Social decisions in MAS

Francisco Grimaldo[1] , Miguel Lozano[2] and Fernando Barber[3]

Resumen— **This paper presents MADeM, a multi-modal agent decision making to provide virtual agents with socially acceptable decisions. We consider multi-modal decisions as those that are able to merge multiple information sources received from a MAS. MADeM performs social decisions since it relies on auctions, a well known market-based coordination mechanism. Our social agents express their preferences for the different solutions considered for a specific decision problem, using utility functions. Therefore, coordinated social behaviors such as task passing or planned meetings can be evaluated to finally obtain socially acceptable behaviors. Additionally, MADeM is able to simulate different kinds of societies (e.g. elitist, utilitarian, etc), as well as social attitudes of their members such as, egoism, altruism, indifference or reciprocity. MADeM agents have been successfully verified in a 3D dynamic environment while simulating a virtual university bar, where different types of waiters (eg. coordinated, social, egalitarian) interact to finally animate complex social scenes.**

Palabras clave— **Social Reasoning, Multiagent Resource Allocation, Welfare Economics.**

I. Introduction and related work

DECISION making is the cognitive process leading to the selection of a course of action among variations. There are several factors that influence this process, although probably the most important could be the amount of information the agent manages when deciding its actions. This is specially important in multiagent contexts, since our aim is to produce socially intelligent agents capable of displaying acceptabe decisions for a specific society model. Socially intelligent agents are autonomous problem solvers that have to achieve their goals by interacting with other similarly autonomous entities [1]. This kind of social agents is required in many complex simulation environments: military/civil simulations, social pedestrians in virtual cities, games, etc.

From virtual agents community, the behavioral animation problem points to the construction of an intelligent system which is able to integrate the different techniques required for the realistic simulation of virtual humans behavior. Among them, we can include perception, motion control, goal selection, action execution, communication between agents, interaction with the environment, etc. Traditionally, designers have sought to make their agents rational so that they can behave efficiently (i.e. the shorter the plan the better). Therefore, multiagent simulations have incorporated group coordination due to the fact that task-oriented agents (i.e. agents devoted to accomplish a set of goals) easily come into conflicts in a resource bounded environment even though their goals are compatible. For example in Generalized Partial Global Planning (GPGP) [2], agents merge the meta-plans describing their operational procedures and figure out the better action in order to maximize the global utility. Collaboration is supported in the RETSINA system [3] thanks to the use of communicative acts that synchronize tasks and occasionally manage conflicts. Team formation and task coordination for heuristic search planning characters is presented in [4] to adapt better to the dynamism of shared environments.

Although this kind of approaches have shown efficient task-oriented behaviors, human behavior is not purely self-interested [5]. Instead, autonomous virtual humans should also display social behaviors such as interchanging information with their partners or grouping and chatting with their friends. According to this, the theories of group dynamics developed in human social psychological sciences have recently inspired several models of interaction in artificial societies (e.g. the SGD model [6] or the social-reasoning based on exchange values [7]).

The purpose of MADeM is to provide a MAS simulation framework with agents managing multi-modal social decisions. We introduce this kind of decisions as those able to consider different focuses of attention coming from different sources. On the one hand, different focuses of attention (i.e. points of view) are required to provide more informed self-interested decisions. For instance, when deciding their actions, humans easily balance several aspects such as: efficiency, tiredness, skillfulness, mood, etc. On the other hand, a MADeM agent integrates social influence by asking external agents about the points of view the former is interested in. This feedback represents the preferences of the others for a specific situation, that basically includes certain resource and task allocations (see section II-A). Internally, MADeM is based on the MARA theory [8] and it uses auctions as a basic procedure to provide the social feedback mentioned (see [9] for a good set of social welfare examples). However, we let the agents manage more than one auction and allocation associated to each decision, so they can simulate multi-modal social decisions in a MAS.

