One-to-Many Multi-agent Negotiation and Coordination Mechanisms to Manage User Satisfaction

Amro Najjar
Department of Computer Science,
Umea University
Umea, Sweden
najjar@cs.umu.se

Yazan Mualla
LE2I, Univ. Bourgogne Franche-Comté, UTBM
Belfort, France
yazan.mualla@utbm.fr

Kamal Singh
Hubert Curien Laboratory,
UMR CNRS, 5516
Saint-Etienne, France
kamal.singh@univ-st-etienne.fr

Gauthier Picard
Hubert Curien Laboratory,
UMR CNRS, 5516
Saint-Etienne, France
gauthier.picard@emse.fr

ABSTRACT

Quality of Experience (QoE) is defined as the measure of end-user satisfaction with the service. 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*. The aim of this article is to go beyond this binary vision and propose a QoE management mechanism. In particular, 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 in order to improve the responsiveness, when the system has to scale up with fast growing number of users. Both are implemented and experimentally evaluated against state-of-the-art one-to-many negotiation frameworks.

KEYWORDS

Multi-agent coordination; negotiation; user satisfaction; SaaS; cloud computing

ACM Reference Format:
Amro Najjar, Yazan Mualla, Kamal Singh, and Gauthier Picard. 2018. One-to-Many Multi-agent Negotiation and Coordination Mechanisms to Manage User Satisfaction. In *Proceedings of IJCAI-ECAI-AAMAS Workshop (ACAN 2018).* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

End-user (or User) satisfaction is a key factor to ensure the success of any online service. Among different user satisfaction determinants, Quality of Experience (QoE) appeared in 2000's as a metric to assess the service quality as perceived subjectively by the user. Since this field is still in early stages, most of the studies dealing with QoE are carried out on the conceptual front. Yet, QoE targets 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 [21]. Nevertheless, most of the existing works are provider-centric since QoE-management decision is taken unilaterally by provider.

Furthermore, they rely on the Mean Opinion Score (MOS) to assess the satisfaction and the service acceptability [31]. MOS is an average of the users' opinions. Thus it hides important information about the users diversity and personal preferences [14].

By definition, agents are self-interested and bound to an individual perspective. For this reason, agents have been proposed to integrate users' personal preferences into the decision loop. However, when assessing users satisfaction, these works rely on service acceptability as a *binary* decision, in which the user decides to accept or reject the proposed service (as in [23], for example). Nevertheless, studies on user satisfaction and QoE suggest that user satisfaction should not be represented as a binary threshold [31], but rather as a *degree* reflecting the user's delight or annoyance of an application or service [8]. To capture this nuanced nature of user satisfaction, recent results of user polls and subjective user tests recommend providers to rely on percentiles, in order to assess the user's subjective estimation of the service quality [15]. 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 a 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. The contribution of this work is threefolds:

(1) We formulate the user satisfaction problem as an optimization problem, whereby the provider seeks to minimize the cost or resource constraint, while meeting a set of business

constraints guaranteeing the subjective satisfaction experienced by the service users;

(2) We propose a heuristic algorithm to solve this problem and experimentally compare its solution with the optimal solution in terms of both cost and user satisfaction;

(3) We implement both solutions and evaluate our contribution against SoTA methods on a Cloud-based case study.

The rest of this article is organized as follows. Section 2 introduces the one-to-many negotiation architecture. Section 3 defines the satisfaction management problem and details the optimal model and proposed heuristic algorithm. Section 4 details the experimental evaluation. Section 5 reviews related works from the literature and Section 6 concludes this article.

2 THE ONE-TO-MANY NEGOTIATION ARCHITECTURE

The one-to-many-negotiation addressed in this article involves a provider negotiating simultaneously with multiple users. The aim of the provider is 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 service users. $Serv$, 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 have issues such as video resolution, video transcoding time, etc. $Goal$ is the provider set of satisfaction goals for the satisfaction categories K. RC is the cost or resource constraint.

