García García, I.; Sebastiá Tarín, L. (2014). A negotiation framework for heterogeneous group recommendation. Expert Systems with Applications. 41(4):1245-1261. doi:10.1016/j.eswa.2013.07.111.

A Negotiation Framework for Heterogeneous Group Recommendation

Inma Garcia [a,1,*], Laura Sebastia [a]

[a] *Universidad Politecnica de Valencia, Spain*

Abstract

Over the last years, some remarkable recommender systems for group of users have been developed. When using most of these systems, each group member communicates his/her preferences to the system, which obtains a group profile as the result of an equal weighting of the individual preferences. This way, no member is particularly dissatisfied with the recommendations. However, this is not a realistic situation, given that not all the members in a group act in the same manner. This paper deals with the problem of recommendation for a group of users, where, besides his/her own preferences, each user may have different expectations about the result of the recommendation and may exhibit a different behaviour with respect to the other group members. Moreover, all this information is private and may be revealed under certain circumstances. In this context, we have opted for building a multi-agent system, where an agent acts on behalf of one group member. We have implemented a *UserAgent* that can be configured in order to exhibit the behaviour desired by the corresponding user. Then, different UserAgents negotiate with the aim of building a group profile that satisfies their particular minimum requirements, while preserving some privacy. Moreover, we have designed a *NegotiatorAgent*, which governs the negotiation and may act as a mediator in order to facilitate the agreement. Finally, we have performed some experiments that show that this mechanism is able to give a response in this heterogeneous environment.

Key words: Group recommender systems, group profile, negotiation, alternative offers protocol.

* Corresponding author. Tel: +34 963 870 000
Email addresses: ingarcia@dsic.upv.es (Inma Garcia), lstarin@dsic.upv.es (Laura Sebastia).

[1] Partial support provided by Consolider Ingenio 2010 CSD2007-00022, Spanish Government Project MICINN TIN2011-27652- C03-01.

1 Introduction

A **Recommender System** (RS) [33] is a personalization tool that attempts to provide people with lists of items that best fit their individual tastes. A RS infers the user's preferences by analyzing the available user data, information about other users and information about the environment. While many RSs are focused on making recommendations to a single user, many daily activities such as watching a movie or going to a restaurant involve a group of users [5], in which case recommendations must take into account the tastes and preferences of all the users in the group [1]. This type of system is called a **Group Recommender System** (GRS). Over the last few years, GRSs have been an active area of research within the field of RS. As a result, some remarkable GRSs have been developed. For example, *Polylens* [29] recommends movies, as an extension of the MovieLens recommender; *MusicFX* [26] selects a radio station among 91 stations; *Intrigue* [1], *The Collaborative Advisory Travel System* [27] and *Travel Decision Forum* [12,14] deal with a tourist domain; *GRSK* [9,10] is a generic GRS that can be used with any application domain.

The main issue in group recommendation is to identify the items that are likely to satisfy all the group members adequately [12,31]. Most of the GRSs lately developed are based on the aggregation of the preference models of individual users into a group model that is then used to elicit a recommendation that is satisfactory for the whole group [12,31]. This *centralized* view has two main implications:

- Most GRSs tend to favour an equal weighting of the individual preferences when recommending an item for the group such that no member is particularly dissatisfied with the decisions. However, some authors criticize these aggregation strategies because the ratings are combined always in the same way without considering how the members in the group interact with each other [6,32]. For example, we can find a person that wants to favor a specific user or that feels more comfortable when accepts the others' proposals instead of trying to impose his/her preferences over the other group members. That is, new trends in GRSs [24,32] argue that it is more realistic to capture the different attitudes of each member in the group with respect to the others.
- Centralized systems need that the user communicates his/her preferences to the system in order to provide a recommendation. However, not all the users feel comfortable in this situation, given that they consider that preferences are delicate information which should not be revealed to everybody. That is, a GRS can be also considered as a *distributed system*, where each user has private information and the system only knows the information that the user wishes to make public.

In summary, the problem we are facing in this work has the following characteristics:

- It is a **group recommendation** problem, where a group of users want to obtain a recommendation of a list of items that match their preferences as a whole.
- Each group member has his/her own preferences and may exhibit a **different behaviour**. For example, different users may have different expectations about the resulting group profile, may want to impose their preferences over the other users' preferences or may like better to agree with the others' proposals.
- Both the preferences and the user expectations are **private information** to each user, who decides which information he/she wants to share with the other users at any given moment.

Therefore, the task of the GRS is to manage, joint with the individual preferences, some other aspects of the users behaviour to come up with a common group profile, only taking into account the information provided by the users until this moment. This may imply, for example, that the final list of recommended items in our case may contain more items suitable for a user than for another and both can be equally satisfied.

In order to deal with this problem, we have opted for **building a multi-agent system (MAS)** [39], where multiple agents work together in order to obtain a recommendation for the whole group. That is, ours is not a centralized system responsible of computing the group preference model; instead, the users themselves (i.e. the agents that represent the users), coordinated by another agent acting as a host, are responsible of obtaining the group profile by means of a negotiation process. Specifically, the tasks performed by these agents are: (1) building the individual preferences model, (2) reaching an agreement about the group profile and (3) selecting the recommended items for all the group members according to the obtained group profile.

The group profile is built by means of a negotiation process ([15,17]), whose goal is to reach an agreement about the preferences that are included in the group model, that is, an agreement about the preferences that are shared by all the group members and at which degree. If the negotiation process has been successful, these preferences are then used to select the list of recommended items for the whole group, which is then shown to the real users. The negotiation process in our GRS is a variation of the model of **alternating offers** [16] in a multi-party setting. Each agent that represents a real user in the MAS (named *UserAgent*) knows the preferences of this user, but may share only some information with the other agents. In this paper, we introduce an example of UserAgent that can be configured by the real user for exhibit a certain behaviour during the negotiation. This different behaviour can be

implemented as different utility functions, different agreement thresholds or different criteria to accept or reject the proposals from other users. Apart from the agents representing the real users, there is a *host agent* that is in charge of controlling the negotiation process, collecting the user proposals and creating a common proposal to all the users. Moreover, this agent may also participate in the negotiation as a mediator to help the agents to reach an agreement.

The advantages of our solution for the resolution of the group recommendation problem are:

(1) The fact that each user may exhibit a different behaviour is easily captured by the behaviour of each agent, who decides which proposals are accepted or rejected. This results in a more flexible system.
(2) Negotiation is a method that captures well the group dynamics when the agents involved have different behaviours.
(3) Users do not need to communicate all their preferences to the system, so that privacy is preserved (at some extent).

The paper is organized as follows. Section 2 summarizes the state of the art on the use of MAS in recommender systems and the management of the particular behaviour of each user in group recommendation. Section 3 gives an outline of the MAS and the agents that participate in the recommendation process. Section 4 introduces the negotiation framework for obtaining the group profile and all the components (protocol, messages, etc.). Section 5 details the strategy of the host agent and Section 6 gives an example of strategy for a user agent. Section 7 shows an example of a negotiation process, summarizes some experiments performed to test our distributed approach and analyzes how different settings for the agents may affect to the result of the recommendation. We finish with some conclusions in Section 8.

2 Background

The main issue in GRS is to identify the items that are likely to satisfy all of the group members adequately [12,31]. Some GRSs build a group profile that considers the tastes and preferences of the whole group by using aggregation methods to elicit the group profile, associating a weight or degree of interest to each preference in the group profile. There are many remarkable GRS based on the elicitation of preferences; examples include *Intrigue* [1], *Polylens* [29], *MusicFX* [26], *Let's Browse* [18], *The Collaborative Advisory Travel System, CATS* [27] and *GRSK* [9]. A description of these GRSs joint with a classification upon different features can be found in [5,7,9,25].

As explained above, these systems collect the preferences of all the group

members and combine them in order to obtain a recommendation that equally satisfies the group as a whole. This implies that every individual is considered as equal to the others and, therefore, how the members in the group interact with each other is ignored. Besides, it can be very difficult to obtain a single recommendation that satisfies every member for heterogeneous groups. Moreover, the general group satisfaction is not always the aggregation of the satisfaction of its members as different people may have different expectations [32]. In summary, the decision of a group member whether or not to accept a given recommendation can depend not only on his/her evaluation of the content of the recommendation but also on his/her beliefs about the evaluations of the other group members and about their motivation [13].

Recently, some GRSs that consider the different behaviours and attitudes of the group members have appeared. For example, [32] proposes a GRS that takes into account the personality of each group member to weight the influence of his/her ratings during the recommendation process. A *conflict situation* is defined as a situation in which concerns of people appear to be incompatible. In these situations, the behaviour of an individual can be described along two basic dimensions: (1) assertiveness, when the person attempts to satisfy his own concerns, and (2) cooperativeness, when the person attempts to satisfy the concerns of the other person. These basic dimensions of behaviour define five different modes for responding to conflict situations: competing (assertive and uncooperative), collaborating (assertive and cooperative), avoiding (unassertive and uncooperative), accommodating (unassertive and cooperative) and compromising (moderately assertive and cooperative). Taking into account group member interactions, this work proposes three versions of a simple group recommendation algorithm inspired in other classical recommendation algorithms described in the literature, where variations based on the conflict mode behaviour of the group members have been included. Generally speaking, assertive conflict mode behaviours penalize negatively the differences with the best choice of another members, given that the other choices do not satisfy his/her personal concerns. On the other hand, cooperative behaviours reward the difference with the best choice of another members, that is, it is not his/her preference choice but it will be good enough for other members and for the group. This approach was tested in the movie recommendation domain, and the results showed that recommendations could be improved when using the conflict personality values.

Another example of GRS that takes into account the users' personality is described in [24]. This GRS deals with some of the emotions that play a role in the recommendation of sequences of items to a group of users. An accurate prediction of individual satisfaction becomes crucial because to keep the rest of the group happy, an individual might need to be confronted occasionally with items he/she does not like. For example, an accurate prediction of satisfaction would help to ensure no individual gets too dissatisfied, by presenting disliked

items at appropriate times. This paper also considers that the feeling of others in the group may influence an individual's satisfaction. For example, it takes into account emotional contagion, the influence of an individual's affective state on that of others in the group, as well as conformity, whereby judgements are influenced by those of others.

The *Travel Decision Forum* [13] is a prototype GRS which supports users in planning a joint vacation. The goal of the GRS is to provide a group profile. At any moment only *one* group member will be interacting with the web-based system and the absent group members are represented by animated characters. In the first phase of the interaction with the system, each member specifies his/her preferences concerning the vacation and the evaluation criteria of a representative with respect to both the absolute utility of proposals and the relative utility of proposals for different group members. The goal of the group in the second phase is to agree on a joint preference model by means of a negotiation between the current user and the representatives of the absent users. There is also another animated character representing the mediator that moderates the negotiation process for each dimension of the preference model. Then, (1) the mediator computes a proposal based on the specified preferences of all members, (2) the representatives accept or reject the proposal according to a threshold defined by the corresponding real user, (3) the current member responds to the proposal by accepting, adapting or rejecting it or by adapting his/her preferences and the process goes back to step 1 again. The interaction with the system continues until the current member has either agreed with the representatives of the other members on a joint preference model for each value dimension or run out of time or interest. Even in the latter case, the positions of the group members could be closer than at the beginning of the interaction, given that the current member may have adapted his/her preferences or may have relaxed the criteria of his/her representative to accept other proposals. When the next group member interacts with the system, there will be opportunities for further convergence. This system differs in two key aspects from the two previous approaches: any decision about the preference model is taken by the real user and the information about preferences is private to each of them.

