IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 19, 2021, accepted March 5, 2021, date of publication April 8, 2021, date of current version April 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3071646

One-to-Many Negotiation QoE Management Mechanism for End-User Satisfaction

AMRO NAJJAR[1], YAZAN MUALLA[2], KAMAL DEEP SINGH[3], GAUTHIER PICARD[4], DAVIDE CALVARESI[5], AVLEEN MALHI[6], STÉPHANE GALLAND[2], AND KARY FRÄMLING[6], (Member, IEEE)

[1]ICR/AI Robolab, DCS, University of Luxembourg, 4365 Esch sur alzette, Luxembourg
[2]Connaissance et Intelligence Artificielle Distribuées (CIAD), Université Bourgogne Franche-Comté, UTBM, 90010 Belfort, France
[3]Laboratoire Hubert Curien UMR CNRS 5516, 42000 Saint-Étienne, France
[4]ONERA/DTIS, Université de Toulouse, 31055 Toulouse, France
[5]Institut Informatique et de Gestion (IIG), University of Applied Sciences and Arts Western Switzerland (HES-SO) Valais-Wallis, 3960 Sierre, Switzerland
[6]Department of Computer Science, Aalto University, 02150 Helsinki, Finland

Corresponding author: Amro Najjar (amro.najjar@uni.lu)

ABSTRACT Quality of Experience (QoE) is defined as the measure of end-user satisfaction with the service. Most of the existing works addressing QoE-management rely on a *binary* vision of end-user satisfaction. This vision has been criticized by the growing empirical evidence showing that QoE is rather a *degree*. This article aims to go beyond the binary vision and propose a QoE management mechanism. We propose a one-to-many negotiation mechanism allowing the provider to undertake *satisfaction management*: to meet fine-grained user QoE goals, while still minimizing the costs. This problem is formulated as an optimization problem, for which a linear model is proposed. For reference, a generic linear program solver is used to find the optimal solution, and an alternative heuristic algorithm is devised to improve the responsiveness when the system has to scale up with a fast-growing number of users. Both are implemented and experimentally evaluated against state-of-the-art one-to-many negotiation frameworks.

INDEX TERMS One-to-many negotiation mechanism, quality of experience, end-user satisfaction, linear model, multiagent systems.

I. INTRODUCTION

End-user (or User) satisfaction[1] is a key factor to ensure the success of any online service. Among different user satisfaction determinants, Quality of Experience (QoE) appeared in the 2000s as a metric to assess the service quality as perceived by the user. Compared with Quality of Service (QoS), a technology-centric metric, research on QoE, a subjective and user centric measure, is still in its early stages of development, and most of the studies dealing with QoE are carried out on the conceptual front.

However, QoE aims to provide a practical measure allowing to quantify user satisfaction and service acceptance. Consequently, *QoE-management* emerged as a process aiming at maximizing QoE while optimizing the used resources [1]. Nevertheless, most of the existing works in the domain of QoE-management are provider-centric since QoE-management decision is taken unilaterally by the provider. Furthermore, they rely on the Mean Opinion Score (MOS) to assess the satisfaction and the service acceptability [2]. MOS is an average of the users' opinions. Thus, it hides important information about the users' diversity and personal preferences [3].

By definition, agents are self-interested and bound to an individual perspective. For this reason, agents have been proposed in the literature to integrate users' personal preferences into the decision loop. Yet, when assessing users' satisfaction, most of these works rely on service acceptability as a *binary* decision in which the user decides to accept or reject the proposed service (as in [4], for example). Nevertheless, studies on user satisfaction and QoE suggest that user satisfaction should not be represented as a binary concept [2], but rather as a *degree* reflecting the user's delight or annoyance of an application or service [5]. To capture this nuanced nature of user

The associate editor coordinating the review of this manuscript and approving it for publication was Danilo Pelusi.

[1]This paper is an updated and extended version of a paper presented at the International Workshop for Agent-based Complex Negotiations (ACAN), a workshop of limited audience and no proceedings.

satisfaction, recent results of user polls and subjective user tests recommend providers to rely on percentiles, to assess the user's subjective estimation of the service quality [6]. This would allow a provider to have a fine-grained management of users' satisfaction by ensuring that a predefined percentage of users' falls in each satisfaction category.

This article proposes a one-to-many negotiation mechanism for user satisfaction management. The proposed mechanism equips the provider with fine-grained control over the user satisfaction levels. Thus, the provider can meet the satisfaction objectives (*e.g., "good"* service delivered to at least 40% of users, *etc.*) while respecting its budget or resource constraint. More specifically, the contribution of this work is threefold:

1) We formulate the user satisfaction problem as an optimization problem, whereby the provider seeks to minimize the cost or resources, while meeting a set of business constraints guaranteeing the subjective satisfaction experienced by the Service Users (SU);
2) We propose a heuristic algorithm to solve this problem and experimentally compare its solution with the optimal solutions in term of, execution time, cost, and user satisfaction;
3) We implement both solutions and evaluate our contribution against SoTA methods in a Cloud-based case study.

The rest of this article is organized as follows. Section II introduces the one-to-many negotiation architecture. Section III defines the satisfaction management problem and details the optimal model and the proposed heuristic algorithm. Section IV details the experimental evaluation. Section V reviews related works from the literature and Section VI concludes this article.

II. THE ONE-TO-MANY NEGOTIATION ARCHITECTURE

The one-to-many-negotiation addressed in this article involves a provider negotiating simultaneously with multiple users. The provider aims to meet a set of predefined user satisfaction goals while respecting the budget constraints. The one-to-many-negotiation can be represented by the following tuple:

$$\langle Provider, Users, Serv, Goal, RC \rangle \tag{1}$$

where *Provider* is the service provider. *Users* is the set $\{u_1, \ldots, u_n\}$ of SU. *Serv*, the service offered by the provider, is defined by a set of attributes (or issues) at_j. For instance, if the service is a video transcoding service, it may involve attributes such as video resolution, video transcoding time, *etc.* *Goal* is the provider set of user satisfaction goals, *RC* is the cost or resource constraint.

To implement this one-to-many negotiation we present a multi-agent architecture aiming at involving the end-user into the satisfaction management process is required to strike a balance between the provider's Quality of Business (QoBiz) and the users' QoE.

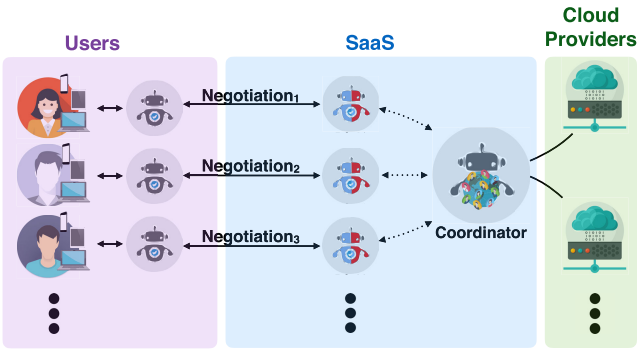

FIGURE 1. The EMan architecture deployed in the cloud ecosystem.

In our earlier works [7], [8], we presented EMan (Figure 1) a one-to-many multi-agent architecture modeling the negotiation between a provider and SU or end-users. EMan follows a binary approach to service acceptability. More specifically, it allows involving the SUs into the QoE-management decision (i.e., how much resources should a Software as a Service (SaaS) provider allocate to SUs to provide an *acceptable* service). However, EMan does not allow for fine-grained user satisfaction management. This paper addresses this limitation by upgrading EMan with a novel user satisfaction mechanism. This section presents the basic components of EMan, while Section III deals with the satisfaction management mechanism.

A. AGENTS

EMan consists of three agent types: service user agents (sa_i), delegate agents (da_i) and a single coordinator agent (ca). The latter two types represent the provider. The choice of one ca is to represent one SaaS provider. However, this does not exclude the fact that this entity could be organized in a group of agents with one representative, or at least multiplied into several entities to provide more robustness and fault tolerance.