Next, we present the MADeM information domain, that basically includes the types of resources being used (section II-A) and the agent preferences representation (section II-B). Then, we explain the decision making procedure, including the auctioning process (section III) and the winner determination problems faced (sections III-A, III-B). Section IV presents the application example created to verify MADeM. Finally, we show the social behaviors obtained according to the main MADeM parameters.

[1]Dpt. d'Informàtica, Univ. València, e-mail: francisco.grimaldo@uv.es
[2]Dpt. d'Informàtica, Univ. València, e-mail: miguel.lozano@uv.es
[3]Dpt. d'Informàtica, Univ. València, e-mail: fernando.barber@uv.es

II. MADeM: Multi-modal Agent Decision Making

The multi-modal agent decision making presented in this paper (MADeM) is based on the MARA theory, thus, it shares a similar domain of definition but making some additions to it. We summarize them as follows:

- A set of agents $A = \{a_1, ..., a_n\}$ where each a_i represents a particular agent involved in the decision. A vector of weights $\overrightarrow{w} =< w_1, ..., w_n >$ is associated to each agent representing the internal attitude of the agent towards other individuals. This information will be used to weigh the information received from other agents (section III-A).
- A set of resources to be allocated by the agents $R = \{r_1, ..., r_m\}$. The definition of resources we use is different from the classical definitions found in the literature and it will be explained in detail in section II-A.
- Instead of having only one utility function as in classical MARA problems, each agent will have a set of utility functions $\{U^1, U^2, ..., U^q\}$. These utility functions will be used to evaluate the allocations from different points of view. We better explain the utility functions for our agents in section II-B. Additionally, each agent will have a vector of utility weights $\overrightarrow{w_u} =< w_{u_1}, ..., w_{u_q} >$ representing the importance given to each point of view in the multi-modal agent decision making.

An example illustrating the functionality of MADeM could be a customer entering a virtual bar and deciding which waiter to place his order to. Different points of view could be considered: to ask the nearer waiter (i.e. tiredness), to ask the less occupied waiter (i.e. utilitarianism) or to ask a waiter who is a friend of him (i.e. sociability). Auctions are used to select the better candidates for each category. In this case, utility weights would express personal tendencies such as laziness, impatience or sociability. Looking at them, MADeM would be able to choose among these three possibilities. The details of the whole winner determination procedure will be explained in sections III-A and III-B.

A. *Types of resources*

In order to obtain social and intelligent behaviors, we provide the agents with the ability to ask for opinion or social feedback about different solutions for a specific decision problem (e.g. whether to pass the execution of a task to another agent or not). Agents do this by following an auctioning mechanism in which the resources being auctioned are tasks. Task auctioning has been widely used in the MAS community but its application to social virtual agents is not so common and needs to take new issues into account. The novelty of our approach is that we do not auction only the execution of a task as found in the literature. Instead, the resource to be auctioned is what we term *task slots*. We consider task slots as slots that need to be assigned in order to execute a task, thus, they can be considered as parameters of the action. When considering any kind of task, we differentiate two main types of task slots: *agent slots* which correspond to agents that play different roles in the task execution, as for example the executant of the task or the beneficiary of the task; and *object slots*, that correspond to objects needed to perform the action.

We also differentiate slots depending on the role they play in the task. There is a slot present in every kind of task which is the agent who carries it out (Ag_e). Besides, we consider as general slots two additional slots present in many tasks: the main object of the action (Obj_m) and the beneficiary or destinatary in any sense of the action (Ag_d). For example, in the *Give* action, Ag_e is the agent who initially has the object, Obj_m is the object being given and Ag_d is the agent who receives the object. In any case, it is always possible for a particular problem to consider more task slots if necessary.

It can be seen that the classical task auctioning is subsumed within our approach as it corresponds to the auctioning of the Ag_e slot. Moreover, the auctioning of objects is also possible by auctioning the appropriate task slot of the considered action (e.g. the Obj_m slot of an action Own).

