A Consensus-based Approach for Selecting Cloud Plans

Ala Arman, Sara Foresti, Giovanni Livraga, Pierangela Samarati
DI - Università degli Studi di Milano, 26013 Crema - Italy
Email: *firstname.lastname*@unimi.it

***Abstract*—An important problem when moving an application to the cloud consists in selecting the most suitable cloud plan (among those available from cloud providers) for the application deployment, with the goal of finding the best match between application requirements and plan characteristics. If a user wishes to move multiple applications at the same time, this task can be complicated by the fact that different applications might have different (and possibly contrasting) requirements. In this paper, we propose an approach enabling users to select a cloud plan that best balances the satisfaction of the requirements of multiple applications. Our solution operates by first ranking the available plans for each application (matching plan characteristics and application requirements) and then by selecting, through a consensus-based process, the one that is considered more acceptable by all applications.**

***Index Terms*—Users preferences, Cloud plan selection, Multi-criteria decision making, Consensus voting**

I. Introduction

Cloud computing represents today the reference paradigm for deploying applications and for storing, managing, and processing large amounts of data. Thanks to the advantages in service availability and economical savings, more and more private and public organizations, as well as individuals, are moving their data and applications to the cloud [1]. The cloud market is growing at a quick pace, offering a variety of opportunities to its users. Indeed, cloud providers available on the market sell plans that differ in the services they offer, the quality of services they guarantee, and the price lists they apply. This variety provides great advantages for users, enabling them to choose the provider (and the plan) that better suits their needs and economical availability.

Choosing one cloud plan over another to deploy an application is a critical task, as it has clear consequences on the quality of the provided service (e.g., a plan/provider with frequent downtimes would cause considerable inconveniences to users trying to interact with the applications deployed over it). The problem of cloud plan selection is even more complicated when a user needs to deploy, on a given cloud plan, multiple applications. In fact, each application can have specific requirements on the characteristics that its "ideal" plan should guarantee. For instance, applications operating with sensitive data will mostly care about security (e.g., encryption algorithms, security auditing) while applications running data-intensive computations on publicly available or non-confidential data will be more interested in performance (e.g., CPU and disk speed, network latency, price lists). Since the requirements of user applications may be contrasting, a single cloud plan satisfying all of them might not exist. The user then needs to properly combine applications' requirements and choose a plan that satisfies them in the best possible way.

A naive approach to choose the most suitable plan would consist in identifying the best plan for each application, and then selecting the one chosen by the majority of the applications. However, such an approach would risk to leave the requirements of some applications completely unsatisfied. In this paper, we propose an approach aimed at balancing requirements satisfaction for multiple applications. To this purpose, our solution first produces, for each application, a ranking of the available plans according to its requirements; it then selects the plan that is globally considered the most acceptable by all applications. To implement these two steps, we put forward the idea of jointly adopting a *multi-criteria decision making technique* (TOPSIS [2]) to rank applications, and a *consensus-based voting technique* (Borda count [3]) to choose a plan that is ranked high by all the applications. The combined adoption of these techniques enables the user to choose the cloud provider (and the plan among the ones it offers) that better balances the requirements of all the applications, reaching a trade-off among their (possibly contrasting) needs. This guarantees that no application is left completely unsatisfied.

The remainder of this paper is organized as follows. Section II presents our problem. Section III illustrates our solution. Section IV discusses related works. Finally, Section V concludes the paper.

II. Problem Definition

We consider a scenario characterized by a user wishing to outsource to the cloud a set $A = \{a_1, \ldots, a_n\}$ of applications. To this aim, she needs to find the most suitable among a set $P = \{p_1, \ldots, p_m\}$ of plans offered by a set of cloud providers. Each plan $p \in P$ might have different characteristics with respect to a set $C = \{c_1, \ldots, c_l\}$ of criteria that the user considers of interest for the set A of applications. For instance, C can include criteria such as the guaranteed availability, the charged costs, or the security guaranteed by the providers. In the definition of C, the user can refer to existing guidelines and classifications (e.g., [4]), combined with her personal needs.

