Balancing social and task oriented behaviors for animation agents*

Francisco Grimaldo, Miguel Lozano, Fernando Barber
Computer Science Department
University of Valencia
Dr. Moliner 50, 46100 Burjassot (Valencia)
{francisco.grimaldo, miguel.lozano, fernando.barber}@uv.es

Abstract

This paper presents a multiagent framework oriented to design groups of virtual agents able to combine task oriented goals while animating social interactions. The social behaviors introduced can be rational (to increase the global performance of the group) and also social (to increase the good relations between friends). As rationality and sociability are often conflicting, the BDI agents created incorporate mechanisms to make socially acceptable decisions, based on social welfare functions and a reciprocity model. We show the results obtained by the system while simulating a virtual bar where groups of waiters and customers are able to efficiently serve/consume different orders and also to animate some social behaviors, as task passing, planned meetings or simple chats.

1 Introduction and related work

The notion of socially acceptable decisions has long been of interest in human societies and also in the multiagent community [10]. Multiagent systems are sometimes referred to as societies of agents and, from the animation point of view, this points to the design of an adequate framework to produce good quality animations for groups of virtual characters. When designing such agents, the main concern is normally with the decision-making mechanism as it is the responsible for the actions that will be finally animated. Traditionally, designers have sought to make their agents rational so that, they can *"do the right thing"* efficiently (i.e. the shorter the plan the better). However, AI-based character animation approaches normally operate in a resource bounded context and obstruction situations appear when characters compete for the use of shared resources (i.e. objects in a virtual environment). Therefore, self interested agents (i.e. agents devoted to accomplish a set of goals) easily come into conflicts even though their goals are compatible.

Many researches have tried to achieve social skills through multiagent coordination. For example in Generalized Partial Global Planning (GPGP) [7], agents merge the meta-plans describing their operational procedures and figure out the better action in order to maximize the global utility. Another example is Multiagent Planning Language (MAPL) [6], that assigns the control over each resource to a unique agent and uses speech-acts to synchronize planned tasks. Collaboration is supported in the RETSINA system [8] thanks to the use of communicative acts (predefined inside Hierarchical Task Networks) that synchronize tasks and occasionally manage conflicts. To adapt better to the dynamism of shared environments, an heuristic planner is used in [9] to support team formation and task coordination between virtual characters. BDI actors have also been applied to character animation. In [4], the reasoning of a virtual human has been implemented in Jason [2] but no social skills

*Supported by the Spanish MEC under grants TIN2006-15516-C04-04 and Consolider Ingenio 2010 CSD2006-00046.

have been modelled. MAS-SOC [1] is a similar system in the literature that aims at creating a platform for multiagent based social simulation. In this context, ongoing work is being done in order to incorporate social-reasoning mechanisms based on *exchange values*, that is, the set of values assigned to the services exchanged among agents during an interaction [3].

All the approaches presented above focus on improving the global efficiency of multiagent simulations, however, an excess of coordination can produce unethical behaviors. For example, an excessive degree of specialization do not contribute to animating realistic behaviors, instead, egalitarian societies of agents can be more interesting when simulating groups of virtual humans. Characters that show purely social behaviors (eg. chatting with other characters or grouping with friends) are required in many complex environments (e.g. virtual cities, shops, bars...), where agents should balance properly rationality and sociability to finally display high quality behavioral animations.

2 Social multiagent animation framework

The social multiagent framework presented here has been developed over Jason [2], which allows the definition of BDI agents using an extended version of AgentSpeak(L) [5]. Figure 1 shows the main modules of our framework.

The animation framework (virtual characters, motion tables, etc) is located at the 3D engine, which can run separately. The agent decision-making is defined in the *Agent Specification File*. This file contains the initial beliefs as well as the set of plans that make up the agent's finite state machine. The *Rational Module* contains the set of plans oriented *"to do the right thing"* according to the character's role. This points directly to efficiency, hence, in our simulation, these plans sequence the actions needed to animate a task. For instance, serving a coffee would require going to the coffee machine to get the coffee and giving it to the customer afterwards. Here, modularity is guaranteed since the *Task library* can be changed depending on the environment being simulated.