2.1 Negotiation Protocol and Strategies

To model the one-to-many negotiation defined in (1), we opt for a one-to-many multi-agent negotiation approach as proposed in [22]. The sections below describe the negotiation process.

2.1.1 Negotiation Protocol. Each $u_i \in Users$ is represented by an agent, noted sa_i. $Provider$ is modeled as a multi-agent system, composed of 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's, and initializes their negotiation strategy. Moreover, it oversees the negotiation process and may intervene on delegates in some sessions to push the provider business goals and budget constraints. On their side, each user agent sa_i seeks to maximize the QoE of 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 $o = \langle at_1, at_2, ..., at_J \rangle$ that defines a value for each one of the at_j of $Serv$. The negotiation protocol is based on the alternate offer protocol. 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[1].

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

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

2.1.2 User Utility Function. If the service under negotiation involves multiple attributes, the user utility function, 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 [30]. Thus, M_{sa_i} is defined as follows:

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

w_{sa_i,at_j} is a weight obtained from the user that indicates how much importance the user gives to attribute at_j. $o^t_{da_i}[at_j]$ is the value assigned to attribute at_j in offer $o^t_{da_i}$. Note that μ_{sa_i,at_j} are functions of the preferred and the reserved values that sa_i accepts for each attribute. The preferred value is the ideal value that yields the maximum utility for this attribute whereas the reservation value is the worst acceptable value (equivalent to MOS $\approx 2.5/5$), beyond which no agreement is better than an agreement [27]. From the perspective of customer expectation management literature, these values correspond respectively to the *desired expectations* and *adequate expectations* [35].

2.1.3 Negotiation Strategies. To reach agreements, agents must make concessions relying on concession strategies. In this article, we assume that both sa_i and da_i can use Time-Based Concession (TBC). This assumption is very common in the literature especially in works aiming to construct a model of the opponent [5]. The concession made by sa_i for the cycle t is computed as follows [32]:

$$\Delta U^t_{sa_i} = U^{t-1}_{sa_i} \cdot \left(\frac{t}{T_{sa_i}}\right)^{\lambda_{sa_i}} \tag{2}$$

Where $U^t_{sa_i}$ is the utility sa_i expects to obtain in the cycle t, T_{sa_i} is the negotiation time deadline. λ_{sa_i} controls the convexity degree of sa_i's concession curve.

2.2 Learning the User Profile

In order 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 existing works, attempting to establish a model of the opponent profile, assume that the opponent employs a TBC strategy [5]. Dealing with opponents employing a Behavior-Based Concession (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[2] 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.

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

2.2.1 TBC Users. The mechanism used to model TBC users is based on our earlier work [22] 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 get $\overline{T_{sa_i}}$ an estimate of sa_i time deadline.

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

$$\overline{T^t_{sa_i}} - t > d \tag{3}$$

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 (c.f. Section 3.1).

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 add them to the $list$.

2.2.2 BBC Users. The mechanism to deal with BBC users has two steps: detection and handling. A delegate da_i is not aware at the beginning of the negotiation whether sa_i is TBC or BBC. Therefore, da_i assumes that sa_i is TBC by dealing with it as explained in Section 2.2.1, while it runs a detector to infer if sa_i is BBC. The BBC detector within da_i tracks the imitation behavior of sa_i by comparing the exchanged offers. Several attributes determine how the detector works. First, the detection starting cycle. Second, the window of how many previous cycles to include in calculating the average concession made by sa_i. Third, 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 that there is a concession or not. These attributes have been defined empirically due to the complexity of defining them formally as been stated in the literature [5].

Once sa_i is detected as BBC, da_i changes how it handles the negotiation. In this case, 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 offering them to sa_i given that for the whole process, da_i does not offer more than the cost allocated to it by ca.