This distributed structure of the GRS gives major flexibility to the system in order to consider different user behaviours. In this sense, the GRS presented in [4] relies on the application of cooperative negotiation mechanisms (based on individual recommendations and user preference models) to generate group recommendations. Each group member is represented by a negotiation agent, all with the same behaviour. First, the individual recommendation (list of ranked items) is obtained by means of the case-based recommender system Trip@dvice [34] and then, the negotiation process starts. Specifically, two negotiation protocols are described: alternating offers, for a direct negotiation between two users; and merging ranks (with mediation) for negotiation

among a greater number of users. In the alternating offers protocol, each negotiation participant has the opportunity to place an offer or to accept one of the previously placed offers by the other negotiation participant. In contrast, in the merging ranks protocol, each negotiation participant supplies its complete ranked list to a negotiation mediator that evaluates each proposal and forms a composite ranked list, called *agreement*. To generate an agreement, the mediator may have a "stock" of strategies to help him choosing among these proposals the one that will be offered to the group. The ranked list of products resulting from the agreement formation phase is the resulting group recommendations produced by the overall process. Unlike Travel Decision Forum, the agent representing each user is autonomous for accepting or rejecting the proposals to finally reach an agreement.

As far as we know, these two last systems are the most similar to our approach. In the remaining, we summarize some other recommender systems that use MAS to retrieve, filter and use information relevant to the recommendation, when a single information source does not contain the complete information needed for the recommendation. The majority of the recommender systems based in a multi-agent architecture are aimed at obtaining recommendations for a single user. Some of them are related to the tourism activity. For example, [37] describes a software travel agent that recommends a trip plan for a tourist to Taipei city by conducting the negotiation with human travelers. *BerlinTainment* [38] is a MAS that can be used to plan comprehensive day itineraries for entertainment on Berlin and determine current locations, points of interest, and routes. *CASIS* [22] is another example of a MAS based on a case-based reasoning approach to recommend the best travel to the user. The work in [20,21] describes a multi-agent recommender approach based on the collaboration of multiple agents exchanging information stored in their local knowledge bases with the global objective of recommending the best travel package to the user. Examples of MAS used in other type of applications include: ANEGSYS [19], which is a RS for product acquisition in local markets based on a MAS that uses automatic bilateral negotiations between buyers and sellers; and [23], which is a multi-agent group recommender system that address the problem of arranging meetings for several participants taking into consideration constraints for personal agendas and transportation schedules. Finally, there is a number of systems that use recommendation techniques as a method of making negotiation smarter and more efficient, such as [17,28,8,40].

3 Description of the Multi-Agent System

This section introduces the main characteristics of the MAS [2] , where different types of agents work together to obtain the group recommendation.

The key aspect of the recommendation is the elicitation of the group preference model, which is obtained by means of a **mediated negotiation** process among a set of agents, each acting on behalf of one member of the group. The behaviour of these agents, named *UserAgents*, is determined by the corresponding real user, who establishes the parameters that define the agent's negotiation strategy. The negotiation model is a multi-party negotiation centralizing the communications through a *NegotiatorAgent*. It receives the proposals of the UserAgents, combines them into a single proposal, which is later broadcasted by the NegotiatorAgent and analyzed by the UserAgents. Moreover, the NegotiatorAgent also acts as a mediator, for facilitating the consensus about the group preferences. Along with the UserAgent and the NegotiatorAgent, there are two support agents involved in the recommendation process. The *PreferencesAgent* calculates the preferences of a user given his/her profile and the *ItemsSelectorAgent* selects the list of items to recommend given the negotiated group profile.

The following sections introduce the main characteristics of our GRS. First, we describe the information that it needs for providing a recommendation. Then, Sections 3.2, 3.3 and 3.4 give an outline of the different types of agents that participate in the recommendation. Finally, the interactions between these agents in the recommendation process are summarized in Section 3.5.

3.1 The GRS Data

Our system relies on the use of a **domain ontology** [11] to describe the user's likes and the items to recommend in the particular domain. This GRS is domain-independent, that is, it has been designed to be able to work with any application domain provided that the domain information is specified through a domain ontology. Classes in the ontology represent the *features* (F) that are commonly managed in the corresponding domain. The leaf nodes of the ontology (instances of classes) represent the *items* to recommend. The edges that link an item to a feature are associated to a value that indicates the *degree of interest* of the item under the corresponding feature, i.e., as a member of the category denoted by the feature. The degree of interest of the item i under the feature f (d^{if}) is the degree of suitability of the item to the

[2] These agents are implemented in Java, over the JADE platform (http://jade.tilab.com/).

feature. *Items* are described by means of a set of tuples which represent all the incoming edges of a leaf node. A tuple that describes an item i is of the form (f, d^{if}), where $f \in F$ is a feature defined in the ontology such that there is an edge connecting f and the item i, and $d^{if} \in [0, 100]$ is the degree of interest of the item i within the category represented by the feature f. A more detailed explanation about the domain ontology can be found in [9].

This ontology is also used to describe the **individual and group preference models**. A *preference* is a tuple of the form (f, d^f), where f is a feature in the ontology and $d^f \in [0, 100]$ is the estimated degree of interest of the user u or the group G in the feature f [9,10]. Then:

- The preference model of an individual user u (P^u) is the set of preferences that are preferred by the user u. In this case, the value d^f of a preference represents the estimated degree of interest of the user u in the feature f (denoted by d^{uf}).
- The preference model of a group G (P^G) is the set of preferences that are preferred by the group G. The value d^f of a preference is the estimated degree of interest of the group G in the feature f (denoted by d^{Gf}).

3.2 The NegotiatorAgent

The NegotiatorAgent plays a double role in the negotiation process. First, it acts as a **coordinator** or *host* during the whole process. This implies to be in charge of the following tasks:

(1) **Managing the group of UserAgents**: The NegotiatorAgent receives the UserAgents' requests to participate in the negotiation to build the P^G. These UserAgents are then included in the set G^{UA}, which represents the UserAgents participating in the negotiation process.
(2) **Lauching the negotiation process** among the UserAgents in G^{UA} by sending a message to these UserAgents.
(3) **Centralizing the communications among the UserAgents**, which implies receiving the UserAgents' proposals, combining them into a single proposal, which is later broadcasted to all the UserAgents. Moreover, the NegotiatorAgent checks the proposals to decide if an agreement has been reached.
(4) **Communicating the result of the negotiation process to the UserAgents in** G^{UA}, that is, whether the negotiation process has been successful or not. In case of agreement, the NegotiatorAgent requests the ItemsSelectorAgent to select the items to recommend. The list of recommended items RI^G is finally sent to the UserAgents in the group.

Besides, the NegotiatorAgent acts as a *mediator* when an agreement has not been reached after the UserAgents have proposed all their preferences. How this mediation to ease the consensus is performed will be further detailed in Section 5.

3.3 The UserAgent

Each user u in the group G is represented by a UserAgent in the MAS. The UserAgent records the data of a user u in a **user profile** [30] which is composed by the *recommendation profile* and the *negotiation profile*:

- The **recommendation profile** (see [9] for further details) contains the information related with the preferences about the items to recommend, which is used to compute the individual preference model P^u.
- The **negotiation profile** allows the user to configure some aspects that determine the UserAgent strategy during the negotiation process. Specifically, the user can define the basic behaviour of the UserAgent (for example, self-interested or collaborative), the minimum requirements to agree with a proposal, etc. This will be detailed in Section 6.

All this information is provided by the user when he/she starts using the system. Once the user has entered his data, the negotiation process is independent from the real user. It is the UserAgent who acts on behalf of this user and participates in the negotiation with the other UserAgents. Finally, the real user is informed about the result of the negotiation: the failure in the negotiation or the list of recommended items for the group. Specifically, the UserAgent is in charge of the following tasks (Figure 1):

(1) **Building and updating the user profile**. When a user enters the system, the first step is to enter the data that compose both the recommendation and the negotiation profile. This user profile can be updated at user's request.
(2) **Obtaining the individual preference model** P^u. The UserAgent sends a request to the PreferencesAgent to compute the individual preference model, that is, the list of preferences that characterize the user within the application domain.
(3) **Participating in the negotiation process**. The UserAgent negotiates with the other UserAgents that act on behalf of the other members of the group to elicit the group preference model that better represents the whole group. This negotiation is carried out by taking into account the behaviour defined by the user in his profile.
(4) **Informing the user about the result of the negotiation**. If the negotiation ends successfully, the UserAgent shows the user the list of

recommended items. Otherwise, the UserAgent informs the user about the failure of the negotiation.

3.4 The Support Agents

Besides the main agents in this architecture, the UserAgent and the NegotiatorAgent, there are two agents that provide support to some of their tasks: the PreferencesAgent and the ItemsSelectorAgent.

The *PreferencesAgent* is the agent in charge of computing the individual preference model P^u. When the PreferencesAgent receives a request from a UserAgent, it analyzes the recommendation profile of the corresponding user and elicits the list of preferences that characterize this user within the application domain. How the PreferencesAgent builds this P^u is out of the scope of this paper, but any method for eliciting preferences given a user profile could be used [9].

The *ItemsSelectorAgent* is the agent in charge of selecting the list of items RI^G that satisfy the group's preferences, given the group preference model P^G. This list is a set of tuples of the form $RI^G = \{(i, d^{Gi})\}$, where i is the recommended item, and d^{Gi} is the estimated degree of interest of the group G in the item i. Specifically, once an agreement is reached by the UserAgents in the negotiation process, the NegotiatorAgent sends a request to the ItemsSelectorAgent to compute the list of items that better match the P^G. Further details about the calculation of RI^G can also be found in [9].

3.5 Interaction of the agents in the MAS

The starting point of the **recommendation process** is a group of users that want a recommendation for the group. This process has been implemented by means of a MAS, where different types of agents work together in order to obtain the recommendation more suitable for the group members. Figure 1 shows an sketch of the interaction between these agents. The purpose of these interactions is the following:

(1) **Step 1: Obtain the individual preference model**. This first step is performed by the PreferencesAgent at each UserAgent's request (step 1 in Figure 1). Given the user profile, the PreferencesAgent builds the set of user preferences that will be used to negotiate (P^u).
(2) **Step 2: Obtain the group preference model**. The negotiation process to compute the group profile P^G is controlled by the NegotiatorAgent. All the UserAgents that want to participate in the negotiation process, send a

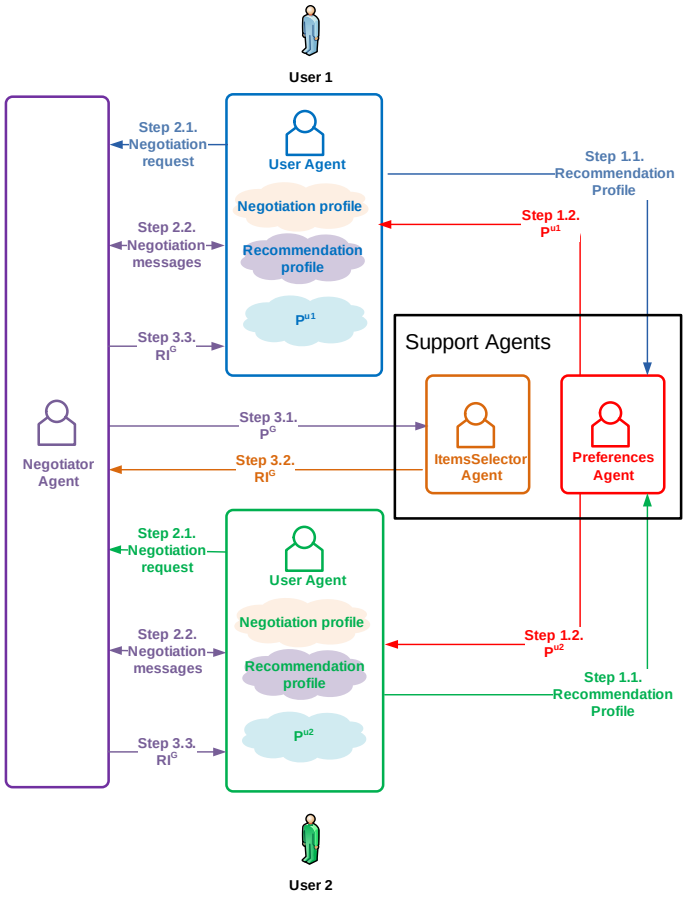

Fig. 1. Steps in the recommendation process.

negotiation request to the NegotiatorAgent (step 2.1 in Figure 1). Then, a flow of messages between the UserAgents and the NegotiatorAgent starts (step 2.2 in Figure 1), that finishes when there is a consensus about the group preference model or when it is not possible to reach an agreement.

(3) **Step 3: Obtain the list of recommended items**. If the UserAgents have reached an agreement during the negotiation process, the group preference model P^G is then used to obtain the list of recommended items to the group. The NegotiatorAgent invokes the ItemsSelectorAgent (step 3.1 in Figure 1). This agent selects the items that better match the list of group preferences and obtains the final list of recommended items RI^G (step 3.2 in Figure 1). The NegotiatorAgent sends this RI^G to the UserAgents involved in the recommendation process (step 3.3 in Figure 1), which in turn send this list to the corresponding users.

4 Negotiation Framework

Agents in MAS share a negotiation mechanism that specifies what possible actions each party can take at any given time, when negotiation terminates, and what the structure of the resulting agreements is [3]. The components of the negotiation framework are the following [15]: **objects, protocols and strategies.** The remaining of this section is devoted to define these components in our MAS.

4.1 The negotiation objects

The negotiation objects are the range of issues over which agreement must be reached, i.e. the **preferences** in the individual preference model P^u of each UserAgent in the group. To participate in the negotiation process, it is mandatory that all the UserAgents have computed the corresponding P^u. Then, the UserAgents negotiate with the preferences in P^u in order to compose an agreed group preference model P^G. If an agreement is reached in the construction of P^G, it will be used to obtain the list of items recommended to the group. If there is no agreement, the negotiation failed and a recommendation cannot be provided.

4.2 The negotiation protocol

The negotiation **protocol** are the set of rules that govern the interaction, types of participants, the negotiation states, the events that cause negotiation states to change and the valid actions of the participants in particular states. The protocol determines the flow of messages between the negotiating parties, dictating who can say what and when and acts as the rules by which the negotiating parties must abide if they are to interact [2]. The protocol must be public and known by all the participants in negotiation.

The definition of negotiation parties in this work follows the model defined by [3], where the roles involved in the negotiation process are *negotiation participant* and *negotiation host*. In our MAS, the role of *participant* is played by the UserAgent and the role of negotiation *host* is played by the NegotiatorAgent. In each negotiation process, there is only one host, whereas there could be an unlimited number of participants.

The first action to be taken is for a participant, i.e. a UserAgent, to require admission to the negotiation. The admission consists of a simple conversation

between the participant and the host, i.e. the NegotiatorAgent, where the participant requests the admission to a particular negotiation process.

Then, the negotiation process itself starts. The negotiation protocol is discrete and synchronous. That is, it is assumed that the agents can take actions in the negotiation only at certain times in the set $\mathcal{T} = \{t_1, t_2, t_3, \ldots\}$ that are determined in advance and known by the agents. These time points can be understood as the **steps** of the negotiation. The interval between these times is δ (i.e. $t_2 = t_1 + \delta$).

We use a generalization of the bilateral **alternating offers protocol** [35] for the multi-party negotiation given that, in our MAS, multiple UserAgents may participate in the negotiation. In this case, the host agent (the NegotiatorAgent) is in charge of supervising the negotiation process, that is, it centralizes the communication among the UserAgents.

4.2.1 The negotiation stages

As explained above, the negotiation process starts when all the UserAgents in the group have requested the NegotiatorAgent to participate in the negotiation. Then, the host sends a message indicating that the negotiation has started. This negotiation process is mainly organized in two sequential stages, as Figure 2 shows:

(1) **Negotiation stage 1 (Proposal Stage)**: Once the negotiation has started, the NegotiatorAgent waits for the UserAgents' proposals until the next negotiation step (that is, it waits during a time interval δ). Then, it analyzes all the responses, builds a combined proposal that is broadcasted to all the UserAgents and, again, it waits for the UserAgents' responses until the next negotiation step. During this interval, the UserAgents evaluate these proposals and accept or reject some proposed preferences. This process is repeated until an agreement is reached or the UserAgents do not have new proposals.

(2) **Negotiation stage 2 (Mediation Stage)**: In case an agreement has not been reached during the proposal stage, the NegotiatorAgent moves to the mediation stage. During this stage, the NegotiatorAgent acts as a *mediator*: it builds proposals that take into account some preferences rejected by the UserAgents during the proposal stage, that have a greater probability to be accepted, in order to facilitate an agreement. This proposal is sent to the UserAgents, which evaluate it and, again, accept or reject some proposed preferences. The UserAgents' responses are analyzed by the NegotiatorAgent, which builds a new proposal. This stage ends when an agreement is reached or when no new proposal can be built. In the latter case, the NegotiatorAgent moves to stage End, that

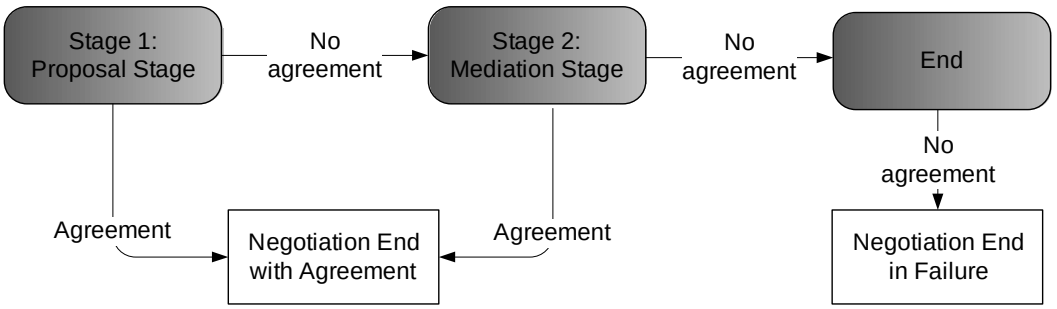

Fig. 2. Negotiation states.

indicates the end of negotiation without agreement [3].

The NegotiatorAgent is in charge of moving from one stage of negotiation to another when certain events occur. Every time it sends a message, the NegotiatorAgent informs the UserAgents about the negotiation stage in which they are. This way, the UserAgent is able to act according to the negotiation stage, if its strategy has been defined to do it.

4.2.2 The negotiation set of rules: the format of messages

All messages follow the specifications of the Agent Communication Language FIPA-ACL [4]. The structure of a message is `(performative : content)`, where `performative` denotes the type of the communicative act of the ACL message, whereas `content` (obviously) denotes the content of the message. It could be a fixed sentence (such as "negotiation end"), but the majority of the messages exchanged between the agents contain **proposals**. Given that the issues under negotiation are the preferences that should be in the group preference model P^G, these proposals comprise two lists of preferences [5]: (1) a list of accepted preferences and (2) a list of proposed preferences. Table 1 shows the types of broadcast messages that the NegotiatorAgent sends to the UserAgents in G^{UA} and Table 2 shows the messages each UserAgent sends to the NegotiatorAgent.

4.3 The decision making model (negotiation strategy)

The decision making model or negotiation strategy are the decision making apparatus the participants employ to act in line with the negotiation protocol in order to achieve their objectives. The sophistication of the model, as well as

[3] We have introduced this additional stage for the sake of clarity of the algorithms that will be detailed below.
[4] http://www.fipa.org/
[5] See Section 3.1. For the sake of simplicity, we call *preference* the feature associated to the preference.

Type	**Perfor.**	**Content**	
Negotiation startup	`cfp`	"send proposal"	The negotiation process has started and the NA is waiting for proposals
Proposal	`propose`	*stage*, $accepted_t$, $proposed_t$	Proposal that combines the UA' proposals previously received by the NA *stage*: stage of the negotiation process (`st1` or `st2`) $accepted_t$: list of preferences accepted by all the UAs in previous iterations of the whole negotiation process $proposed_t$: the list of preferences proposed to the UAs
Negotiation end with agreement	`agree`	P^G, RI^G	P^G: group preferences model RI^G: list of recommended items
Negotiation end in failure	`failure`	"negotiation end"	

Table 1
The NegotiatorAgent messages.

the range of decisions that have to be made, are influenced by the protocol in place, by the nature of the negotiation object, and by the range of operations that can be performed on it [15]. Within a negotiation process, each party adopts a **strategy** which determines exactly which actions they make (in response to actions by other parties or external events) in an effort to maximize their individual or collective gain [3]. Sections 5 and 6 explain the strategies followed by the NegotiatorAgent and the UserAgents, respectively.

5 The NegotiatorAgent strategy

As explained in previous sections, the NegotiatorAgent is the agent that acts as the *host* in the negotiation process. This implies that it is in charge of managing the group of UserAgents, launching and controlling the negotiation process and informing the UserAgents about the result of this negotiation. Moreover, the NegotiatorAgent centralizes the communication among the UserAgents. On the other hand, this agent is also concerned with the fact that the UserAgents reach an agreement. At the beginning of the negotiation (proposal stage), it remains in the background, leaving the UserAgents negotiate on their own. However, if this is not possible, it acts as a *mediator* in the second stage

Type	**Perfor.**	**Content**	
Participate in negotiation	`request`	"negotiation"	The UA wants to participate in negotiation.
Proposal	`propose`	$user - accepted_t^u$, $user - proposed_t^u$	The UA does not agree with the preferences currently accepted by all the UAs and it wants to continue with the negotiation process. $user - accepted_t^u$: the preferences in the NA's proposal that the UA accepts. $user - proposed_t^u$: the preferences that the UA proposes in this iteration.
Proposal and agreement	`agree`	$user - accepted_t^u$, $user - proposed_t^u$	The UA agrees with the list of accepted preferences sent by the NA in the last message.

Table 2
The UserAgent messages.

(mediation stage) in order to facilitate such an agreement.

As the negotiation host, the NegotiatorAgent controls the whole negotiation process. First, it receives the requests of the UserAgents for participating in the negotiation. All the UserAgents are accepted and included in the set G^{UA}. In case that a UserAgent leaves the negotiation on its own or it is expelled (because, for example, it did not answer on time), it is removed from G^{UA} and the NegotiatorAgent does not interact with it anymore. Once all the UserAgents have been registered, the NegotiatorAgent sends a broadcast message to start the negotiation. Then, it waits for the UserAgents' messages. After receiving these messages, it follows these steps: (1) analyzing the UserAgents' responses, (2) preparing and sending a new message and (3) deciding in which stage the negotiation is. After the third step, if the negotiation process continues, the NegotiatorAgent waits again for the UserAgents' messages.

5.1 Step 1. Analysis of the UserAgents' responses

Let's consider the following time points:

- t_0 is the time point when the NegotiatorAgent sent its last message
- t_1 is the current time point
- t_2 is the time point when the next message from the NegotiatorAgent is due

Algorithm 1 The NegotiatorAgent's algorithm.

Step 1. Analyze the UserAgents' responses

Step 1.1. Analyze the UserAgents' message performatives

if $\forall u \in G^{UA}$: `performative`^{u}==`agree`

{Negotiation end with agreement}

$P^G \leftarrow accepted_{t_0}$

Obtain the RI^G from the ItemsSelectorAgent

Send message: `(agree :` P^G`,` RI^G`)`

Return

end if

if *stage* ==`end`

{Negotiation end in failure}

Send message: `(failure : "negotiation end")`

Return

end if

Step 1.2. Analyze the UserAgents' answers to the previous proposal

Compute $accepted_{t_2}$ as equation 1 shows

Update RP with the non-accepted preferences

if *stage* ==`st2`

Update RP (remove accepted or the first N^{remove} preferences)

end if

Step 2. Build the NegotiatorAgent's new proposal

if *stage* ==`st1`

Compute $proposed_{t_2}$ as equation 2 shows

end if

if *stage* ==`st2`

$proposed_{t_2} \leftarrow N^{mediation}$ top preferences in RP

end if

Send message: `(propose :` *stage*`,` $accepted_{t_2}$`,` $proposed_{t_2}$`)`

Step 3. Decide about moving from one stage to another

if *stage* ==`st1` $\wedge$ $proposed_{t_2} == \emptyset$

stage $\leftarrow$ `st2`

end if

if *stage* ==`st2` $\wedge$ $RP == \emptyset$

stage $\leftarrow$ `end`

end if

After the NegotiatorAgent receives the messages from the UserAgents, which have the format `(performative`^{u} `:` $user - accepted^u_{t_1}$`,` $user - proposed^u_{t_1}$`)`, it proceeds as follows:

- **Step 1.1. Analysis of the UserAgents' performatives**. There are two

possibilities:

· *All the UserAgents agree with the previous proposal*: This implies that the negotiation ends with an agreement. Then, the group preference model P^G is composed by the list of accepted preferences in this last proposal, joint with the degree of interest for the whole group for each preference (feature) p, which is computed as [6]:

$$d_p^G = \underset{\forall u \in G^{UA}}{avg} \left(d_p^u \right)$$

Afterwards, the NegotiatorAgent makes a request to the ItemsSelectorAgent, which uses this P^G to obtain the list of recommended items RI^G, which is sent to the UserAgents, and the negotiation process finishes.

· *Not all the UserAgents agree with the previous proposal*: In this case, there is no agreement at this moment and the negotiation process continues with the successive steps, except if the negotiation *stage* is `end`, in whose case, the negotiation process fails because there are not new proposal from the NegotiatorAgent when acting as a mediator.

- **Step 1.2. Analysis of UserAgents' answers**. At this moment, the NegotiatorAgent updates the list of preferences accepted by all the UserAgents as follows:

$$accepted_{t_2} = accepted_{t_0} \cup \left(\bigcap_{\forall u \in G^{UA}} user - accepted_{t_1}^u \right) \quad (1)$$

The remaining tasks in this step are related with the NegotiatorAgent acting as a mediator and will be explained later in this section.

5.2 Step 2. Building the NegotiatorAgent's new proposal

How the NegotiatorAgent builds a new proposal depends on the stage the negotiation process is, which, in turn, determines whether this agent is acting as a host or as a mediator.

5.2.1 The NegotiatorAgent as a host (proposal stage)

The NegotiatorAgent composes a new proposal ($proposed_{t_2}$) as the aggregation of the latest proposed preferences by all the UserAgents:

[6] Recall that this list of preferences only contains the features the UserAgents are interested in, but the NegotiatorAgent also knows the degree of interest associated to these features for each UserAgent.

Management of the list of rejected preferences (RP)	
$N^{mediation}$	Number of preferences proposed in a negotiation step during stage 2.
N^{remove}	Number of not accepted preferences removed from the RP in a negotiation step during the stage 2. It must not be greater than $N^{mediation}$ so that a preference is not removed before being proposed at this stage.

Table 3
The negotiation profile of the NegotiatorAgent.

$$proposed_{t_2} = \bigcup_{\forall u \in G^{UA}} user - proposed^u_{t_1} \quad (2)$$

Even in the case that the new proposal is empty, which indicates that no new proposal was received from the UserAgents, the corresponding message is sent, because there is a new list of accepted preferences ($accepted_{t_2}$) that could make the UserAgents to reach an agreement in the next negotiation step.

5.2.2 The NegotiatorAgent as a mediator (mediation stage)

At stage 2, the NegotiatorAgent acts as a mediator with the aim of helping the UserAgents to reach an agreement. In short, it is in charge of building new proposals. Specifically, it progressively proposes the preferences rejected at stage 1, that is, it proposes first the preferences that it considers that may be more interesting for the group of users, taking into account the experience of the first stage. For doing so, the NegotiatorAgent maintains a **ranked list of rejected preferences** (denoted by RP), where each rejected preference has the following form:

$$(p, d_p^G, users)$$

where p denotes the feature associated to the preference, $users$ is the list of UserAgents that have proposed the preference p at any time and d_p^G denotes the average of the degree of interest of the preference p for the UserAgents in the list $users$.

During stage 1, when a UserAgent u proposes a preference p and it is not accepted by the group, it is registered in RP (see Algorithm 1, Step 1.2): if p is not in RP already, a tuple $(p, d_p^u, \{u\})$ is included in RP; otherwise, the associated values are updated accordingly. As stage 1 continues, a preference p that was previously rejected by the group (and, therefore, included in RP), can be accepted afterwards, in whose case, the corresponding tuple is removed from the set RP. In our current implementation, the list RP is ranked according to the number of UserAgents that have proposed each preference, given that it is easier that the group accepts a preference proposed by many UserAgents. The

preferences with the same number of UserAgents in the list $users$ are ranked according to the degree of interest d_p^G. When the list is ranked, it is balanced according to the UserAgents that proposed each preference, so that no user is benefitted in detriment of the others.

The strategy of the NegotiatorAgent when acting as a mediator (which can be configured in order to adapt its behaviour -see Table 3-), consists in progressively propose the preferences in the list RP until this list is empty or an agreement is reached. Specifically, each proposal during the mediation stage contains the first $N^{mediation}$ preferences. The preferences from the NegotiatorAgent's proposal that are accepted by all the UserAgents are removed from RP. However, if no preference is accepted, then they are not removed and the same $N^{mediation}$ preferences will be proposed in the next message. In order to avoid deadlocks, N^{remove} preferences are removed from RP in case that no preference in the last proposal has been accepted. Therefore, in any case, the number of preferences in the list RP decreases at each iteration of the negotiation process.

The values of $N^{mediation}$ and N^{remove} ensure that all the preferences in RP are eventually proposed to the UserAgents (unless an agreement is reached before) and that a deadlock cannot be encountered. In particular, if $N^{mediation}$ is assigned a high value (w.r.t. the number of preferences in RP), the negotiation is intended to go faster in the sense that more preferences are included in each proposal. However, this does not necessarily imply that an agreement is reached faster because the preferences in a proposal are accepted by unanimity, which is more difficult to achieve if the UserAgents have more preferences to choose. Moreover, if N^{remove} is low (w.r.t. the number of preferences in RP), the negotiation process may take longer but it is more likely that the negotiation is successful, given that the preferences are slowly removed from RP and, consequently, they can be proposed several times, which gives the UserAgents more opportunities of accepting them. In other words, if N^{remove} is high, it may cause that preferences that would be accepted in subsequent iterations, are removed and, therefore, cannot be accepted. Given that the strategy of the UserAgents is not known by the NegotiatorAgent in advance, it is difficult to decide (during the configuration process) the most appropriate values of these parameters in order to ease the agreement. For this reason, as future work, we are interested on studying the inclusion of learning techniques to be able to adapt the strategy of the NegotiatorAgent (the criteria to rank the list RP and the values of $N^{mediation}$ and N^{remove}), depending on how the UserAgents act during the negotiation process.

5.3 Step 3. Deciding about moving from one stage to another

Finally, the NegotiatorAgent considers whether the negotiation must move from the current stage to a different one. The events that define the end of each stage are the following:

- The stage 1 ends when the messages from all the UserAgents do not contain any new preferences. In this case, the new proposal built by the NegotiatorAgent is empty.
- The stage 2 ends when the NegotiatorAgent (acting as a mediator) does not have any new proposal to make.

If the negotiation is at stage End, then the process ends in failure. It is important to remark that the NegotiatorAgent is the only agent that can decide when the negotiation finishes. In other words, the UserAgents can decide to leave the negotiation but they cannot decide when it finishes.

6 The UserAgent strategy

As explained above, this section is devoted to describe the implementation of the UserAgent. Nevertheless, any other implementation could be used, as long as it follows the same negotiation protocol. All in all, our UserAgent can be configured in order to exhibit a behaviour as closer as possible to the real user's behaviour. The goal of each participant, i.e. each UserAgent, in the negotiation process is to obtain an agreed P^G that at least contains a set of preferences enough to satisfy its minimum requirements. Regarding the classification of behaviours introduced in [32], our UserAgent exhibits an *assertive* behaviour, in the sense that it attempts to satisfy the corresponding user's concerns. However, this behaviour may cause that a consensus cannot be reached if the users have heterogeneous tastes. For this reason, the user may prefer the UserAgent to be more collaborative, so that, it may make some concessions in order to reach an agreement. All these characteristics about the behaviour of the UserAgent are defined in the negotiation profile. This profile contains parameters that are used:

(1) To check when a proposal is considered to be satisfactory, namely, the **proposal scoring function** F^u and the **agreement threshold** T^u_{agree}.
(2) To model the progressive negotiation mechanism, namely, the **levels of negotiation** L^u.
(3) To model the UserAgent behaviour during the mediation stage, namely, the **degree of cooperativeness** and the **concession tactics** $N^u_{concession}$.

Proposal Evaluation	
Proposal scoring function F^u	$F^u(preferences)$ defines an utility function to evaluate the proposal
Agreement threshold T^u_{agree}	if $F^u(accepted_t) \geq T^u_{agree}$ then u agrees with $accepted_t$
Progressive Negotiation Mechanism	
Levels of negotiation L^u	$l^u = \{0, 100\} \cup \{l_i \in]0, 100[\ \}$ $L_1 = [l_n, 100], L_2 = [l_{n-1}, l_n[, \ldots, L_N = [0, l_1[$ where $l_n, l_{n-1}, \ldots, l_1 \in l^u$ and $L_1, L_2, \ldots, L_N \in L^u$
Behaviour	
Degree of cooperativeness	**Self-interested**: u will only accept the preferences that are included in its own P^u. **Collaborative**: u considers the preferences proposed by other UAs in order to ease an agreement.
Concession tactics $N^u_{concession}$	Number of preferences proposed by other UAs (and not included in its own CP^u) that it accepts at each iteration of the mediation stage.

Table 4
The negotiation profile of the UserAgent u.

It is important to remark that both the behaviour and the strategy of each UserAgent are private; in other words, the UserAgent strategy and behaviour are unknown for the other UserAgents in G^{UA} and for the NegotiatorAgent.

The UserAgent negotiation strategy is a **progressive negotiation mechanism** both for evaluating the NegotiatorAgent proposal (see section 6.1) and for composing the new proposal (see section 6.2); that is, only those preferences with certain degree of interest in P^u are accepted/proposed at a given moment. The user defines a parameter that governs this process: the **levels of negotiation** L^u (see table 4). The UserAgent accepts/offers the most-valued preferences and, if there is no agreement, it incrementally accepts/offers other preferences with a lower degree of interest. In order to apply this progressive mechanism, the user indicates the UserAgent in which order his/her preferences should be accepted/proposed by means of the definition of the levels of negotiation. The levels of negotiation are a set of intervals of degrees of interest in which the preferences in P^u are organized. This results into the **classified user preferences** CP^u. The number of levels in the CP^u may imply a different performance of the UserAgent. For example, if a user defines only one level, this implies that the UserAgent will accept all the preferences proposed

Algorithm 2 The UserAgent strategy.

Step 1. Analysis of the NegotiatorAgent message

Step 1.1. Analyze the list of preferences accepted by all the UserAgents

if $F^u(accepted_{t_1}) \geq T^u_{agree}$

`performative` $\leftarrow$ `agree`

else

`performative` $\leftarrow$ `propose`

end if

Step 1.2. Analyze the NegotiatorAgent's new proposal

if $stage ==$ `st1`

Compute $user - accepted_{t_2}$ as equation 3 shows

end if

if $stage ==$ `st2`

$user - accepted_{t_2} \leftarrow N^u_{concession}$ top preferences in $proposed_{t_1}$

end if

Step 2. Build the UserAgent's new proposal $user - proposed_{t_2}$

if $stage ==$ `st1`

Compute $user - proposed_{t_2}$ as equation 4 shows

Update $L_{current}$ to the next non-empty level in CP^u

end if

if $stage ==$ `st2`

$user - proposed_{t_2} \leftarrow \emptyset$

end if

Send message: (`performative` : $user - accepted_{t_2}$, $user - proposed_{t_2}$)

by other UserAgents that are in its own P^u, which would lead to a quicker agreement. However, if many levels are defined, this UserAgent will need more iterations to reach an agreement. On the other hand, this mechanism allows the UserAgent partially preserve the privacy of its preferences, given that it is not necessary communicating all the preferences in P^u at the beginning of the negotiation. On the contrary, it is possible that an agreement is reached before proposing all the preferences of a UserAgent.

Algorithm 2 shows the UserAgent strategy, detailing how it processes the NegotiatorAgent's message and how it builds the new proposal. Recall that the NegotiatorAgent's message is of the form (`performative` : $stage$, $accepted_{t_1}$, $proposed_{t_1}$) and that the new message of the UserAgent will have the format (`propose/agree` : $user - accepted_{t_2}$, $user - proposed_{t_2}$), where $t_2 = t_1 + \delta$.

Initially, when the UserAgent requests to participate in the negotiation and it is accepted, it sets $L_{current}$ to L_1 or to the first non-empty level of negotiation. Then, at each iteration, it waits for the next message from the NegotiatorAgent. The strategy of the UserAgent when this message arrives can be decomposed into two steps: (1) the analysis of the NegotiatorAgent's message

and (2) the construction of the new proposal of the UserAgent.

6.1 Step 1. Analysis of the NegotiatorAgent message

The analysis of the NegotiatorAgent's message gives as a result the performative of the new message (`agree` or `proposal`) and the list $user - accepted^u_{t_2}$. This process is performed in two steps:

- **Step 1.1. Analyze the list of preferences accepted by all the UserAgents**. By means the scoring function [7] $F^u(accepted_{t_1})$, the UserAgent decides whether the preferences accepted by all the UserAgents in G^{UA} in previous iterations of the negotiation process (contained in the list $accepted_{t_1}$), are good enough to satisfy its requirements (T^u_{agree}).
- **Step 1.2. Analyze the NegotiatorAgent's new proposal**. In this step, the UserAgent evaluates the new proposal of the NegotiatorAgent ($proposed_{t_1}$) and it computes the list $user - accepted^u_{t_2}$. This process depends on the negotiation stage.
  - **Proposal stage** ($stage$ == `st1`): The UserAgent only accepts the preferences included in its own P^u. This means that a preference p included in $proposed_{t_1}$ is accepted by the UserAgent if it belongs to CP^u in a level higher or equal to its current level of negotiation. This way, only those preferences already or about to be proposed by the UserAgent are accepted and, therefore, the idea of negotiating first with the most-valued preferences is maintained when analyzing a received proposal and deciding which preferences should be accepted. More formally:

$$user - accepted^u_{t_2} = \{p/p \in proposed_{t_1} : level(p) \geq L_{current}\} \qquad (3)$$

    where $level(p)$ denotes the level of preference p.
  - **Mediation stage** ($stage$ == `st2`): During stage 2, the UserAgent will act according to the behaviour defined by the corresponding user. That is, if the user has configured the UserAgent to be self-interested, only those preferences proposed by the NegotiatorAgent that belong to the P^u are accepted (this is equivalent to $N^u_{concession} = 0$). Otherwise, the UserAgent will accept some preferences proposed by other UserAgents during stage 1. Specifically, it will accept (at each iteration) $N^u_{concession}$ preferences in $proposed_{t_1}$. Therefore, a higher value of $N^u_{concession}$ determines a higher level of cooperativeness. Given that the UserAgent knows that the preferences in the mediator's proposal are ordered according to their degree of

[7] Note that the list used to check whether the requirements of the UserAgent are satisfied is the list $accepted_{t_1}$, instead of the new proposal of the NegotiatorAgent, which has not already been consensued by the UserAgents.

$l^u = \{0, 50, 75, 100\}$	
L^u	CP^u
$L_1 = [75, 100]$	(Sport,75),(Beach,80),(Open Spaces,77)
$L_2 = [50, 75[$	(Architecture,50)
$L_3 = [0, 50[$	(Museums,20)

Table 5
Example of levels of negotiation in the Tourism domain (Movielens dataset).

interest for the whole group, this UserAgent is implemented [8] for accepting the first $N^u_{concession}$ preferences in $proposed_{t_1}$.

6.2 Step 2. Building the UserAgent new proposal

The second part of this algorithm consists in **building the UserAgent's new proposal**. The preferences in P^u are organized in levels, according to the levels of negotiation defined by the user in the negotiation profile, which result in the **classified user preferences** CP^u (see table 5). Then, the first proposal of the UserAgent only contains preferences from the highest level of negotiation L_1. As the negotiation progresses, the proposals will include preferences from lower levels. In general, the new proposal will contain the preferences in the non-empty highest level that has not been yet proposed, except those preferences already accepted by all the UserAgents. Considering that t_1 represents the current time point (when the NegotiatorAgent's message has been received) and t_2 is the time point when the next message from the UserAgent is due, the new proposal is built as follows:

$$user - proposed^u_{t_2} = \{p/(p, d^{up}) \in CP^u(L_{current})\} - accepted_{t_1} \qquad (4)$$

where $CP^u(L_{current})$ denotes the preferences in the current level of negotiation and $accepted_{t_1}$ is the list of accepted preferences in the last message from the NegotiatorAgent. Initially, $L_{current} = L_1$ (or to the first non-empty level of negotiation) and, after sending the proposal, it is updated to the next non-empty level of preferences in CP^u. That is, $user - proposed^u_{t_2}$ are the preferences in the current level of negotiation in the CP^u that have not be previously accepted by all the UserAgents.

Note that, although the UserAgent agrees with the accepted preferences, it builds a new proposal. The reason behind is that the other UserAgents could

[8] However, if more complex behaviours are defined, for example, to favor a certain user during the negotiation, this may not be valid.

	Proposal Stage		**Mediation Stage**	
	Negotiator-Agent	UserAgent	Negotiator-Agent	UserAgent
New Proposal	Joins the UAs' new proposals	Uses the CP^u	Uses the RP	–
Accepted Preferences	Pref. accepted by all the UAs	Pref. in CP^u at $L_{current}$	Pref. accepted by all the UAs	Self-interested: pref. in P^u Collaborative: pref. in P^u and $N^u_{concession} \notin P^u$
Agree	All UAs agree	$F^u(accepted_t) \geq T^u_{agree}$	All UAs agree	$F^u(accepted_t) \geq T^u_{agree}$
Negotiation End Agreement	All UAs agree	–	All UAs agree	–
Negotiation End Failure	–	–	$RP = \emptyset$	–

Table 6
Summary of the behavior of the NegotiatorAgent and the UserAgent (UA).

reject the current proposal and, in this case, a new proposal is necessary to continue the negotiation process. If there are no preferences to propose or the negotiation is at stage 2, then the list $user - proposed^u_{t_2}$ is empty.

Finally, the message that will be sent to the NegotiatorAgent is:

`(agree/propose :` $user - accepted^u_{t_2}$`,` $user - proposed^u_{t_2}$`).`

This progressive mechanism allows the UserAgent partially preserve the privacy of its preferences, given that it is not necessary communicating all the preferences in P^u at the beginning of the negotiation. On the contrary, it is possible that an agreement is reached before proposing all the preferences of a UserAgent. An additional advantage of this mechanism is that the user can decide the initial preferences to be used in the negotiation process; then, if no agreement is reached, other preferences can be considered. Currently, this is an static mechanism, but the set of preferences could be dynamically defined.

7 Results

This section first shows a simple example, where this negotiation protocol is applied (see Table 6 for a summary). Then, we detail some experiments we have performed with the aim of testing our negotiation process with heterogenous groups. Finally, we analyze some aspects of this protocol by considering different values of the UserAgent and NegotiatorAgent parameters.

	UserAgent u_1	UserAgent u_2	UserAgent u_3
Degree of cooperativeness	Collaborative	Collaborative	Collaborative
F^u	$F^u(acc) = \left(100 - \frac{\sum_{f \in acc} \lvert d^{Gf} - d^{uf} \rvert}{\lvert acc \rvert}\right) * min\left(1, \frac{\lvert acc \rvert}{\lvert P^u \rvert} + \alpha\right)$		
T^u_{agree}	0.75	0.7	0.5
$N^u_{concession}$	1	1	1
Levels of negotiation	CP^u: (1,75) (2,80) (3,77); (4,50) (5,60); (6,20)	CP^u: (6,90) (3,90) (7,80); (1,50) (2,50); (8,30)	CP^u: (9,70) (5,90); (3,50) (2,50) (6,60)

Table 7
Negotiation profile of three UserAgents.