Since plans in P differ in the characteristics of the offered services, we assume the user to rate the degree to which a criterion $c_i \in C$ is "satisfied" by a plan $p_j \in P$. Intuitively,

	R_1	R_2	R_3	R_4	R_5
Availability	0.40	0.50	0.90	0.40	0.20
Performance	0.50	0.60	0.97	0.30	0.30
Security	0.60	0.70	0.80	0.20	0.40
Costs	0.50	0.40	0.10	0.60	0.70
...	...	...	...	...	...
Backup	0.40	0.60	0.30	0.30	0.20
StorageSpace	0.50	0.30	0.40	0.30	0.20
MobileSupport	0.60	0.80	0.30	0.40	0.50

Fig. 1. Sample rating vectors $R_1, \ldots, R_5$

this degree expresses how much the services offered by p_j are close to an ideal scenario that maximizes the satisfaction of c_i (e.g., a cloud provider offering its services for free would have the maximum rating for the *Cost* criterion). Each plan p_j is then associated with a *rating vector* $R_j[1,\ldots,l]$, where $0 \leq R_j[i] \leq 1$ represents the rating of p_j with respect to criterion c_i, where higher ratings represent better satisfaction of the criterion. For instance, a plan p_j providing more than 10 synchronized replicas, sophisticated authentication mechanisms and encryption algorithms, high CPU rates and network bandwidth, but applying expensive price lists, will have a high rating w.r.t. security, availability, and performance, and a low rating w.r.t. cost. Figure 1 illustrates sample rating vectors $R_1, \ldots, R_5$ for plans $p_1, \ldots, p_5$ respectively, over different criteria. For instance, p_5 is rated lower for criterion *Availability* ($R_5[Availability] = 0.20$) than for criterion *Costs* ($R_5[Costs] = 0.70$).

The set C of criteria is defined by the user considering all the requirements of all her applications in A. Indeed, as mentioned in Section I, not *all* criteria in C may be relevant to *all* applications in A. We denote with $C_k \subseteq C$ the set of criteria relevant to application $a_k \in A$. For instance, with reference to our running example, the set C_1 of criteria relevant to a_1 is $C_1 = \{Availability, Performance, Security, Cost\}$, and the set C_2 relevant to a_2 is $C_2 = \{Backup, StorageSpace, MobileSupport\}$. Also, given an application a_k with its set C_k of relevant criteria, not all criteria $c_i \in C_k$ can be assumed to be equally important to a_k. For instance, with reference to the example above, a_1 might value *Security* more than *Availability*. A natural way to express the requirements of an application $a_k \in A$ consists in associating a weight to each criterion in C_k. In fact, this permits to model applications having different (and possibly disjoint) relevant criteria, with different relevance for different applications. Considering an application a_k, the importance of each criterion $c_i \in C_k$ is modeled by associating c_i with a *weight*, where higher weights model higher importance of the criterion for a_k. Formally, the requirement for an application a_k is expressed as a *weight vector* $W_k[1,\ldots,|C_k|]$, where $W_k[i]$ represents the weight (i.e., the relative importance) of criterion c_i for application a_k. To enable comparison among the weights, we assume the weight vectors to be normalized (i.e., $\sum_{i=1}^{|C_k|} W_k[i]=1,\ k = 1,\ldots,n$). Figure 2 illustrates two sample weight vectors for two applications, a_1 and a_2, where $C_1 = \{Availability, Performance, Security, Costs\}$ and $C_2 = \{Backup, StorageSpace, MobileSupport\}$. For instance, the weight vector $W_2 = [0.30, 0.40, 0.30]$ of

	W_1
Availability	0.04
Performance	0.02
Security	0.04
Costs	0.90

	W_2
Backup	0.30
StorageSpace	0.40
MobileSupport	0.30

Fig. 2. Sample weight vectors for applications a_1 and a_2 over different criteria

application a_2 states that criteria *Backup* and *MobileSupport* have the same relative importance (0.30 each), while criterion *StorageSpace* is more relevant (0.40).