According to this, we represent resources or task slots as $r = task(slot)$, where we consider the slot as a typed parameter according to an ontology defined for the problem. An assignment of an element (agent or object) to a slot is then represented as $task(slot) \leftarrow element$, and an allocation P of elements to task slots has the following representation:

$$P = \{t_1(s_1) \leftarrow e_1, t_1(s_2) \leftarrow e_2, \ldots, t_1(s_n) \leftarrow e_n, t_2(s_1) \leftarrow e_{n+1}, \ldots, t_m(s_n) \leftarrow e_{n*m}\}$$

Therefore, we represent each one of the solutions considered for a decision problem as an allocation of this type.

For example, in the bar environment presented in section IV we have modelled the task $Give(Waiter, Object, Customer)$ as the action used by waiters to give a product to a customer. In this task we have 2 slots that may be auctioned: the Ag_e, which is the waiter that gives, and the Ag_d, which is the customer that receives the product. These slots would be represented as $Give(Ag_e)$ and $Give(Ag_d)$. Hence, a possible allocation for these slots could be $< Give(Ag_e) \leftarrow a_1, Give(Ag_d) \leftarrow a_5 >$, given that a_1 is a waiter and a_5 is a customer.

B. *Agent preferences*

In the auctioning process, the auctioneer asks the agents to bid for one or more task slot allocations, each bidder then uses its utility functions to evaluate and score the different allocations being considered.

MADeM agents use non-negative 2-additive utility functions of the form:

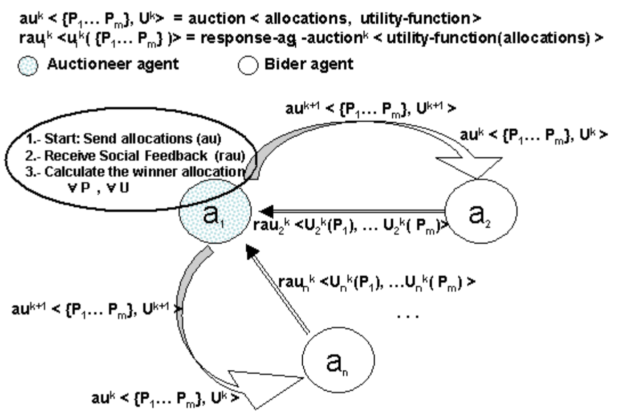
$au^k < \{P_1 \dots P_m\}, U^k >$ = auction < allocations, utility-function >
$rau_i^k < u_i^k(\{P_1 \dots P_m\}) >$ = response-ag$_i$ -auction^k < utility-function(allocations) >
Auctioneer agent
Bider agent
1.- Start: Send allocations (au)
2.- Receive Social Feedback (rau)
3.- Calculate the winner allocation $\forall P$, $\forall U$
$au^{k+1} < \{P_1 \dots P_m\}, U^{k+1} >$
$au^k < \{P_1 \dots P_m\}, U^k >$
a_1
a_2
$rau_2^k < U_2^k(P_1), \dots U_2^k(P_m) >$
$rau_n^k < U_n^k(P_1), \dots U_n^k(P_m) >$
...
$au^{k+1} < \{P_1 \dots P_m\}, U^{k+1} >$
a_n
$au^k < \{P_1 \dots P_m\}, U^k >$

Fig. 1. MADeM Procedure

$$U(P) = \sum_{p \in P} u(p) + \sum_{p_1, p_2 \in P} u(p_1, p_2) \qquad (1)$$

where $u(p)$ is a utility function for a unique slot assignment and $u(p_1, p_2)$ is the extra utility value given to the situation in which both assignments are done. In order to obtain the utility functions $(U(P))$ normalized in the range $[0, 1]$, the utility functions of all agents must be divided by the same constant, which will depend on the particular problem.

Despite the fact that the types of resources being auctioned are tasks, utility functions express benefit. Therefore, agents would aim at maximizing their utility. Using cost functions could also be possible but agents would aim at minimizing their costs.

