, for each deployment entity, $\frac{18}{25}, \frac{17}{26}, \frac{17}{25}, \frac{18}{26}$, we obtain a recall average $R = 0.68$.

It is worth pointing out that in some cases, depending on particular requirements, a high precision at the cost of recall or high recall with lower precision can be chosen. Thus evaluating a selection framework must be related to the purpose of the selection and the search process. In our case a compromise between the recall and the precision values is necessary. Therefore, we can announce that ISA provides accurate results for IaaS services selection.

TABLE IV. APPLICATIONS PROFILES

Application profile								
Functional requirements							QoS	
Deployment Entities	Compute			Storage	Network		Response time	Availability
	vCPU	CPU events/s	RAM	Hard drive's size	Bandwidth	Throughput		
E 1 Weights	0.25		0.3	0.25	0.2		0.5	0.5
E 1 Values	2	$6< v \leq 12$	8	60	4	-	$v \leq 900$	90%
E 2 Weights	0.2		0.2	0.5	0.1		0.7	0.3
E 2 Values	2		10	400	2	42	≤ 900	95%
E 3 Weights	0.25		0.25	0.25	0.25		0.5	0.5
E 3 Values	2	$10 < v \leq 20$	8	30	6	-	≤ 900	95%
E 4 Weights	0.5		0.3	0.1	0.1		0.8	0.2
E 4 Values	16	$50 < v \leq 80$	32	300	10	60	700	95%

TABLE VI. AMAZON DECISION MATRIX [25]

Model	Family	vCPU	CPU Cred-its/hr	RAM GB	Hard drive GB	Bandwidth Gbit s^{-1}	Throughput Mbit s^{-1}	Price h^{-1}	Response time ms	Availability
t2.nano	General purpose	1	3	0.5	30	-	4	$0.0058	63	99%
c5d.4xlarge	Compute optimized	16	81	32	400	5.5	435.7	$0.768	22	99%
m5a.large	General purpose	2	36	8	30	3.12	256	$0.086	53	99%
t2.large	General purpose	2	36	8	30	-	42	$0.0928	50	99%
i3.large	Storage optimized	2	54	19.25	475	-	53.13	$0.156	42	99%
c5d.xlarge	Compute optimized	4	54	8	100	3.5	437.5	$0.192	31	99%

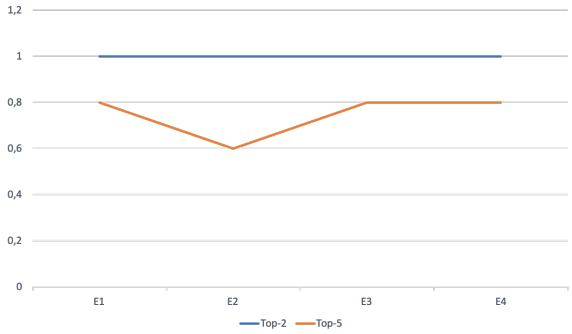

Figure 3. Top-k precision for retrieved services

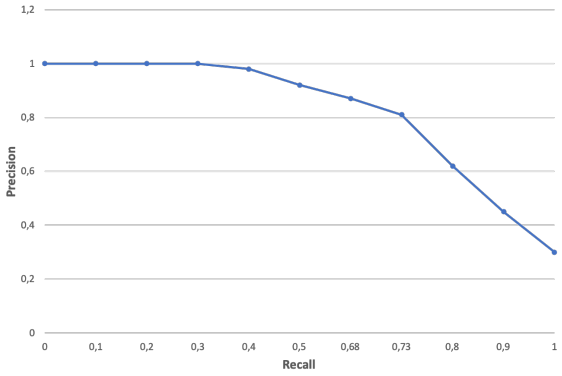

Figure 4. R-P Curves of ISA

The second step of our evaluation consist on comparing our framework to classic RS based on CF technique. Although the number of users and items is relatively small compared to commercial RS, it proves to be sufficient for the purpose of these experiments. For each deployment entity, we present the predicted ratings for each deployment entity described in Table IV.