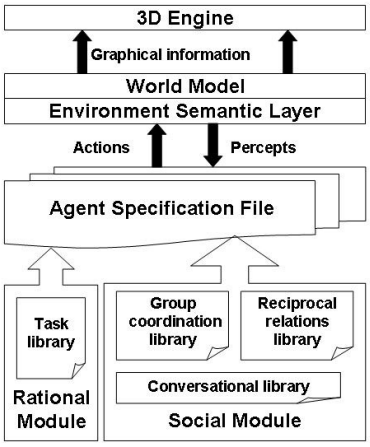

Figure 1: Social multiagent animation framework.

As stated above, only rational behaviors are not enough to simulate agent societies, therefore, our framework also introduces a *Social Module* to animate different types of social situations. Firstly, a *Group coordination library* has been included to avoid conflicts between self-interested agents and to allow them to cooperate while executing their tasks. This library includes an auction model that uses a utilitarian social welfare function. Secondly, the *Reciprocal relations library* can be used to create agents that promote social interactions. Auctions are also used here, but a reciprocity model has been included in order to implement a more sophisticated social welfare function. Finally, the *Conversational library* contains the set of plans that handle the animation of interactions between agents (e.g. task passing, planned meetings, chats between friends...). The environment is handled by a *Semantic Layer* which acts as an interface between the agent and the world. It is in charge of perceiving the state of the world and executing the actions requested by the agents, while ensuring the consistency of the *World Model*.

To illustrate the presented social multiagent animation framework, we have created a

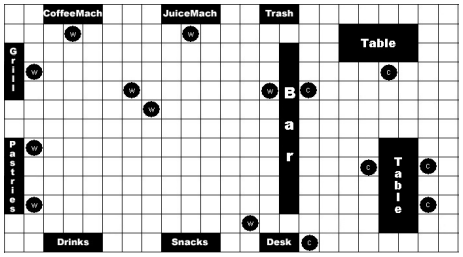

Figure 2: Virtual university bar environment.

virtual university bar where waiters take orders placed by customers without neglecting their social capabilities (i.e. group coordination and partner friendship relations). On the other hand, customers place orders and consume them when served. Here, sociability is achieved since customers try to sit with their friends. The environment models six typical locations in a bar: a juice machine, a coffee machine, a grill, a shelf with snacks, a refrigerator with drinks and a larder with pastries (see figure 2). These locations behave like resources that have an associated time of use to provide their products (e.g. an orange juice) and they can only be occupied by one agent at a time. Additionally, there is a cash desk for the customers to place orders, a bar for the waiters to serve the orders to the corresponding customer and tables with chairs to consume the served orders. To simulate different workload situations, customers can be configured with the desired order frequency. Moreover, product probabilities (i.e. probability of coffee, probability of snack...) can also be set to model the typical scenarios in a bar: breakfast time, lunch time, etc.

3 Animation agents

The simulation of worlds inhabited by interactive virtual actors normally involves facing a set of problems related with the use of shared limited resources and the need to animate pure social behaviors. In this section, we present two types of agents that manage these issues:

3.1 Socially rational agents

The exchange of tasks can increase the group performance in environments where, despite the use of a resource being limited to one single character, it is possible to use a resource to perform several tasks simultaneously (e.g. a coffee machine in our virtual university bar can be used by a waiter to make more than one coffee at the same time). According to this, we can define socially rational agents in our framework that use the *Group Coordination library* to exchange tasks between them using a Multiagent Resource Allocation approach with the following characteristics:

- *Type of resource:* We use tasks (t) as the type of resource to be allocated. In this context, tasks are indivisible and not sharable (e.g. use the juice machine to get an orange juice, give a drink to a customer, etc.).