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

2.3 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 (c.f. Section 3), 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 into the ongoing negotiation sessions in order to undertake satisfaction management. This process is detailed in the following sections.

3 USER SATISFACTION MANAGEMENT MECHANISM

This section details the core contribution of this article. Section 3.1 describes the satisfaction management. Section 3.2 models the user satisfaction management problem as an optimization problem. Section 3.3 details the proposed heuristic algorithm.

3.1 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:

$$M_{sa_i}(\hat{o}) \geq h_k \tag{4}$$

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 (4) differs from one user to another since M_{sa_i} is personal.

In order to minimize the cost and know which user could be 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 bellow. If the *optimalMode* is active, the algorithm relies on the optimal solution denoted as `OptimalSM` described in Section 3.2. Otherwise, it relies on the heuristic algorithm denoted as `HeuristicSM` (Section 3.3). 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 $k$
1  At the outset of each iteration ;
2  Profiles ← getUserProfiles() ;
3  if (optimalMode ==true) then
4      tailoredOffers ← OptimalSM()
5  else tailoredOffers ← HeuristicSM();
6  foreach Offer $o_i$ of tailoredOffers do
7      if (propose($o_i$, $sa_i$)==true) then
8          $rating_i$ ← requestRating()
9  foreach Satisfaction Category $k$ of $K$ do
10     updateAlready($k$, Rating)
```

3.2 User Satisfaction Optimization Problem

The 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:

$$\textbf{Minimize} \sum_{k=1}^{K}\sum_{i=1}^{N^t} C_i^k \cdot X_i^k \tag{5}$$

Subject to

$$\sum_{k=1}^{K} X_i^k \leq 1, \ \forall i \tag{6}$$

$$\sum_{m=1}^{k}\left(\frac{\sum_{i=1}^{N^t} X_i^m + Already[m]}{\#TerminatedSessions}\right) \geq Goal[k], \ \forall k \tag{7}$$

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.

The constraint in (6) 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.

The constraint in (7) 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 case of infeasibility, a rare case that occurs when the number of users in $list^t$ is not enough to satisfy the constraint in (7), 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. Thus, users remain in $list$ until they accept their tailored offer $\overline{o_{sa_i}}$ or until their negotiation session is terminated because of reaching their time deadline. 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.

3.3 Heuristic Satisfaction Management Algorithm

In 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 3.3.1). Second, it picks up users and assigns them to what it considers the best category they fit in.

3.3.1 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 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.

3.3.2 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 go down.

- *DISTANCE_TO_GOAL*: A category k, whose distance to achieve its goal (*i.e.* $Goal[k]$) is greater, has the priority over other categories. These distances are calculated using the $Already$ array and $\#TerminatedSessions$ defined in Section 3.2. 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 3.2), 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.

4 EXPERIMENTAL EVALUATION