1) SERVICE USER AGENTS

Service user agents (on the left-hand side of Figure 1) participate in the negotiation process on behalf of service users. To do so, they rely on a utility function and a set of concession strategies discussed below respectively.

a: USER AGENT UTILITY FUNCTION

The agent sa_i encodes its preferences using the utility function M_{sa_i}. At every cycle t, M_{sa_i} is used to assess the utility of offers $o^t_{da_i}$ received from the corresponding delegate da_i. M_{sa_i} is a weighted sum of μ_{sa_i,at_j}, the attribute-wise utility functions which measure the utility obtained from one attribute. This choice is supported by the literature of QoE and QoE-management [9]. Thus, M_{sa_i} is defined as follows:

$$M_{sa_i}(o^t_{da_i}) = \sum_{j=1}^{j=J} w_{sa_i,at_j} \cdot \mu_{sa_i,at_j}(o^t_{da_i}[at_j]) \tag{2}$$

where J is the number of attributes, w_{sa_i,at_j} represents the weight associated to the at_j attribute specifying the importance given to this attribute by the user (these weights should satisfy $\sum_{j=1}^{J} w_{sa_i,at_j} = 1$), $o^t_{da_i}[at_j]$ represents the value offered in $o^t_{da_i}$ for at_j, and μ_{sa_i,at_j} represents the utility function of the attribute at_j. To determine the form of this function, EMan relies on evidence from QoE literature [2] where it has been shown that for some attributes μ_{sa_i,at_j} is likely a logarithmic relationship [10] derived from the Weber-Fechner Law (WFL). The latter is a famous law in Psychophysics, it stipulates that the human perception of a stimulus is estimated by a logarithmic function. WFL has been validated by empirical QoE studies where it has been shown that a logarithmic function applies for attributes such as waiting time and bandwidth [10], [11]. Thus, when the WFL applies, μ_{sa_i,at_j} has the following form:

$$\mu_{sa_i,at_j}(o^t_{da_i}[at_j]) = \alpha_{sa_i,at_j} \cdot ln(o^t_{da_i}[at_j]) + \gamma_{sa_i,at_j} \quad (3)$$

where $o^t_{da_i}[at_j]$ represents the value of attribute at_j offered by the provider. The user-specific coefficients α_{sa_i,at_j} and γ_{sa_i,at_j} are derived by her agent sa_i from her attributes' reservation and preferred values. The preferred value (denoted as pv_{sa_i,at_j}) is the ideal value that yields the maximum utility for this attribute whereas the reservation value (denoted as rv_{sa_i,at_j}) is the worst acceptable value (equivalent to MOS $\approx 2.5/5$), beyond which no agreement is better than an agreement [12]. The user, as has been confirmed by empirical studies, is capable of verbalizing the values of pv_{sa_i,at_j} and rv_{sa_i,at_j} [7], [13], [14]. Therefore, they can be obtained by sa_i from the user. However, for privacy reasons, they are not divulged to the provider. From the perspective of customer expectation management literature, these values correspond respectively to the *desired expectations* and *adequate expectations* [14].

Note that in addition to the logarithmic relationship, other relationships have been proposed by existing works to model QoE. They include the exponential IQX hypothesis [15] and other formulas derived from Steven's power law (*cf.* [16]). The formulation (and justification) of the user utility function is beyond the scope of this article. This function can rely on a QoE hypothesis or can be learned by the user agent. Furthermore, we consider that the user utility function should involve attributes concerning the user, the application, the provider, and underlying network [9].

For attributes that don't satisfy the logarithmic relationship, we assume its utility function to be linear:

$$\mu_{sa_i,at_j}(o^t_{da_i}[at_j]) = \frac{rv_{sa_i,at_j} - o^t_{da_i}[at_j]}{rv_{sa_i,at_j} - pv_{sa_i,at_j}} \quad (4)$$

where $\mu_{sa_i,at_j}(o^t_{da_i}[at_j]) \in [0, 1]$.

b: USER AGENT CONCESSION STRATEGIES

To make a decision on whether to accept or reject a given offer received from the provider, sa_i relies on its current Aspiration Rate (AR) and its utility function. $AR^t_i \in [0, 1]$ expresses the utility sa_i expects to obtain at this negotiation cycle t. AR is variable since, to reach agreements, sa_i makes concessions by reducing its AR. In this paper we assume that sa_i follow either a Time-Based Concession (TBC) strategy or Behavior-Based Concession (BBC) strategy [17]. TBC are a very common assumption in the literature (*cf.* [18]). In this case, $\Delta AR^t_{sa_i}$, the concession made by sa_i for cycle t is computed as follows [19]:

$$\Delta AR^t_{sa_i} = AR^{t-1}_{sa_i} \cdot \left(\frac{t}{T_{sa_i}}\right)^{\lambda_{sa_i}} \quad (5)$$

where T_{sa_i} is sa_i's negotiation time deadline. λ_{sa_i} controls the convexity degree of the concession curve thereby determining the behavior of sa_i (conciliatory, linear or conservative) [19]. As for BBC strategies, they are basically tit-for-tat strategies where the concession made by sa_i is proportional to the concession made by the da_i. Section II-C2 details how the architecture handle sa_i using a BBC strategy.

2) DELEGATE AGENTS

When a new sa_i enters the system, a new da_i is created (the agents in the middle of Figure 1). It enters into a bilateral negotiation session with sa_i. Like sa_i, da_i relies on a utility function M_{da_i}. However, the utility of an offer $o^t_{sa_i}$ from da_i's standpoint is determined by the cost required to provide the service if this offer is accepted. M_{da_i} is defined as:

$$M_{da_i}(o^t_{sa_i}) = \frac{RC - Cost(\Pi(o^t_{sa_i}))}{RC - PC} \quad (6)$$

where M_{da_i} is $\in [0, 1]$, Π is used by da_i to estimate the resources required to serve $o^t_{sa_i}$, *Cost* helps estimate the cost of renting these resources from the cloud. *RC* and *PC* are the reservation (maximal) and preferred (minimal) costs of da_i. These values are initialized by *ca* when da_i is created. The value of *RC* is of particular importance because it allows the provider to impose its budget constraint *i.e.,* the average cost spent on a user should not exceed *RC*.

To facilitate agreements with sa_i, da_i also relies on concession strategies. In particular, two types of concession strategies may be used by da_i:

- *TBC* computed as follows:

$$\Delta AR^t_{da_i} = \frac{1}{T_{da_i}} \quad (7)$$

  where T_{da_i} is the delegate's time deadline. When da_i reaches T_{da_i}, it no longer makes any concession.
- *BBC or tit-for-tat* in which da_i's concession is proportional to the concession made by sa_i in the previous cycle. As da_i doesn't have access to the real concession of sa_i, it relies on its own utility function to assess the concession of sa_i:

$$\Delta AR^t_{da_i} = M_{da_i}(o^t_{sa_i}) - M_{da_i}(o^{t-1}_{sa_i}) \quad (8)$$

  where $(o^t_{sa_i})$ and $(o^{t-1}_{sa_i})$ are the previous couple of offers received from sa_i.

Most of the time da_i agents are autonomous in their negotiations with sa_i. Once a negotiation session is terminated,

da_i notifies ca about the outcome of the negotiation session. This helps ca keep track of the results of the sessions.

B. NEGOTIATION PROTOCOL

As discussed earlier, each $u_i \in Users$ is represented by an agent, denoted as sa_i. *Provider* is modeled by two types of agents, a coordinator agent ca and a set of delegate agents $\{da_1, \ldots, da_n\}$. Each da_i is responsible for assuming the bilateral negotiation sessions with user agent sa_i. ca spawns da_i agents, and initializes their negotiation strategy. Moreover, it oversees the negotiation process and may intervene in some sessions to push the provider business goals and impose its budget constraints. On their side, each user agent sa_i seeks to maximize the QoE of the user u_i by relying on a utility function derived from this user's preferences and expectations.

The object exchanged during the negotiation process is an offer $o = \langle at_1, at_2, \ldots, at_J \rangle$ that defines a value for each one of the attributes at_j of the service *Serv*. The negotiation protocol is based on the alternate offer protocol [7]. At each cycle, an agent can accept the opponent's offer, reject it and propose a counter-offer, or leave the session in case it reaches its time-deadline. After the end of the session, if the user agent accepts the service, it may choose to rate the service and send this rating to the provider.[2]

Note that the negotiation process is non-synchronous, some sessions can be already finished, and some can be ongoing, while others may not have started yet. Moreover, negotiators (*i.e.,* sa_i and da_i agents) do not disclose their preferences, their strategies, or their reservation values.

Note also that we consider that both parties seek win-win settlements. Furthermore, exploitation is considered beyond the scope of this article.

C. LEARNING THE USER PROFILE

To undertake the satisfaction management process, the provider has to construct a model of the user's profile including the reservation and preferred values of each attribute and the negotiation time deadline. Most of the existing works, attempting to establish a model of the opponent profile, assume that the opponent employs a TBC strategy [18]. Dealing with opponents employing a BBC strategy is less studied. However, since the proposed satisfaction management mechanism is designed for open providers, a considerable portion of its users might be BBC. For this reason, we propose a BBC detection and handling mechanism allowing the provider to deal with BBC users. Although this mechanism does not enable da_i to establish a model of the BBC users' profiles[3] that can be fed to the satisfaction management process, da_i can still use this mechanism to negotiate with BBC users, provide them with a satisfactory service without violating the provider's cost constraints.

[2]Rating is not used if the same user requests the same service in future.

[3]Establishing such a model of BBC users is a future research perspective.

The following sections discuss the user profile learning used with TBC and BBC users.

1) TBC USERS

The mechanism used to model TBC users is based on our earlier work [20] where each delegate da_i keeps track of the offers proposed by sa_i in the ongoing sessions and uses them to estimate the derivative of the concessions made by sa_i. When the derivative goes into negative values, da_i starts making a non-linear regression every cycle to estimate the time deadline $\overline{T_{sa_i}}$ of sa_i.

This process is repeated as long as the following condition holds:

$$\overline{T^t_{sa_i}} - t > d \tag{9}$$

where t is the current cycle, $\overline{T^t_{sa_i}}$ is da_i estimate of sa_i time deadline. Thus, this condition means that at least d cycles remain before sa_i reaches $\overline{T^t_{sa_i}}$. Once the condition is violated, da_i considers that sa_i is nearing its time deadline. Consequently, it considers that $o^t_{sa_i}$, the latest offer received from sa_i contains, approximately, its reservation values for each attribute. This is justified by the fact that since sa_i uses a TBC strategy, when it is near its T_{sa_i}, it sends offers containing values very close to its reservation value for each attribute.

To get the preferred values for the service attributes, we assume that these values are those included in the first counter-offer sent from sa_i to da_i. After getting these values, da_i uses its cost function to estimate the cost required for different satisfaction categories for sa_i (*cf.* Section III-A).

Finally, da_i notifies ca that sa_i is nearing its time deadline and sends it all the values estimated above. ca creates a new record for each new user and adds them to the *list*.

2) BBC USERS

The mechanism used by da_i to deal with BBC users involves two steps: detection and handling. At the beginning of the negotiation session, da_i does not know whether sa_i is TBC or BBC. Therefore, da_i assumes that sa_i is TBC and deals with it as explained in Section II-C1. Simultaneously da_i runs a detector to detect if sa_i is BBC. This BBC detector within da_i tracks the imitation behavior of sa_i by comparing the exchanged offers (*e.g.,* if da_i did not make any concession in cycle $t-1$, sa_i does the same in cycle t). The BBC detectors have several parameters that need to be tuned including the detection starting cycle, the window of how many previous cycles to include in calculating the average concession made by sa_i, and the sensitivity of the detector, *i.e.,* the difference in utility value between two consecutive offers (or between the last offer and the average offer in the window) is used to determine if a concession has been made by sa_i. These attributes have been tuned empirically due to the complexity of defining them formally, a choice also confirmed in the literature [18].

Once sa_i is detected as BBC, da_i changes how it handles the negotiation. In this case, because of sa_i tit-for-tat, there

is no way of reaching an agreement unless da_i starts making considerable concessions. To efficiently exploit the range of concessions it can make, da_i distributes these concessions across several offers and start sending them to sa_i given that for the whole process, da_i does not offer more than the cost allocated to it by *ca*.

D. COORDINATION STRATEGIES

In the bilateral negotiation sessions, each da_i negotiates autonomously. Once a session is terminated either successfully or not, the corresponding delegate notifies the coordinator. This helps the coordinator keep track of the acceptance rate of the service. For the sake of the satisfaction management process (*cf.* Section III), once the user accepts the service, the coordinator must know what is the obtained satisfaction level. To do so, it can either:

1) Rely on da_i estimation of the corresponding user profile to estimate the satisfaction perceived by the user; or
2) Rely on the user's explicit feedback in form of a rating sent from the user to the provider. In the experiments section, we will study the impact of $f\%$, the percentage of sa_i rating the service after accepting it.

Note that, in addition to the coordination strategies discussed above, *ca* may intervene in the ongoing negotiation sessions to undertake satisfaction management. This process is detailed in the following sections.

III. USER SATISFACTION MANAGEMENT MECHANISM

This section details the core contribution of this article. Section III-A describes the satisfaction management. Section III-B models the user satisfaction management problem as an optimization problem. Section III-C details the proposed heuristic algorithm.

A. SATISFACTION MANAGEMENT OVERVIEW

The satisfaction obtained by the user from the service offered by the provider falls into one of K satisfaction levels or categories. Each category represents a satisfaction level perceived by users *e.g., excellent, very good, good, fair, acceptable, etc.,*. To be in the satisfaction category k, the following condition must be satisfied:

$$M_{sa_i}(\hat{o}) \geq h_k \tag{10}$$

where $\hat{o}$ is the offer accepted by sa_i, h_k is the threshold defining the category k. h_k is defined by the provider as a common threshold applied to all users, but the condition in (10) differs from one user to another since M_{sa_i} is personal. To minimize the cost and know which user could be efficiently placed in which category, da_i keeps track of sa_i's sequence of offers. Based on this sequence, da_i constructs a model of sa_i's profile. When da_i infers that sa_i is nearing its time deadline, it sends the estimated profile to *ca*. The latter adds them to a *priority-list* (denoted as *list*) on which it undertakes the satisfaction management mechanism detailed in the algorithm below. If the *optimalMode* is active, the algorithm relies on the optimal solution denoted as `OptimalSM` and described in Section III-B. Otherwise, it relies on the heuristic algorithm denoted as `HeuristicSM` (Section III-C). Both functions return a list of offers (denoted as tailoredOffers) tailored for each user in the list. Then, the function `propose()` proposes the tailored offer to the corresponding user. In case sa_i accepts the offer, it may choose to rate the service it obtained.

Algorithm 1: Satisfaction Management Mechanism

input : *list*, a list of learned user profiles
input : *Already*[k], users already in the category k

```
At the outset of each iteration ;
Profiles ← getUserProfiles() ;
if (optimalMode ==true) then
    tailoredOffers ← OptimalSM()
else tailoredOffers ← HeuristicSM();
foreach Offer o_i of tailoredOffers do
    if (propose(o_i, sa_i) ==true) then
        rating_i ← requestRating()
foreach Satisfaction Category k of K do
    updateAlready(k, Rating)
```

B. USER SATISFACTION OPTIMIZATION PROBLEM

User satisfaction management can be seen as an optimization problem where, at each iteration t, *ca* seeks to assign the N^t users in $list^t$ to the K satisfaction categories, while minimizing the cost and respecting a set of goals defining the percentage of users that should be in each category, where a category represents a satisfaction level perceived by users. Note that, for the optimization problem, we consider the cost as a soft constraint and the satisfaction goals as hard constraints. This choice is justified by the recommendations from QoE literature and by the fact that the provider business model should be flexible. Thus, user satisfaction management can be formalized as follows:

$$\textbf{Minimize} \sum_{k=1}^{K}\sum_{i=1}^{N^t} C_i^k \cdot X_i^k \tag{11}$$

$$\textbf{Subject to} \sum_{k=1}^{K} X_i^k \leq 1, \quad \forall i \tag{12}$$

$$\sum_{m=1}^{k}\left(\frac{\sum_{i=1}^{N^t} X_i^m + Already[m]}{\#TerminatedSessions}\right) \geq Goal[k], \ \forall k \tag{13}$$

where N^t is the number of users in $list^t$ in the iteration t. X_i^k is a 2D array of binary variables. $x_i^k = 1$ means that the user x_i is in category k. C_i^k is a 2D array containing the estimation of the cost of putting x_i in the category k. These costs are derived from the profile estimates sent by the delegates.

Constraint (12) ensures that at most a user is assigned to one category. Note that a user can be assigned to no category (*i.e.,* $\sum_{k=1}^{K} X_i^k = 0$), in this case, the user remains in *list* for the next iteration.

Constraint (13) ensures that at least $Goal[k]$ percent of users are assigned to either a given category k or a category better than k. In other words, the number of users assigned to category k combined with the number of users assigned to each category m that is better than k (*i.e.,* $m >= 1$ *and* $m < k$) satisfy the provider goals for this category k. $Already[k]$ is the number of users that have been assigned to the category k in previous iterations, *#TerminatedSessions* is the number of negotiation sessions that have been terminated either successfully or unsuccessfully at the outset of this iteration and $Goal[k]$ is the provider's predefined goal for the category k. It represents the percentage of users that should be put in the satisfaction category k.

In the case of infeasibility, a rare case that occurs when the number of users in $list^t$ is not enough to satisfy constraint (13), or when the cost of the solution exceeds the current budget of the provider, users, whose time is not up, remain in *list* for the next iteration.

The obtained optimal solution assigns a user sa_i to a category k. The provider proposes $\overline{o_{sa_i}}$, a tailored offer that is supposed to give sa_i the satisfaction of the category it was assigned to. If sa_i does not accept the offer, it remains in *list* for the next iteration where it may get a new tailored offer. Otherwise, if the offer is accepted by sa_i, it may choose to send its rating of the service quality to the provider. This rating helps the provider arrange sa_i in one of the K categories and thereby allows it to have a more precise estimation of $Already[k]$ for the next iteration. Otherwise, if sa_i chooses not to give its rating to the provider, the latter assumes that sa_i actually perceived a satisfaction level that corresponds to the category it was assigned to. At the end of each iteration, the solution of the optimizer at iteration t is fed as an input (*i.e., Already*$[k]$ and *#TerminatedSessions*) to the next iteration.

C. HEURISTIC SATISFACTION MANAGEMENT ALGORITHM

At each iteration, the proposed heuristic algorithm must strike a balance between two considerations: *(i)* assign enough users to category k to get it as close as possible to its $Goal[k]$, and *(ii)* respect the budget constraint by choosing a less costly assignment. To do so, the heuristic algorithm works as follows. First, it sorts the priority list, namely *list* (Section III-C1). Second, it picks up users and assigns them to what it considers the best category they fit in.