- *Agent preferences:* Each agent (i) has a performance utility function ($U^i_{perf}(t)$) that quantifies its will to perform a task (t). An important characteristic of tasks as opposed to resources is the fact that tasks are coupled with constraints regarding their coherent combination. Thus, utility functions must take into account the agent and world states as well as the task being evaluated.

- *Solution method:* Tasks are exchanged between agents using a first-price sealed-bid (FPSB) auction model. The *Group coordination library* implements this kind of auctions so that an agent can auction any task during its animation.

- *Social Welfare:* Social rationality has been modelled through a social welfare function that is used during the winner determination problem. The aim of this function is to represent the collective utility function from a multiagent point of view. The socially rational agent presented at this point uses the following elitist function:

$$sw_{el}(t) = \max\{U^i_{perf}(t) | i \epsilon Agents\} \quad (1)$$

That is, the winner of an auction will be the agent that bid the maximum performance utility for the task t. Note that sw_{el} has t as its domain but not a resource allocation, as it can be usually found in the social welfare theory. The reason for this change in the notation is that we are only interested in the allocation of the next task. Negotiation of long sequences of actions is not very interesting for interactive characters, as plans will probably be broken due to the dynamism of the environment and to other unpredictable events (e.g. a meeting with a friend).

Figure 3 depicts the finite state machine that describes a socially rational waiter in the application example introduced in section 2[1]. Waiters serve an order basically in two steps: first, using the corresponding resource (e.g. the grill to produce a sandwich) and second, giving the product to the customer at the bar. Tasks are always auctioned before their execution but once an agent has committed to do a task to an auctioneer, that task cannot be reallocated again. Finally, the result of a subcontracted task is informed to the referred auctioneer agent.

The performance utility function ($U^i_{perf}(t)$) employed by these agents is shown in Algorithm 1. This function aims to maximize the number of parallel tasks being performed. Therefore, cooperation emerges since all agents want to serve orders as fast as possible. Socially rational agents can reach high performances but they can also be unrealistic for the animation of artificial human societies. For example, since they work as much as they can, they will display unethical or robotic behaviors. To rectify this situation, agents should compensate for this elitism with pure social behaviors. These behaviors are oriented to animate normal social relations between the members of a society (e.g. friendliness).

[1]The finite state machine is specified by means of plans in Jason's extended version of AgentSpeak(L)

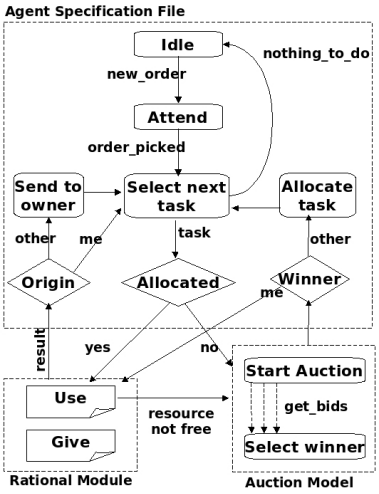

Figure 3: Socially rational waiter specification.

```
R=GetResourceUsedBy(t)
if t is Use then
   if i is Auctioneer then
      if R is free then return(1)
      else return(0)
   else
      if (IsUsing(i,R) and
         not(R is complete)) then
         return(1)
      else return(0)
else if t is Give then
   if i is Auctioneer then
      if nextAction is NULL then
         return(1)
      else return(0)
   else
      if (currentTask is Give and
         not(handsBusy<2) then
         return(1)
      else return(0)
else return(0)
```

Algorithm 1: PerformanceUtility(i,t)

3.2 Reciprocal social agents

These kinds of agents introduce a relational model between the members of the group. Whereas performance utility functions mod-

elled the interest of an agent to exchange a task still from a rational point of view, we introduce two additional social utilities to represent the social interest in exchanging a task. The aim of social utilities is to promote task allocations that, in the short term, lead the agents to perform social interactions with their friends. Hence, agents can now decide whether to adopt this kind of social allocations or to be rational as explained previously.