Given an application $a_k \in A$ and a set P of plans, the user can identify the plan $p \in P$ that best matches the requirements of a_k by using classical multi-criteria decision making approaches (e.g., [5]). Because of the heterogeneity of the requirements of the applications, however, the plan maximizing the satisfaction of the requirements of an application a_k may not be the plan maximizing satisfaction of the requirements of another application $a_x \neq a_k$. It would instead be desirable to combine the requirements of all the applications, to select a plan that satisfies all of them in the best possible way. A simple solution would choose the plan that better satisfies the majority of the application requirements. However, such a trivial approach may select a solution that fully satisfies the requirements of applications $A \setminus \{a_k\}$ while not satisfying the requirements of a_k at all. This solution might then be considered not desirable as it would strongly penalize application a_k. To prevent such a situation, we propose to adopt a consensus-based approach aimed at choosing the plan that balances the preferences of all the applications, hence enabling the user to determine a solution that provides a good trade-off in the satisfaction of the requirements of all her applications.

III. Consensus for Cloud Plan Selection

Our approach to choose the plan that best fits the user requirements operates in two steps (see Figure 3): *1) rank*, for each application, the providers on the basis of their compliance with the application requirements; *2) reach a consensus* in the choice of the provider that better suits the application requirements, based on the rankings obtained in the first step. In the following, we present our approach, based on TOPSIS for computing rankings, and Borda count for reaching a consensus.

A. Ranking Cloud Plans for an Application

The first step of our solution aims at producing a ranking of the plans in P for each application $a_k \in A$. Such a ranking reflects the satisfaction of the requirements of a_k by the different plans, being the first plan in the ranking the one better satisfying all the requirements of a_k.

To rank the plans for an application, we propose to adopt traditional multi-criteria decision making (MCDM) techniques. In fact, MCDM approaches effectively identify, in a pool of alternative solutions, the one that optimizes a set of objective functions (i.e., application requirements in our terminology). Among several MCDM techniques, a possible approach relies on adopting TOPSIS [2] as it showed to provide good results

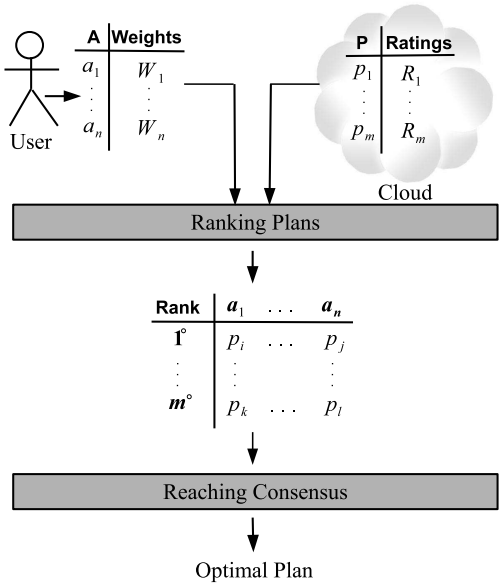

Fig. 3. Working of the approach

when applied to cloud scenarios, traditionally characterized by many alternatives compared to the number of criteria [6].

Given an application $a_k \in A$, a set P of alternative solutions (plans, in our scenario), a set C_k of criteria relevant to a_k, the weights W_k assigned by a_k to the criteria in C_k, and the ratings $R_j[i], i = 1,\ldots,|C_k|, j = 1,\ldots,|P|$ assigned to plan p_j for criterion c_i, TOPSIS produces a ranking of the alternatives in P, ordering them according to how well they satisfy the criteria (from the best to the worst). To produce such a ranking, TOPSIS evaluates the distance of each plan in P from the *ideal* and *anti-ideal* solutions, ranking higher those plans that are closer to the ideal solution and farthest from the anti-ideal solution. Intuitively, the ideal solution p_k^+ for a_k is a plan (which may not belong to P) that satisfies in the best possible way all the criteria relevant to a_k. On the contrary, the anti-ideal solution p_k^- for a_k is a plan (which may not belong to P) that satisfies in the worst possible way the criteria relevant to a_k. For each application in A, TOPSIS works in three steps: *i)* it first computes a weighted decision matrix, based on weights and ratings; *ii)* it then identifies the ideal and anti-ideal solutions; and *iii)* finally, it ranks the plans based on their distance from the ideal and anti-ideal solutions.