As illustrated in Figures 5,6, 7, and 8 the highest predicted ratings given by our approach to the deployment entities E_1, E_2, E_3 and E_4 are, respectively, 0.8379, 0.8979, 0.9039, 0.9798. The recommended IaaS services are, respectively, m5d.large i3.large, m5a.large and c5d.2xlarge. For clarity and visibility purposes, we did not display all instances' predicted ratings of Tables III and VI.

The metrics used to evaluate our approach are the Root-Mean Square Error (RMSE) and The Normalized Discounted Cumulative Gain (NDCG).

The RMSE is a metric widely used to evaluate predicted ratings [27]. It represents the sample standard deviation of the differences between predicted values and expected values. RMSE is the square root of the average of squared

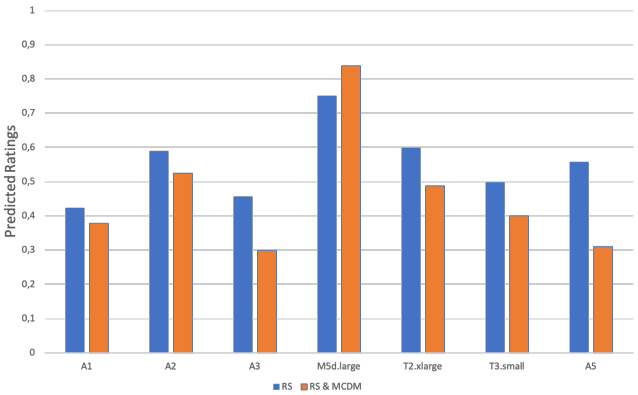

Figure 5. Predicted ratings for the deployment entity E_1

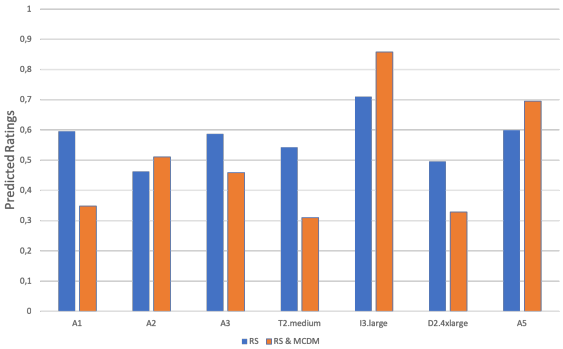

Figure 6. Predicted ratings for the deployment entity E_2

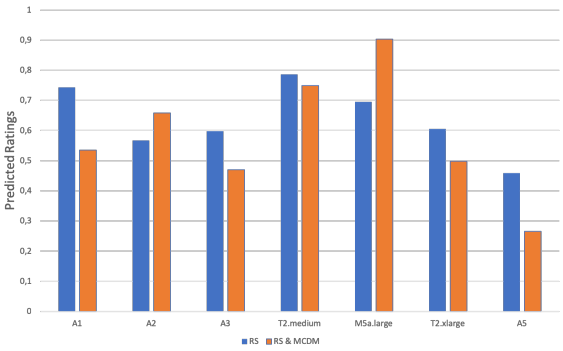

Figure 7. Predicted ratings for the deployment entity E_3

errors.

$$RMSE = \frac{\sqrt{\sum_{i=1}^{n}(p_{A,i} - \widehat{p}_{A,i})^2}}{N}$$

where $p_(A, i)$ is a predicted value by user "A" for item i, $\widehat{p}_{A,i}$ is the expected value of user "A" for item i, and N is the number of predicted values. In order to be able to calculate RMSE values, we assume that users introduce their expected rating values.