Reciprocal social agents express their preferences through a tuple of the form $[U^i_{perf}(t), U^i_{int}(j,t,t_{next}), U^i_{ext}(j,t,t_{next})]$, where the definitions of the new social utilities are:

- Internal social utility ($U^i_{int}(j,t,t_{next})$): is the utility that a bidder agent i assigns to a situation where i commits to do the auctioned task t so that the auctioneer agent j can execute his next task t_{next}.
- External social utility ($U^i_{ext}(j,t,t_{next})$): is the utility that a bidder agent i assigns to a situation where the auctioneer agent j executes the auctioned task t while i continues his current action.

Based on these utilities, we define the social winner of an auction as the agent that maximizes the following indicators:

$$SW_{rec}(t) = \max\{U^*_{int}(t), U^*_{ext}(t)\} \quad (2)$$

$U^*_{int}(t)$ represents the maximum social utility given by one bidder to get the auctioned task (see equation 3). Besides, in order to balance task exchange, internal social utilities are weighted with a reciprocity matrix (see equation 4). We define the reciprocity factor w_{ij} for two agents i and j, as the ratio between the number of favours (i.e. tasks) that j has made to i and vice versa.

$$U^*_{int}(t) = \max\{U^i_{int}(j,t,t_{next}) * w_{ji} | i \epsilon Agents\} \quad (3)$$

$$w_{ij} = \frac{Favours_{ji}}{Favours_{ij}} \quad (4)$$

On the other hand, $U^*_{ext}(t)$ represents the maximum social utility given by all bidders to the situation where the task is not exchanged but performed by the auctioneer. It is calculated following equation 5 and the same reasoning applies to reciprocity.

$$U^*_{ext}(t) = \max\{U^i_{ext}(j,t,t_{next}) * w_{ij} | i \epsilon Agents\} \quad (5)$$

At this point, the winner determination problem has two possible candidates coming from equations 1 and 2. Agents choose between them in accordance with their *Sociability* factor, which is the probability to get the social winner instead of the rational winner. *Sociability* can be varied in the range $[0,1]$ to model intermediate behaviors between efficiency and total reciprocity. This can provide great flexibility when animating characters, since *Sociability* can be dynamically changed thus producing different behaviors depending on the world state (e.g. the workload in our virtual bar example).

```
if j is friend then
   endCurTask=RemainTime(currentTask)
   intervalA=[endCurTask,
         endCurTask + ExecTime(t)]
   intervalB=[0,ExecTime(tNext)]
   if Overlap(intervalA,intervalB) then
      if Near(t,tNext) then
         return(1)
      else return(0)
   else return(0)
else return(0)
```

Algorithm 2: InternalSocialUtility(j,t,tNext)

```
if j is friend then
   endCurTask=RemainTime(currentTask)
   intervalA=[0,endCurTask]
   intervalB=[0,ExecTime(t)]
   if Overlap(intervalA,intervalB) then
      if Near(currentTask,t) then
         return(1)
      else return(0)
   else return(0)
else return(0)
```

Algorithm 3: ExternalSocialUtility(j,t,tNext)

We have implemented reciprocal social waiters in our virtual bar. Their actuation is governed by a finite state machine similar to that shown in 3 but changing the winner determination process to include social utilities. Algorithms 2 and 3 implement the internal and external social utility functions respectively. The function NEAR computes the distance between the agents while they are executing a pair of tasks. Therefore, these functions evaluate social interest as the chance to meet a friend in the near future.

4 Results

In order to test our approach, we have animated the virtual university bar example with up to 10 agents serving 100 orders placed by different customers in a situation equivalent to breakfast time (i.e. $P_{coffee} = 0.4$, $P_{juice} = 0.15$, $P_{sandwich} = 0.2$, $P_{pastry} = 0.05$, $P_{drink} = 0.15$, $P_{snack} = 0.05$). For space reasons, we focus on the results obtained for the waiter agents previously presented.