In the remainder of this section, we illustrate more in details the working of TOPSIS in our scenario. For simplicity, in the following, we refer our discussion to one application only (a_k), with the note that the process described is executed for all applications in A (as clearly illustrated in the pseudo-code in Figure 4).

Weighted decision matrix. To determine the weighted decision matrix for each application a_k, TOPSIS uses a *decision matrix* $\mathbb{R}_k$, with a row for each criteria $c \in C_k$ and a column for each plan $p \in P$. Basically, the decision matrix for application a_k is composed of the rating vectors R_j (restricted to the

INPUT
$A = \{a_1,\ldots,a_n\}$ /* set of applications */
$P = \{p_1,\ldots,p_m\}$ /* set of plans */
$C = \{c_1,\ldots,c_l\}$ /* set of criteria */
$W_1,\ldots,W_n$ /* weight vectors */
$R_1,\ldots,R_m$ /* rating vectors */

OUTPUT
$p \in P$ /* optimal plan */

MAIN

/* **Step 1**: rank plans for each application in A */

1: **for each** $a_k \in A$ **do**
2: let $\mathbb{R}_k$ be the *decision matrix* of size $|C_k| \times |P|$
3: **for each** $i = 1,\ldots,|C_k|$ **do**
4: **for each** $j = 1,\ldots,|P|$ **do**
5: $\mathbb{R}_k[i][j] := R_j[i]$ /* fill $\mathbb{R}_k$ with values in the rating vectors */
6: let $\mathbb{D}_k$ be the *weighted decision matrix* of size $|C_k| \times |P|$ for a_k
7: **for each** $i = 1,\ldots,|C_k|$ **do**
8: **for each** $j = 1,\ldots,|P|$ **do**
9: $\mathbb{D}_k[i][j] := \mathbb{R}[i][j] \cdot W_k[i]$ /* fill $\mathbb{D}_k$ with weighted ratings */
10: let D_k^+ be the *weighted rating vector* of size $|C_k|$ for p_k^+
11: let D_k^- be the *weighted rating vector* of size $|C_k|$ for p_k^-
12: **for each** $i = 1,\ldots,|C_k|$ **do**
13: $D_k^+[i] := max\{\mathbb{D}_k[i][j] \mid j = 1,\ldots,|P|\}$ /* p_k^+ rating for c_i */
14: $D_k^-[i] := min\{\mathbb{D}_k[i][j] \mid j = 1,\ldots,|P|\}$ /* p_k^- rating for c_i */
15: let S_k be a vector of size $|P|$ for a_k /* to store closeness values */
16: **for each** $j = 1,\ldots,|P|$ **do**
17: let $dist_j^+$ be the distance between p_j and p_k^+
18: let $dist_j^-$ be the distance between p_j and p_k^-
19: $S_k[j] := \frac{dist_j^-}{dist_j^+ + dist_j^-}$ /* closeness between p_j and ideal solutions */
20: let O_k be a list of size $|P|$ to contain plans in ranked order for a_k
21: **for each** $j = 1,\ldots,|P|$ **do**
22: insert p_j in O_k in decreasing order of $S_k[j]$

/* **Step 2**: reach consensus */

23: **for each** $a_k \in A$ **do**
24: let B_k be the Borda vector of size $|P|$ for a_k
25: **for each** $j = 1,\ldots,|P|$ **do**
26: let x be the position of p_j in O_k /* TOPSIS ranking of p_j */
27: $B_k[j] := |P| + 1 - x$ /* Borda score for plan p_j and application a_k */
28: **for each** $j = 1,\ldots,|P|$ **do**
29: $Borda(p_j) := \sum_{k=1}^{n} B_k[j]$ /* Borda score for plan p_j*/
30: **return** $p \in P$ s.t. $\nexists p' \in P, p' \neq p : Borda(p') > Borda(p)$

Fig. 4. Algorithm for selecting the consensus-based optimal plan

	p_1	p_2	p_3	p_4	p_5
Availability	0.40	0.50	0.90	0.40	0.20
Performance	0.50	0.60	0.97	0.30	0.30
Security	0.60	0.70	0.80	0.20	0.40
Costs	0.50	0.40	0.10	0.60	0.70

Fig. 5. Decision matrix $\mathbb{R}_1$ for application a_1

criteria C_k relevant to a_k): each cell $\mathbb{R}_k[i][j]$ in the decision matrix represents the rating $R_j[i]$ assigned to plan $p_j \in P$, for criteria $c_i \in C_k$ (lines 2–5). Figure 5 illustrates the decision matrix for a_1, obtained from the rating vectors in Figure 1 restricted to the first four criteria (i.e., those relevant for a_1). The original TOPSIS proposal normalizes the decision matrix, to guarantee that values in different cells can be properly compared. Since the rating values assigned to plans are already a-dimensional values between 0 and 1, in our scenario, it is not necessary to normalize the decision matrix $\mathbb{R}_k$.
To properly take into consideration the importance of the

	p_1	p_2	p_3	p_4	p_5	p_1^+	p_1^-
Availability	0.016	0.020	0.036	0.016	0.008	0.036	0.008
Performance	0.010	0.012	0.019	0.006	0.006	0.019	0.006
Security	0.024	0.028	0.032	0.008	0.016	0.032	0.008
Costs	0.450	0.360	0.090	0.540	0.630	0.630	0.090
$dist_j^+$	0.182	0.271	0.540	0.096	0.035		
$dist_j^-$	0.360	0.271	0.039	0.450	0.540		
S_1	0.665	0.500	0.068	0.824	0.939		

Fig. 6. Weighted decision matrix $\mathbb{D}_1$, ideal solution p_1^+, anti-ideal solution p_1^-, distances $dist_j^+$ and $dist_j^-$ of each plan p_j from p_1^+ and p_1^-, and relative closeness S_1 of each plan to the ideal solutions for application a_1

different criteria in C_k for the considered application a_k, the decision matrix $\mathbb{R}_k$ is composed with vector W_k (i.e., with the weights assigned to each criteria to reflect the application needs). Each cell in the *weighted decision matrix* $\mathbb{D}_k$ for application a_k is computed as the product $\mathbb{D}_k[i][j] = \mathbb{R}_k[i][j] \cdot W_k[i]$ of the rating obtained by plan p_j for criterion c_i, and the weight of criterion c_i for application a_k (lines 6–9). Figure 6 illustrates the weighted decision matrix for application a_1 of our running example. For instance, $\mathbb{D}_1[Availability][p_1]$ is obtained as $\mathbb{R}_1[Availability][p_1] \cdot W_1[Availability]$=0.4 · 0.04=0.016. Note that the weighted decision matrix permits to identify, for each criterion c_i singularly taken, the best and the worst plan, which correspond to the highest and lowest values in the row representing c_i. As an example, the best plan w.r.t. the *Security* criterion for application a_1 is p_3, while the worst is p_4.