7.1 Case of Study

This case shows an example of the full negotiation process with three UserAgents, which have been implemented as explained in Section 6. The negotiation profile of these UserAgents is summarized in Table 7. The utility function we have chosen is aimed at obtaining a P^G that contains a significant subset of preferences in the corresponding user P^u with a degree of interest (d^{Gf}) closer to the user degree of interest. α is a correction factor to give more or less weight to the number of preferences in the P^u that must be included in the P^G. The parameters of the NegotiatorAgent are $N^{mediation} = 3$ and $N^{remove} = 1$. This example allows us to show the flow of the agents messages, the change from negotiation stage 1 to negotiation stage 2 and the strategy of the agents. Each step in this process matches with a time interval.

Figure 3 shows the flow of messages that both the NegotiatorAgent and the UserAgents exchange during the negotiation process. At each step, the NegotiatorAgent sends a message to all the UserAgents or viceversa. Note that actually the message from the UserAgent contains both the feature and the degree of interest of each preference, but in Figure 3 we only show the feature index for the sake of clarity. We are not going to explain this example in detail; we will remark only the aspects that are not completely clear in the figure and that it is important to emphasize.

When the NegotiatorAgent receives the UserAgents' messages, first it decides whether there is agreement or not. In case of no agreement (for example,

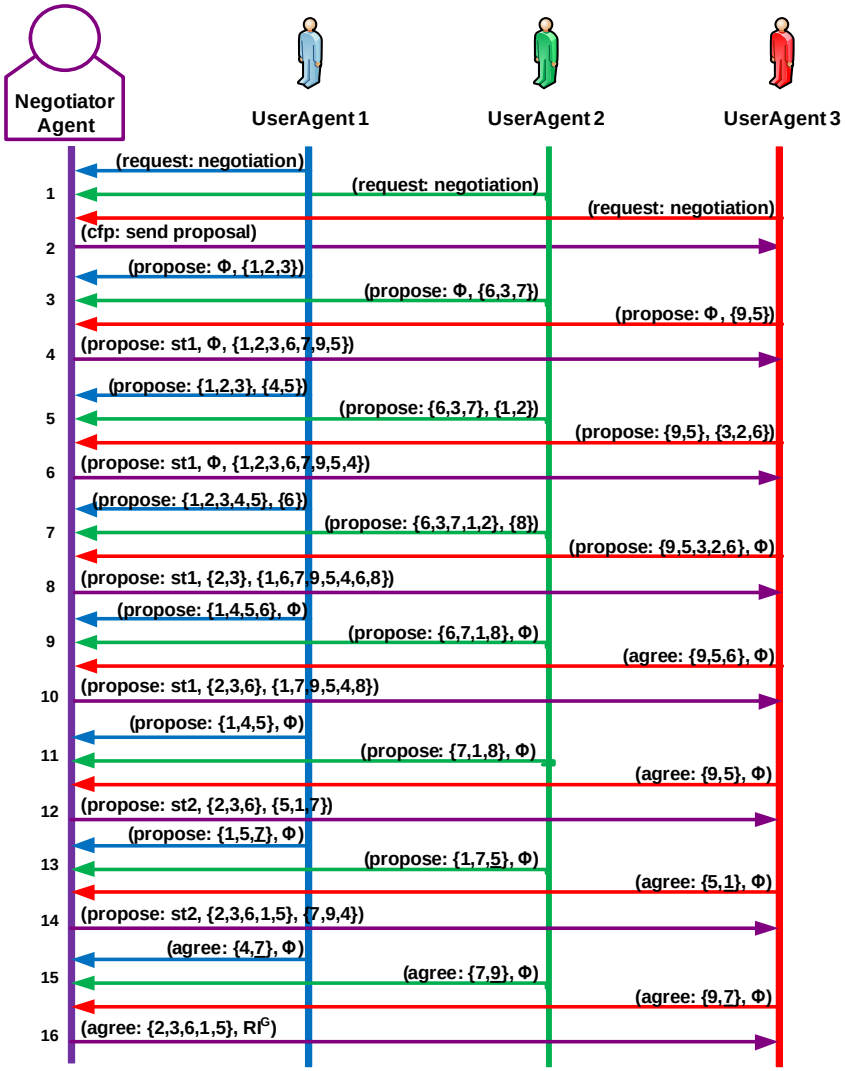

Fig. 3. Example of a negotiation process

at step 8), the list of accepted preferences is updated in preferences 2 and 3, because they have been accepted by all the UserAgents. The remaining preferences proposed by the NegotiatorAgent are added to the list of rejected preferences RP. For example, the tuple $(1, (75 + 50)/2, \{u_1, u_2\})$ is added to RP. Finally, the corresponding message is sent.

When the UserAgents receive a proposal, the list of accepted preferences is evaluated with the corresponding scoring function to determine if they agree or not. For example, the application of the scoring function at step 9 results in [9]:

$$F^{u_1}(\{2,3\}) = \Big(100 - (|60-80| + |72-77|)/2\Big) * min\Big(1, 2/6 + 0.3\Big) = 0.55 <$$

[9] α is set to 0.3 and the degree of interest of the group (d^{Gf}) is the average of the corresponding d^{uf}.

Preference	Degree of interest	Users
5	75	u_1,u_3
1	62.5	u_1,u_2
7	80	u_2
9	70	u_3
4	50	u_1
8	30	u_2

Table 8
Ranked list of rejected preferences RP.

$0.75 \rightarrow no - agree$

$F^{u_2}(\{2,3\}) = \Big(100 - (|60-50| + |72-90|)/2\Big) * min\Big(1, 2/6 + 0.3\Big) = 0.54 < 0.7 \rightarrow no - agree$

$F^{u_3}(\{2,3\}) = \Big(100 - (|60-50| + |72-50|)/2\Big) * min\Big(1, 2/5 + 0.3\Big) = 0.59 < 0.5 \rightarrow agree$

At step 10, the NegotiatorAgent determines that there is not agreement with the current proposal and adds the preference 6 to the list of accepted preferences by all the UserAgents. It sends this information, also indicating that the negotiation is still in stage 1, so that the UserAgents evaluate this new proposal. However, the negotiation (internally) moves to stage 2 because the proposals of all the UserAgents were empty.

Given that there is not agreement with the current proposal, the negotiation process continues at stage 2, where the NegotiatorAgent builds a proposal that contains the first $N^{mediation}$ (i.e. 3) preferences in the list of rejected preferences RP (see Table 8).

As the negotiation is at stage 2, each UserAgent acts according its own behaviour. In this case, all of them are collaborative and, therefore, some preferences not included in its own CP^u will be accepted at each negotiation round [10] , specifically, the number of preferences defined by $N^u_{concession}$. For example, the UserAgent u_3 accepts preference 5 because it belongs to CP^{u_3}; given that $N^{u_3}_{concession} = 1$ and there are two remaining preferences, it chooses the first one in the list, because the NegotiatorAgent proposes the preferences in order of interest for the group.

[10] The preferences that a UserAgent accepts and are not included in its CP^u are underlined in Figure 3.

At step 15, the UserAgents evaluate the new list of accepted preferences and in this case, all UserAgents agree with the proposal. Finally, at step 16, the NegotiatorAgent determines that an agreement has been reached. The final list of accepted preferences is $P^G = \{(2, 60), (3, 72), (6, 57), (1, 42), (5, 50)\}$, which is used to obtain the list of recommended items RI^G.

7.2 Experimental Results

Now, we summarize some experiments that we have carried out to test the performance of our negotiation mechanism when dealing with heterogeneous groups. Specifically, we have used a Tourism data set, which was developed in our research group[36]. It contains information about leisure and tourist activities in the city of Valencia (Spain). The ontology comprises 115 features structured in two levels. Information about the users was collected from 58 real users, who directly filled out a questionnaire through a web service. Specifically, each user gave personal details and rated the 15 features of the first level in the ontology. Moreover, in order to have data to compare the results obtained by the RS, all the users were requested to rate every item in the data set through a form that was independent from the web site. If the users had not visited the place, the rate indicated whether or not they would be interested in visiting it. In this way, we have complete feedback information about the users.

We built 100 groups of different size ranging from 2 to 6 randomly selected users (20 groups for each group size). Then, we added the negotiation profile (see Table 4) to the users in these groups. Namely, for each group, we built variants with all the possible combinations of self-interested/collaborative users and two T_{agree} levels: 50 and 75. This resulted in more that 100000 groups for testing the negotiation procedure. Some other aspects of the negotiation profile remained unchanged: $N^u_{concession}$ is set to 2 and the l_u levels are 0, 50, 70, 90 and 100 for all the users. We used the same utility function introduced in the previous section.

In order to make an initial comparison between the results obtained with our negotiation process, we selected four variants for each group that result from the combination of (1) all the users are collaborative/self-interested and (2) with a T_{agree} of 50/75. These results are shown in Figure 4. In particular, these plots show the minimum F^u value of all the users in each group in the four situations described above. The vertical lines separate the groups of different size (the first 20 groups have 2 users, the next 20 groups have 3 users, and so on). The horizontal line indicate the threshold for reaching an agreement (that is, the T_{agree} value). This implies that only if the minimum F^u value is above this threshold, the group has reached an agreement with

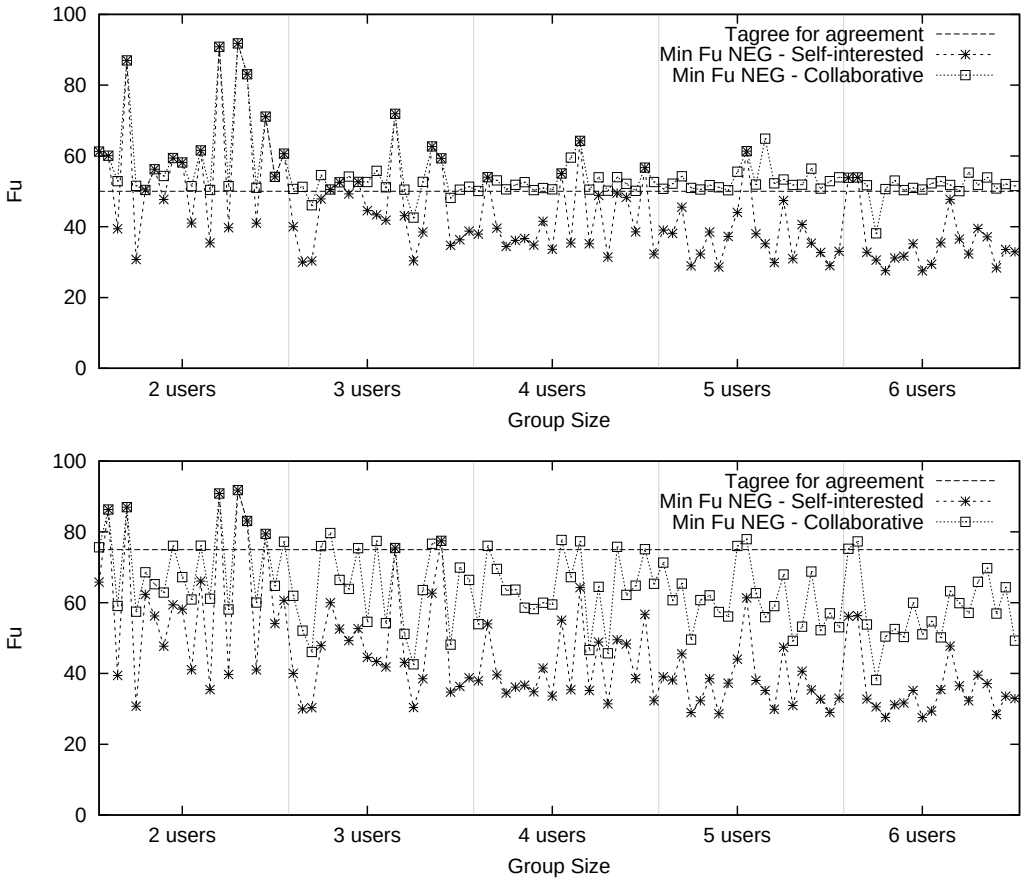

Fig. 4. Comparison of the minimum utility for each group self-interested and collaborative users and a T_{agree} of 50 (above) and 75 (below).

the corresponding P^G. Therefore, in Fig. 4 (above) we can see that, when all the users are collaborative, the negotiation procedure is able to reach an agreement in most of the groups, because, during the collaborative stage, users accept preferences that are not present in their own P^u, which allows the other users to increase their F^u. Obviously, when the T_{agree} is 75, it is much more difficult to reach such an agreement and this is reflected in the results. It is interesting to see how the minimum F^u value in Fig. 4 (above) is closer to the threshold line. This is due to the fact that the negotiation process finishes when all the users obtain a F^u value above their own T_{agree} (which in this case is 50). Moreover, this can also be observed in some groups when all the users are self-interested. For example, the second group in Fig. 4 (above) has a F^u value of 60 (approx.), whereas in Fig. 4 (below), this value is higher than 80, which means that the group reaches an agreement in both cases.

Plots in Fig. 5 show a comparison between the minimum and the maximum F^u obtained in each group, that is, the difference in utility for the least and the most satisfied users in the group, only when all the users are collaborative. It can be observed that, when the T_{agree} is higher (namely, 75), the maximum

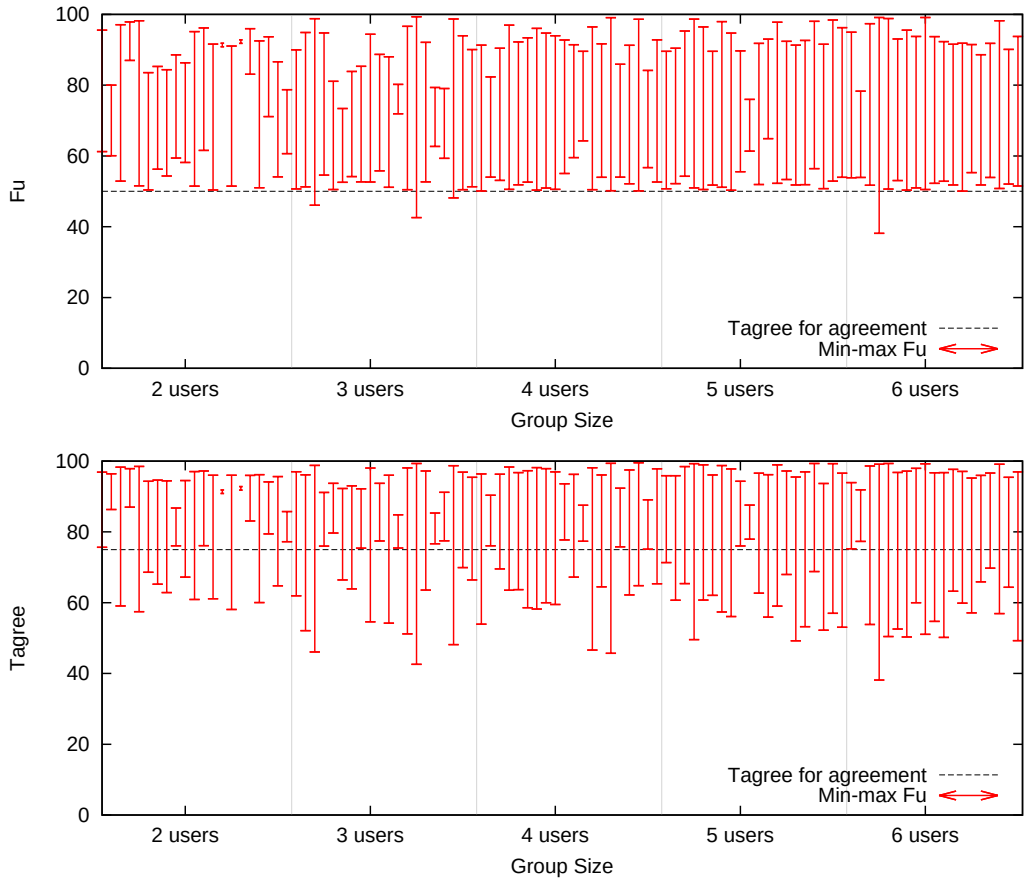

Fig. 5. Range of utility for each group with a T_{agree} of 50 (above) and 75 (below) with collaborative users.

utility is also higher than in the other situation (with a T_{agree} of 50), because, in order to reach an agreement, the P^G must be more satisfactory for all the users in the group.

Plots in Fig. 6 show the percentage of agreements reached in heterogeneous groups of 3 and 5 users [11], that is, with all the possible combinations of behaviours. These combinations are represented in the X and Y axis. The X axis indicates the T_{agree} value in average of the group members; the Y axis indicates the degree of collaboration, calculated as the number of collaborative users with respect to the total number of users in the group. As the previous figures show, when all the users are collaborative (degree of collaboration equal to 1), almost the 100% of agreements are reached, when the T_{agree} value is 50 (top-left circle). Obviously, as the degree of collaboration decreases and the T_{agree} value and the number of users in the group increase, it is more difficult to reach such an agreement.

[11] We only show these figures because they are representative of what happens with groups of other sizes.

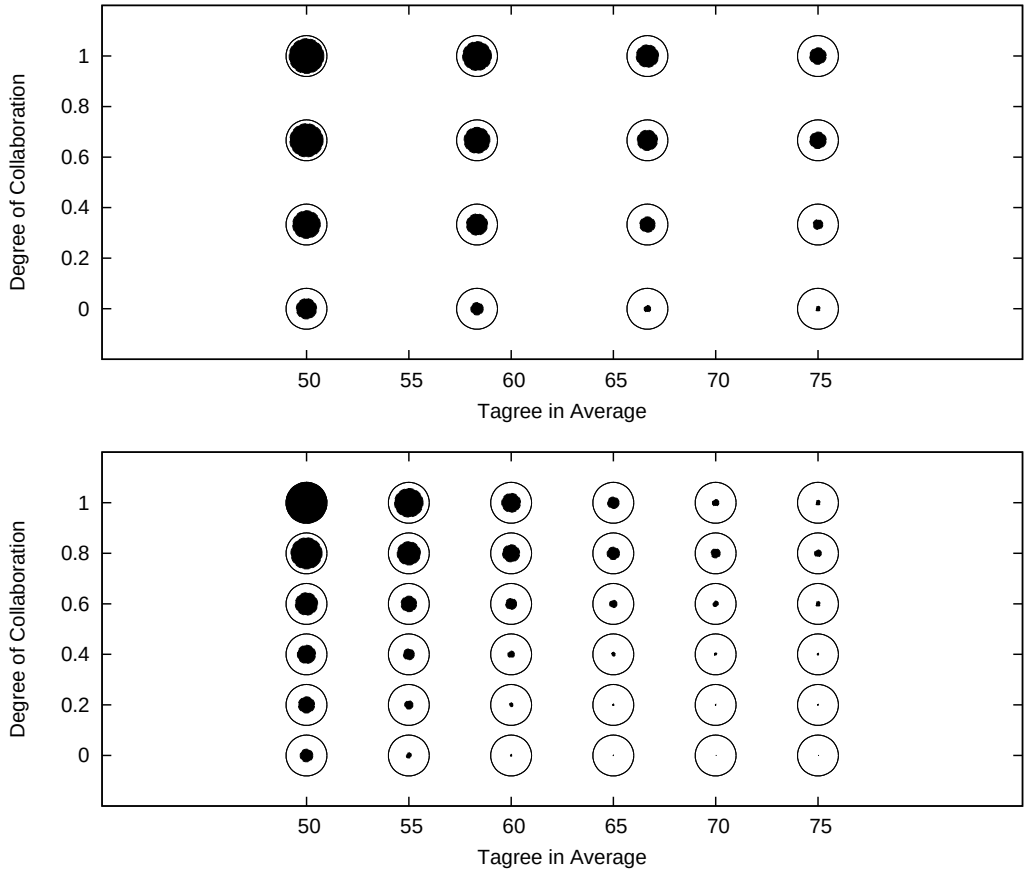

Fig. 6. Percentage of agreements reached in heterogenous groups of 3 (above) and 5 (below) users.

Regarding the computational cost of the negotiation process, it took only few milliseconds to perform it (we did not consider the communication cost). It is only slightly more costly than a centralized mechanism. We also observed that the number of iterations needed in the negotiation process depended on the following aspects:

- If all the users are self-interested and an agreement is reached, the maximum number of iterations is usually given by the number of levels of negotiation in the negotiation profile (4 in our experiments). Even when the sets of preferences P^u of all the users are similar, it is difficult that the distribution in the CP^u is exactly the same.
- If there is at least one collaborative user and an agreement is reached during the mediation stage, the number of iterations is 7, in average.
- If an agreement is not reached, usually 20 iterations are needed in average, although when the P^u of some users contains many preferences, it could be necessary to perform up to 26 iterations, given that, during the mediation stage, the rejected preferences are proposed progressively.

In summary, in the success of the negotiation process, two main aspects have

an influence:

- Users' behaviour: an agreement is easier to reach when the number of collaborative users in the group is high and their T_{agree} is low.
- Groups homogeneity: when the P^u of the group members have a high number of preferences in common, it is easier to reach an agreement.

On the other hand, we have observed that the number of preferences in the P^u of the group members may also have an influence in this success. In fact, it seems more difficult to reach an agreement when there is a user whose P^u size is significantly different from the other users. For this reason, one of our current works is focused on analyzing which internal characteristics of the groups ease or make it more difficult to reach an agreement.

7.3 Discussion

In this section, we analyze some aspects of the negotiation process. During the proposal stage (stage 1), the preferences that are accepted are a subset of those that belong to the intersection of the P^u of all the users. This could lead us to think that this process is quite inefficient. However, recall that we are facing a group recommendation problem where the users do not want to reveal all their preferences to the system, if it is not necessary. That is, it could be the case that after few negotiation rounds an agreement is reached and, therefore, the UserAgents have not communicated all their preferences, unlike most centralized systems that need all the preferences from the very beginning of the recommendation process. Moreover, in our GRS, the UserAgent can be configured to propose only some preferences or to propose them in a slower or faster manner (depending on the number of negotiation levels).

The order in which the preferences are accepted depends on the degree of interest d^{uf} in each P^u. That is, preferences with higher d^{uf} for all the UserAgents are accepted sooner. This has an influence in the final P^G in case an agreement is reached during stage 1, because the P^G will only contain the preferences most valued by all the users.