III. Decision making procedure

MADeM uses one-round sealed-bid combinatorial auctions to choose among different solutions to a decision problem. As mentioned above, a solution is represented by an allocation of one or many task slots. Auctioneer and bidder roles are not played by fixed agents throughout the simulation. Instead of that, every agent can dynamically adopt each role depending on his/her needs or interests. For example, an agent would be the auctioneer when he wanted to pass a task to another agent following a social behavior. On the other hand, agents receiving the auction would bid their utility values provided that they were interested in the task slots being auctioned. Our model allows more than one auction to be running at the same time. Similarly, agents can participate in several auctions simultaneously. Thus, this approach lies in between centralized and distributed market-based allocation.

An overview of the multi-modal decision making procedure followed by the agents to generate socially acceptable decisions is shown in figure 1. This procedure is mainly based on the following steps:

1. *Auctioning phase*: This phase is carried out by an agent (a_1) who wants to socially solve a decision problem (e.g. where to sit). This agent then constructs the set of allocations representing all the possible solutions for the problem $(< P_1, P_2, ..., P_m >)$. These allocations have the form of task slots assignations such as $SitAt(Obj_m) \leftarrow table_1$. Next, he auctions them to a particular group of agents, that we call the target agents. Each auction also includes a single type of utility function that the agent is interested in evaluating from the others $(au^k(< P_1, P_2, ..., P_m >, U^k))$. As complex decisions require to take into consideration more than one point of view, the auctioneer agent can start different auctions for the same set of allocations (au^1 through au^q).
   The process to select the target agents for an auction varies depending on the kinds of task slots being allocated. When dealing with agent slots, the target agents can be extracted from the type of the task slot being auctioned. For example, in our bar environment, the task slot $Make\_Coffee(Ag_e)$ should only be auctioned to agents of the class *Waiter*. On the other hand, when allocating object slots, the target agents could be those agents that are somehow related to objects of the same type as the task slot being auctioned. For instance, the task slot $SitAt(Obj_m)$, where Obj_m corresponds with the table where to sit at, could be auctioned to all the agents related to a table in the environment. When there is no type system available, the target agents would be the whole set of agents.
2. *Bidding phase*: Since the auctioneer informs about both the task slot allocations and the utility functions being considered, bidders simply have to compute the requested utility functions and return the values corresponding to each auction back to the auctioneer $(ra_i^k =< U_i^k(P_1), ..., U_i^k(P_m) >)$.
3. *Winner determination phase*: In this phase, the auctioneer selects a winner allocation for each launched auction, that is, for each point of view being considered. To do this, he uses a classical winner determination problem, as explained in section III-A. Afterwards, he chooses one final winner allocation among these auction winners using a multi-modal decision making process. The details of this calculation are fully described in section III-B. Thus, the final winner allocation will represent an acceptable decision for the society being simulated.

A. Winner Determination Problem

Once bidders have answered to an auction call (no answering means no preference, therefore, utility zero) the auctioneer agent has the utility values $(U_i(P_j))$ given by each bidder $(i \in A)$ to every allocation being evaluated (P_j). Equation 2 groups these utility values in a set of vectors, one for each allocation.

$$\overrightarrow{U(P_j)} =< U_1(P_j), ..., U_n(P_j) > \quad \forall j \in [1..m] \quad (2)$$

Remember that every agent had an associated vector of weights representing its attitude towards the other individuals ($\overrightarrow{w}$). According to it, the auctioneer weighs the utility vectors in equation 2 doing a component by component multiplication with the attitude vector as shown in equation 3.

$$\overrightarrow{U_w(P_j)} = \overrightarrow{U(P_j)} * \overrightarrow{w} \ \forall j \in [1..m] \tag{3}$$

Attitude weights are used to model the social behavior of the auctioneer agent. For example, egoism is modelled giving weight 1 to himself and 0 to all other agents. In fact, a whole range of behaviors between egoism and altruism can be modelled using the vector of equation (4), where $p = 0$ represents the previous behavior, $p = 1$ represents total altruism and $p = 0.5$ represents an egalitarian behavior or indifference between oneself and the rest of the agents.

$$\begin{array}{l} Egoism - Altruism: \quad \overrightarrow{w} = < p, ..., p, 1-p, p, ..., p > \\ w_i = 1-p, w_{j \neq i} = p, i = Myself \end{array} \tag{4}$$

It is also possible to model reciprocal attitudes by means of the vector $\overrightarrow{w}$. A simple example is shown in equation 5, where weights are based on the interchange of favors between agents.

$$Reciprocity: \quad w_i = \frac{Favors\_from(i)}{Favors\_to(i)} \tag{5}$$

In order to behave socially, the auctioneer attends to the social welfare value when selecting the winner allocation of an auction. Thus, the winner determination problem chooses the allocation that maximizes the welfare of the society (see equation 6).

$$Winner = P_w \longleftrightarrow sw(P_w) = \max_{j \in [1..m]} sw(P_j) \tag{6}$$

To compute the social welfare of an allocation, the auctioneer uses Collective Utility Functions (CUFs) and the weighted utilities defined in equation 3 (as shown in equation 7). MADeM allows to select among different CUFs when evaluating the social welfare of an allocation, each one related to a kind of society: utilitarian, egalitarian, elitist and nash.

$$sw(P) = cuf(\overrightarrow{U_w(P)}) \tag{7}$$

$$\begin{array}{ll} cuf_{util} = \Sigma u_w(i) & cuf_{egal} = \min\{u_w(i)\} \\ cuf_{elitist} = \max\{u_w(i)\} & cuf_{nash} = \Pi u_w(i) \end{array} \tag{8}$$

B. Multi-modal decision making

An agent can ask other agents about different points of view (e.g. efficiency, tiredness, etc). In order to do this, he performs several auctions with different types of utility functions (see parameter U^k in figure 1). Once all these auctions have been resolved, the auctioneer has the winner allocation for each point of view and the social welfare obtained provided that allocation is adopted (see equation 9).

$$\begin{array}{l} auction_1(\{P_1, P_2, ..., P_m\}, U^1) \longrightarrow (P_{w1}, sw(P_{w1})) \\ \qquad ... \\ auction_k(\{P_1, P_2, ..., P_m\}, U^k) \longrightarrow (P_{wk}, sw(P_{wk})) \end{array} \tag{9}$$

The final winner allocation is then chosen using the vector of utility weights $\overrightarrow{w_u}$. MADeM taked the values in $\overrightarrow{w_u}$ as weights assigned to each utility function. Hence, the final winner allocation is that which maximizes the welfare of the society after having multiplied it by the corresponding utility weight (see equation 10),

$$P_w = P_{wi} \longleftrightarrow sw(P_{wi}) = \max_{i \in [1..k]} w_u(i) * sw(P_{wi}) \tag{10}$$

IV. Application example

In this section we show how we have integrated MADeM into a multi-agent framework oriented to simulate socially intelligent characters in 3D virtual environments. This framework is developed over Jason [10], which allows the definition of BDI agents using an extended version of AgentSpeak(L) [11]. Here, we have created a virtual university bar where waiters take orders placed by customers. The typical objects in a bar, such as a juice machine, behave like dispensers that have an associated time of use to supply their products (e.g. 2 minutes to make an orange juice) and they can only be occupied by one agent at a time. Therefore, waiters should coordinate to avoid conflicts. Besides, waiters are social linked with their friends and this social network is used when deciding whether to do favors, to promote social meetings, etc.