We measure group efficiency through a *Througput* value defined as the ratio between the optimal simulation time (T^*_{sim}) and the real simulation time (T_{sim}):

$$Throughput = \frac{T^*_{sim}}{T_{sim}} \quad (6)$$

Througput is an indicator in the range $[0, 1]$ that estimates how close a simulation is to the ideal situation in which the workload can be divided by the agents and no collisions arise. The simulation time for this hypothetical situation is estimated using T^*_{sim} as follows:

$$T^*_{sim} = \frac{N_{tasks} * \overline{T_{task}}}{N_{agents}} \quad (7)$$

, where N_{tasks} is the total number of tasks being animated, $\overline{T_{task}}$ is the mean time needed to complete a task in a single-agent simulation and N_{agents} is the number of agents.

Figure 4 compares the *Throughput* obtained by different types of waiters in our application example. As expected, self-interested agents do not perform well when the number of agents grow. Even though agent goals are compatible, self-interested agents collide as they do not cooperate but compete for the use of the resources. These collisions produce high waiting times and low quality animations. We enhance this low performance result introducing socially rational mechanisms in the agent decision cycle. As it can be observed, elitist agents (*Sociability* = 0) perform better, since resource waiting times are reduced due to the exchange of tasks with agents that can carry them out in parallel. Nevertheless, they produce unrealistic outcomes since they are continuously working if they have the chance, leaving aside their social relationships. To solve this problem, reciprocal social agents can use the *Sociability* factor to balance rationality and sociability. These kinds of agents spend some time on their social interactions (e.g chats between friends), so it is normal to observe a decrease in their efficiency. Therefore, the *Throughput* value for the sort of animations we are pursuing should be placed somewhere in between elitist and fully reciprocal agents (*Sociability* = 1).

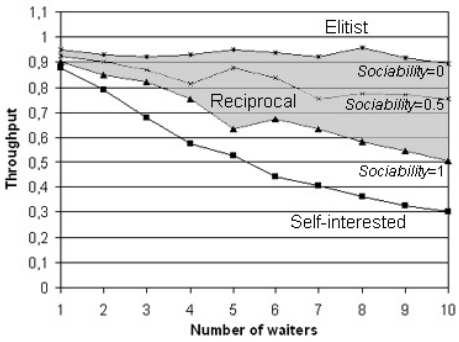

Figure 4: Efficiency obtained by reciprocal social waiters.

Throughput is an estimator for the behavioral performance but, despite being a basic requirement when animating groups of virtual characters, it is not the only criterion to evaluate when we try to create high quality animations. A function to measure the quality of a simulation has to take into account also the amount of time spent on animating social interactions. According to this, we define the

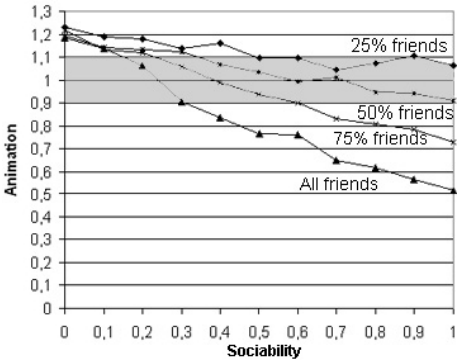

Figure 5: Animation obtained by reciprocal social waiters.