If an agreement is not reached during stage 1, the strategy of the NegotiatorAgent when acting as a mediator, has a limited influence in the result of the negotiation, because it mainly depends on the behaviour of the agents. However, as the behaviour of the UserAgents is private, nobody knows what it is going to happen. In case all the UserAgents have been configured to be self-interested, stage 2 will not have any effect to reach an agreement, given that no concessions will be made at this stage. On the contrary, if some UserAgents are collaborative, an agreement can be reached during stage 2, depending on the value of $N^u_{concession}$ (which indicates the degree of cooperativeness of the

agent) and the heterogeneity of the group.

8 Conclusions and further work

Group Recommendation can be defined as the problem of the selection, among a set of items, of those that are likely of interest to the group of users. There are different aspects that define this problem. We are interested in solving the group recommendation problem for a group of users where each user has his/her own preferences and expectations about the resulting group profile, may want to impose their preferences over the other users' preferences or may like better to agree with the others' proposals. Besides, both the preferences and the particular behaviour are private information.

In this context, we have opted for building a MAS, where some agents act on behalf of the group members. This way, each user can define a different behaviour for the agent, which results in a more realistic social model. Moreover, both preferences and behaviour are private information that can be revealed under certain circumnstances. We have implemented a UserAgent that can be configured with different parameters to model the behaviour that the user wants the agent to exhibit. For example, degree of cooperation, concession tactics, etc., determine the reasoning process to decide whether to accept or reject a proposal and to compose a new offer. These UserAgents work together (coordinated by the NegotiatorAgent) with the aim of building a group profile that satisfies the particular requeriments of each group member. This group profile is then used to obtain the list of recommended items.

The negotiation process to obtain the group profile is a multi-party variation of the alternating-offers protocol, where the NegotiatorAgent centralizes the communication among the UserAgents. The negotiation comprises two stages: during the first stage, the UserAgents negotiate on their own to try to reach an agreement; however, if this is not possible, the NegotiatorAgents acts as a mediator during the second stage in order to facilitate such an agreement.

The advantages of our solution for the resolution of the group recommendation problem are:

(1) Each UserAgent may exhibit a different behaviour, which is easily configured by setting some parameters.
(2) UserAgents do not need to communicate all their preferences to the system, because an agreement can be reached before. Moreover, they do not reveal what their behaviour is with respect to the other UserAgents. Therefore, privacy is preserved (at some extent).
(3) The negotiation protocol we propose captures well the group dynamics

when the agents involved have different behaviours and allows them to obtain a group profile according to the preferences and expectations of all the group members.

Finally, we have performed some experiments that show that this mechanism is able to give a response in this heterogeneous environment.

As for further work, we are working on defining new, more complex behaviours for the UserAgent and analyzing how these behaviours affect the agreement. Additionally, we are working on incorporating techniques based on trust during the negotiation process, so that proposals are accepted or rejected depending on the UserAgent that proposed them. This way, related users are favored to the detriment of others.

References

[1] L. Ardissono, A. Goy, G. Petrone, P. Torasso, Intrigue: personalized recommendation of tourist attractions for desktop and handset devices, Applied AI, Special Issue on Artificial Intelligence for Cultural Heritage and Digital Libraries 17 (8) (2003) 687–714.

[2] C. Bartolini, C. Preist, N. Jennings, A generic software framework for automated negotiation, in: Proceedings of the AAMAS'02, 2002.

[3] C. Bartolini, C. Preist, N. Jennings, A software framework for automated negotiation, in: Software Engineering for Multi-Agent Systems III, vol. 3390 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2005, pp. 213–235.

[4] P. Bekkerman, K. Sarit, F. Ricci, Applying cooperative negotiation methodology to group recommendation problem, in: ECAI Workshop on Recommender systems, 2006.

[5] L. Boratto, S. Carta, State-of-the-art in group recommendation and new approaches for automatic identification of groups, in: A. G. Soro A., Vargiu E., P. G. (eds.), Information Retrieval and Mining in Distributed Environments, vol. 324 of Studies in Computational Intelligence, Springer Berlin / Heidelberg, 2011, pp. 1–20.

[6] Y.-L. Chen, L.-C. Cheng, C.-N. Chuang, A group recommendation system with consideration of interactions among group members, Expert Systems with Applications 34 (3) (2008) 2082–2090.

[7] L. M. de Campos, J. M. Fernandez-Luna, J. F. Huete, M. A. Rueda-Morales, Managing uncertainty in group recommending processes, User Modeling and User-Adapted Interaction 19 (3) (2009) 207–242.

[8] J. de la Rosa, N. Hormazbal, S. Aciar, G. Lopardo, A. Trias, M. Montaner, A negotiation-style recommender based on computational ecology in open negotiation environments, IEEE Transactions on Industrial Electronics 58 (6) (2011) 2073 – 2085.

[9] I. Garcia, S. Pajares, L. Sebastia, E. Onaindia, Preference elicitation techniques for group recommender systems, Information Sciences 189 (2012) 155–175.

[10] I. Garcia, L. Sebastia, E. Onaindia, On the design of individual and group recommender systems for tourism, Expert Systems with Applications 38 (6) (2011) 7683–7692.

[11] T. R. Gruber, A translation approach to portable ontology specifications, Knowledge Acquisition 5 (2) (1993) 199–220.

[12] A. Jameson, More than the sum of its members: Challenges for group recommender systems, in: Proceedings of the International Working Conference on Advanced Visual Interfaces. Gallipoli, Italy, New York, NY, USA, 2004.

[13] A. Jameson, S. Baldes, K. Thomas, Enhancing mutual awareness in group recommender systems, in: B. Mobasher, E. Sarabjot S. Anand (eds.), Proceedings of the International Joint Conference on Artificial Intelligence. Workshop on Intelligent Techniques for Web Personalization. Acapulco, Mexico, AAAI, 2003.

[14] A. Jameson, S. Baldes, K. Thomas, Two methods for enhancing mutual awareness in a group recommender system, in: Proceedings of the International Working Conference on Advanced Visual Interfaces. Gallipoli, Italy, New York, NY, USA, 2004.

[15] N. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, M. Wooldridge, C. Sierra, Automated negotiation: Prospects, methods and challenges, Group Decision and Negotiation 10 (2) (2001) 199–215.

[16] S. Kraus, Automated negotiation and decision making in multiagent environments, in: ACAI, LNAI 2086, 2001.

[17] M. Lenar, J. Sobecki, Using recommendation to improve negotiations in agent-based systems, Journal of Universal Computer Science 13 (2) (2007) 267–286.

[18] H. Lieberman, N. W. Van Dyke, A. Vivacqua, Let's browse: A collaborative browsing agent, Elsevier Science B. V 12 (1999) 427–431.

[19] M. A. Lopez-Carmona, I. Marsa-Maestre, B. Alarcos, Anegsys: An automated negotiation based recommender system for local e-marketplaces, IEEE Latin America Transactions 5 (6) (2007) 409–416.

[20] F. Lorenzi, A. Bazzan, M. Abel, An architecture for a multiagent recommender system in travel recommendation scenarios, in: A. Felfernig, M. Zanker (eds.), Proceedings of 3rd Workshop on Model-Based Systems (ECAI 2006). Riva del Garda, Italy, ECAI, 2006.

[21] F. Lorenzi, A. Bazzan, M. Abel, Truth maintenance task negotiation in multiagent recommender system for tourism, in: Proceedings of the AAAI Workshop on Intelligent Techniques for Web Personalization and Recommender Systems in E-Commerce (AAAI 2007), 2007.

[22] F. Lorenzi, D. S. dos Santos, A. Bazzan, Negotiation for task allocation among agents in case-base recommender systems: a swarm-intelligence approach., in: E. Aimeur (ed.), Proceedings of the Workshop Multi-Agent Information Retrieval and Recommender Systems - Nineteenth International Conference on Artificial Intelligence (IJCAI 2005). Edimburgh, Scotland, 2005.

[23] S. Macho-Gonzalez, M. Torrens, B. Faltings, A multi-agent recommender system for planning meetings, in: Fourth International Conference on Autonomous Agents, Workshop on Agent-based Recommender Systems (WARS 2000), 2000.

[24] J. Masthoff, The pursuit of satisfaction: Affective state in group recommender systems, User Modeling (2005) 297–306.

[25] J. Masthoff, Recommender Systems Handbook, chap. Group Recommender Systems: Combining Individual Models, Springer, 2011, pp. 677–702.

[26] J. F. McCarthy, T. D. Anagnost, Musicfx: An arbiter of group preferences for computer supported collaborative workouts, in: Proceedings of the ACM 1998 Conference on CSCW, 1998.

[27] K. McCarthy, L. McGinty, B. Smyth, M. Salamo, Social interaction in the cats group recommender, in: Workshop on the Social Navigation and Community based Adaptation Technologies. Dublin, Ireland, 2006.

[28] A. A. Niknafs, H. B. Band, Improved win-win quiescent point algorithm: A recommender system approach, Journal of Applied Sciences 10 (23) (2010) 3084–3090.

[29] M. O'Connor, D. Cosley, J. A. Konstan, J. Riedl, Polylens: a recommender system for groups of users, in: Seventh European Conference on Computer Supported Cooperative Work ECSCW, 2001.

[30] M. Pazzani, D. Billsus, The Adaptive Web, chap. Content-based recommendation systems, Springer Berlin / Heidelberg, 2007, pp. 325–341.

[31] C. Plua, A. Jameson, Collaborative preference elicitation in a group travel recommender system, in: Proceedings of the AH Workshop on Recommendation and Personalization in eCommerce. Malaga, Spain., 2002.

[32] J. A. Recio-Garcia, G. Jimenez-Diaz, A. A. Sanchez-Ruiz, B. Diaz-Agudo, Personality aware recommendations to groups, in: Proceedings of the third ACM conference on Recommender systems (RecSys'09), 2009.

[33] P. Resnick, H. R. Varian, Recommender systems, Communications of the ACM 40 (3) (1997) 56–58.

[34] F. Ricci, D. Cavada, Q. N. Nguyen, Integrating travel planning and on-tour support in a case-based recommender system, in: Proceedings of the Workshop on Mobile Tourism Systems (in conjunction with Mobile HCI'02), 2002.

[35] A. Rubinstein, Perfect equilibrium in a bargaining model, Journal of the Econometric Society (1982) 97–109.

[36] L. Sebastia, I. Garcia, E. Onaindia, C. Guzman, e-Tourism: a tourist recommendation and planning application, International Journal on Artificial Intelligence Tools (WSPC-IJAIT) 18 (5) (2009) 717–738.

[37] V.-W. Soo, S.-H. Liang, Recommending a trip plan by negotiation with a software travel agent, in: International Workshop on Cooperative Information Agents (CIA), 2001.

[38] J. Wohltorf, R. Cissee, A. Rieger, H. Scheunemann, Berlintainment: An agent-based serviceware framework for context-aware services, in: Proceedings of the International Symposium on Wireless Communication Systems. Mauritius., 2004.

[39] M. Wooldridge, An Introduction to Multiagent Systems, John Wiley and Sons Ltd, 2002.

[40] R. Zhang, S. Zhang, S. Ye, Y. Zhao, J. Ford, J. Makedon, Providing recommendations in scens, The electronic Journal for e-commerce tools and Applications 2 (4) (2009) 9.