Ideal and anti-ideal solutions. Based on the weighted decision matrix $\mathbb{D}_k$, TOPSIS is able to identify both the ideal and the anti-ideal solutions p_k^+ and p_k^- for application a_k. For the ideal solution p_k^+, the weighted rating for criterion c_i (denoted $D_k^+[i]$) is the maximum weighted rating obtained by a plan in P for c_i (i.e., $D_k^+[i]$=$max\{\mathbb{D}_k[i][j] : p_j \in P\}$, line 13). For instance, the ideal solution for application a_1, considering the weighted decision matrix in Figure 6, has weighed ratings D_1^+=$[0.036, 0.019, 0.032, 0.630]$. Similarly, for the anti-ideal solution p_k^-, the weighted rating for criterion c_i (denoted $D_k^-[i]$) is the minimum weighted rating obtained by a plan in P for c_i (i.e., $D_k^+[i]$=$min\{\mathbb{D}_k[i][j] : p_j \in P\}$, line 14). For instance, the anti-ideal solution for application a_1, considering the weighted decision matrix in Figure 6, has weighted ratings D_1^+=$[0.008, 0.006, 0.008, 0.090]$.

Ranking. To produce a ranking, TOPSIS then computes the Euclidean *distance* of each plan $p_j \in P$ from the ideal p_k^+ and anti-ideal p_k^- solution in an l-dimensional space (with l the number of criteria in C_k). Then, it computes the relative closeness of each plan p_j to the ideal solutions as $\frac{dist_j^-}{dist_j^+ + dist_j^-}$, where $dist_j^+$ and $dist_j^-$ the distance of p_j from p_k^+ and p_k^-, respectively (lines 15–19, where such closeness values are maintained in a score vector S_k). For instance, the relative closeness values of the plans in P to p_1^+ for application a_1 considering the weighted decision matrix in Figure 6, is S_1=$[0.665, 0.500, 0.068, 0.824, 0.939]$. The higher the value S_k, the better the plan satisfies the requirements of application a_k. Then, TOPSIS produces a ranking of the plans for application a_k by ordering them in decreasing order of S_k. For instance, with reference to our running example, the ratings in Figure 1, and the weights in Figure 2, the ranking of plans produced by TOPSIS for application a_1 is $\langle p_5, p_4, p_1, p_2, p_3 \rangle$ (see column a_1 in Figure 7).

	a_1	a_2	a_3	a_4	a_5	a_6	a_7
1°	p_5	p_3	p_5	p_1	p_3	p_4	p_3
2°	p_4	p_2	p_4	p_2	p_2	p_5	p_2
3°	p_1	p_1	p_1	p_5	p_1	p_1	p_1
4°	p_2	p_4	p_2	p_3	p_5	p_2	p_5
5°	p_3	p_5	p_3	p_4	p_4	p_3	p_4

Fig. 7. Sample rankings of the plans for each application

B. Reaching Consensus

The second step of our solution aims at choosing a cloud plan that balances the preferences of all user applications. A straightforward approach to maximize requirements satisfaction would adopt a majority voting, that is, it would choose the cloud plan ranked first by most applications. For instance, consider a scenario characterized by a set $A = \{a_1, \ldots, a_7\}$ of applications and a set $P = \{p_1, \ldots, p_5\}$ of plans, where the rankings computed by TOPSIS for each application are illustrated in Figure 7. The plan that would win with the majority voting approach would be p_3. However, as already noted, this solution might be not desirable as p_3 is ranked last by three applications a_1, a_3, and a_6, which would then be strongly penalized.

We then propose to adopt a consensus-based voting technique, that permits to choose an alternative that is acceptable for a broad set of voters (applications, in our scenario), rather than simply counting majority. While noting that there are different approaches that can be applied (e.g., [7], [8]), we consider - as an example - the Borda count method [3]. In our cloud scenario, alternatives correspond to plans and the applications play the role of voters. To express its vote, each application $a_k \in A$ associates a Borda score $B_k[j]$ with each plan $p_j \in P$. Such a score reflects the rankings computed by TOPSIS (or, more in general, by the chosen MCDM technique) by assigning $m = |P|$ points to the first ranked plan, 1 to the last ranked plan, and $m+1-x$ to the x-th ranked plan. The overall Borda score $Borda(p_j)$ of a plan $p_j \in P$ is then obtained by summing the scores assigned to the plan by each application, that is, $\sum_{k=1}^{n} B_k[j]$. The plan with the highest Borda score is the one that is chosen by the user, since it has the consensus of all the applications. As an example, Figure 8 illustrates the Borda scores assigned by each application to each plan, and the overall score of each plan. It is interesting to see that the chosen plan is p_1, which is ranked first by one application only, while p_3 is only the third choice, even though it is ranked first by three applications.