To evaluate the proposed satisfaction management mechanism, we implement it as a Software as a service (SaaS) using Repast Simphony [25]. The latter is a Java-based multi-agent simulation environment. The optimal solution is computed using IBM's CPLEX optimizer [16]. In this implementation and as per (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 is organized as follows. Section 4.1 introduces the parameters of the experiments. Section 4.2 compares the optimal solution with the proposed heuristics. The experiment in Section 4.3 studies the impact of the percentage of users rating the service after being served. The experiment in Section 4.4 analyzes the impact of the percentage of users employing tit-for-tat (i.e. BBC) strategies.

4.1 Experiments Parameters

$|SU|$, the total number of users entering the simulation is 10000 users (except for Section 4.2 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 that are generated randomly. $T_{sa_i} \in [40 : 120]$ cycle. *RC*, the delegate reservation cost

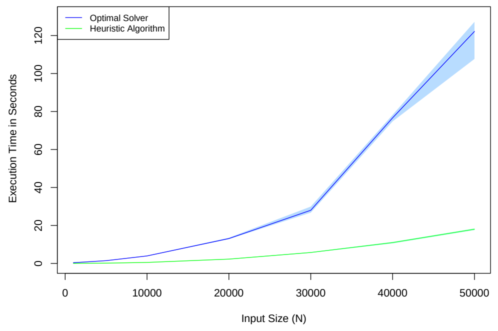

Figure 1: The average execution time of the optimal solution (blue curve) and the heuristic algorithm (green curve) with minimum and maximum values.

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. Due to space constraints, we will not discuss the impact of RC.

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, etc. 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 service, the utility perceived by the user agent should be greater than 0.75 etc. $f\%$ is the percentage of users choosing to give their rating after they accept the service. The impact of this parameter is studied in Section 4.3. *BBC%* stands for the percentage of BBC users in the simulation, its impact is studied in Section 4.4. The results of the experiments are obtained by averaging the outcomes of at least 10 simulation runs.

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

4.2 Optimal Solution vs. Heuristic Algorithm

In order to evaluate the computational cost of the optimal solution, Section 4.2.1 compares the execution time of CPLEX optimizer with the proposed heuristic algorithm. Then, Section 4.2.2 evaluates the heuristic strategies. We compare the cost they spend on served users with the minimal cost calculated by the optimal solver.

4.2.1 Execution Time. Figure 1 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

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%

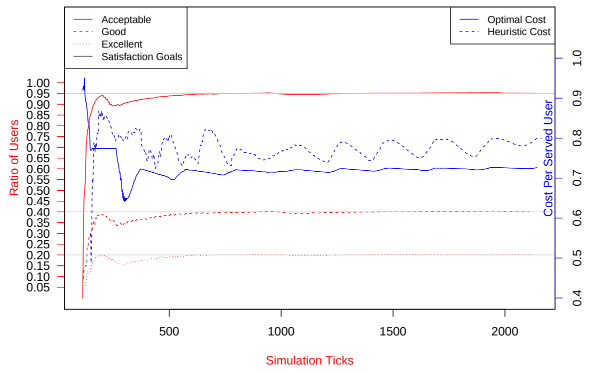

Figure 2: The user satisfaction achieved by the heuristic strategies (red curves) and their cost (blue dashed curve) compared with the optimal solution cost (blue solid curve).

solution becomes impractical. On the other hand, the green curve shows that the execution time of the heuristic algorithm is much shorter even with bigger N.

4.2.2 Cost of the Heuristic Strategies. Table 1 compares the cost spent per served user by each one of the heuristic strategies (Section 3.3.2) 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 1 and Table 1 show that the provider must address a trade-off between cost optimality and responsiveness. For instance, it can use the optimal solution with a relatively low influx of users. On the other hand, with higher influx, it becomes expensive in terms of responsiveness. This may degrade the user experience and lead to client churn.

Figure 2 studies the results achieved by the chosen heuristic strategy in one run as a function of 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), the cost constraint

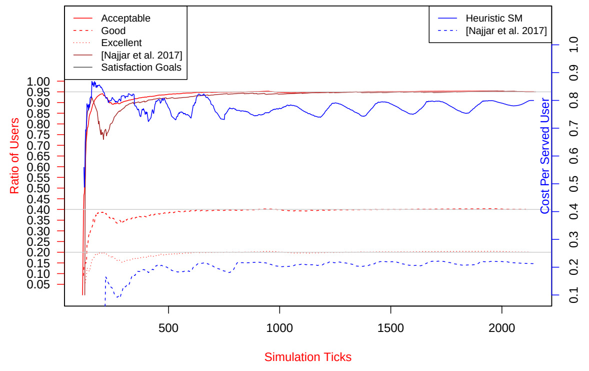

Figure 3: User acceptability rate (left y-axis) and its cost (right y-axis) using the proposed heuristic Satisfaction Management (SM) algorithm compared to [23].

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. However, this is compensated by the shorter execution time.

Note that while Figure 2 plots the results obtained during the simulation run, Figure 4 and 5 plot an average of results obtained at the end of multiple simulation runs.

4.2.3 Heuristic Algorithm vs. State-of-the-art. According to our knowledge, other works, such as AQUAMan [23], 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 3 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. 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.

4.3 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 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 4 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

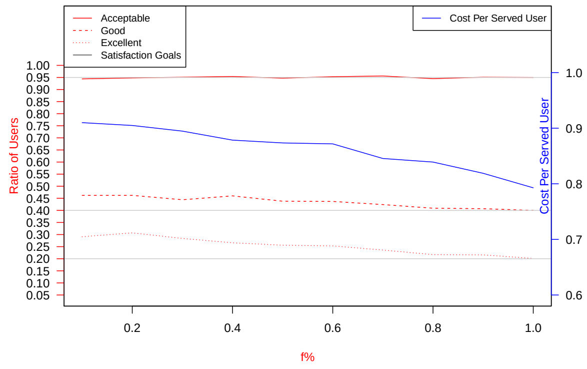

Figure 4: The impact of $f\%$ (x-axis) on the ratio of users assigned to each satisfaction category (left y-axis) and on the cost spent per served user (right y-axis).

no rating from 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 are 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 higher $f\%$, the cost drops significantly. This is explained as follows. Using the feedback from 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, in order to encourage users to give their feedback (and hence increasing $f\%$), the provider can choose to share this 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[3].

4.4 Impact of BBC Users

This section studies the impact of adding BBC users. Figure 5 compares the user satisfaction results obtained by the BBC detection and handling mechanism proposed in Section 2.2.2 (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 3.3.2). This strategy prioritizes categories whose distance to their goal is furtherest. 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.

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 in order to compensate the loss of the fine-grained learning users profiles. Nevertheless, these results show that the mechanism managed to achieve a 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 less users are getting *excellent* and *good* services.

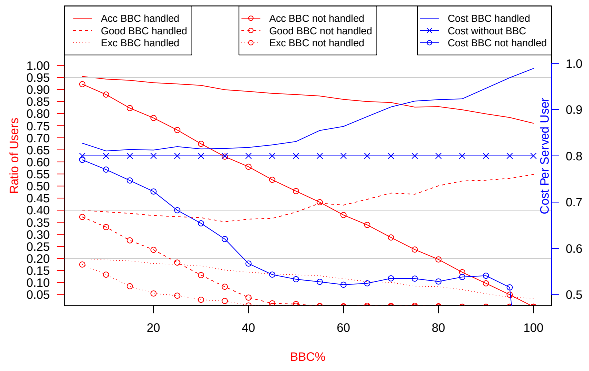

Figure 5: The Impact of BBC Users.

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

5 RELATED WORKS

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

5.1 Quality of Experience

Most of the works dealing with QoE are carried out on the conceptual front. These works have identified the 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 [15, 18]. The *E-Model* [17] 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 Mean Opinion Score (MOS). The latter has received a growing criticism in the QoE community since it fails to account for user diversity [15]. According to our knowledge, these are the only works attempting to equip the provider with a fine-grained satisfaction management.

5.2 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 [3, 26], predict the opponent's bids [9], avoid exploitation [13], maximize the agent's own utility [29, 34]. 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 [33]. Furthermore, most of these works address a scenario of one-to-one negotiation [5] 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 [6, 7, 10], most of works dealing with opponent modeling assume that the opponent follows TBC strategies [5]. Some recent works such as [9] and [12] 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.

5.3 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 maximize the buyer's utility whereas other sessions are aborted as in [1, 19, 24]. 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 agreement with multiple providers, each offering a distinct atomic service [20, 28]. Nevertheless, these works assume a closed set of atomic sellers all known before the outset of the negotiation.

5.4 Cloud Elasticity Management

In the cloud computing context, multi-agent negotiation has been used for service composition and elasticity management [11]. For instance, An *et al.* [2] 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.

Our earlier architecture, AQUAMan [22, 23], 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. AQUAMan's mechanism is detailed in [23]. Yet, this work's main contribution seems to be focused on analyzing the cost paid by the SaaS to the cloud provider.

6 CONCLUSIONS 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 the cost optimality and the 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 [4] and use it to determine what information can be communicated to the provider during the negotiation process.

REFERENCES

[1] Bedour Alrayes, Özgür Kafalı, and Kostas Stathis. 2017. Concurrent bilateral negotiation for open e-markets: the Conan strategy. *Knowledge and Information*

Systems (2017), 1–39.

[2] Bo An, Victor Lesser, David Irwin, and Michael Zink. 2010. Automated negotiation with decommitment for dynamic resource allocation in cloud computing. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 981–988.

[3] Reyhan Aydoğan and Pınar Yolum. 2012. Learning opponent's preferences for effective negotiation: an approach based on concept learning. *Autonomous Agents and Multi-Agent Systems* 24, 1 (2012), 104–140.

[4] Tim Baarslag, Alper T Alan, Richard Gomer, Muddasser Alam, Charith Perera, Enrico H Gerding, et al. 2017. An Automated Negotiation Agent for Permission Management. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 380–390.

[5] Tim Baarslag, Mark JC Hendrikx, Koen V Hindriks, and Catholijn M Jonker. 2015. Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems* (2015), 1–50.

[6] Tim Baarslag, Koen Hindriks, and Catholijn Jonker. 2013. A Tit for Tat Negotiation Strategy for Real-Time Bilateral Negotiations. *Complex Automated Negotiations: Theories, Models, and Software Competitions* (2013), 229–233.

[7] Arash Bahrammirzaee, Amine Chohra, and Kurosh Madani. 2013. An adaptive approach for decision making tactics in automated negotiation. *Applied intelligence* 39, 3 (2013), 583–606.

[8] Kjell Brunnström, Sergio Ariel Beker, Katrien De Moor, Ann Dooms, Sebastian Egger, Marie-Neige Garcia, Tobias Hossfeld, Satu Jumisko-Pyykkö, Christian Keimel, Mohamed-Chaker Larabi, et al. 2013. Qualinet white paper on definitions of quality of experience. (2013).

[9] Siqi Chen and Gerhard Weiss. 2014. OMAC: a discrete wavelet transformation based negotiation agent. In *Novel Insights in Agent-based Complex Automated Negotiation*. Springer, 187–196.

[10] Peyman Faratin, Carles Sierra, and Nick R Jennings. 1998. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems* 24, 3 (1998), 159–182.

[11] Mohamed Galal Hafez and Mohamed Shaheen Elgamel. 2016. Agent-Based Cloud Computing: A Survey. In *Future Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference on*. IEEE, 285–292.

[12] Jianye Hao and Ho-Fung Leung. 2012. ABiNeS: An adaptive bilateral negotiating strategy over multiple items. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on*, Vol. 2. IEEE, 95–102.

[13] Koen Hindriks, Catholijn M Jonker, and Dmytro Tykhonov. 2009. The benefits of opponent models in negotiation. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, Vol. 2. IEEE, 439–444.

[14] Tobias Ho*ß*feld, Raimund Schatz, and Sebastian Egger. 2011. SOS: The MOS is not enough!. In *Quality of Multimedia Experience (QoMEX), 2011 Third International Workshop on*. IEEE, 131–136.

[15] Tobias Hoßfeld, Poul E Heegaard, Martín Varela, and Sebastian Möller. 2016. Qoe beyond the mos: an in-depth look at qoe via better metrics and their relation to mos. *Quality and User Experience* 1, 1 (2016).

[16] IBM. [n. d.]. CPLEX Optimizer. https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