following animation function:

$$Animation = \frac{T^*_{sim} + T_{social}}{T_{sim}} \qquad (8)$$

, where T_{social} represents the time devoted to chat and to animate social agreements between friends. In our virtual bar we have empirically estimated T_{social} as the 35% of T^*_{sim}. Figure 5 shows the animation values for 10 reciprocal social waiters with 4 degrees of friendship: all friends, 75% of the agents are friends, half of the agents are friends and only 25% of the agents are friends. As we have already mentioned, low values of *Sociability* produce low quality simulations since the values obtained for the animation function are greater than the reference value ($Animation^* = 1$). On the other hand, high values of *Sociability* also lead to low quality simulations, specially when the degree of friendship is high. In these cases, the number of social conversations being animated is too high to be realistic and animation is far from $Animation^*$. The animation function can be used to extract the adequate range of values for the *Sociability* factor, depending on the situation being simulated. For example, in our virtual bar we consider as good quality animations those which fall inside $\pm 10\%$ of $Animation^*$ (see shadowed zone in figure 5). Hence, when all the waiters are friends, good animations emerge when $Sociability \in (0.1, 0.3)$.

Finally, table 1 allows to evaluate the effect of our social mechanism over the actuation of each particular agent. This table shows the amount of time devoted to execute each type of task in executions with 10 socially rational waiters ($Sociability = 0$) and 10 fully reciprocal social waiters ($Sociability = 1$). The irregular values in the columns T_{use} and T_{give} on the left side of the table demonstrate how some socially rational agents have specialized in certain tasks. For example, agents 2, 5, 9 and 10 spend most of their time giving products to the customers at the bar. On the other hand, agents 3 and 7 are mainly devoted to use the resources of the bar (e.g. coffee machine, etc) to get the products ordered. Although specialization is a desirable outcome in many multiagent systems, egalitarian human societies need also to balance the workload assigned to each agent. On the bottom part of the table, fully reciprocal social waiters achieve an equilibrium between the time they are giving products and the time they are using the resources of the environment (see balance column, *Bal*). A collateral effect of this equilibrium is the increase in the waiting times, since social agents will sometimes prefer to meet his friends in a resource than to reallocate the task (compare columns T_{wait}). As a consequence, a new percentage of the execution time appears (T_{chat}) within which agents can animate pure social interactions (e.g. chats between waiters that are friends).

5 Conclusions and future work

The animation of groups of intelligent characters is a current topic with a great number of behavioral problems to be tackled. We aim at incorporating human style social reasoning in character animation. Therefore, this paper presents a technique to properly balance social with task-oriented plans in order to produce realistic social animations. The multiagent animation framework presented allows the definition of different types of social agents: from elitist agents (that only use their interactions to increase the global performance of the group) to fully reciprocal agents. These

$Sociability = 0$					
Ag	T_{wait}	T_{chat}	T_{use}	T_{give}	Bal
1	0	0	32	19	-6
2	3	0	4	26	-3
3	14	0	52	1	28
4	3	0	16	28	-3
5	0	0	7	30	-16
6	3	0	37	17	-1
7	0	0	67	4	21
8	0	0	45	17	1
9	7	0	5	23	-11
10	1	0	6	41	-10
$Sociability = 0$					
1	16	36	69	34	-2
2	18	62	58	24	-2
3	41	66	45	16	0
4	48	61	60	27	3
5	34	68	58	12	-1
6	48	74	64	14	-2
7	18	66	48	24	1
8	33	76	45	24	4
9	46	58	36	21	0
1	27	69	56	20	-1

Table 1: Time distribution for reciprocal social waiters (time values are in seconds)

latter agents extend the theory of social welfare with a reciprocity model that allow controlling the emergence of unethical agent specialization and promote social interactions between the members of a group.

Ongoing work is being done to provide the agents with mechanisms to self-regulate their *Sociability* factor depending on their social relations (e.g. degree of friendship) and on their previous intervention (e.g. amount of time already devoted to chat). Thus, agents will be able to dynamically adjust to the situation in order to stay between the boundaries of good quality animations at any time.

References

[1] R. H. Bordini, A. C. da Rocha, J. F. Hübner, A. F. Moreira, F. Y. Okuyama and R. Vieira. MAS-SOC: a Social Simulation Platform Based on Agent-Oriented Programming. *Journal of Artificial Societies and Social Simulation*, vol. 8, no. 3, 2005.

[2] R. H. Bordini and J. F. