DOI: 10.1109/TSC.2017.2777977

Supporting User Requirements and Preferences in Cloud Plan Selection

Sabrina De Capitani di Vimercati, *Senior Member, IEEE,* Sara Foresti, *Senior Member, IEEE,* Giovanni Livraga, *Member, IEEE,* Vincenzo Piuri, *Fellow, IEEE,* and Pierangela Samarati, *Fellow, IEEE*

Abstract—With the cloud emerging as a successful paradigm for conveniently storing, accessing, processing, and sharing information, the cloud market has seen an incredible growth. An ever-increasing number of providers offer today several cloud plans, with different guarantees in terms of service properties such as performance, cost, or security. While such a variety naturally corresponds to a diversified user demand, it is far from trivial for users to identify the cloud providers and plans that better suit their specific needs. In this paper, we address the problem of supporting users in cloud plan selection. We characterize different kinds of requirements that may need to be supported in cloud plan selection and introduce a very simple and intuitive, yet expressive, language that captures different requirements as well as preferences users may wish to express. The corresponding formal modeling permits to reason on requirements satisfaction to identify plans that meet the constraints imposed by requirements, and to produce a preference-based ranking among such plans.

Index Terms—Cloud plan selection, cloud plan requirements and preferences, cloud plan ranking

1 Introduction

Cloud computing is undeniably on its way to become the de facto standard adopted by public and private companies, governmental agencies, and individuals for data storage and computation. The benefits of moving to the cloud –in contrast to owning and locally managing a computing infrastructure– are many and diverse, including reduced economic costs, increased service availability, and high flexibility, as users can freely scale up and down resource demand according to their needs by renting payper-use cloud infrastructures and computation capabilities. As the ICT advances and makes fast network connections ubiquitously available, these benefits become more and more evident to potential customers, and make cloud-based outsourcing more appealing than ever (e.g., [1], [2], [3]). A recent study from Gartner, Inc. testifies this trend by forecasting that by 2020 the "cloud shift" (i.e., the shift from traditional IT offerings to cloud-based ones) will involve direct and indirect spending for more than USD 1 Trillion [4].

The increasing demand of cloud services has also introduced opportunities for a variety of offers (plans), with different characteristics and guarantees (e.g., in terms of performance, Quality of Service, and security). While such variety naturally corresponds to a diversified market demand, choosing the "best" cloud service plan is often not an easy task for users. The research and industrial community has recognized this problem and the need for approaches supporting users in evaluating, comparing, ranking different cloud plans or choosing the plan that best suits their needs, though research in this respect is still in early stages.

• *Sabrina De Capitani di Vimercati, Sara Foresti, Giovanni Livraga, Vincenzo Piuri, and Pierangela Samarati are with the Computer Science Department, Università degli Studi di Milano, 26013 Crema – Italy. E-mail:* firstname.lastname@*unimi.it*

Some of the existing solutions focus on the problem of comparing different cloud plans based on their cost and performance (e.g., to select a plan that completes a task in the shortest time within a cost budget, or that has the lowest cost with satisfying performance [5]). Other approaches supporting users in cloud plan selection consider other properties characterizing available plans (e.g., the type of provided services, QoS values, supported operating systems, instance sizes [6]). These solutions, however, typically do not allow users to express conditions on arbitrary properties of their interest, or to specify expressive combinations among them (e.g., they either consider pre-defined KPIs such as response time, transparency, reliability [7] or only allow the definition of basic conjunctions among requirements, such as 'supported operating system x and instance size y' [6]). However, users may be interested in defining more expressive requirements. For instance, a user might want to consider only cloud plans offered by servers located in a specific geographical area and certified by given authorities or that protect data using a specific encryption algorithm.

In this paper, we address this problem and propose a flexible and expressive framework for characterizing cloud plans, for supporting users in expressing requirements and preferences over plan characteristics, and for taking into account requirements and preferences in determining plans that are acceptable as well as preferable. For the definition of requirements, we provide a high-level and intuitive language that users can adopt for specifying restrictions on plans as well as an underlying formal modeling to reason about requirement satisfaction. Our approach is based on a simple and intuitive, yet quite expressive, characterization of requirements, which enables users to conveniently specify constraints on the values that plans should (or should not) assume for their attributes. Our modeling of requirements includes simple conditions on attribute values, combinations or alternatives thereof, as well as combinations

A	$Dom(A)$	
prov	Mhard, GoGo, Ghost, Amaron	*cloud provider*
loc	EU, US, JP	*physical location of servers*
encr	AES, 3DES, DES	*encryption algorithm*
avail	VH, H, MH, M, ML, L, VL	*level of availability*
test	authA, authB, authC, authD	*authority running penetration tests*
cert	certA, certB, certC, certD, certE	*security certification*
aud	3M, 6M, 9M, 1Y	*frequency of security auditing*

Fig. 1. An example of attributes and their domains

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
prov	Mhard	Ghost	Ghost	GoGo	GoGo	GoGo	GoGo
loc	JP	US	US	EU	US	EU	US
encr	DES	3DES	AES	3DES	AES	AES	AES
avail	ML	M	H	VH	H	VH	VH
test	authC	authC	authB	authB	authA	authA	authB
cert	certC	certB	certC	certB	certC	certA	certC
aud	–	1Y	–	–	–	–	–

Fig. 2. An example of plan descriptors of seven cloud plans

that should be forbidden. Similarly, our modeling of preferences is intuitive and flexible, enabling users to express a preference relationship among attribute values and also among attributes themselves. Ranking of acceptable plans is then based on users preferences. The contribution of the paper is threefold. First, we identify and characterize possible requirements that users may wish to express on cloud plans for them to be considered acceptable. Second, we provide a characterization of preferences on plans, enabling users to specify attribute values that are to be preferred over others as well as preferences among attributes. Third, we illustrate possible approaches to rank acceptable plans based on users' preferences.

Our approach assumes a set of plans to be evaluated against user requirements and preferences and is agnostic with respect to specific scenarios in which such an evaluation can be needed. In particular, it can naturally be used in brokerage services, to evaluate offers by different providers, as well as in single-provider scenarios, to evaluate the different plans (possibly resulting by customization) offered by a provider.

The remainder of this paper is organized as follows. Section 2 introduces the considered scenario and summarizes basic concepts. Section 3 presents our characterization, as well as formal modeling, of user requirements. Section 4 illustrates our characterization of user preferences, at both the value and attribute level. Section 5 describes our approach for ranking cloud plans according to user preferences. Section 6 discusses the related work. Finally, Section 7 concludes the paper.

2 Scenario and basic concepts

The goal of this paper is to provide a framework for supporting users in selecting cloud plans that better respond to their needs on the data and applications they wish to outsource. We consider such needs to be expressed as requirements and preferences over attributes (and their values) that represent characteristics of the plans of interest to the user. In our work, we do not restrict our approach to any specific predefined set of such attributes and assume to refer to a generic set $\mathcal{A}$ of attributes encompassing all possible properties that users may wish to consider (e.g., availability, cost, implemented security measures), with attributes names and values shared between users and providers as a common ontology [8]. With such a general and flexible approach, attributes in $\mathcal{A}$ (and their values) can represent Service Level Objectives guaranteed in the Service Level Agreement (SLA) to be signed by the provider of a plan and the user (e.g., the guaranteed availability), as well as any generic property/metadata of a plan that the provider can guarantee or that the user can measure (e.g., the physical location where the servers are placed). In this regard, our work can nicely complement existing approaches that aim to provide uniform frameworks for expressing and evaluating the characteristics of the cloud plans available on the market (e.g., the CSA–CAI [9], [10]).

Each attribute $A \in \mathcal{A}$ takes values over a given domain $Dom(A)$, representing all the values of interest to the user. Figure 1 illustrates an example of attributes, together with their domains, characterizing plans in terms of their provider (prov), the physical location where the servers are placed (loc), the encryption algorithm offered by the provider specified in the plan for protecting data (encr), the guaranteed availability (avail, from very high to very low), the authority running penetration testing (test), the security certification (cert), and the frequency of security auditing (aud). This set of attributes models a variety of characteristics of the considered plans, ranging from security-related features (e.g., encr), to performance-related ones (e.g., avail), to more generic metadata (e.g., loc).

Every cloud plan can then be characterized by the value that the plan assumes for each attribute in the considered set $\mathcal{A}$. While for simplicity we assume attributes to take only scalar values, we note that ranges of values can be easily modeled with two attributes, representing the extremes of the range (e.g., the length of the encryption key between 128 and 512 bits can be modeled as min_key_len=128 and max_key_len=512). One aspect that should be taken into consideration is that plans may not be complete in their specification, that is, the value of some attributes may be unknown (e.g., not specified or not supported by the specific plan). We address this possibility by allowing attributes to assume a special value '–', expressing the fact that the value is not known or the attribute is not applicable for the plan. With this said, we characterize every cloud plan with a plan descriptor as follows.

Definition 2.1 (Plan descriptor)**.** *Let* P_j *be a cloud plan and* $\mathcal{A} = \{A_1, \ldots, A_m\}$ *be the set of attributes describing properties of interest. The* plan descriptor *of* P_j *is a vector* $P_j[A_1, \ldots, A_m]$*, where* $P_j[A_i] \in Dom(A_i) \cup \{-\}$*,* $i = 1, \ldots, m$*, with* $Dom(A_i)$ *the domain of attribute* A_i *and '–' the special value.*

With reference to the attributes in Figure 1, Figure 2 illustrates the plan descriptor of seven cloud plans, offered by three providers (Mhard, Ghost, GoGo). Note that only plan P_2 provides explicit guarantees on security audit, declaring the frequency with which it is performed.

3 User requirements

Requirements express constraints over the values that the attributes of a cloud plan P should satisfy to be considered

acceptable for the user. Our modeling allows users to specify such restrictions in a simple, yet expressive, way. The building block in the specification of requirements is the *attribute term*, which permits to evaluate whether the value of an attribute belongs (or does not belong) to a given set of values. An attribute term is formally defined as follows.

Definition 3.1 (Attribute term)**.** *Let A be an attribute in $\mathcal{A}$, $Dom(A)$ be its domain, and $\{v_i, \ldots, v_j\} \subseteq Dom(A) \cup \{-\}$ be a set of values. An* attribute term *t over A is an expression of the form: 'A IN $\{v_i, \ldots, v_j\}$' or 'A NOT IN $\{v_i, \ldots, v_j\}$'.*

A positive (i.e., IN) term is satisfied if the attribute has a value included in the set explicitly specified, while a negative (i.e., NOT IN) term is satisfied if the attribute has a value not included in the set explicitly specified. In the remainder of the paper, we use $A(v_i, \ldots, v_j)$ as a shorthand for 'A IN $\{v_i, \ldots, v_j\}$', and $\neg A(v_i, \ldots, v_j)$ as a shorthand for 'A NOT IN $\{v_i, \ldots, v_j\}$'. For instance, $\texttt{loc}(\text{EU}, \text{US})$ is a positive term evaluating true if the location of servers is either Europe or US, while $\neg\texttt{encr}(\text{DES})$ is a negative term evaluating true if the encryption algorithm is known and is different from DES (i.e., it is AES or 3DES for our example – Figure 1). In the following, given an attribute term t, we use $t.attr$ to denote the attribute over which t is defined and $t.values$ to denote the set of values that make term t true, that is:

$$t.values = \begin{cases} \{v_i, \ldots, v_j\} & \text{if } t = A \text{ IN } \{v_i, \ldots, v_j\} \\ Dom(A) \setminus \{v_i, \ldots, v_j\} & \text{if } t = A \text{ NOT IN } \{v_i, \ldots, v_j\} \end{cases}$$

3.1 Requirements specification

One of the motivations of our work is to provide a convenient and simple approach for the identification and specification of requirements (as well as preferences, as we will see later on) that users may have with respect to cloud plans to consider them acceptable (or preferred). Having defined the basic building block (Definition 3.1) for the specification of user requirements, we now identify possible requirements that users might wish to express. The simplest requirement (to which we refer as *base requirement*) corresponds to an attribute term. More complex requirements allow users to specify combinations of terms that must be jointly satisfied (ALL); alternatives among terms that must be satisfied (ANY); and combinations of terms that: are not considered acceptable (FORBIDDEN), must be satisfied only under certain circumstances (IF-THEN), or for which at most (AT_MOST) or at least (AT_LEAST) a certain number must be satisfied. We now describe the different requirements that users can formulate. We use notations r and $\mathcal{R}$ to refer to a user requirement and to a set of user requirements, respectively. Examples refer to the requirements in Figure 3.

- t
  A base requirement of the form $r{=}t$ demands that term t be satisfied, that is, that attribute $t.attr$ over which t is defined assume (or do not assume) a value in $t.values$. Requirements r_1 and r_2 are examples of base requirements. Positive requirement r_1 demands that the provider be either Ghost, GoGo, or Mhard. Negative requirement r_2 demands availability to be different from very low (VL) and low (L).

r_1 :$\texttt{prov}(\text{Ghost}, \text{GoGo}, \text{Mhard})$
r_2 :$\neg\texttt{avail}(\text{VL}, \text{L})$
r_3 :ALL$(\{\texttt{loc}(\text{EU}, \text{US}), \neg\texttt{encr}(\text{DES})\})$
r_4 :ANY$(\{\texttt{test}(\text{authA}, \text{authB}), \texttt{cert}(\text{certA}, \text{certB})\})$
r_5 :ANY$(\{\texttt{loc}(\text{EU}), \texttt{cert}(\text{certC})\})$
r_6 :IF ALL$(\{\texttt{loc}(\text{US}), \texttt{encr}(\text{3DES}))$ THEN
ANY$(\texttt{audit}(\text{3M}, \text{6M}), \texttt{cert}(\text{certA}))$
r_7 :IF ALL$(\texttt{test}(-))$ THEN ANY$(\texttt{cert}(\text{certA}))$
r_8 :FORBIDDEN$(\{\neg\texttt{loc}(\text{EU}), \texttt{test}(\text{authC})\})$
r_9 :AT_MOST$(2, \{\texttt{prov}(\text{Ghost}), \texttt{avail}(\text{M, MH}), \texttt{encr}(\text{3DES})\})$
r_{10} :AT_LEAST$(2, \{\texttt{loc}(\text{EU}), \texttt{encr}(\text{AES}), \texttt{prov}(\text{Gogo, Ghost})\})$

Fig. 3. Well-defined set $\mathcal{R}$ of user requirements

- ALL$(\{t_1, \ldots, t_h\})$
  A requirement of the form $r{=}$ALL$(\{t_1,\ldots,t_h\})$ demands that all terms in $\{t_1, \ldots, t_h\}$ be satisfied. Requirement r_3 is an example of an ALL requirement demanding that servers be physically located either in Europe or in US *and* that the encryption used for storage be different from DES.

- ANY$(\{t_1, \ldots, t_h\})$
  A requirement of the form $r{=}$ANY$(\{t_1,\ldots,t_h\})$ demands that at least one term among the set $\{t_1, \ldots, t_h\}$ be satisfied. In other words, $\{t_1, \ldots, t_h\}$ represents a set of alternatives, and the satisfaction of (at least) one term (no matter which one) suffices. Requirements r_4 and r_5 are examples of ANY requirements. Requirement r_4 demands that the penetration testing be done by either authority authA or authority authB, *or* that the security certification be either certA or certB. Requirement r_5 demands that the servers be located in Europe *or* that the security certification be certC.

- IF ALL$(\{t_{p_1}, \ldots, t_{p_h}\})$ THEN ANY$(\{t_{p_1}, \ldots, t_{p_k}\})$
  A requirement of the form $r =$ IF ALL$(\{t_{p_1}, \ldots, t_{p_h}\})$ THEN ANY$(\{t_{p_1}, \ldots, t_{p_k}\})$ demands that if all terms in $\{t_{p_1}, \ldots, t_{p_h}\}$ are satisfied, then at least one term in $\{t_{p_1}, \ldots, t_{p_k}\}$ be also satisfied. In other words, it imposes to satisfy an ANY requirement (consequence), whenever an ALL requirement (premise) is satisfied. Requirements r_6 and r_7 are examples of IF-THEN requirements. Requirement r_6 demands that *if* servers are located in US and 3DES encryption is used, *then* either security certification must be certA or security auditing must be performed every 3 or 6 months. Requirement r_7 instead demands that *if* the authority running penetration testing is not declared, *then* the security certification must be certA.

- FORBIDDEN$(\{t_1, \ldots, t_h\})$
  A requirement of the form $r =$ FORBIDDEN$(\{t_1,\ldots,t_h\})$ demands that *not* all terms in $\{t_1, \ldots, t_h\}$ be satisfied (or, equivalently, that *at least* one of the terms in $\{t_1, \ldots, t_h\}$ be not satisfied). A FORBIDDEN requirement can therefore be used to specify that a given combination of values for the attributes is considered not acceptable. Requirement r_8 is an example of FORBIDDEN requirement stating that the user considers *not*

acceptable a plan for which servers are outside Europe *and* penetration testing is done by authority authC.

- AT_MOST$(n, \{t_1, \ldots, t_h\})$
  A requirement of the form $r =$ AT_MOST$(n, \{t_1, \ldots, t_h\})$ demands that at most a given number n (with $n \leq h$) of terms in set $\{t_1, \ldots, t_h\}$ be satisfied. Intuitively, it expresses combinations of attribute values that must be controlled and accepted only up to some limit (expressed by the number of terms whose satisfaction can be tolerated). Requirement r_9 is an example of AT_MOST requirement, stating that at most two conditions can be satisfied among: *i)* the provider being Ghost; *ii)* the availability being MH or H; and *iii)* the encryption being 3DES.

- AT_LEAST$(n, \{t_1, \ldots, t_h\})$
  A requirement of the form $r =$ AT_LEAST$(n, \{t_1, \ldots, t_h\})$ is complementary to an AT_MOST$(n, \{t_1, \ldots, t_h\})$ requirement, and demands that, among a set $\{t_1, \ldots, t_h\}$ of possible terms, at least a given number n (with $n \leq h$) be satisfied. Requirement r_{10} is an example of AT_LEAST requirement, demanding that at least two requirements be satisfied among *i)* being run on a server in Europe; *ii)* providing AES encryption; and *iii)* being provided by Gogo or Ghost.

Note that base, ALL, and ANY requirements, which we distinguish for clarity, could be seen as of the same (IF-THEN) form. In fact, an ALL requirement can be seen as a set of base requirements. In turn, a base requirement can be seen as an ANY requirement composed of a single term, and an ANY requirement corresponds to an IF-THEN requirement whose premise is bound to 'true'. We also note that the assumption of having IF-THEN requirements with only one ALL requirement in the premise and only one ANY requirement in the consequence is motivated by clarity and simplicity, and does not affect expressiveness of our model. An IF-THEN requirement with an ANY$(t_{p_1}, \ldots, t_{p_h})$ requirement in the premise could in fact be easily expressed by a set of h IF-THEN requirements, all with the same consequence and each with a different term t_{p_i} (with $i = 1, \ldots, h$) in the premise. Similarly, an ALL$(t_{c_1}, \ldots, t_{c_k})$ requirement in the consequence corresponds to a set of k IF-THEN requirements, all with the same premise and each with one term t_{c_j} (with $j = 1, \ldots, k$) in the consequence. We also note that the specification of an AT_LEAST$(n, \{t_1, \ldots, t_h\})$ requirement and an AT_MOST$(n, \{t_1 \ldots, t_h\})$ requirement with the same value for parameter n and the same set of terms corresponds to require that *exactly* n terms in $\{t_1, \ldots, t_h\}$ be satisfied.

These observations on the expressive power of our modeling with respect to the ability of capturing different user needs show that the same needs could be expressed by different sets of requirements (e.g., requiring `encr` to be AES and `test` to be authB could be modeled with two base requirements, or equivalently with a single ALL requirement). A user can then formulate her set of requirements in a flexible way by choosing the kinds of requirements she feels more appropriate and convenient for her.

When formulating requirements, special care has to be taken in using special value '—' in attribute terms, due to its semantics of unknown/undeclared value. For instance, a base requirement of the form $A(-)$ would mean that the user is interested in a cloud plan P only if attribute A assumes an unknown/undeclared value (i.e., only if $\mathrm{P}[A] = -$), which clearly makes little sense in practice. The special value could instead be used in a base requirement corresponding to a *negative* term (i.e., $\neg A(-)$): this is perfectly legitimate as such requirement, while not explicitly specifying any particular value that A must assume, imposes the value of A to be explicitly declared in the plan. In summary, since value '—' denotes absence of a declared value for an attribute in a plan, we generally expect it to be used in negative terms. Exceptions to this rule allow value '—' to be used (only) in a *positive* term t when t is included in a FORBIDDEN or AT_MOST requirement, or in the premise of an IF-THEN requirement. In fact, FORBIDDEN and AT_MOST requirements state that value '—' belongs to configurations that the user does not appreciate (in line with its semantics). In the premise of an IF-THEN requirement, terms of the form $A(-)$ can be used to identify those plans for which additional requirements (specified in the consequence) must be satisfied by plans for which attribute A is not declared.

In the remainder of this paper we assume that: *i)* the set $\mathcal{R}$ of requirements defined by the user includes, for each attribute $A \in \mathcal{A}$, at most one term over A in the set of base and ALL requirements; and *ii)* each requirement $r \in \mathcal{R}$ includes at most one term over each attribute A (if not, the multiple terms over A would be either in conflict, or redundant). A set $\mathcal{R}$ of requirements satisfying these two conditions is said to be *well-defined*, as defined in the following.

Definition 3.2 (Well-defined set of requirements)**.** *Let $\mathcal{A}$ be a set of attributes, T be a set of terms over $\mathcal{A}$, and $\mathcal{R}$ be a set of requirements over T. $\mathcal{R}$ is* well-defined *iff:*

1) $\forall A \in \mathcal{A} : |T^*_A| \leq 1$, *with T^*_A the set of terms appearing in base and* ALL *requirements in $\mathcal{R}$ over A;*
2) $\forall A \in \mathcal{A}, \forall r \in \mathcal{R} : |T^r_A| \leq 1$, *with T^r_A the set of terms over A appearing in r.*

Condition 1 ensures that each attribute A appears in at most one term in base and ALL requirements in $\mathcal{R}$ all together. Condition 2 ensures that each requirement in $\mathcal{R}$ includes at most one term for each attribute A. It is easy to see that the set of requirements in Figure 3 is well-defined.

3.2 Acceptable plans

Requirements impose conditions that a user requires to be satisfied by a cloud plan to consider such a plan acceptable. As already mentioned, *all* the requirements in the specified set $\mathcal{R}$ must be satisfied and therefore plans that do not satisfy at least one requirement are not considered to be acceptable to the user. To evaluate whether a plan P is acceptable, we interpret the terms over which $\mathcal{R}$ is defined as Boolean variables, and each requirement $r \in \mathcal{R}$ as one (or more) Boolean formula(s) b over such variables. Figure 4 illustrates the translation of the requirements discussed in Section 3.1 into Boolean formulas. Base requirements do not need special translations, as they simply correspond to attribute terms (and hence Boolean variables). An ALL$(\{t_1, \ldots, t_h\})$ requirement translates to a set of h attribute terms. ANY and IF-THEN requirements translate to Boolean disjunctions

Requirement	Translation
t	t
ALL$(\{t_1, \ldots, t_h\})$	$\{t_1, \ldots, t_h\}$
ANY$(\{t_1, \ldots, t_h\})$	$t_1 \vee \ldots \vee t_h$
IF ALL$(\{t_{p_1}, \ldots, t_{p_h}\})$THEN ANY$(\{t_{c_1}, \ldots, t_{c_k}\})$	$(t_{p_1} \wedge \ldots \wedge t_{p_h}) \Longrightarrow (t_{c_1} \vee \ldots \vee t_{c_k})$
FORBIDDEN$(\{t_1, \ldots, t_h\})$	$(\neg t_1) \vee \ldots \vee (\neg t_n)$
AT_MOST$(n, \{t_1, \ldots, t_h\})$	$\forall T \in \mathcal{T}^{n+1} : \bigvee_j (\neg t_j)$, with $t_j \in T$
AT_LEAST$(n, \{t_1, \ldots, t_h\})$	$\bigvee_{i=1}^{\lvert\mathcal{T}^n\rvert} (\bigwedge_j t_j)$, with $t_j \in T_i^n, T_i^n \in \mathcal{T}^n$

Fig. 4. Translation of user requirements in their equivalent Boolean formulas (T denotes an arbitrary set of terms, T^n a set of n terms, $\mathcal{T}^n$ the set of all T^n)

b_1 :prov(Ghost, GoGo, Mhard)
b_2 :$\neg$avail(VL, L)
$b_{3.1}$:loc(US, EU)
$b_{3.2}$:$\neg$encr(DES)
b_4 :test(authA, authB) $\vee$ cert(certA, certB)
b_5 :loc(EU) $\vee$ cert(certC)
b_6 :loc(US) $\wedge$ encr(3DES) $\Longrightarrow$ audit(3M, 6M) $\vee$ cert(certA)
b_7 :test(−) $\Longrightarrow$ cert(certA)
b_8 :loc(EU) $\vee \neg$test(authC)
b_9 :$\neg$prov(Ghost) $\vee \neg$avail(M, MH) $\vee \neg$encr(3DES)
b_{10} :(loc(EU) $\wedge$ encr(AES)) $\vee$ (loc(EU) $\wedge$ prov(Gogo, Ghost))$\vee$ (encr(AES) $\wedge$ prov(Gogo, Ghost))

Fig. 5. Set $\mathcal{B}^{\mathcal{R}}$ or Boolean formulas representing the set $\mathcal{R}$ of requirements in Figure 3

and implications among terms, respectively. The translation of FORBIDDEN, AT_MOST, and AT_LEAST requirements deserves more explanation. A FORBIDDEN$(\{t_1, \ldots, t_h\})$ requirement corresponds to the Boolean disjunction of the *negation* of the terms. The semantics of $\neg t$, in our problem, translates a positive (i.e., IN) term into a negative (i.e., NOT IN) term, and viceversa (i.e., if $t = A$ IN $\{v_i, \ldots, v_j\}$ then $\neg t = A$ NOT IN $\{v_i, \ldots, v_j\}$, and viceversa). The translation of an AT_MOST$(n, \{t_1, \ldots, t_h\})$ requirement corresponds to a set of disjunctions, one for each subset $T^{(n+1)}$ of $(n+1)$ terms in $\{t_1, \ldots, t_h\}$, each demanding that such a combination be not satisfied (i.e., the terms are negated). An AT_LEAST$(n, \{t_1, \ldots, t_h\})$ requirement translates into a disjunction among the conjunctions, one for each subset T^n of n terms in $\{t_1, \ldots, t_h\}$, of the terms in T^n (i.e., the satisfaction of any combination of n terms in $\{t_1, \ldots, t_h\}$ satisfies the requirement). Figure 5 illustrates the result of the translation of the set $\mathcal{R}$ of requirements in Figure 3 into an equivalent set $\mathcal{B}^{\mathcal{R}}$ of Boolean formulas ($\mathcal{B}^{\mathcal{R}}$ denotes the set of Boolean formulas obtained from a set $\mathcal{R}$ of requirements). Formulas $b_1, b_2, b_{3.1}, b_{3.2}$ represent the base and ALL requirements (r_1, r_2, and r_3). The Boolean disjunctions in b_4 and b_5 represent the ANY requirements (r_4 and r_5), the implications in b_6 and b_7 the IF-THEN requirements (r_6 and r_7), and the disjunction in b_8 the FORBIDDEN requirement (r_8). Formula b_9 represents the AT_MOST requirement (r_9). Lastly, b_{10} represents the AT_LEAST requirement (r_{10}).

The interpretation of terms as Boolean variables permits to easily check whether a plan P is acceptable to the user. Intuitively, to verify whether P is acceptable it is sufficient to check whether the corresponding truth assignment to terms satisfies all the formulas $\mathcal{B}^{\mathcal{R}}$ resulting from the translation of the set $\mathcal{R}$ of requirements. More precisely, given a set T of terms and a plan P with descriptor $\mathrm{P}[A_1, \ldots, A_m]$, we consider P as a truth assignment $\mathrm{P} : T \rightarrow \{0, 1\}$ assigning a Boolean value to the terms in T. Given a term $t \in T$, the truth value assigned by P to t, denoted $\mathrm{P}(t)$, is determined as follows:

$$\mathrm{P}(t) = \begin{cases} 1, \text{if } \mathrm{P}[t.attr] \in t.values \\ 0, \text{otherwise} \end{cases}$$

A cloud plan P *satisfies* a term t iff t is assigned value 1 by P as a truth assignment, as formally defined in the following.

Definition 3.3 (Term satisfaction)**.** *Let* P *be a plan with descriptor* $\mathrm{P}[A_1, \ldots, A_m]$*, t be an attribute term over attribute A_i, with $A_i \in \{A_1, \ldots, A_m\}$. Plan* P satisfies *term t iff* $\mathrm{P}(t) = 1$.

For instance, consider plan P_1 in Figure 2 and two terms of the form t_h = loc(EU, US) and t_k = prov(Mhard). We have that $\mathrm{P}_1(t_h) = 0$, since $\mathrm{P}_1[\mathtt{loc}] = \mathrm{JP}$, and $\mathrm{JP} \notin t_h.values$. On the contrary, $\mathrm{P}_1(t_k) = 1$, since $\mathrm{P}_1[\mathtt{prov}] = \mathrm{Mhard}$, and $\mathrm{MHard} \in t_k.values$. We then say that P_1 does not satisfy t_h, while it satisfies t_k.

Extending the definition above, we say that a plan P satisfies a Boolean formula b over a set of terms defined over attributes $A_x, \ldots, A_y$ iff the evaluation of b over the truth values assigned by P to the terms returns value 1. We denote the evaluation of b over the truth values assigned by P with $\mathrm{P}(b)$. To illustrate, consider formula b_4 : test(authA, authB) $\vee$ cert(certA, certB) in Figure 5 and plan P_3 in Figure 2. Since $\mathrm{P}_3(b_4) = (\mathrm{P}_3(\mathtt{test}(\mathrm{authA}, \mathrm{authB}))) \vee (\mathrm{P}_3(\mathtt{cert}(\mathrm{certA}, \mathrm{certB}))) = (1) \vee (0) = 1$, we have that P_3 satisfies b_4.

We are now ready to formally define an acceptable cloud plan, that is, a plan that satisfies all the user requirements.

Definition 3.4 (Requirement satisfaction)**.** *Let* P *be a plan with descriptor* $\mathrm{P}[A_1, \ldots, A_m]$*, $\mathcal{R}$ be a set of requirements, and $\mathcal{B}^{\mathcal{R}}$ be the set of Boolean formulas representing $\mathcal{R}$. Plan* P satisfies *$\mathcal{R}$, and is said to be* acceptable*, iff* $\forall b \in \mathcal{B}^{\mathcal{R}} : \mathrm{P}(b) = 1$.

A plan P satisfies a set $\mathcal{R}$ of requirements if it satisfies *all* the Boolean formulas $\mathcal{B}^{\mathcal{R}}$ resulting from the translation of $\mathcal{R}$. It is easy to see that, among the plans in Figure 2, only plans $\mathrm{P}_3, \ldots, \mathrm{P}_7$ satisfy the requirements in Figure 3 and are then acceptable. Plan P_1 in fact does not satisfy Boolean formulas $b_{3.1}$, $b_{3.2}$, b_4, b_8, and b_{10} (i.e., it does not satisfy requirements r_3, r_4, r_8, and r_{10}). Plan P_2 instead does not satisfy b_5, b_6, b_8, b_9, and b_{10} (i.e., it does not satisfy requirements r_5, r_6, r_8, r_9, and r_{10}).

Since plans that do not satisfy requirements are not acceptable for the user, we remove them from consideration and, in the following, we will consider only acceptable plans.

As a last remark on our Boolean interpretation of the problem, we note that with the possibility of reasoning in terms of Boolean formulas, we can identify cases in which a set $\mathcal{R}$ of requirements, while being well-defined (Definition 3.2), includes requirements that are in conflict. This can be done by evaluating whether there exists a truth assignment (i.e., a plan in our framing of the problem) respecting the condition in Definition 3.4. As a simple example, consider a well-defined set of requirements $\mathcal{R} = \{r_1, \ldots, r_4\}$, with $r_1 = t_1$, $r_2 = t_2$, $r_3 = t_3$, and $r_4 = \text{AT\_MOST}(2, \{t_1, t_2, t_3\})$. In this case, no cloud plan could ever be considered acceptable. The problem of verifying the existence of an acceptable plan translates into the problem of checking the satisfiability of a set of Boolean formulas, solvable by adopting any off-the-shelf SAT/CSP solver. Indeed, the Boolean formulas to be checked for satisfiability will include all formulas in $\mathcal{B}^{\mathcal{R}}$, and additional formulas that constrain, for each attribute $A \in \mathcal{A}$, at most one term over A (among the terms over A appearing in the constraints) to assume value 1. For example, a truth assignment satisfying all formulas representing a set $\mathcal{R} = \{r_1, r_2\}$ of requirements such that $r_1 = A_1(v_1)$ and $r_2 = \text{ANY}(\{A_2(v_2), A_1(v_3)\})$ could assign value 1 to all the three terms, but would of course not correspond to any meaningful plan since, in a plan, A_1 can assume either value v_1 or v_2 or v_3, but not all of them.

4 USER PREFERENCES

Requirements establish constraints that plans should satisfy to be considered acceptable for the user. Clearly, more plans might exist satisfying such constraints. For instance, plans $\text{P}_3, \ldots, \text{P}_7$ in Figure 2 all satisfy the requirements in Figure 3. Among the acceptable plans, we can expect that some might be preferred over others. Our next contribution is to provide a way for users to express preferences. Our approach will then take preferences into account to produce a preference-based ranking of acceptable plans. Also for preferences, our aim is to be expressive while at the same time maintain simplicity and intuitiveness of specifications.

We consider two levels of specification for preferences, on attribute values in the first place, and then on attributes themselves.

4.1 Preferences on attribute values

Preferences on attribute values allow users to specify, among the values that can be assumed by an attribute, which ones are to be preferred over others. Before defining preferences, we need to define values that can be assumed by an attribute. Such values can be all values of the attribute domain, in case the attribute is not involved in a base or ALL requirement, or the values that the user has specifically indicated as acceptable, otherwise. Given a set of requirements, we then define attribute domains as restricted by requirements as follows.

A	$Dom^{\mathcal{R}}(A)$
`prov`	Mhard, GoGo, Ghost
`loc`	EU, US
`encr`	AES, 3DES
`avail`	VH, H, MH, M, ML
`test`	authA, authB, authC, authD, —
`cert`	certA, certB, certC, certD, certE, —
`aud`	3M, 6M, 9M, 1Y, —

Fig. 6. Restricted domains for the attributes in Figure 1 according to requirements in Figure 3

Definition 4.1 (Restricted domain)**.** *Let $\mathcal{R}$ be a set of requirements over a set $\mathcal{A}$ of attributes. The restricted domain $Dom^{\mathcal{R}}(A)$ of an attribute $A \in \mathcal{A}$ is defined as:*

$$Dom^{\mathcal{R}}(A) = \begin{cases} t.values & \text{if } \exists\, r \in \mathcal{R},\ r = t \text{ or } r = \text{ALL}(T) \\ & \text{with } t \in T \text{and } t.attr = A \\ Dom(A) \cup \{-\} & \text{otherwise} \end{cases}$$

Figure 6 reports the restricted domains for the attributes in Figure 1 assuming the set of requirements in Figure 3.

Among the values that can be assumed by an attribute, we can imagine that a user may consider some values to be - among themselves - equally preferable, while having instead preferences of some values over others. For instance, authB might be preferred over authC and authD for attribute `test`, but either of the latter two may be equally preferable for the user.

Such generic form of preferences can be easily and simply expressed by defining a totally order relationship among sets of attribute values, as captured by the following definition.

Definition 4.2 (Preference relationship)**.** *Let A be an attribute in $\mathcal{A}$ and $\mathcal{P}(A)$ be a partition over the restricted domain $Dom^{\mathcal{R}}(A)$ of A. A* preference relationship *over $Dom^{\mathcal{R}}(A)$, denoted $\succ_A$, is a total order relationship over $\mathcal{P}(A)$.*

For instance, a possible preference relationship over attribute `test` can define partition $\mathcal{P}(\texttt{test}) = \langle\{\text{authA}\}, \{\text{authB}\}, \{\text{authC}, \text{authD}\}, \{-\}\rangle$, with preference relationship $\{\text{authA}\} \succ_{\texttt{test}} \{\text{authB}\} \succ_{\texttt{test}} \{\text{authC}, \text{authD}\} \succ_{\texttt{test}} \{-\}$. Given two partitions $V_i, V_j \in \mathcal{P}(A)$ if $V_i \succ_A V_j$, then all values in V_i are considered preferable over all values in V_j (for simplicity, we apply notation $\succ_A$ also to values, and we write $v_i \succ_A v_j$ to indicate that $v_i \in V_i, v_j \in V_j$ and $V_i \succ_A V_j$). Note that, when special value '—' is included in $Dom^{\mathcal{R}}(A)$, we expect it to appear as the least preferred value, since an unknown value cannot certainly be more preferred than the least preferred among the ones known to the user.

Figure 7 illustrates, via the corresponding Hasse diagrams, an example of preference relationships for the attributes in Figure 1 (we will discuss numbers appearing at the sides of nodes shortly). For instance, considering attribute `loc`, we have that $\{\text{EU}\} \succ_{\texttt{loc}} \{\text{US}\}$, meaning that the user prefers a European location over one in US.

Taking into consideration preferences on values when considering different attributes (as it is the case of our plans) it might happen that, given two plans, none of them has all values preferred over the ones in the other but, while one plan has a more preferred value for an attribute (e.g., authB against authC for testing), the other might have a *much more* preferred value for another attribute (e.g., certA

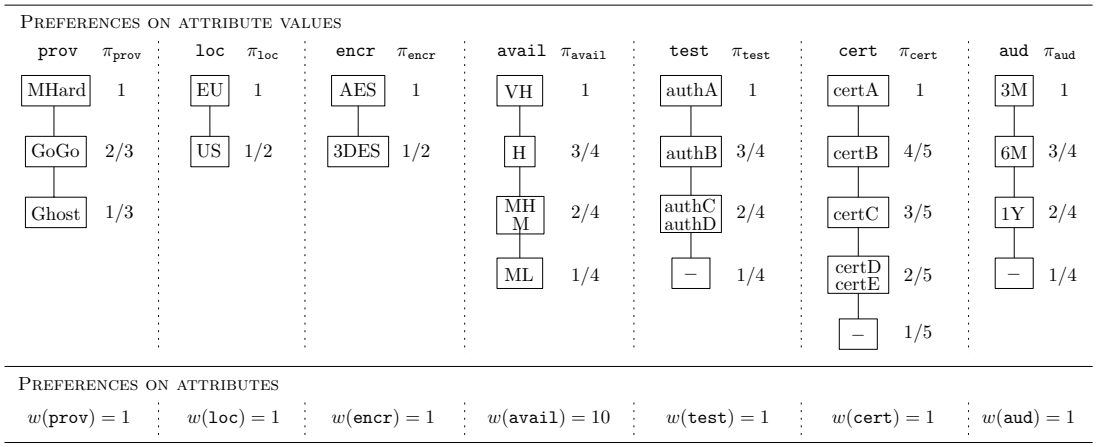


Fig. 7. Preference relationships and score functions over the restricted domains in Figure 6, and weight function over the attributes in Figure 1

over certD for certification). To easily determine when a values is *much more* preferred than another one, we translate preferences over attribute values into *numerical scores* by simply assigning to each value a score reflecting the relative position of the value in the ordering dictated by the preference relationship defined for the attribute. More precisely, the score associated with a value represents the relative closeness of the value to the most preferred value(s) in the restricted domain: the most preferred value(s) is (are) assigned score 1, while the least preferred value(s) is (are) assigned score $1/k$, with k the number of sets in partition $\mathcal{P}(A)$. Formally, such score function is defined as follows.

Definition 4.3 (Score function)**.** *Let A be an attribute in $\mathcal{A}$, and $\succ_A$ be a preference relationship over $Dom^{\mathcal{R}}(A)$. The* score function $\pi_A : Dom^{\mathcal{R}}(A) \rightarrow \mathbb{Z}^+$ *over A associates with each value $v \in Dom^{\mathcal{R}}(A)$ a positive score $\pi_A(v) = 1 - \frac{i}{k}$, with k the cardinality of $\mathcal{P}(A)$ and i the number of sets in $\{V_j \in \mathcal{P}(A) : V_j \succ_A V_l, v \in V_l\}$.*

To illustrate, consider attribute prov and its preference relationship $\succ_{\texttt{prov}}$ in Figure 7. The preference relationship distinguishes $k = 3$ sets, and scores will then be: $\pi_{\texttt{prov}}(\text{MHard}) = 1 = 1 - \frac{0}{3}$ ($i = 0$, most preferred), $\pi_{\texttt{prov}}(\text{GoGo}) = 2/3 = 1 - \frac{1}{3}$ ($i = 1$, second most preferred), $\pi_{\texttt{prov}}(\text{Ghost}) = 1/3 = 1 - \frac{2}{3}$ ($i = 2$, third and least preferred).

4.2 Preferences on attributes

Preferences on values, and corresponding score functions, allow us to reason about how (and how much) better a plan is with respect to another one with reference to each specific attribute. To reason about which plans can be considered better than others, it is clearly important to take into consideration also the specific attributes on which plans assume better values. For instance, a plan might be better than another one with respect to availability but worse with respect to encryption. Which one is to be preferred depends then, besides on how much better the two are for such attributes, on which attribute (either availability or encryption) the user cares more. For instance, a user outsourcing sensitive data may give more importance to security-related attributes (e.g., encryption) while a user outsourcing high-performance applications may consider performance-related attributes (e.g., availability) more important than all the other attributes. To take this into consideration, we allow users to specify how important attributes are for them. For attributes, we consider importance to be specified as a *weight* (number) that the user assigns to each attribute: attributes with equal weights are considered equally important, while attributes with greater weights are considered more important. Weights are formally defined as follows[1].

Definition 4.4 (Weight function)**.** *Let $\mathcal{A}$ be a set of attributes. The* weight function $w : \mathcal{A} \rightarrow \mathbb{N}^+$ *associates a positive weight with each attribute in $\mathcal{A}$.*

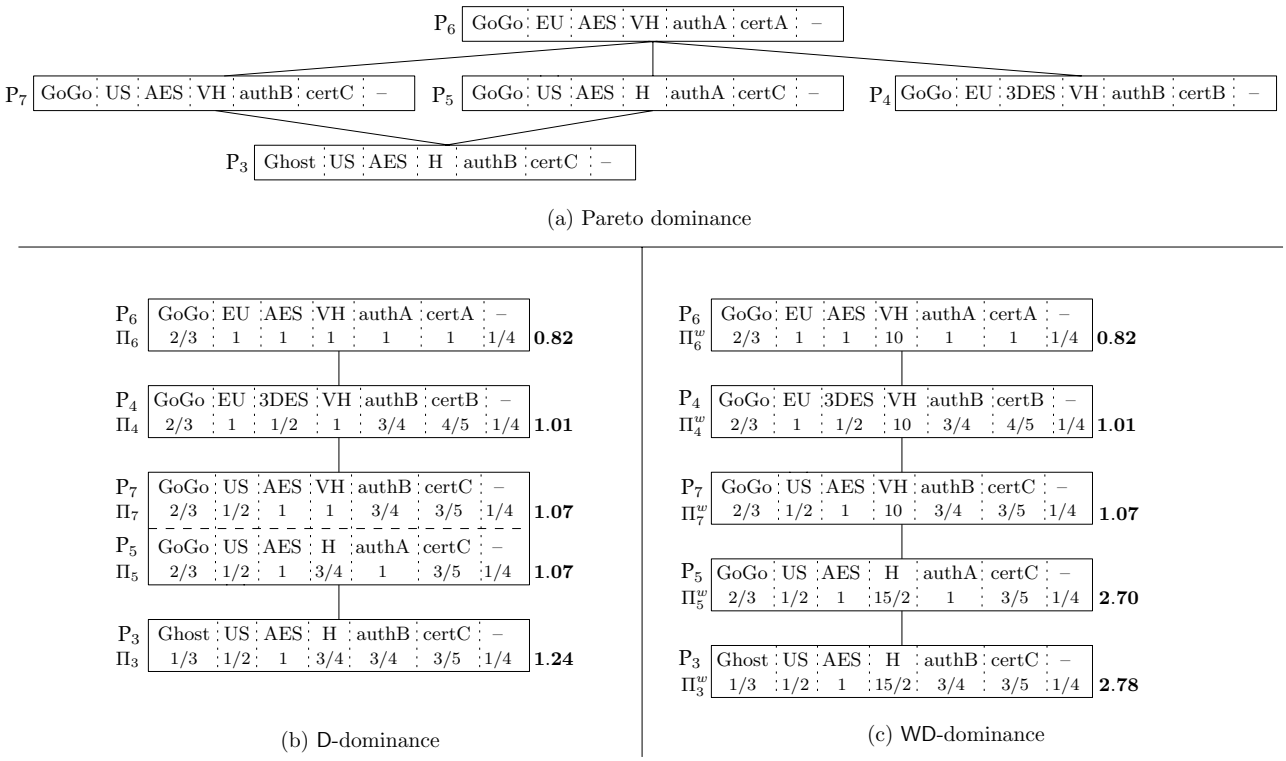
The reason for considering weights instead of a simple preference relationship like for values is that, while for values generic weights might be cumbersome and probably not that intuitive (also considering the different cardinalities of attribute domains), for attributes explicit support for user-defined weights adds expressiveness without introducing complexity. Note also that weights could be either explicitly specified or derived from an order relationship specified among attributes. This can be supported, for example, by simply assigning weight 1 to the least relevant attribute and increasing the weight by 1 at each step of such order relationship.

Figure 7 reports a possible weight function for the attributes of our example, identifying availability as the most (by and large) relevant attribute, with $w(\texttt{avail}) = 10$, and giving equally low importance to the other attributes (all with weight 1). Note that the case where all the attributes are considered equally important, or their relative importance is not a significant aspect for the user, can be modeled simply by assigning the same weight to all attributes.

5 PLAN RANKING

Having defined value and attribute preferences, we are now ready to discuss how to take them into consideration for

1. For simplicity, we assume the weight function to assume values in $\mathbb{N}^+$. We note that other co-domains could be used, with the only restriction that the smallest weight be greater than or equal to 1.



(a) Pareto dominance

(b) D-dominance

(c) WD-dominance

Fig. 8. Rankings of plans $P_3, \ldots, P_7$ in Figure 2 that satisfy the requirements in Figure 3

ranking plans. We introduce our approach to ranking incrementally, identifying and introducing concepts that dictate dominance among plans. We first identify the following Pareto dominance relationship as the natural extension of value preferences.

Definition 5.1 (Pareto dominance)**.** *Let P_i, P_j be two acceptable plans with respect to a set $\mathcal{R}$ of requirements over set $\mathcal{A}$ of attributes and, $\forall A \in \mathcal{A}$, $\succ_A$ be the preference relationship over $Dom^{\mathcal{R}}(A)$.* P_i Pareto-dominates P_j*, denoted $P_i \succ_P P_j$, iff $\forall A \in \mathcal{A}, P_i[A] \succeq_A P_j[A]$, and $\exists A_l \in \mathcal{A}$ such that $P_i[A_l] \succ_{A_l} P_j[A_l]$.*

According to Definition 5.1, P_i dominates P_j (in other words, P_i is preferred –higher in ranking– over P_j) if P_i has, for all attributes, values that are equally or more preferred than those in P_j and, for at least one attribute, a more preferred value than the one in P_j.

Figure 8(a) illustrates the Pareto dominance relationship among plans $P_3, \ldots, P_7$ of our running example.

Pareto dominance, while natural, provides only an initial step for defining a ranking since all plans that have reversed relationships for at least two attributes will remain not ordered (incomparable). For instance, in Figure 8(a), plans P_3 and P_4 remain incomparable (i.e., $P_3 \not\succ_P P_4$ and $P_4 \not\succ_P P_3$), since one has a more preferred availability ($P_3[\texttt{avail}] \succ_{\texttt{avail}} P_4[\texttt{avail}]$) while the other has a more preferred security certification ($P_4[\texttt{cert}] \succ_{\texttt{cert}} P_3[\texttt{cert}]$). A similar reasoning also applies to P_7 and P_5, P_7 and P_4, and P_5 and P_4.

We then define a strategy for ranking plans taking into consideration the importance of values (i.e., their scores), and possibly also the importance of attributes (i.e., their weights) as specified by the user. Our approach for ranking is based on the distance of a plan from an *ideal plan*, that is, a plan where all attributes assume the most preferred values.

An *ideal* plan is defined as follows.

Definition 5.2 (Ideal plan)**.** *Let $\mathcal{A}$ be a set of attributes and $\forall A \in \mathcal{A}$, $\succ_A$ be the preference relationship over $Dom^{\mathcal{R}}(A)$. An* ideal plan*, denoted $P_\top$, is a plan such that $\forall A \in \mathcal{A}$, $\forall v \in Dom^{\mathcal{R}}(A)$, $P_\top[A] \succeq_A v$.*

For instance, with reference to our running example, there is a single ideal plan, that is $[\text{Mhard}, \text{EU}, \text{AES}, \text{VH}, \text{authA}, \text{certA}, \text{3M}]$. Note that more than one ideal plan may exist (as many as the possible different combinations of most preferred values for the attributes in $\mathcal{A}$).

To define how distant a generic plan is with respect to an ideal one, we associate with each plan the score vector containing the scores of the plan's attribute values as follows.

Definition 5.3 (Score vector)**.** *Let $\mathcal{A}$ be a set of m attributes, P_j be an acceptable plan with respect to a set $\mathcal{R}$ of requirements over a set $\mathcal{A}$ of attributes, and π_A be a score function over A, $\forall A \in \mathcal{A}$. The* score vector *associated with P_j is a vector $\Pi_j[A_1, \ldots, A_m]$, with $\Pi_j[A_i] = \pi_{A_i}(P_j[A_i]), \forall A_i \in \mathcal{A}$.*

Note that, by definition, the score vector $\Pi_\top$ of an ideal plan $P_\top$ has all values equal to 1.

We are now ready to introduce our ranking based on the distance from an ideal plan. For simplicity, we first introduce

the ranking without taking into account attribute weights and then extend it to consider them.

To measure how much a cloud plan P differs from an ideal plan $P_\top$, we interpret P as a point in an m-dimensional space (with m the number of attributes in $\mathcal{A}$), and its score vector Π as its Cartesian coordinates in the space. For each cloud plan P, we then measure the *distance* between the point corresponding to P and the one corresponding to $P_\top$. While noting that any notion of distance in an m-dimensional space would work, we use the *Euclidean distance*. Let us recall that, in our framing of the problem, the Euclidean distance between a pair of generic points P_h and P_k characterized (i.e., located in the m-dimensional space) by score vectors Π_h and Π_k is defined as:

$$\mathsf{dist}(\Pi_h, \Pi_k) = \sqrt{\sum_{i=1}^{m} (\Pi_h[A_i] - \Pi_k[A_i])^2}$$

We then define a dominance relationship D-dominance (D for distance) among plans based on such a distance from $P_\top$ as follows.

Definition 5.4 (D-dominance)**.** *Let P_i and P_j be two acceptable plans with respect to a set $\mathcal{R}$ of requirements. P_i D-dominates P_j, denoted $P_i \succ_\mathsf{D} P_j$, iff $\mathsf{dist}(\Pi_i, \Pi_\top) < \mathsf{dist}(\Pi_j, \Pi_\top)$, where dist is the Euclidean distance of the score vectors Π_i, Π_j of the given plans, from $\Pi_\top$.*

Figure 8(b) illustrates the D-dominance among the acceptable plans of our running example where, for each plan P_i, we report value $\mathsf{dist}(\Pi_i, \Pi_\top)$ in boldface on the right of the node representing P_i and Π_i. The D-dominance among the acceptable plans implies a ranking from the plan closest (P_6, with $\mathsf{dist}(\Pi_6, \Pi_\top) = 0.82$) to the farthest ($P_3$, with $\mathsf{dist}(\Pi_3, \Pi_\top) = 1.24$) from the ideal one. Note that for P_5 and P_7 the D-dominance is not defined, both being at distance 1.07 from the ideal plan. On this latter observation, let us note that (unlike the Pareto dominance) whenever $\succ_\mathsf{D}$ in Definition 5.4 is not defined for a pair of plans P_x and P_y (i.e., $P_x \not\succ_\mathsf{D} P_y$ and $P_y \not\succ_\mathsf{D} P_x$), we have the guarantee that the distances of the score vectors of P_x and P_y from that of the ideal plan are *equal*, and we can therefore conclude that P_x and P_y can be regarded as being *equivalent* w.r.t. the value preferences formulated by the user. In fact, the distances can be equal only if: *i)* P_x and P_y, while being different plans, have the same descriptor; or *ii)* P_x and P_y have different descriptors but the 'good' values for some attributes are compensated by the 'bad' values for other attributes (i.e., the reversed relationships that caused incomparable results for the Pareto dominance). Referring to our running example, since $\mathsf{dist}(\Pi_5, \Pi_\top) = \mathsf{dist}(\Pi_7, \Pi_\top) = 1.07$, both P_5 and P_7 have the same position in the ranking induced by $\succ_\mathsf{D}$. In fact, it is easy to see that, despite having the same values for attributes `prov`, `loc`, `encr`, `cert`, and `aud`, the better value assumed by P_7 for `avail` (i.e., VH as opposed to H for P_5) is 'compensated' by the worse value assumed for `test` (i.e., authB as opposed to authA for P_5).

D-dominance captures the distance among plans considering all attributes equally valuable. To account for priority among attributes, as specified by the user with corresponding weights (Definition 4.4), we need to adjust the definition of distance accordingly. We do so by scaling the

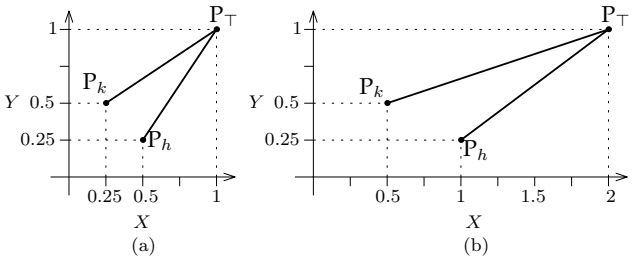


Fig. 9. Graphical representation of the Euclidean distance between P_h, P_k and $P_\top$ before (a) and after (b) the scaling by a factor 2 on the x-axis

m-dimensional space using, as a scaling factor along each dimension (attribute), the weight assigned by the user to the attribute. Scaling stretches the axes representing the attributes according to their weights (i.e., anisotropic scaling). Every point in the original m-dimensional space moves in the scaled space accordingly. More precisely, each element of a score vector is multiplied by the weight assigned by the user to the corresponding attribute. In the following, we use notation Π^w to denote the score vector after the scaling operation performed according to weight function w. This is captured by the following definition of *weighted score vector*.

Definition 5.5 (Weighted score vector)**.** *Let $\mathcal{A}$ be a set of m attributes, P_j be an acceptable cloud plan with respect to a set $\mathcal{R}$ of requirements over a set $\mathcal{A}$ of attributes, Π_j be the score vector of P_j, and w be a weight function over $\mathcal{A}$. The* weighted score vector *associated with P_j is a vector $\Pi_j^w[A_1, \ldots, A_m]$, with $\Pi_j^w[A_i] = \Pi_j[A_i] \cdot w(A_i), \forall A_i \in \mathcal{A}$.*

To illustrate the concept, consider a simple hypothetical bi-dimensional space and two plans P_h and P_k with $\Pi_h = [1/2, 1/4]$ and $\Pi_k = [1/4, 1/2]$. Figure 9(a) illustrates their position in the corresponding bi-dimensional space, where the x-axis corresponds to the first attribute X, the y-axis corresponds to the second attribute Y, and point $\Pi_\top = [1, 1]$ corresponds to ideal plan $P_\top$. The two plans have the same distance (0.90) from $P_\top$, hence neither dominates the other with respect to D-dominance. Let us now take into consideration attribute weights and assume that $w(X) = 2$ and $w(Y) = 1$, meaning that attribute X is considered more important than (twice as important as) attribute Y. Using $w(X)$ and $w(Y)$ as scaling factors along the dimensions corresponding to attributes X and Y, respectively, the points in the space are moved along the x-axis by a factor of 2. The weighted score vectors representing the positions of the plans in the anisotropic scaled space are: $\Pi_h^w = [1, 1/4]$, $\Pi_k^w = [1/2, 1/2]$, with $\Pi_\top^w = [2, 1]$ (Figure 9(b)). The distances between the corresponding plans are then $\mathsf{dist}(\Pi_h, \Pi_\top) = 1.25$ and $\mathsf{dist}(\Pi_k, \Pi_\top) = 1.58$, with dist now computed over the weighted score vectors, differentiating P_h as closer to the ideal plan, and hence preferable over P_k. This consistently with the fact that P_h is closer to $P_\top$ on the dimension representing attribute X, which is considered more important. Taking into account the weighted score vectors, we formally define a WD-dominance (W for weight) among plans based on weighted score vectors as follows.

Definition 5.6 (WD-dominance)**.** *Let P_i and P_j be two acceptable plans with respect to a set $\mathcal{R}$ of requirements, and w be*

a weight function. P_i **WD**-*dominates* P_j, *denoted* $\mathrm{P}_i \succ_{\mathsf{WD}} \mathrm{P}_j$, *iff* $\mathsf{dist}(\Pi_i^w, \Pi_\top^w) < \mathsf{dist}(\Pi_j^w, \Pi_\top^w)$, *where* **dist** *is the Euclidean distance of the weighted score vectors* Π_i^w, Π_j^w *of the given plans, from* $\Pi_\top^w$.

Figure 8(c) reports the **WD**-dominance relationships among the acceptable plans of our running example assuming attribute weights as in Figure 7, where $w(\texttt{avail}) = 10$ while the weight of all other attributes is 1. The **WD**-dominance relationship now identifies P_7 as closer to the ideal plan than P_5, and hence preferable.

We note that when all attributes have weight 1, no scaling is performed, the weighted score vectors Π^w are exactly the original score vectors Π, and **WD**-dominance reduces to **D**-dominance. Note also that, clearly, both relationships respect the Pareto dominance. In fact, given any two plans P_i and P_j, any preference relationship, and any weight function, if P_i Pareto-dominates P_j, then plan P_i **D**-dominates and **WD**-dominates P_j. Also, if P_i **D**-dominates or **WD**-dominates P_j, then it cannot be that P_j Pareto-dominates P_i. This property is formalized by the following theorem.