Waiters serve orders basically in two steps: first, using the corresponding dispenser (e.g. the grill to produce a sandwich); and second, giving the product to the customer. Though, tasks are always auctioned using MADeM before their execution in order to find good social allocations. Hence, the slot being auctioned will be the executor of the task (Ag_e) for both of them (see table I). The classes of objects used to describe task slots are extracted from the object taxonomy defined in the world ontology. Therefore, the set of allocations among which to decide is represented in equation 11. That is, for each task, a waiter evaluates whether to carry out the task against the chance to pass it to another waiter and perform his next task.

$$\begin{cases} P_0 = \{t(Ag_e) \leftarrow Myself\} \\ P_i = \{t(Ag_e) \leftarrow a_i, t_{next}(Ag_e) \leftarrow Myself\} \end{cases} \tag{11}$$

Waiters take into account three points of view when calling MADeM: *performance*, *chatting* and

TABLA I
TASKS AND SLOTS BEING CONSIDERED.

Tasks/ Slots	Ag_e	Obj_m	Ag_d
Use	Waiter	Dispenser	-
Give	Waiter	Product	Customer

tiredness. Equations 12 and 13 define the utility values returned by the performance utility function for the tasks *Use* and *Give*. This function aims at maximizing the number of tasks being performed at the same time and represents the waiters' willingness to serve orders as fast as possible. Social behaviors defined for a waiter are oriented to animate chats among his friends at work. Therefore, waiters implement the social utility function detailed in equations 14 and 15, where *Near* computes the distance between the agents while they are executing a pair of tasks. These functions evaluate social interest as the chance to meet a friend in the near future, thus performing a planned meeting. Finally, equation 16 defines the tiredness utility function for a waiter. This later function implements the basic principle of minimum energy, widely applied by humans at work. The type of society being simulated for waiters is elitist, thus, waiters will choose those allocations that maximize the utility functions previously defined.

$$U^{perf}(\text{Use}(Ag_e) \leftarrow a) = \begin{cases} 0 & \text{if } a \neq Myself \\ 1 & \text{if } (a = Myself) \wedge \\ & IsUsing(Myself, Obj_m) \wedge \\ & not(IsComplete(Obj_m))] \\ \frac{1}{t_{tobefree} + t_{queue}} & \text{Otherwise} \end{cases} \tag{12}$$

$$U^{perf}(\text{Give}(Ag_e) \leftarrow a) = \begin{cases} 0 & \text{if } a \neq Myself \\ 1 & \text{if } (a = Myself) \wedge \\ & CurrentTask = \text{'Give'} \wedge \\ & not(HandsBusy(Myself) < 2)] \\ \frac{1}{t_{tobefree}} & \text{Otherwise} \end{cases} \tag{13}$$

$$U^{soc}(t_1(Ag_e) \leftarrow a_1, t_2(Ag_e) \leftarrow a_2) = \begin{cases} 0 & \text{if } a_1 \neq Myself \vee a_2 \neq Auctioneer \\ 1 & \text{if } (a_1 = Myself) \wedge a_2 = Auctioneer \wedge \\ & IsFriend(a_1, a_2) \wedge Near(Pos(t_1), Pos(t_2)) \wedge \\ & ExecTime(t_2) > RemainTime(CurrentTask) \\ 0 & \text{Otherwise} \end{cases} \tag{14}$$

$$U^{soc}(t(Ag_e) \leftarrow a) = \begin{cases} 0 & \text{if } a \neq Auctioneer \\ 1 & \text{if } a = Auctioneer \wedge IsFriend(a, Myself) \\ & \wedge Near(Pos(CurrentTask), Pos(t)) \wedge \\ & \wedge TimeToStart(t, a) = Now \wedge \\ 0 & \text{Otherwise} \end{cases} \tag{15}$$

TABLA II
RESULTS FOR DIFFERENT TYPES OF WAITERS.

Agent	σ_{NTasks}	$\overline{NChats}$	$Tasks/s$
Coordinated	6.73	5	0,91
Social	4.37	29.4	0.65
Egalitarian	2.74	6.6	0.62
Self-interested	-	-	0.17

$$U^{tir}(t(Ag_e) \leftarrow a) = \begin{cases} 1 - \frac{tasks\_done}{total\_tasks} & \text{if } a = Myself \\ 0 & \text{Otherwise} \end{cases} \tag{16}$$

V. Results

In order to verify the social outcomes obtained with MADeM agents, we have simulated different types of waiters serving customers. The results shown in this section correspond to simulations where 10 waiters attend 100 customers.