The Normalized Discounted Cumulative Gain (NDCG) is

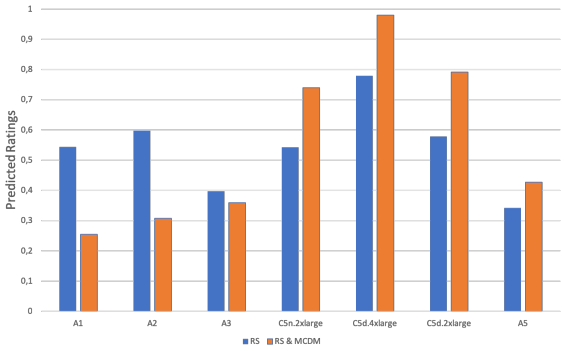

Figure 8. Predicted ratings for the deployment entity E_4

a measure of ranking quality. NDCG is defined as

$$NDCG_N = \frac{DCG_N}{IDCG_N}$$

where DCG_N and $IDCG_N$ are the Discounted Cumulative Gain (DCG) of top-N items of a predicted ranking and the ideal ranking, respectively. DCG_N is calculated by

$$DCG_N = \sum_{i=1}^{N} \frac{2^{(rel_i)} - 1}{log_2(i+1)}$$

where rel_i is the value of the item at position i of a ranking and $IDCG_N$ is calculated by

$$IDCG_N = \sum_{i=1}^{REL} \frac{2^{(rel_i)} - 1}{log_2(i+1)}$$

where REL represents the list of relevant items (ratings ≥ 0.5). The value of NDCG is between 0 and 1. where a larger value means a better ranking, and 1 implies the ideal ranking.

We illustrate the result of comparing the CF technique to our work in Table VII.

TABLE VII. RMSE & NDCG AVERAGE

Deployment Entities	RS		RS & MCDM	
	RMSE	NDCG	RMSE	NDCG
E1	0.041	0.571	0.032	0.71
E2	0.052	0.43	0.034	0.81
E3	0.033	0.62	0.038	0.76
E4	0.045	0.65	0.031	0.79
Average	0.04275	0.567	0.033	0.767

When conducting the CF approach, we obtained respectively 0.04275 and 0.567 as RMSE and NDCG average. However, the RS & MCDM approach gave us 0.033 and 0.767 as RMSE and NDCG average as illustrated in Figure 9. So, in terms of RMSE (i.e., 0.04275 vs. 0.033), the merging of MCDM & RS performs better than RS only. In terms of NDCG (i.e., 0.567 vs. 0.767), RS & MCDM present better result than the CF approach.

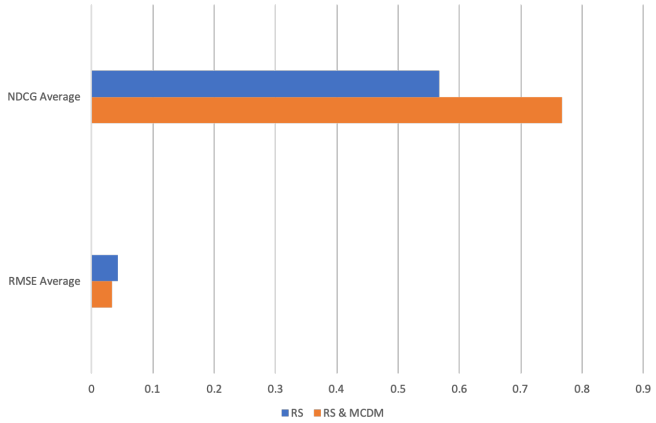

Figure 9. RMSE & NDCG Average

It is worth pointing that the use of CF algorithm only conducts to calculate predicted ratings for all items in our knowledge base, which can be time consuming. However, by applying the step one of our approach we can reduce the number of candidate services by providing only services related to the selected IaaS provider. In addition, the selection of IaaS services using CF algorithm will be associated with previous users experiences in our knowledge base. Although we identify the most similar users, their application profiles must be more or less different to the active user application profile. Consequently, the predicted IaaS services are less accurate. In conclusion, these experiments show that our approach performs better than using RS only.

After identifying suitable IaaS services, we aim to optimize the application deployment cost. To do so, we apply our consolidation algorithm to integrate potential services. It is worth pointing that the cost of the recommended services is estimated to 2.123 $ per hour ($0.113\$ + 0.156\$ + 0.086\$ + 1.768\$$). We consider the result of the consolidation algorithm is acceptable if it provides a cost $\leq 1.123\$$.

As described in Section V, the first step of the consolidation algorithm is identifying the largest service recommended by our framework, which is c5d.4xlarge. Second, we verify if this service can perform the largest deployment entities (E_2) added to its assigned deployment entity (E_4), which is not the case (the performance evaluation is based on parallel computing [28]). We continue applying the steps of our Algorithm 2 to conclude that the deployment entities E_1 and E_3 can be consolidated and performed by the upgraded service c5d.xlarge. The total cost for the application dropped to $1.116\$/h$ (compared to the nonconsolidated services). Thus, we consider that the consolidation algorithm provides acceptable results that optimized the application deployment cost.

Following the process of service selection using our proposed framework shows the feasibility and the effectiveness of our approach in IaaS service selection.

VII. Conclusion

The motivation of our research stems from the need to assist users in selecting appropriate cloud infrastructure services. Although the market growth provides economic benefits to the users due to increased competition between IaaS providers, the lack of similarity with respect to how IaaS services are described and priced by different providers makes the decision on the best option challenging. The decision also needs to consider the user's preferences over different features. To raise this challenge, we proposed a new hybrid approach based on MCDM and RS techniques that transform the IaaS services selection from an ad-hoc task that involves manually reading the provider documentation to a structured and guided process. By generalizing our previous work [1], we take into consideration medium and large application profiles, which cannot be fulfilled by a single IaaS service. Thus, several services are recommended to satisfy the user requirements. In order to to improve the selected services' utilization and optimize deployment costs we introduce a consolidation method inspired from the knapsack algorithm.

Although we believe that our approach leaves scope for a range of enhancements, yet it provides suitable results. The experimental evaluation conducted against typical RS technique highlights the main benefits of the proposed approach.

For our ongoing works, we are focusing on studying the relation between the deployment entities of the user's application. In fact, the deployment of an application's component as independent deployment entities entails communications between these entities. This communication may introduce new networks requirements and add several constraints such as data flow management.

References

[1] H. Gabsi, R. Drira, and H. H. B. Ghezala, "Personalized iaas services selection based on multi-criteria decision making approach and recommender systems," *International Conference on Internet and Web Applications and Services (ICIW 2018), IARIA, Barcelona, Spain*, pp. 5–12, 2018, ISBN: 978-1-61208-651-4 ISSN: 2308-3972.

[2] "Analytical Research Cognizance," 2019, URL: http://www.arcognizance.com [accessed: 2019-01-23].

[3] M. Eisa, M. Younas, K. Basu, and H. Zhu, "Trends and directions in cloud service selection," *IEEE Symposium on Service-Oriented System Engineering*, 2016, ISBN: 978-1-5090-2253-3.

[4] S. Soltani, K. Elgazzar, and P. Martin, "Quaram service recommender: a platform for iaas service selection," *International Conference on Utility and Cloud Computing*, pp. 422–425, 2016, ISBN: 978-1-4503-4616-0.

[5] J. Lu, D. Wu, and G. Zhang, "Recommender system application developments: A survey," *Decision Support Systems*, vol. 74, pp. 12–32, 2015.

[6] T. Zain, M. Aslam, M. Imran, and Martinez-Enriquez, "Cloud service recommender system using clustering," *Electrical Engineering, Computing Science and Automatic Control (CCE)*, vol. 47, pp. 777–780, 2014.

[7] A. J. Ruby, B. W. Aisha, and C. P. Subash, "Comparison of multi criteria decision making algorithms for ranking cloud renderfarm services," *Indian Journal of Science and Technology*, vol. 9, p. 31, 2016.

[8] Z. Rehman, F. Hussain, and O. Hussain, "Towards multi-criteria cloud service selection," *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2013, ISBN: 978-1-61284-733-7.

[9] J. Lv, X. Wang, M. Huang, H. Cheng, and F. Li, "Solving 0-1 knapsack problem by greedy degree and expectation efficiency," *Applied Soft Computing*, vol. 41, pp. 94–103, 2016.

[10] M. Zhang, R. Ranjan, S. Nepal, M. Menzel, and A. Haller, "A declarative recommender system for cloud infrastructure services selection," *GECON*, vol. 7714, pp. 102–113, 2012.

[11] L. Sun, H. Dong, F. Khadeer, Hussain, O. K. Hussain, and E. Chang, "Cloud service selection: State-of-the-art and future research directions," *Journal of Network and Computer Applications*, vol. 45, pp. 134–150, 2014.

[12] C. JatothG and U. Fiore, "Evaluating the efficiency of cloud services using modified data envelopment analysis and modified super-efficiency data envelopment analysis," *Soft Computing, Springer-Verlag Berlin Heidelberg*, vol. 7221-7234, p. 21, 2017.

[13] F. Aqlan, A. Ahmed, O. Ashour, A. Shamsan, and M. M. Hamasha, "An approach for rush order acceptance decisions using simulation and multi-attribute utility theory," *European Journal of Industrial Engineering,*, 2017, ISSN: 1751-5254.

[14] C. B. Do and S. K. Kyu, "A cloud service selection model based on analytic network process," *Indian J Sci Technol*, vol. 8, no. 18, 2016.

[15] ——, "A hybrid multi-criteria decision-making model for a cloud service selection problem using bsc, fuzzy delphi method and fuzzy ahp," *Indian J Sci Technol*, vol. 86, pp. 57–75, 2016.

[16] M. Whaiduzzaman, A. Gani, N. B. Anuar, M. Shiraz, M. N. Haque, and I. T. Haque, "Cloud service selection using multicriteria decision analysis," *The Scientific World Journal*, vol. 2014, p. 10, 2014.

[17] L. D. Ngan and R. Kanagasabai, "Owl-s based semantic cloud service broker," *International conference on web services (ICWS)*, pp. 560–567, 2013, ISBN: 978-1-4673-2131-0.

[18] F. K. Hussain, Z. ur Rehman, and O. K. Hussain, "Multi-criteria iaas service selection based on qos history," *International Conference on Advanced Information Networking and Applications*, 2014, ISSN: 1550-445X.

[19] Z. Li, L. OBrien, H. Zhang, and R. Cai, "On the conceptualization of performance evaluation of iaas services," *IEEE Transactions on Services Computing*, vol. 7, pp. 628 – 641, 2014.

[20] Q. Yu, "Cloudrec: a framework for personalized service recommendation in the cloud," *Knowledge and Information Systems*, vol. 43, pp. 417–443, 2014.

[21] S. Sukhpal and I. Chana, "A survey on resource scheduling in cloud computing: Issues and challenges," *Journal of grid computing*, vol. 14, pp. 217–264, 2016.

[22] "Microsoft Azure," 2019, URL: https://azure.microsoft.com/en-us/pricing/details/cloud-services [accessed: 2019-02-28].

[23] S. Soltani, P. Martin, and K. Elgazzar, "A hybrid approach to automatic iaas service selection," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 7, pp. 1–18, 2018.

[24] "Mahout Apache," 2019, URL: http://mahout.apache.org/ [accessed: 2019-02-28].

[25] "Amazon Instance Types," 2019, URL: https://aws.amazon.com/ec2/instance-types/?nc1=h_ls [accessed: 2019-02-28].

[26] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," *International conference on Machine learning*, pp. 233–240, 2006.

[27] Z. Zheng, X. Wu, Y. Zhang, M. R.lyu, and J. Wang, "Qos ranking prediction for cloud services," *European Journal of Industrial Engineering,*, vol. 24, pp. 1213–1222, 2013.

[28] X. Liu, C. Wang, B. B. Zhou, J. Chen, T. Yang, and A. Y. Zomaya, "Priority-based consolidation of parallel workloads in the cloud," *IEEE Transactions on parallel and distributed systems*, vol. 24, pp. 1874–1883, 2013.