1) SORTING LIST

$list^t$ can be sorted according to one of the following criteria:

- Cost $\overline{c_i}$: this is an estimation of the cost required to serve the worst possible service that is likely to be acceptable by sa_i. The intuition here is that less costly users should be preferred since they can be satisfied easily.
- Utility $\overline{u_{sa_i}}$: the estimated utility range of sa_i. This range is calculated as follows: $\overline{u_{sa_i}} = p_i - \overline{c_i}$ where p_i is the cost required to provide sa_i with an ideal service that maximize its utility. The intuition here is that p_i should be taken into account as well as $\overline{c_i}$ since, for instance, offering an *excellent* service to a user whose p_i is lower, is much cheaper.
- Deadline $\overline{T_{sa_i}}$: the estimated number of cycles remaining before sa_i reaches its time deadline. The goal here is to avoid users reaching their time deadline thereby quitting the service without being served.

Note that, at each iteration, the coordinator obtains all the criteria mentioned above from delegates as a result of the user profile learning process.

2) SELECTING USERS FROM LIST

Once the list is sorted, the next decision is to choose which user goes to which category. Note that users can be assigned to a k category as long as this category has not met its $Goal[k]$ and as long as non-assigned users are still present in the list. The proposed heuristic algorithm can rely on *DISTANCE_TO_GOAL* strategy or on *CATEGORY_COST* strategy. Both of these strategies pick users up from the top of the list and then iterate to the bottom.

- *DISTANCE_TO_GOAL*: A category k, whose distance to achieve its goal (*i.e.,* $Goal[k]$) is greater, has priority over other categories. These distances are calculated using the *Already* array and *#TerminatedSessions* defined in Section III-B. Since sometimes *list* does not contain users enough to meet the satisfaction goals in this iteration, this strategy aims at getting closer to the goals as much as possible.
- *CATEGORY_COST*: The more satisfactory a category, the more priority it has over others. For instance, the *excellent* category will have priority over the *very good* category. The *very good* category will have more priority than the *good* category, *etc.*

Note that like the optimal solution (Section III-B), the solution of the heuristic algorithm creates a tailored offer for each user sa_i, as per its category, $\overline{o_{sa_i}}$. At the end of each iteration, the solution of the heuristic algorithm (not the optimal) at iteration t (*i.e., Already*$[k]$ and *#TermuinatedSessions*) is fed as an input for the next iteration.

IV. EXPERIMENTAL EVALUATION

To evaluate the proposed satisfaction management mechanism, we implement it within SaaS provider simulated using Repast Simphony [21]. The latter is a Java-based multi-agent simulation environment that holds significant operational and executional features [22], [23]. The optimal solution is computed using IBM's CPLEX optimizer [24]. In this implementation and as per tuple (1), *Provider* is the SaaS provider, *Users* are the SaaS users, *Serv* is the SaaS service, *Goal* are the SaaS provider predefined satisfaction goals, and *RC* is the

maximum cost the SaaS provider may pay to rent resources from the cloud to serve a user.

This section proceeds as follows. Section IV-A introduces the parameters of the experiments. Section IV-B compares the optimal solution with the proposed heuristics. The experiment in Section IV-C studies the impact of the percentage of users rating the service after being served. The experiment in Section IV-D analyzes the impact of the percentage of users employing tit-for-tat (*i.e.,* BBC) strategies.

A. EXPERIMENTS' PARAMETERS

$|SU|$, the total number of users entering the simulation is 10000 (except for Section IV-B where we examine the execution time with $|SU|$ going up to 50000). Users enter the simulation following a Poisson process whose mean is A (the arrival rate). The service in the experimental scenario involves more than one attribute. User profiles, including the reservation and preferred values, the time deadline, and the weights for each attribute, are generated randomly. T_{sa_i}, the negotiation time deadline of SU is generated randomly within the following range [40 : 120] cycles. RC, the delegate reservation cost imposed on the heuristic algorithm, is set to $RC = 0.75 \times PX$ where PX is the minimal cost required to serve the preferred service for users. Discussing the impact of RC on the provider business goals is beyond the scope of this article.

In the following experiments, we define 3 satisfaction categories ($K = 3$). These categories are *acceptable*, *good* and *excellent*. The goals set for these categories are 0.95, 0.4 and 0.2 respectively. This means that at least 95% of users should obtain an *acceptable* service or better, 40% should obtain *good* service or better, and 20% should obtain *excellent* service. h, the thresholds for these satisfaction categories are defined as 0.75, 0.5, and 0.0 for excellent, good and acceptable service respectively. This means that to get an *excellent, good, and acceptable* service, the utility perceived by the user agent should be greater than 0.75, 0.5, and 0.0 respectively. $f\%$ is the percentage of users choosing to give their rating after they accept the service. The impact of this parameter is studied in Section IV-C. $BBC\%$ stands for the percentage of BBC users in the simulation, its impact is studied in Section IV-D. The results of the experiments are obtained by averaging the outcomes of at least 10 simulation runs.

Note that whereas users can have logarithmic utility functions, due to space constraints, these users will not be included in the following experiments.

B. OPTIMAL SOLUTION VS. HEURISTIC ALGORITHM

To evaluate the computational cost of the optimal solution, Section IV-B1 compares the execution time of CPLEX optimizer with the proposed heuristic algorithm. Then, Section IV-B2 evaluates the heuristic strategies. We compare the cost they spend on served users with the minimal cost calculated by the optimal solver.

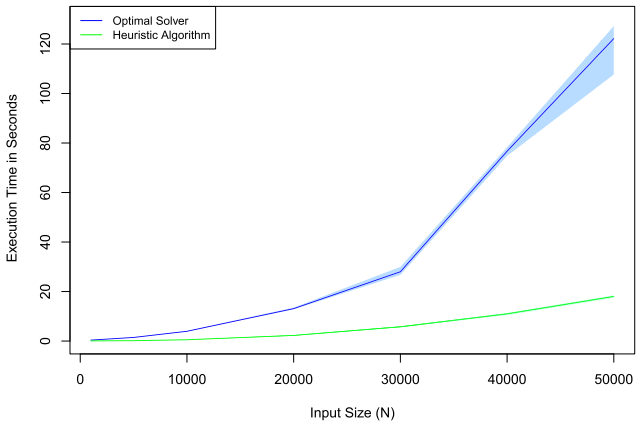

FIGURE 2. The average execution time of the optimal solution (blue curve) and the heuristic algorithm (green curve) with minimum and maximum values.

TABLE 1. The cost of heuristic strategies vs. optimal cost.

	Utility_First	*Cost_First*	*Deadline_First*
DISTANCE_TO_GOAL	+10.2%	+16.5%	+10.9%
CATEGORY_COST	+12.9%	+12.9%	+13.1%

1) EXECUTION TIME

Figure 2 compares the execution time of the optimal solver with that of the heuristic algorithm, both as a function of N, the size of *list*. As can be seen from the blue curve in the figure, the execution time of the optimal solution increases exponentially with N. With relatively small $N < 10000$ users, the execution time is still feasible. With $N = 10000$, the solver takes almost 4 seconds to calculate the optimal solution for each iteration. However, in today's market, popular SaaS services are often subject to load-spikes or surges of users requesting the service (*e.g.,* a video transcoding service used during a football match). Yet, as can be seen from the figure, with a relatively big N, using the optimal solution becomes impractical. The green curve shows that the execution time of the heuristic algorithm is much shorter even with bigger N.

2) COST OF THE HEURISTIC STRATEGIES

Table 1 compares the cost spent per served user by each one of the heuristic strategies (Section III-C2) with the cost spent per served user by the optimal solution.

All these heuristic strategies managed to achieve the satisfaction goals and, as can be seen from the table, their cost is not significantly more expensive than the optimal solution. In particular, the cost spent by the strategy Utility/*DISTANCE_TO_GOAL* is only 10.2% more expensive than the optimal solution and is thereby the least costly among other heuristic strategies. For this reason, this strategy is chosen for the rest of the experiments.

The results in Figure 2 and Table 1 show that the provider must address a trade-off between cost optimality and responsiveness. For instance, a provider can use the optimal solution with a relatively low influx of users. Yet, with a higher influx, it becomes expensive in terms of response time. This may degrade the user experience and lead to client churn.

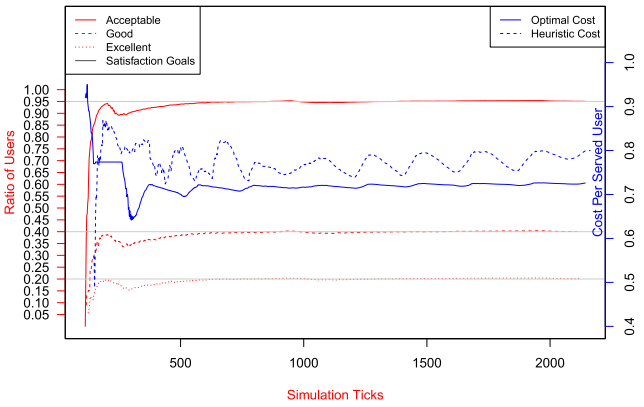

FIGURE 3. The user satisfaction achieved by the heuristic strategies (red curves) and their cost (blue dashed curve) compared with the optimal solution cost (blue solid curve).

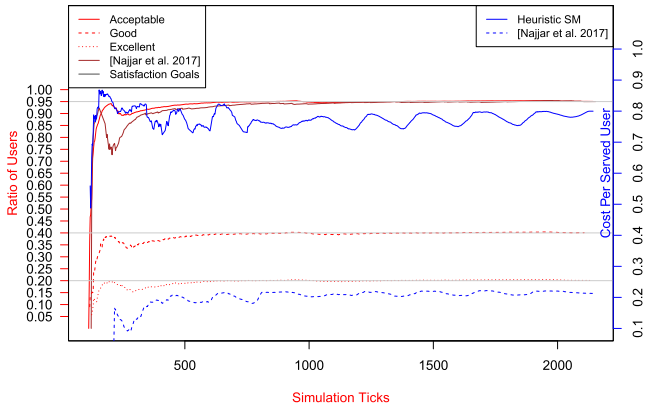

FIGURE 4. User acceptability rate (left y-axis) and its cost (right y-axis) using the proposed heuristic Satisfaction Management (SM) algorithm compared to [4].

Figure 3 studies the results achieved by the chosen heuristic strategy in one run as a function of the time ticks of the simulation. The left y-axis represents the ratio of users assigned to each satisfaction category. The red curves plots the ratio of users receiving acceptable, good, and excellent satisfaction. The gray lines delineate the provider satisfaction goals. As can be seen from the figure, after a short period of oscillation, each satisfaction category managed to get the ratio of users it needed. For this reason, each red curve is slightly above the corresponding gray line.

To normalize the cost spent per served user, its value is divided by *RC* (right y-axis). As can be seen from the figure, both the optimal solution and the heuristic algorithm did not exceed *RC* (*i.e.,* it does not exceed the value 1.0 on the right y-axis), where the cost constraint is defined by the provider. Furthermore, the cost spent per served user by the heuristic algorithm is slightly higher than the cost spent by the optimal solution. This confirms the results of Table 1 and as discussed above, this extra cost is compensated by the shorter execution time.

Note that while Figure 3 plots the results obtained during the simulation run, Figure 5 and 6 plot an average of results obtained at the end of multiple simulation runs.

3) HEURISTIC ALGORITHM VS. STATE-OF-THE-ART

According to our knowledge, other works, such as AQUAMan [4], propose service acceptability rate management mechanisms where client satisfaction is viewed as a binary measure (accept/not accept), hence, no fine-grained satisfaction management control. Figure 4 compares the proposed Satisfaction Management (SM) heuristic with AQUAMan. As can be seen from the figure, both achieve 95% acceptability rate. However, in contrast to the proposed SM, AQUAMan achieved 0% for the ratio of good and excellent users (40% and 20% respectively with the proposed SM). Furthermore, the average satisfaction obtained by users with the proposed SM is 0.37 whereas it drops to 0.127 with AQUAMan. This better service quality comes with a considerable cost increase, as can be seen from the figure.

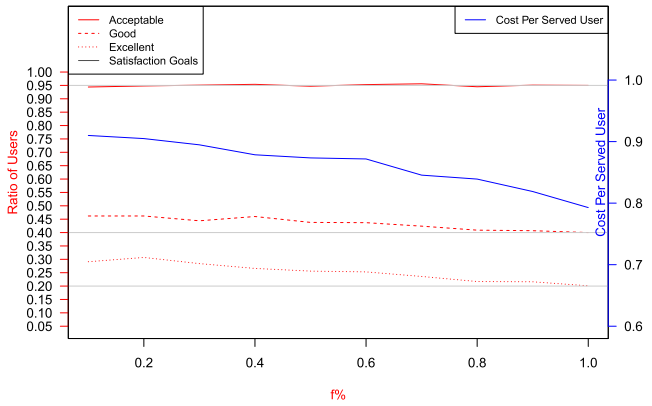

FIGURE 5. The impact of $f\%$ (x-axis) on the ratio of users assigned to each satisfaction category (left y-axis) and on the cost spent per served user (right y-axis).

Yet, as has been shown by Table 1, the proposed heuristic is only 10% more expensive than a comparable optimal solution for the satisfaction management problem.

C. IMPACT OF USER FEEDBACK

This experiment studies the impact of $f\%$, the percentage of sa_i rating the service after accepting it. In the previous experiment, it was assumed that all users who accepted the service gave their rating to the provider (*i.e.,* $f\% = 100\%$). To understand the impact of this parameter both on the ratio of users assigned to each satisfaction category and on the cost spent per served user, Figure 5 plots them as a function of $f\%$. As can be seen from the figure, even with $f\% \approx 10\%$, each satisfaction goal retained *at least* the ratio of users provided by the satisfaction goals (gray lines). Furthermore, since accepting or rejecting the service is an explicit decision that requires no rating from the user, the ratio of users finding the service acceptable (red solid curve) is not impacted by $f\%$. However, as $f\%$ increases, the other two categories get closer to their goals. For instance, with $f\% \approx 10\%$, about 30% of users perceive the service as excellent even though the provider goal for this category was 20%. Thus, the ratio of users getting excellent service is considerably above the

goal. This leads to higher costs since extra users are getting excellent service.

To understand the impact of $f\%$ on the cost, the blue solid curve plots the normalized average cost spent per served user (right y-axis). As can be seen from the figure, even with low $f\%$, the provider managed to meet its cost constraints since the cost does not exceed RC (1.0 on the right y-axis). Furthermore, with a higher $f\%$, the cost drops significantly. This is explained as follows. Using the feedback from the users accepting the service, the provider can better steer its satisfaction management thereby approaching the goals for each satisfaction category and minimizing the cost spent per served user.

These results suggest that, to encourage users to give their feedback (and hence increasing $f\%$), the provider can choose to share these profit margins ($\approx$ 12% with $f\%$ going from 10% to 100%) it achieves with users who accept to give their feedback by offering them, for instance, a discount on the fees they pay or other advantages for the next time they request another service from the provider. This is a win-win business strategy that saves costs for the provider and is also advantageous for the cooperative users.[4]

D. IMPACT OF BBC USERS

This section studies the impact of adding BBC users. Figure 6 compares the user satisfaction results obtained by the BBC detection and handling mechanism proposed in Section II-C2 (red curves), with the results of the same users but when this mechanism is deactivated and BBCs are not handled (red curves with circle markers). On the one hand, even when BBCs are not handled, with relatively low $BBC\%$, the heuristic algorithm manages to remain close to the goal since it can still handle TBC users. As $BBC\%$ increases, the user satisfaction drops down dramatically for all the three categories. Yet, the ratio of users getting acceptable service witnesses a significantly slower decay than the other two categories. This is explained by the used *DISTANCE_TO_GOAL* strategy (Section III-C2). This strategy prioritizes categories whose distance to their goal is the furthermost. For this reason, in this example, since the goal of the ''acceptable'' category is considerably higher than the goals of the other two categories, it is prioritized and it decays relatively slower. This behavior can be useful for the provider since it allows for graceful degradation.

On the other hand, the BBC detection and handling mechanism shows a more robust behavior. The *acceptable* and the *excellent* categories decrease is significantly slower than the case when BBC is not handled. Concerning the results of the *good* category, the figure shows that unexpectedly, this category witnesses a considerable increase as $BBC\%$ rises. This is explained by the fact that the mechanism can no longer propose tailored offers to guarantee a fine-grained excellent service.

[4]Dealing with users who give misleading feedback is considered beyond the scope of this article.

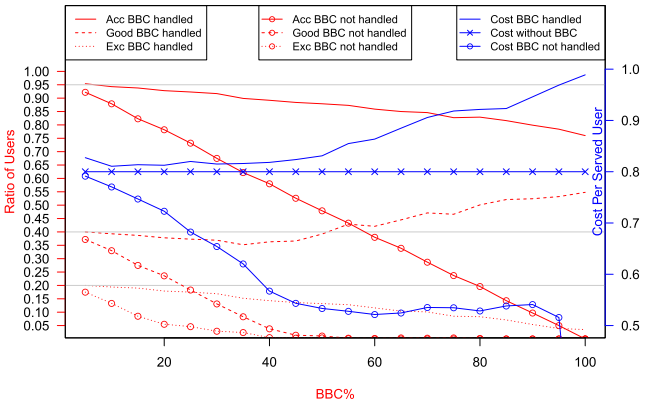

FIGURE 6. The impact of BBC users.