As we have previously mentioned, we have modelled an elitist society of waiters within which agents attend to three points of view (i.e. performance, sociability and tiredness), each of them represented by its own utility function. In this context, utility weights can be adjusted to create different types of social waiters. For example, a *coordinated* waiter could be an agent that chooses its decisions following performance 75% of the times and following sociability or tiredness in the rest of the situations. The vector of utility weights for a *coordinated* waiter would then be $\overrightarrow{w_u} =< 0.75, 0.125, 0.125 >$, where each component represents the importance given to each utility function being evaluated. Similarly, we have defined *social* waiters as agents with the following vector of utility weights $\overrightarrow{w_u} =< 0.125, 0.75, 0.125 >$ and *egalitarian* waiters as agents with $\overrightarrow{w_u} =< 0.125, 0.125, 0.75 >$. Table II summarizes some results obtained with *coordinated*, *social* and *egalitarian* waiters against *self-interested* waiters with no social mechanism included. *Coordinated* waiters perform better (see column $Tasks/s$) since the majority of conflicts caused by the use of the same dispenser (e.g. the coffee machine) are resolved with specialization, that is, by passing the task to another waiter already using the dispenser. On the other hand, *social* waiters take more time to serve customers but animate a greater number of chats among friends (see the mean number of chats being animated in column $\overline{NChats}$). *Egalitarian* waiters look at the tiredness utility function and try to allocate the task to the least tired waiter, therefore, the standard deviation in the number of tasks performed by each agent tends to zero (see column σ_{NTasks}). Finally, *self-interested* waiters demonstrate to perform worse than any kind of social waiter. As this agents are unable to do task passing nor chatting, columns σ_{NTasks} and $\overline{NChats}$ are not considered.

Besides the possibility to define the importance of each point of view through the vector of utility weights $\overrightarrow{w_u}$, MADeM allows for the definition of a

TABLA III

Results for different personal weights.

Coordinated	$\sigma_{Favours}$	$\overline{Favours}$
Indifference	7.57	6.9
Reciprocity	1.15	8.8
Altruism	5.94	17
Egoism	1.41	0.7
Social	$\sigma_{Favours}$	$\overline{Favours}$
Indifference	3.52	8.7
Reciprocity	1.76	7.8
Altruism	6.66	12.7
Egoism	0.81	0.4
Egalitarian	$\sigma_{Favours}$	$\overline{Favours}$
Indifference	7.58	13.6
Reciprocity	2.4	15.5
Altruism	4.44	17.9
Egoism	0.47	0.1

vector of personal weights $\overrightarrow{w}$ that models the attitude of an agent towards the other individuals. Table III shows the results obtained for the previously defined waiters using the models of attitude introduced in section III-A. Agents using *indifference* do not apply any modification over the utilities received, therefore, we consider the results of this attitude as the base values to compare with for each type of waiter. *Reciprocity* weights utilities attending to the ratio of favors already done between the agents. This attitude produces equilibrium in the number of favors exchanged as it can be seen in column $\sigma_{Favours}$. Altruism has been implemented in such a way that the weight given to oneself utilities is 0.25 whereas the weights for the rest of the agents is 0.75. As expected, altruist agents do more favors, since the importance given to the other's opinions is three times the importance given to their own opinion (see high values for the mean number of favors exchanged $\overline{Favours}$). On the other hand, egoism weights are 0.75 to oneself and 0.25 to the others, thus, agents rarely do favors (see low values in column $\overline{Favours}$).