IV. Related Work

Different issues have to be considered when moving applications and/or data to the cloud, and the scientific community has recently addressed different problems (e.g., [1], [9]). The line of work closest to ours deals with the problem

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	Tot
p_1	3	3	3	5	3	3	3	**23**
p_2	2	4	2	4	4	2	4	22
p_3	1	5	1	2	5	1	5	20
p_4	4	2	4	1	1	5	1	18
p_5	5	1	5	3	2	4	2	22

Fig. 8. Borda scores assigned to each plan by each application

of selecting the most suitable (set of) cloud provider(s) for data and application outsourcing (e.g., [10], [11], [12]). The work in [10] proposes a solution allowing a set of cloud providers to publish the characteristics of their services, and a user to select the one that better satisfies her requirements for an application (differently from our scenario where we must accommodate contrasting requirements of a set of applications). The proposal in [11], while sharing with our work the assumption that users might have contrasting needs when moving to the cloud, significantly differs from ours since it solves conflicts among requirements by composing multiple cloud services. The proposal in [12] focuses instead on selecting a single cloud plan specifically focusing on cloud data storage, while we aim at selecting the best plan on which deploying a set of applications. The approach in [13] pursues the goal of selecting the most suitable providers to store data replicas, considering performances and providers load. MCDM techniques have been also proposed to solve the problem of cloud services/plans selection. For instance, the work in [6] compares different MCDM techniques to select the best cloud service based on performance measurements. Our scenario is instead more general and supports generic requirements on arbitrary criteria, with the specific support for contrasting requirements to be jointly satisfied. The proposal in [14] adopts an Analytical Hierarchical Process (AHP) based ranking mechanism to properly weight the requirements of an application to be outsourced to the cloud. While having the same goal of selecting the most suitable cloud plan, this proposal focuses on the requirements of a single application.

Other related works include proposals aimed at enabling cloud providers to choose the best suite of services to offer to their customers, considering the users requests and the need of the providers to save on allocated resources (e.g., [15], [16], [17]). This problem is orthogonal to the one addressed here, since we consider the user (in contrast to the provider) point of view.

Another major line of work focuses on security issues in multi-cloud scenarios, proposing solutions to protect integrity (e.g., [18], [19], [20], [21]) and confidentiality of accesses (e.g., [22], [23]) to data outsourced to multiple providers. While sharing with us a scenario characterized by multiple cloud providers, these proposals are complementary to ours as they specifically focus on the enforcement of protection mechanisms.

V. Conclusions

We presented a solution enabling cloud users to choose the plan, among the ones available in the cloud, that aims at reaching consensus among plans preferred by different applications. In our approach, each application individually ranks the available cloud plans depending on how well they satisfy the application requirements. The choice on the final plan is then taken adopting a consensus-based approach on the different rankings. Our work leaves space for further extensions and improvements. In particular, in our work we consider all applications, and rankings produced by them, to be equally important when computing a solution to reach consensus. This implies that the relative distance among plans is not considered, only their position in the rankings is. This observation is consistent with our assumptions, as we are operating with multiple applications with independent requirements, and independent desire for a most suitable plan, and what is assumed to be important for finding consensus is what is the best choice for each individual applications. An interesting alternative to be investigated can consider the application requirements by different applications as a single ecosystem to be globally optimized, considering then not only the rankings but the distance among plans, possibly evaluating also preferences among applications.

Acknowledgements

This work was supported in part by the EC within the 7FP under grant agreement 312797 (ABC4EU) and within the H2020 under grant agreement 644579 (ESCUDO-CLOUD).

References

[1] P. Samarati and S. De Capitani di Vimercati, "Cloud security: Issues and concerns," in *Encyclopedia on Cloud Computing*, S. Murugesan and I. Bojanova, Eds. Wiley, 2016, to appear.

[2] H. Ching-Lai and K. Yoon, *Multiple attribute decision making: methods and applications*. Springer-Verlag, 1981.

[3] J. C. de Borda, *Memoire sur les Elections au Scrutin*. Histoire de l'Academie Royale des Sciences de Paris, 1781.

[4] M. Galster and E. Bucherer, "A taxonomy for identifying and specifying non-functional requirements in service-oriented development," in *Proc. of IEEE SERVICES 2008*, Honolulu, HI, USA, July 2008.

[5] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[6] Z. Rehman, O. Hussain, and F. Hussain, "IaaS cloud selection using MCDM methods," in *Proc. of IEEE ICEBE 2012*, Hangzhou, China, September 2012.

[7] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the web," in *Proc. of ACM WWW 2001*, Hong Kong, China, May 2001.

[8] W. D. Cook and L. M. Seiford, "On the Borda-Kendall consensus method for priority ranking problems," *Management Science*, vol. 28, no. 6, pp. 621–637, 1982.

[9] S. De Capitani di Vimercati, S. Foresti, and P. Samarati, "Managing and accessing data in the cloud: Privacy risks and approaches," in *Proc. of CRiSIS 2012*, Cork, Ireland, October 2012.

[10] A. Goscinski and M. Brock, "Toward dynamic and attribute based publication, discovery and selection for cloud computing," *Future Generation Computer Systems*, vol. 26, no. 7, pp. 947–970, 2010.

[11] A. V. Dastjerdi and R. Buyya, "Compatibility-aware cloud service composition under fuzzy preferences of users," *IEEE TCC*, vol. 2, no. 1, pp. 1–13, 2014.

[12] A. Ruiz-Alvarez and M. Humphrey, "An automated approach to cloud storage service selection," in *Proc. of ACM ScienceCloud 2011*, San Jose, CA, USA, June 2011.

[13] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, "DONAR: Decentralized server selection for cloud services," in *Proc. of ACM SIGCOMM 2010*, New Delhi, India, August/September 2010.

[14] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012–1023, 2013.

[15] M. Anisetti, C. Ardagna, P. Bonatti, E. Damiani, M. Faella, C. Galdi, and L. Sauro, "e-Auctions for multi-cloud service provisioning," in *Proc. of IEEE SCC 2014*, Anchorage, AL, USA, June–July 2014.

[16] M. Anisetti, C. Ardagna, F. Gaudenzi, and E. Damiani, "A cost-effective certification-based service composition for the cloud," in *Proc. of IEEE SCC 2016*, San Francisco, CA, USA, June–July 2016.

[17] R. Jhawar, V. Piuri, and P. Samarati, "Supporting security requirements for resource management in cloud computing," in *Proc. of IEEE CSE 2012*, Paphos, Cyprus, December 2012.

[18] Y. Zhu, H. Hu, G. J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE TPDS*, vol. 23, no. 12, pp. 2231–2244, 2012.

[19] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A high-availability and integrity layer for cloud storage," in *Proc. of ACM CCS 2009*, Chicago, IL, USA, November 2009.

[20] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, and P. Samarati, "Integrity for distributed queries," in *Proc. of IEEE CNS 2014*, San Francisco, CA, USA, October 2014.

[21] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Efficient integrity checks for join queries in the cloud," *JCS*, vol. 24, no. 3, pp. 347–378, 2016.

[22] E. Stefanov and E. Shi, "Multi-cloud oblivious storage," in *Proc. of ACM CCS 2013*, Berlin, Germany, November 2013.

[23] S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, and P. Samarati, "Three-server swapping for access confidentiality," *IEEE TCC*, 2015, pre-print.