This relative robustness exhibited by the BBC detection and handling mechanism comes at a cost. The right-y axis represents the cost paid per served user. On the one hand, The blue curve with no markers shows the cost of the BBC detection and handling mechanism. It increases significantly as $BBC\%$ rises to compensate for the loss of the fine-grained learning users profiles. Nevertheless, these results show that the mechanism managed to achieve relative stability for the system with low $BBC\%$. The figure shows that with $BBC\% < 40\%$, the system manages to keep the user satisfaction ratios close to the goals without spending more costs than the costs spent when no BBC user is included (blue line with star markers). With $BBC\% > 40\%$, a coarse-grained and more costly service is offered to users.

On the other hand, the blue curve with circle markers plots the cost when BBC users are not handled. As BBC% increases, it decreases significantly since fewer users are getting *excellent* and *good* services.

This section presented the empirical evaluation of the proposed mechanism, highlighted its gains, and discussed its limitations. Next section goes through the related works.

V. RELATED WORKS

This section reviews respectively the related works in the domains of QoE, opponent modeling in multi-agent negotiation, and one-to-many negotiations and multi-agent negotiation for elasticity management.

A. QUALITY OF EXPERIENCE

Most of the works dealing with QoE is carried out on the conceptual front. These works have identified satisfaction management as a key process for providers to win the customer's loyalty, avoid client churn and survive in the open and competitive online environment. Few works attempted to provide the working frameworks allowing to integrate the satisfaction management. In these works, service providers base their planning on (estimated) percentages of users judging a service as ''poor or worse'' (%PoW), ''good or better'' (%GoB), or the percentage of users abandoning a service, ''Terminate Early'' (%TME). Bellecore is one of the first service providers to adopt this approach [6]. The *E-Model* [25]

is another project aimed at estimating the percentage of dissatisfied users from the ratings of subjective user tests. However, these tests were carried out offline and no guarantees have been provided regarding their applicability in real-life scenarios. Furthermore, to assess customer satisfaction, they relied on the MOS. The latter has received growing criticism in the QoE community since it fails to account for user diversity [6].

Nourikhah and Akbari [26] study the relationship between the perceived QoS by users and their level of satisfaction. The experiments conducted by them show certain latencies with photo viewing service on the mobile device therefore, the experience sampling method was used for recording their satisfaction level on a scale from one to five. It is not meant to treat the user data as metric as it is ordinal therefore, Bayesian data analysis with a generalized linear model is used for estimation of overall user satisfaction. Hence, it was proposed that opinion score distribution can be used to represent the quality of user experience instead of the MOS. The shortcomings of the current QoE management approaches are highlighted in [27] where user behavior is not taken into account. Their proposed study insights that there is a decrease of average QoE in the system with an increase of user patience because the user consumes resources without any wait of being served. Hence, the incorporation of these aspects of quality is often ignored in QoE modeling and management.

More recently, several works have started to rely on crowd-sourcing to measure and assess QoE [28]–[30]. For instance, in [28] the authors discuss how crowd-sourcing can support vendors, operators, and regulators to determine the QoE in the context of the new 5G networks that enable various new applications and network architectures. In another related work [29], the authors show that crowd-sourcing is a suitable way of measuring music streaming QoE by relating the results of crowd-sourcing experiments with lab results and highlighting the high correlation.

According to our knowledge, these are the only works attempting to equip the provider with fine-grained satisfaction management.

B. OPPONENT MODELING

Relying on machine learning techniques to establish a model of the opponent's negotiation behavior is a rich literature. The model is typically used to minimize the negotiation cost [31], [32], predict the opponent's bids [33], avoid exploitation [34], maximizes the agent's own utility [35], [36]. However, a fine-grained satisfaction management is not well studied since the opponent's interests are rarely taken into account except when the goal is to reach win-win settlements [37]. Furthermore, most of these works address a scenario of one-to-one negotiation [18] which is not applicable in the cloud computing ecosystem where the SaaS provider may negotiate with thousands of users simultaneously.

Moreover, while multiple works propose BBC strategies to model opponent imitation behavior [17], [38], [39], most of the works dealing with opponent modeling assume that the opponent follows TBC strategies [18]. Some recent works such as [33] and [40] propose a generic model enabling to predict the next offer of the opponent even when the latter adopts a BBC strategy. However, modeling BBC users in a satisfaction management mechanism is not covered in the literature. Recently, [41] considers both the current concession behavior of the proposing agent and the concession of its opponent in the last counteroffer to create a new offer. The resulting mechanism is a kind of imitating offer generation tactic applied in bilateral negotiation settings.

C. ONE-TO-MANY NEGOTIATIONS

One-to-many negotiation is a mature body of research. However, the goal of most of the exiting works is to reach one *atomic* agreement between, for instance, a buyer and several *concurrent* sellers. This agreement is the one that maximizes the buyer's utility whereas other sessions are aborted as in [42]–[45]. These solutions are not applicable for an open provider seeking to find agreement with the majority (maybe all) of its users.

This limitation is addressed by recent works on service composition where one-to-many negotiation is used to reach an agreement with multiple providers, each offering a distinct atomic service [46], [47]. Nevertheless, these works assume a closed set of atomic sellers all known before the outset of the negotiation.

D. CLOUD ELASTICITY MANAGEMENT

In the cloud computing context, multi-agent negotiation has been used for service composition and elasticity management [48]. For instance, An *et al.* [49] propose a one-to-many negotiation solution to handle resource allocation in the cloud. However, the provider accepts an offer only if it gains immediate payoff. For this reason, user satisfaction management or service acceptability rate are not taken into account. Another paper [50] proposes a Controlling Elasticity (ControCity) framework to control the elasticity of the resources using two essential components called buffer management to control the input queue of user's request at the application layer and elasticity management to control the elasticity of the cloud platform with learning automata technique at the middleware layer.

The services are required by cloud consumers for providing them with a certain level of QoS in addition to meeting their business requirements. A cloud computing service negotiation (CCSN) is proposed for providing simple or composite services and strategies to consumers [51]. It proposed a process for aggregation of the results of negotiations on simple task requirements for ensuring end-to-end service requirements.

A new model is proposed for QoS attributes which can be easily used and understood by cloud consumers [52]. The QoS attributes are classified into four major categories of technical, strategic & organizational, economic, and political & legislative. These QoS attributes can be fed into the multi-criteria decision to compare and rank these attributes

which helps in deciding the suitable services for cloud customers.

Our earlier architecture, AQUAMan [4], [20], is an adaptive mechanism for elasticity management. AQUAMan enables the provider to control the service acceptability rate. Nevertheless, it adopts a binary vision of acceptability (yes/no). Fine-grained satisfaction levels are not taken into account. For this reason, the user profile estimation in AQUAMan can only deal with acceptability rate control and is not adapted for the satisfaction management process.

VI. CONCLUSION AND FUTURE WORKS

In this article, we proposed a satisfaction management mechanism enabling the provider to achieve a set of user satisfaction goals while minimizing the cost it pays to the cloud provider. This problem is modeled as a linear optimization problem for which a linear solver is used. An alternative heuristic algorithm was also devised and compared with the optimal solution. The results showed that the provider should strike a balance between cost optimality and service responsiveness. This balance is influenced by the current influx of users and the priorities of the provider. Furthermore, the results showed that encouraging users to provide their subjective rating to the provider may lead to win-win outcomes since the service may become less costly for the provider and the user. While the proposed BBC detection and handling mechanism managed to relatively stabilize the user satisfaction and the cost with a low percentage of BBC users, When more users adopt BBC strategy, it provides a coarse-grained and costly service. One future work is to update the model towards achieving a fine-grained costless satisfaction management for BBC users by relying on advanced machine learning techniques.

If user agents choose to share some of their preferences with the provider, a less costly and more satisfactory outcome would be reached. Thus, another future research perspective is to improve the user agent to become capable of capturing the user's privacy and permission policy [53] and use it to determine what information can be communicated to the provider during the negotiation process.

A third future research perspective is to rely on Holonic MAS (HMAS) [54], [55] to define sub-layers of coordination to achieve scalability in large-scale situations if the number of agents scales largely. HMAS is a suitable paradigm for dynamic multilevel modeling and simulation [56]. HMAS structures a community of hierarchical agents, called holons, that could be composed of other agents [54]. The holon concept allows agents to group to create a higher-level agent named super-holon or be broken down into several lower-level agents named sub-holons. This would be helpful to deal with the bottleneck possibly introduced with the singularity of *ca*.

REFERENCES

[1] S. Möller and A. Raake, *Quality of Experience: Advanced Concepts, Applications and Methods*. Berlin, Germany: Springer, 2014.

[2] R. Schatz, T. Hoßfeld, L. Janowski, and S. Egger, "From packets to people: Quality of experience as a new measurement challenge," in *Data Traffic Monitoring and Analysis*. Berlin, Germany: Springer, 2013, pp. 219–263.

[3] T. Hossfeld, R. Schatz, and S. Egger, "SOS: The MOS is not enough!" in *Proc. 3rd Int. Workshop Qual. Multimedia Exper.*, Sep. 2011, pp. 131–136.