Agent's preferences can sometimes go against personal attitudes. For example, whereas *reciprocity* tries to balance the number of favors, tiredness tends to assign tasks to the least tired waiter (see the greater $\sigma_{Favours}$ for egalitarian waiters). Another example is *egoism* applied to *egalitarian* waiters, in this case no task at all is passed among the agents ($\overline{Favours} = 0.1$). However, agent's preferences can also empower personal attitudes. For instance, *altruism* applied to *coordinated* waiters produces a high level of specialization. This type of agents produces big values for $\sigma_{Favours}$ as the agents already using a dispenser (e.g. a juice machine) keep on getting products from the dispenser following both an altruist and a coordinated behavior that reduces collisions for the use of an exclusive resource. Despite that, personal weights have demonstrated to produce similar effects on the agents regardless of the kind of waiter being considered (i.e. *coordinated*, *social* or *egalitarian*).

VI. Conclusions

The decision making approach presented in this paper aims at incorporating human style social reasoning for character animation. According to this, we have presented MADeM as a market based multimodal agent decision making for social MAS that is able to obtain different kind of coordinated behaviors for the agents involved. On the one hand, MADeM decisions allow the agents to manage several points of view related to its actions. This social feedback is modelled via utility functions that express the different preferences of each agent for the allocations received for each solution being considered. As an example, a group of socially intelligent waiters has been created that consider different points of view in their decision making: performance, sociability and tiredness. On the other hand, MADeM allows the agents to model specific attitude towards the others. The attitudes modelled include indifference (do not modify the other's utility), reciprocity (use the ratio of favors between the agents involved), altruism (the more work for me the better) or egoism (the more work for the others the better).

Acknowledgments

This work has been jointly supported by European Commission FEDER funds and the Spanish MEC under grants Consolider Ingenio-2010 CSD2006-00046 and TIN2006-15516-C04-04.

Referencias

[1] L. M. Hogg and N. Jennings, "Socially intelligent reasoning for autonomous agents," *IEEE Transactions on System Man and Cybernetics*, vol. 31, no. 5, pp. 381–393, 2001.

[2] K. S. Decker and V. R. Lesser, *Readings in Agents*, chapter Designing a family of coordination algorithms, 1997.

[3] J. A. Giampapa and K. Sycara, "Team-oriented agent coordination in the RETSINA multi-agent system," Tech. report CMU-RI-TR-02-34, Robotics Institute-Carnegie Mellon University, 2002.

[4] F. Grimaldo, M. Lozano, F. Barber., and J.M. Orduña, "Integrating social skills in task-oriented 3D IVA," in *IVA'05*. 2005, Springer-Verlag LNAI.

[5] Steven De Jong, Karl Tuyls, and Katja Verbeeck, "Artificial agents learning human fairness," in *AAMAS '08*, 2008, vol. 2, pp. 863–871.

[6] Rui Prada and Ana Paiva, "Believable groups of synthetic characters," in *AAMAS '05*, 2005, pp. 37–43.

[7] M. Ribeiro, A. C. da Rocha, and R. H. Bordini, "A system of exchange values to support social interactions in artificial societies," in *AAMAS'03*, ACM, Ed., 2003.

[8] Yann Chevaleyre, Paul E. Dunne, Ulle Endriss, Jerome Lang, Michel Lemaitre, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A. Rodriguez-Aguilar, and Paulo Sousa, "Issues in multiagent resource allocation," in *Informatica*, 2006, pp. 3–31.

[9] Yann Chevaleyre, Ulrich Endriss, Sylvia Estivie, and Nicolas Maudet, "Welfare engineering in practice: On the variety of multiagent resource allocation problems," in *ESAW*, 2004, pp. 335–347.

[10] R. H. Bordini and J. F. Hübner, "Jason," Available at http://jason.sourceforge.net/, March 2007.

[11] A. S. Rao, "AgentSpeak(L): BDI agents speak out in a logical computable language," in *Proc. of MAAMAW'96*, Springer Verlag, Ed., 1996, number 1038 in LNAI, pp. 42–55.