Theorem 5.1. *Let* $\mathcal{A}$ *be a set of attributes, and* P_i, P_j *be two acceptable plans with respect to a set* $\mathcal{R}$ *of requirements. For any score function and for any weight function, the following properties hold:*

1) $\mathrm{P}_i \succ_{\mathsf{P}} \mathrm{P}_j \Longrightarrow \mathrm{P}_i \succ_{\mathsf{D}} \mathrm{P}_j$ *and* $\mathrm{P}_i \succ_{\mathsf{WD}} \mathrm{P}_j$*;*
2) $\mathrm{P}_i \succ_{\mathsf{D}} \mathrm{P}_j$ *or* $\mathrm{P}_i \succ_{\mathsf{WD}} \mathrm{P}_j \Longrightarrow \mathrm{P}_j \not\succ_{\mathsf{P}} \mathrm{P}_i$.

Proof. We prove the two properties separately.

1) $\mathrm{P}_i \succ_{\mathsf{P}} \mathrm{P}_j \Longrightarrow \mathrm{P}_i \succ_{\mathsf{D}} \mathrm{P}_j$.
Suppose, by contradiction, that $\exists \mathrm{P}_i, \mathrm{P}_j$ such that $\mathrm{P}_i \succ_{\mathsf{P}} \mathrm{P}_j$ and $\mathrm{P}_i \not\succ_{\mathsf{D}} \mathrm{P}_j$. Since $\mathrm{P}_i \succ_{\mathsf{P}} \mathrm{P}_j$, we have that (Definition 5.1): *i)* $\forall A \in \mathcal{A}, \mathrm{P}_i[A] \succeq_A \mathrm{P}_j[A]$; and *ii)* $\exists A_l \in \mathcal{A}$ such that $\mathrm{P}_i[A_l] \succ_{A_l} \mathrm{P}_j[A_l]$. By Definition 4.3, this in turn implies that: *i)* $\forall A \in \mathcal{A}, \Pi_i[A] \geq \Pi_j[A]$; and *ii)* $\exists A_l \in \mathcal{A}, \Pi_i[A_l] > \Pi_j[A_l]$. Therefore, $\mathsf{dist}(\Pi_i, \Pi_\top) < \mathsf{dist}(\Pi_j, \Pi_\top)$, resulting into $\mathrm{P}_i \succ_{\mathsf{D}} \mathrm{P}_j$ and hence contradicting our hypothesis.

$\mathrm{P}_i \succ_{\mathsf{P}} \mathrm{P}_j \Longrightarrow \mathrm{P}_i \succ_{\mathsf{WD}} \mathrm{P}_j$.
Suppose, by contradiction, that $\exists \mathrm{P}_i, \mathrm{P}_j, w$ such that $\mathrm{P}_i \succ_{\mathsf{P}} \mathrm{P}_j$ and $\mathrm{P}_i \not\succ_{\mathsf{WD}} \mathrm{P}_j$. Again, since $\mathrm{P}_i \succ_{\mathsf{P}} \mathrm{P}_j$, we have that (Definition 5.1): *i)* $\forall A \in \mathcal{A}, \mathrm{P}_i[A] \succeq_A \mathrm{P}_j[A]$; and *ii)* $\exists A_l \in \mathcal{A}$ such that $\mathrm{P}_i[A_l] \succ_{A_l} \mathrm{P}_j[A_l]$. By Definition 4.3, this in turn implies that: *i)* $\forall A \in \mathcal{A}, \Pi_i[A] \geq \Pi_j[A]$; and *ii)* $\exists A_l \in \mathcal{A}, \Pi_i[A_l] > \Pi_j[A_l]$. Since by Definition 4.4 $\forall A \in \mathcal{A}, \forall w : w(A) \in \mathbb{N}^+$ and by Definition 4.3 $\forall A \in \mathcal{A}, \forall v \in Dom^{\mathcal{R}}(A) : \pi_A(v) \in \mathbb{Z}^+$, we have that $\forall A \in \mathcal{A}, \forall v \in Dom^{\mathcal{R}}(A), \forall w : \Pi_i[A] = \Pi_j[A] \Longrightarrow w(A) \cdot \Pi_i[A] = w(A) \cdot \Pi_j[A]$, and that $\forall A \in \mathcal{A}, \forall v \in Dom^{\mathcal{R}}(A), \forall w : \Pi_i[A] > \Pi_j[A] \Longrightarrow w(A) \cdot \Pi_i[A] > w(A) \cdot \Pi_j[A]$. Therefore, $\mathsf{dist}(\Pi_i^w, \Pi_\top^w) < \mathsf{dist}(\Pi_j^w, \Pi_\top^w)$, resulting into $\mathrm{P}_i \succ_{\mathsf{WD}} \mathrm{P}_j$ and hence contradicting our hypothesis.

2) $\mathrm{P}_i \succ_{\mathsf{D}} \mathrm{P}_j \Longrightarrow \mathrm{P}_j \not\succ_{\mathsf{P}} \mathrm{P}_i$.
Suppose, by contradiction, that $\exists \mathrm{P}_i, \mathrm{P}_j$ such that $\mathrm{P}_i \succ_{\mathsf{D}} \mathrm{P}_j$ and $\mathrm{P}_j \succ_{\mathsf{P}} \mathrm{P}_i$. Since $\mathrm{P}_j \succ_{\mathsf{P}} \mathrm{P}_i$, then (as proved for Property 1 of the theorem) we have that $\mathrm{P}_j \succ_{\mathsf{D}} \mathrm{P}_i$. This would result in $\mathrm{P}_i \succ_{\mathsf{D}} \mathrm{P}_j \Longrightarrow \mathrm{P}_j \succ_{\mathsf{P}} \mathrm{P}_i \Longrightarrow \mathrm{P}_j \succ_{\mathsf{D}} \mathrm{P}_i$, which is a contradiction.

$\mathrm{P}_i \succ_{\mathsf{WD}} \mathrm{P}_j \Longrightarrow \mathrm{P}_j \not\succ_{\mathsf{P}} \mathrm{P}_i$.
Suppose, by contradiction, that $\exists \mathrm{P}_i, \mathrm{P}_j, w$ such that $\mathrm{P}_i \succ_{\mathsf{WD}} \mathrm{P}_j$ and $\mathrm{P}_j \succ_{\mathsf{P}} \mathrm{P}_i$. Again, since $\mathrm{P}_j \succ_{\mathsf{P}} \mathrm{P}_i$, then (as proved for Property 1 of the theorem) we have that $\mathrm{P}_j \succ_{\mathsf{WD}} \mathrm{P}_i$. This would result in $\mathrm{P}_i \succ_{\mathsf{WD}} \mathrm{P}_j \Longrightarrow \mathrm{P}_j \succ_{\mathsf{P}} \mathrm{P}_i \Longrightarrow \mathrm{P}_j \succ_{\mathsf{WD}} \mathrm{P}_i$, which is a contradiction. □

We also note that if P_i **D**-dominates or **WD**-dominates P_j, we cannot say anything about whether P_i also Pareto-dominates P_j (in fact, Property 2 of Theorem 5.1 elaborates on whether P_j Pareto-dominates P_i). As an example, consider plans P_4 and P_5. It is easy to see that P_4 **D**-dominates and **WD**-dominates P_5 but they are not comparable with respect to the Pareto dominance relationship. We finally note that neither **D**-dominance implies **WD**-dominance nor the opposite. Indeed, a plan that is preferred according to **D**-dominance could be considered worse according to **WD**-dominance and viceversa. To illustrate, consider a simple hypothetical bi-dimensional space and two plans P_h and P_k such that $\Pi_h = [0.24, 0.5]$ and $\Pi_k = [0.5, 0.25]$. By evaluating the Euclidean distance between these plans and the ideal plan $\mathrm{P}_\top$, it is easy to see that $\mathrm{P}_k \succ_{\mathsf{D}} \mathrm{P}_h$, since $\mathsf{dist}(\Pi_k, \Pi_\top) = 0.9013 < \mathsf{dist}(\Pi_h, \Pi_\top) = 0.9097$. Now consider a weight function w such that $w(A_1) = 1, w(A_2) = 2$. We have now that $\mathsf{dist}(\Pi_k^w, \Pi_\top^w) = 1.5811 > \mathsf{dist}(\Pi_h^w, \Pi_\top^w) = 1.2560$. Hence, we have that $\mathrm{P}_k \succ_{\mathsf{D}} \mathrm{P}_h$, and $\mathrm{P}_h \succ_{\mathsf{WD}} \mathrm{P}_k$.

We close this section with a note on the complexity of the process of ranking a set of acceptable plans. The final ranking to be returned to the user reflects the dominance relationships existing among the plans. The complexity of computing such a ranking then depends on the adopted dominance relationship. Ranking a set of n plans according to the Pareto dominance (Definition 5.1) requires, in the worst case, to establish the dominance relationship between all pairs of plans, and hence its computational cost is quadratic in the number n of acceptable plans (i.e., $O(n^2)$). Ranking a set of n plans according to the **D**-dominance (Definition 5.4) or the **WD**-dominance (Definition 5.6) requires instead to compute, for each plan, the Euclidean distance between its (weighted) score vector and the (weighted) score vector of the ideal plan $\mathrm{P}_\top$. The computation of such distances has linear cost in the number n of acceptable plans, and the computation of the ranking itself can be performed simply by sorting the plans based on the computed distances (i.e., $O(n\log(n))$).

We also note that, while computation of acceptable plans and their ranking are assumed to happen once (i.e., at the time of cloud plan selection by the user), changes in specifications can be taken into consideration incrementally. For requirements: *i)* deletion of a requirement simply requires evaluating the remaining requirements against the plans not considered acceptable before to see if some of them are now acceptable, and placing them in the ranking depending on the adopted dominance relationship (nothing needs to be done on the acceptable plans); *ii)* insertion of a requirement simply requires to evaluate the new requirement on the acceptable plans and discard plans that do not

satisfy it (as they are not acceptable anymore); *iii)* update of a requirement can be interpreted as a deletion followed by an insertion. For preferences, changes clearly require to recompute dominance relationships among plans, but do not change the acceptability of plans (and then do not require to re-evaluate requirements). For plans: *i)* deletion of a plan simply requires deleting it from the solution; *ii)* insertion of a new plan simply requires evaluating the plan against the requirements and, if it is acceptable, placing it in the ranking according to dominance relationships; *iii)* changes in a plan can be interpreted as a deletion followed by an insertion.

6 Related work

The problem of cloud provider or cloud plan selection has been widely recognized and several solutions have been proposed to support users in their choice. Before ranking and selecting a cloud plan (or cloud provider), users need to measure and compare the guarantees offered by cloud providers. Different works have addressed this problem, proposing techniques for measuring, for example, performance, costs, and Quality of Service offered to final users (e.g., [5], [7], [11]), possibly using also feedbacks by users and/or by a trusted third party (e.g., [12], [13], [14]). These proposals are complementary to the work presented in this paper as the information produced by them (i.e., the value of some properties) could be used as input to our approach.

The line of work closest to ours focuses on the problem of ranking and/or selecting cloud services that satisfy user requirements. Some proposals are based on the adoption of a third party (broker [15]) that helps users in selecting the "best service" according to their requirements. In [6] the authors present a brokerage-based approach for selecting cloud services that takes into consideration which properties and values a user desires from service providers as well as the order of importance of the properties. Apart from the similarity with our approach in considering user requirements and preferences, our approach is more general and flexible. As a matter of fact, we consider a variety of requirements in contrast to the proposal in [6] that considers only conjunctions of requirements defined over a limited number of properties. Also, our approach supports different kinds of preferences. In [16], the authors present a cloud broker service that uses high-level requirements expressed as objectives for selecting cloud elements/services from multiple providers to deploy an application. Our work has however a different focus. In [17], the authors propose a solution enabling users to express their requirements through an SLA template that is then matched to the SLAs of a set of candidate providers. Our approach differs from it since we consider generic requirements as well as preferences, and propose different strategies for ranking the cloud plans. In [18], the authors propose a brokerage-based system supporting user requirements on QoS levels (e.g., response time and throughput, which should be minimized or maximized) to find optimal compositions of web services. The work shares with ours the aim to provide a broker supporting users in service selection. However, the scenario, final goal, and requirements (QoS levels in [18] and generic requirements and preference in our work) are different.

Some proposals reduce the problem of ranking cloud services based on multiple attributes (in particular, QoS requirements) to a problem of Multi-Criteria Decision-Making (MCDM) and use specific approaches (e.g., the Analytic Hierarchy Process) to solve it (e.g., [7], [19]), possibly taking into consideration also the implicit vagueness in certain requirements (e.g., [20], [21]). Although these solutions consider some forms of user requirements and preferences, our approach is more expressive as we support requirements expressed as Boolean formulas and preferences at both attribute and value level.

The problem addressed in this paper also resembles the more general problem of selecting cloud providers under specific circumstances (e.g., [22], [23], [24], [25], [26]). In [22], the authors present an approach for ranking cloud providers according to their ability in satisfying the requirements of multiple applications. In [23], the authors present a solution for allocating virtual machines to cloud providers so that all user and cloud provider requirements on their placement are satisfied. In [24], the authors illustrate a solution for selecting a combination of cloud services according to users' preferences expressed though fuzzy requirements. In [25], [26], the authors show a solution for identifying an SLA between a user and a cloud provider that satisfies user requirements as well as dependencies among different properties. While interesting, all these proposals are complementary to our work, which focuses on the definition of an expressive and flexible model for user requirement specification and for cloud plan ranking.

7 Conclusions

To fully benefit from the wide availability of a multitude of cloud providers and plans in the growing cloud market, it is important to provide users with support for expressing their needs and taking them into consideration in cloud plan selection. Our work makes a step forward in this direction by providing a flexible and expressive approach that enables users to specify requirements they want to be satisfied on plans as well as preferences they may have on plan characteristics. Our approach identifies different kinds of requirements, which users can define in a simple and intuitive manner, and ensures their enforcement and consideration in the identification of acceptable plans, as well as in producing a preference-based ranking.

Acknowledgments

This work was supported in part by the EC within the H2020 under grant agreement 644579 (ESCUDO-CLOUD), and within the FP7 under grant agreement 312797 (ABC4EU).

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.

[2] H. Takabi, J. B. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Security & Privacy*, vol. 8, no. 6, pp. 24–31, 2010.

[3] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *IEEE TSC*, vol. 5, no. 2, pp. 220–232, 2012.

[4] Gartner, Inc., "Gartner says by 2020 "cloud shift" will affect more than $1 trillion in it spending," http://www.gartner.com/newsroom/id/3384720, 2016.

[5] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: Comparing public cloud providers," in *Proc. of ACM IMC 2010*, Melbourne, Australia, November 2010.

[6] S. Sundareswaran, A. Squicciarini, and D. Lin, "A brokerage-based approach for cloud service selection," in *Proc. of IEEE CLOUD 2012*, Honolulu, HI, USA, June 2012.

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

[8] 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.

[9] Cloud Security Alliance – Consensus Assessments Initiative, https://cloudsecurityalliance.org/group/consensus-assessments/.

[10] S. De Capitani di Vimercati, S. Foresti, G. Livraga, and P. Samarati, "Supporting users in data outsourcing and protection in the cloud," in *Cloud Computing and Services Science*, M. Helfert, D. Ferguson, V. Munoz, and J. Cardoso, Eds. Springer, 2017, pp. 3–15.

[11] A. Lenk, M. Menzel, J. Lipsky, S. Tai, and P. Offermann, "What are you paying for? Performance benchmarking for Infrastructure-as-a-Service offerings," in *Proc. of IEEE CLOUD 2011*, Washington DC, USA, July 2011.

[12] N. Ghosh, S. K. Ghosh, and S. K. Das, "SelCSP: A framework to facilitate selection of cloud service providers," *IEEE TCC*, vol. 3, no. 1, pp. 66–79, 2015.

[13] Z. Zheng, X. Wu, Y. Zhang, M. R. Lyu, and J. Wang, "QoS ranking prediction for cloud services," *IEEE TPDS*, vol. 24, no. 6, pp. 1213–1222, 2013.

[14] L. Qu, Y. Wang, and M. A. Orgun, "Cloud service selection based on the aggregation of user feedback and quantitative performance assessment," in *Proc. of IEEE SCC 2013*, Santa Clara, CA, USA, June-July 2013.

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

[16] P. Pawluk, B. Simmons, M. Smit, M. Litoiu, and S. Mankovski, "Introducing STRATOS: A cloud broker service." in *Proc. of IEEE CLOUD 2012*, Honolulu, HI, USA, June 2012.

[17] C. Redl, I. Breskovic, I. Brandic, and S. Dustdar, "Automatic SLA matching and provider selection in grid and cloud computing markets," in *Proc. of ACM/IEEE GRID 2010*, Beijing, China, September 2012.

[18] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end QoS constraints," *ACM TWEB*, vol. 1, no. 1, 2007.

[19] R. Karim, C. Ding, and A. Miri, "An end-to-end QoS mapping approach for cloud service selection," in *Proc. of IEEE SERVICES 2013*, Santa Clara, CA, USA, June-July 2013.

[20] I. Patiniotakis, Y. Verginadis, and G. Mentzas, "PuLSaR: Preference-based cloud service selection for cloud service brokers," *JISA*, vol. 6, no. 26, pp. 1–14, 2015.

[21] I. Patiniotakis, S. Rizou, Y. Verginadis, and G. Mentzas, "Managing imprecise criteria in cloud service ranking with a fuzzy multi-criteria decision making method," in *Proc. of ESOCC 2013*, Málaga, Spain, September 2013.

[22] A. Arman, S. Foresti, G. Livraga, and P. Samarati, "A consensus-based approach for selecting cloud plans," in *Proc. of IEEE RTSI 2016*, Bologna, Italy, September 2016.

[23] 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.

[24] 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.

[25] S. De Capitani di Vimercati, G. Livraga, V. Piuri, P. Samarati, and G. Soares, "Supporting application requirements in cloud-based IoT information processing," in *Proc. of IoTBD 2016*, Rome, Italy, April 2016.

[26] S. De Capitani di Vimercati, G. Livraga, and V. Piuri, "Application requirements with preferences in cloud-based information processing," in *Proc. of IEEE RTSI 2016*, Bologna, Italy, September 2016.

Sabrina De Capitani di Vimercati is a professor at the Computer Science Department, Università degli Studi di Milano, Italy. Her research interests are in data security and privacy. She has published more than 210 papers in journals, conference proceedings, and books. She has been a visiting researcher at SRI International, CA (USA), and George Mason University, VA (USA). She chairs the IFIP WG 11.3 on Data and Application Security and Privacy.
http://www.di.unimi.it/decapita

Sara Foresti is an associate professor at the Computer Science Department, Università degli Studi di Milano, Italy. Her research interests are in data security and privacy. She has published more than 90 papers in journals, conference proceedings, and books. She has been a visiting researcher at George Mason University, VA (USA). She has been serving as General chair and PC chair of several conferences. She chairs the IEEE CS Italy Chapter.
http://www.di.unimi.it/foresti

Giovanni Livraga is an assistant professor at the Computer Science Department, Università degli Studi di Milano, Italy. His research interests are in data privacy and security in emerging scenarios. His PhD thesis received the ERCIM STM WG 2015 award. He has been a visiting researcher at SAP Labs, France and George Mason University, VA (USA). He has been serving as PC member for several conferences.
http://www.di.unimi.it/livraga

Vincenzo Piuri is a professor at the Computer Science Department, Università degli Studi di Milano, Italy. His main research interests are in signal and image processing, biometrics, intelligent systems, digital and signal processing architectures. He has published in more than 350 papers in journals, conference proceedings, and books. He has been named IEEE Fellow (2001) and ACM Distinguished Scientist (2008).
http://www.di.unimi.it/piuri

Pierangela Samarati is a professor at the Computer Science Department, Università degli Studi di Milano, Italy. Her main research interests are in data protection, security, and privacy. She has participated in/coordinated several projects. She has published more than 260 papers in journals, conference proceedings, and books. She has been named ACM Distinguished Scientist (2009) and IEEE Fellow (2012).
http://www.di.unimi.it/samarati