[4] A. Najjar, Y. Mualla, O. Boissier, and G. Picard, "AQUAMan: QoE-driven cost-aware mechanism for SaaS acceptability rate adaptation," in *Proc. Int. Conf. Web Intell.*, Aug. 2017, pp. 331–339.

[5] K. Brunnström, S. A. Beker, K. De Moor, A. Dooms, S. Egger, M. N. Garcia, T. Hossfeld, S. Jumisko-Pyykkö, C. Keimel, M. C. Larabi, and B. Lawlor, "Qualinet white paper on definitions of quality of experience," Eur. Netw. Qual. Exper. Multimedia Syst. Services, Brussels, Belgium, Tech. Rep., 2013.

[6] T. Hoßfeld, P. E. Heegaard, M. Varela, and S. Möller, "QoE beyond the MOS: An in-depth look at QoE via better metrics and their relation to MOS," *Qual. User Exper.*, vol. 1, no. 1, pp. 1–23, Dec. 2016.

[7] A. Najjar, C. Gravier, X. Serpaggi, and O. Boissier, "Modeling user expectations & satisfaction for SaaS applications using multi-agent negotiation," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, Oct. 2016, pp. 399–406.

[8] A. Najjar, X. Serpaggi, C. Gravier, and O. Boissier, "Multi-agent systems for personalized QoE-management," in *Proc. 28th Int. Teletraffic Congr. (ITC)*, vol. 3, Sep. 2016, pp. 1–6.

[9] R. Schatz, M. Fiedler, and L. Skorin-Kapov, "Qoe-based network and application management," in *Quality of Experience*. Cham, Switzerland: Springer, 2014, pp. 411–426.

[10] P. Reichl, S. Egger, R. Schatz, and A. D'Alconzo, "The logarithmic nature of QoE and the role of the weber-fechner law in QoE assessment," in *Proc. IEEE Int. Conf. Commun.*, May 2010, pp. 1–5.

[11] S. Egger, P. Reichl, T. Hoßfeld, and R. Schatz, "'Time is bandwidth'? Narrowing the gap between subjective time perception and quality of experience," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 1325–1330.

[12] D. G. Pruitt, *Negotiation Behavior*. New York, NY, USA: Academic, 2013.

[13] A. Sackl and R. Schatz, "Evaluating the impact of expectations on end-user quality perception," in *Proc. 4th Int. Workshop Perceptual Qual. Syst. (PQS)*, Sep. 2013, pp. 122–128.

[14] V. A. Zeithaml, L. L. Berry, and A. Parasuraman, "The nature and determinants of customer expectations of service," *J. Acad. Marketing Sci.*, vol. 21, no. 1, pp. 1–12, Jan. 1993.

[15] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," *IEEE Netw.*, vol. 24, no. 2, pp. 36–41, Mar. 2010.

[16] S. Khorsandroo, R. M. Noor, and S. Khorsandroo, "A generic quantitative relationship to assess interdependency of QoE and QoS," *KSII Trans. Internet Inf. Syst.*, vol. 7, no. 2, pp. 327–346, 2013.

[17] P. Faratin, C. Sierra, and N. R. Jennings, "Negotiation decision functions for autonomous agents," *Robot. Auto. Syst.*, vol. 24, nos. 3–4, pp. 159–182, Sep. 1998.

[18] T. Baarslag, M. J. Hendrikx, K. V. Hindriks, and C. M. Jonker, "Learning about the opponent in automated bilateral negotiation: A comprehensive survey of opponent modeling techniques," *Auto. Agents Multi-Agent Syst.*, vol. 30, pp. 849–898, Sep. 2015.

[19] S. Son and K. M. Sim, "A price- and-time-slot-negotiation mechanism for cloud service reservations," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 3, pp. 713–728, Jun. 2012.

[20] A. Najjar, O. Boissier, and G. Picard, "AQUAMAN: An adaptive QoE-aware negotiation mechanism for SaaS elasticity management," in *Proc. Int. Conf. Auto. Agents Multi-Agent Syst. (AAMAS)*, 2017, pp. 1655–1657.

[21] M. J. North, T. R. Howe, N. T. Collier, and J. R. Vos, "The repast simphony runtime system," in *Proc. Agent Conf. Generative Social Processes, Models, Mech.*, vol. 10. Chicago, IL, USA: Argonne National Laboratory and The Univ. of Chicago, 2005, pp. 13–15.

[22] Y. Mualla, W. Bai, S. Galland, and C. Nicolle, "Comparison of agent-based simulation frameworks for unmanned aerial transportation applications," *Procedia Comput. Sci.*, vol. 130, pp. 791–796, Jan. 2018.

[23] Y. Mualla, A. Najjar, A. Daoud, S. Galland, C. Nicolle, A.-U.-H. Yasar, and E. Shakshuki, "Agent-based simulation of unmanned aerial vehicles in civilian applications: A systematic literature review and research directions," *Future Gener. Comput. Syst.*, vol. 100, pp. 344–364, Nov. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18328462

[24] *Cplex Optimizer*. Accessed: Jan. 10, 2021. [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio

[25] "Transmission and multiplexing (TM); speech communication quality from mouth to ear for 3,1 kHz handset telephony across networks," Eur. Telecommun. Standards Inst., ETSI Tech. Rep. ETR 250, 1996.
[26] H. Nourikhah and M. K. Akbari, "Impact of service quality on user satisfaction: Modeling and estimating distribution of quality of experience using Bayesian data analysis," *Electron. Commerce Res. Appl.*, vol. 17, pp. 112–122, May 2016.
[27] T. Hossfeld, L. Atzori, P. E. Heegaard, L. Skorin-Kapov, and M. Varela, "The interplay between QoE, user behavior and system blocking in QoE management," in *Proc. 22nd Conf. Innov. Clouds, Internet Netw. Workshops (ICIN)*, Feb. 2019, pp. 112–117.
[28] T. Hoßfeld, S. Wunderer, A. Beyer, A. Hall, A. Schwind, C. Gassner, F. Guillemin, F. Wamser, K. Wascinski, M. Hirth, M. Seufert, P. Casas, P. Tran-Gia, W. Robitza, W. Wascinski, and Z. Ben Houidi, "White paper on crowdsourced network and QoE measurements—Definitions, use cases and challenges," 2020, *arXiv:2006.16896*. [Online]. Available: http://arxiv.org/abs/2006.16896
[29] A. Schwind, C. Moldovan, T. Janiak, N. D. Dworschak, and T. Hosfeld, "Don't stop the music: Crowdsourced QoE assessment of music streaming with stalling," in *Proc. 12th Int. Conf. Qual. Multimedia Exper. (QoMEX)*, May 2020, pp. 1–6.
[30] K. Borchert, A. Seufert, E. Gamboa, M. Hirth, and T. Hoßfeld, "*In vitro* vs *in vivo*: Does the study's interface design influence crowdsourced video QoE?" *Qual. User Exper.*, vol. 6, no. 1, pp. 1–16, Dec. 2020.
[31] R. Aydoğan and P. Yolum, "Learning opponent's preferences for effective negotiation: An approach based on concept learning," *Auto. Agents Multi-Agent Syst.*, vol. 24, no. 1, pp. 104–140, Jan. 2012.
[32] Y. Oshrat, R. Lin, and S. Kraus, "Facing the challenge of human-agent negotiations via effective general opponent modeling," in *Proc. 8th Int. Conf. Auto. Agents Multiagent Syst.*, vol. 1. Tokyo, Japan: Springer, 2009, pp. 377–384.
[33] S. Chen and G. Weiss, "OMAC: A discrete wavelet transformation based negotiation agent," in *Novel Insights in Agent-based Complex Automated Negotiation*. Tokyo, Japan: Springer, 2014, pp. 187–196.
[34] K. Hindriks, C. M. Jonker, and D. Tykhonov, "The benefits of opponent models in negotiation," in *Proc. IEEE/WIC/ACM Int. Joint Conf. Web Intell. Intell. Agent Technol.*, vol. 2, Sep. 2009, pp. 439–444.
[35] S. Saha, A. Biswas, and S. Sen, "Modeling opponent decision in repeated one-shot negotiations," in *Proc. 4th Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2005, pp. 397–403.
[36] C. Yu, F. Ren, and M. Zhang, "An adaptive bilateral negotiation model based on Bayesian learning," in *Complex Automated Negotiations: Theories, Models, and Software Competitions*. Berlin, Germany: Springer, 2013, pp. 75–93.
[37] N. van Galen Last, "Agent smith: Opponent model estimation in bilateral multi-issue negotiation," in *New Trends in Agent-Based Complex Automated Negotiations*. Berlin, Germany: Springer, 2012, pp. 167–174.
[38] A. Bahrammirzaee, A. Chohra, and K. Madani, "An adaptive approach for decision making tactics in automated negotiation," *Int. J. Speech Technol.*, vol. 39, no. 3, pp. 583–606, Oct. 2013.
[39] T. Baarslag, K. Hindriks, and C. Jonker, "A tit for tat negotiation strategy for real-time bilateral negotiations," in *Complex Automated Negotiations: Theories, Models, and Software Competitions*. Berlin, Germany: Springer, 2013, pp. 229–233.
[40] J. Hao and H.-F. Leung, "ABiNeS: An adaptive bilateral negotiating strategy over multiple items," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.*, vol. 2, Dec. 2012, pp. 95–102.
[41] K. Mansour, "A hybrid concession mechanism for negotiating software agents in competitive environments," *Int. J. Artif. Intell. Tools*, vol. 29, no. 6, Sep. 2020, Art. no. 2050016.
[42] K. Mansour and R. Kowalczyk, "A meta-strategy for coordinating of one-to-many negotiation over multiple issues," in *Foundations of Intelligent Systems*. Berlin, Germany: Springer, 2012, pp. 343–353.
[43] T. D. Nguyen and N. R. Jennings, "Coordinating multiple concurrent negotiations," in *Proc. 3rd Int. Joint Conf. Auto. Agents Multiagent Syst.*, vol. 3. Washington, DC, USA: IEEE Computer Society, 2004, pp. 1064–1071.
[44] B. Alrayes, O. Kafalı, and K. Stathis, "Concurrent bilateral negotiation for open e-markets: The CONAN strategy," *Knowl. Inf. Syst.*, vol. 56, no. 2, pp. 1–39, 2017.
[45] P. Bagga, N. Paoletti, B. Alrayes, and K. Stathis, "A deep reinforcement learning approach to concurrent bilateral negotiation," 2020, *arXiv:2001.11785*. [Online]. Available: http://arxiv.org/abs/2001.11785
[46] J. Richter, M. B. Chhetri, R. Kowalczyk, and Q. B. Vo, "Establishing composite SLAs through concurrent QoS negotiation with surplus redistribution," *Concurrency Comput., Pract. Exper.*, vol. 24, no. 9, pp. 938–955, Jun. 2012.
[47] K. Mansour and R. Kowalczyk, "On dynamic negotiation strategy for concurrent negotiation over distinct objects," in *Novel Insights in Agent-Based Complex Automated Negotiation*. Tokyo, Japan: Springer, 2014, pp. 109–124.
[48] M. G. Hafez and M. S. Elgamel, "Agent-based cloud computing: A survey," in *Proc. IEEE 4th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2016, pp. 285–292.
[49] B. An, V. Lesser, D. Irwin, and M. Zink, "Automated negotiation with decommitment for dynamic resource allocation in cloud computing," in *Proc. 9th Int. Conf. Auto. Agents Multiagent Syst. (AAMAS)*, vol. 1, 2010, pp. 981–988.
[50] M. Ghobaei-Arani, A. Souri, T. Baker, and A. Hussien, "ControCity: An autonomous approach for controlling elasticity using buffer management in cloud computing environment," *IEEE Access*, vol. 7, pp. 106912–106924, 2019.
[51] B. Shojaiemehr, A. M. Rahmani, and N. N. Qader, "Automated negotiation for ensuring composite service requirements in cloud computing," *J. Syst. Archit.*, vol. 99, Oct. 2019, Art. no. 101632.
[52] M. Eisa, M. Younas, and K. Basu, "Analysis and representation of QoS attributes in cloud service selection," in *Proc. IEEE 32nd Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, May 2018, pp. 960–967.
[53] T. Baarslag, A. Alper, R. Gomer, M. Alam, P. Charith, and E. Gerding, "An automated negotiation agent for permission management," in *Proc. 16th Conf. Auto. Agents MultiAgent Syst.* Kaiserslautern, Germany: DFKI, 2017, pp. 380–390.
[54] C. Gerber, J. Siekmann, and G. Vierke, "Holonic multi-agent systems," Res. Center Artif. Intell. (DFKI), Berlin, German, Tech. Rep. 681, 1999.
[55] S. Galland and N. Gaud, "Organizational and holonic modelling of a simulated and synthetic spatial environment," in *Proc. 10 Years Later E MAS*, in Lecture Notes in Computer Science, vol. 9068, 2015, pp. 1–23.
[56] H.-J. Bürckert, K. Fischer, and G. Vierke, "Holonic transport scheduling with teletruck," *Appl. Artif. Intell.*, vol. 14, no. 7, pp. 697–725, Aug. 2000.

AMRO NAJJAR received the Ph.D. degree in computer science from Mines Saint Étienne, France, in 2015. His Ph.D. Thesis was Multi-Agent Negotiation for QoE-Aware Cloud Elasticity Management. From 2018 to 2019, he was working as a Postdoctoral Researcher with Umeå University, Sweden. He is currently a Research Associate with the University of Luxembourg. Since 2013, he has published more than 45 peer-reviewed articles. His research interests include multi-agent systems, explainable AI (XAI), social robots, and human–computer interaction.

YAZAN MUALLA received the Ph.D. degree in computer science from Université Bourgogne Franche-Comté, France. From 2010 to 2014, he was a Field Engineer with the Drilling and Measurements Segment, Schlumberger, an international company. He is currently a Researcher and a Teacher with the Belfort-Montbéliard University of Technology (UTBM), France. His research interests include multi-agent systems, explainable AI, human–computer interaction, and cyber physical systems.

KAMAL DEEP SINGH received the B.Tech. degree in electrical engineering from the IIT Delhi, New Delhi, India, in 2002, and the Ph.D. degree in computer science from the University of Rennes 1, Rennes, France, in 2007. He then joined the Dionysos Group, National Research Institute in Computer Science (INRIA), as a Postdoctoral Researcher, where he co-developed many components of quality-of-experience estimation tools and worked mainly on the analysis of video-based applications. He was a Postdoctoral Researcher with the Telecom Bretagne, Rennes, where he worked on the Internet of Things and cognitive radio. He is currently an Associate Professor with the Université de Saint-Étienne, Saint-Étienne, France, where he is a part of the research team Data Intelligence, Laboratoire Hubert Curien. His research interests include the Internet of Things, mobile networks, AI, quality of experience, and software-defined networking.

GAUTHIER PICARD is Senior Research Fellow, PhD, HDR at ONERA, Information Processing and Systems Department, University of Toulouse, Smart and decision-making systems Unit. His research topics include cooperation and adaptation in multiagent systems and distributed optimization, which are mainly applied to satellite mission planning, aircraft design and ambient intelligence, as part of various industrial and academic projects at national and European level.

DAVIDE CALVARESI is currently a Senior Researcher with the University of Applied Sciences and Arts Western Switzerland (HES-SO). His main endeavor currently focuses on bridging sub-symbolic and symbolic AI to foster explainability in Multi-Agent Systems in the context of the European Project EXPECTATION. He is also a co-founder of the startup Wriggle Solutions pursuing safety on the street with the project SMARTtyre–real-time monitoring of the tires' consumption and structure–holding two patents. His research interests include real-time multi-agent systems, explainable AI, blockchain, and assistive technologies. He has been the chair of several workshops in those areas, such as EXTRAAMAS from 2019 to 2021, BCT4MAS from 2018 to 2020, and RTcMAS2018.

AVLEEN MALHI received the Ph.D. and M.Sc. degrees in computer science engineering from Thapar University, India, in 2012 and 2016, respectively. She is currently working as a Postdoctoral Researcher with the Department of Computer Science, Aalto University. She is also working on H2020 Smart Cities-based project and plays a leading role in Business Finland Project, eParkly for smart parking management system. She worked on the security of Vehicular Ad-Hoc Networks on an Industry Sponsored Project by TATA Consultancy Services for four years during her Ph.D. She has numerous journal and conference articles mainly in the area of machine learning, the IoT, and security. Her research interests include the IoT, machine learning, and information security.

STÉPHANE GALLAND received the Ph.D. degree from the National High School of Mines of Saint-Etienne, France, in 2001, with an agent-oriented approach for the design and the implementation of simulations of distributed manufacturing companies. He is currently the Deputy Director of the Research Laboratory on Distributed Knowledge and Artificial Intelligence (CIAD), University of Technology of Belfort-Montbéliard, France. Since 2002, he has been a member of the University of Technology of Belfort-Montbéliard, where he is doing his research on agent-oriented software engineering and agent-based simulation, both applied to the modeling and the simulation of complex systems (smart cities and crowds). He is also one of the major contributors of the ASPECS methodology, SARL agent-programming language, and Janus agent framework.

KARY FRÄMLING (Member, IEEE) is currently a Professor in data science with Umeå University and an Adjunct Professor in computer science with Aalto University. He is also the founder of several companies and the former Chair of the IoT Work Group, The Open Group. He is the author of over 100 papers published in scientific journals and conferences, including the (presumably) first article that mentions the IoT and describes an operational implementation in 2002. Since then, he has been the main architect of the IoT systems in various domains, such as buildings, HVAC equipment, vehicles, supply chain management, and smart cities, under the umbrella concept of intelligent products. His current core interest is developing new methods for explainable artificial intelligence and the related machine learning technologies.