[17] European Telecommunications Standards Institute. 1996. ETSI Technical Report ETR 250 (1996) Transmission and multiplexing (TM); speech communication quality from mouth to ear for 3,1 kHz handset telephony across networks.

[18] ITU. 1993. TU-T P.Suppl. 3 to ITU-T Series P Recommendations, models for predicting transmission quality from objective measurements.

[19] Khalid Mansour and Ryszard Kowalczyk. 2012. A meta-strategy for coordinating of one-to-many negotiation over multiple issues. In *Foundations of Intelligent Systems*. Springer, 343–353.

[20] Khalid Mansour and Ryszard Kowalczyk. 2014. On dynamic negotiation strategy for concurrent negotiation over distinct objects. In *Novel Insights in Agent-based Complex Automated Negotiation*. Springer, 109–124.

[21] Sebastian Möller and Alexander Raake. 2014. *Quality of experience: advanced concepts, applications and methods*. Springer International Publishing.

[22] Amro Najjar, Olivier Boissier, and Gauthier Picard. 2017. AQUAMan: An Adaptive QoE-Aware Negotiation Mechanism for SaaS Elasticity Management (extended abstract). In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

[23] Amro Najjar, Yazan Mualla, Olivier Boissier, and Gauthier Picard. 2017. AQUAMan: QoE-Driven Cost-Aware Mechanism for SaaS Acceptability Rate Adaptation. In *2017 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*.

[24] Thuc Duong Nguyen and Nicholas R Jennings. 2004. Coordinating multiple concurrent negotiations. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*. IEEE Computer Society, 1064–1071.

[25] Michael J North, Thomas R Howe, Nick T Collier, and Jerry R Vos. 2005. The repast simphony runtime system. In *Proceedings of the agent 2005 conference on generative social processes, models, and mechanisms*, Vol. 10. ANL/DIS-06-1, co-sponsored by Argonne National Laboratory and The University of Chicago, 13–15.

[26] Yinon Oshrat, Raz Lin, and Sarit Kraus. 2009. Facing the challenge of human-agent negotiations via effective general opponent modeling. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 377–384.

[27] Dean G Pruitt. 2013. *Negotiation behavior*. Academic Press.

[28] Jan Richter, M Baruwal Chhetri, Ryszard Kowalczyk, and Q Bao Vo. 2012. Establishing composite SLAs through concurrent QoS negotiation with surplus redistribution. *Concurrency and Computation: Practice and Experience* 24, 9 (2012), 938–955.

[29] Sabyasachi Saha, Anish Biswas, and Sandip Sen. 2005. Modeling opponent decision in repeated one-shot negotiations. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*. ACM, 397–403.

[30] Raimund Schatz, Markus Fiedler, and Lea Skorin-Kapov. 2014. QoE-based network and application management. In *Quality of Experience*. Springer, 411–426.

[31] Raimund Schatz, Tobias Hoßfeld, Lucjan Janowski, and Sebastian Egger. 2013. From packets to people: Quality of experience as a new measurement challenge. In *Data traffic monitoring and analysis*. Springer, 219–263.

[32] Seokho Son and Kwang Mong Sim. 2012. A price-and-time-slot-negotiation mechanism for cloud service reservations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42, 3 (2012), 713–728.

[33] Niels van Galen Last. 2012. Agent Smith: Opponent Model Estimation in Bilateral Multi-issue Negotiation. *New Trends in Agent-Based Complex Automated Negotiations* (2012), 167–174.

[34] Chao Yu, Fenghui Ren, and Minjie Zhang. 2013. An Adaptive Bilateral Negotiation Model Based on Bayesian Learning. In *Complex Automated Negotiations: Theories, Models, and Software Competitions*. Springer, 75–93.

[35] Valarie A Zeithaml, Leonard L Berry, and Arantharanthan Parasuraman. 1993. The nature and determinants of customer expectations of service. *Journal of the academy of Marketing Science* 21, 1 (1993), 1–12.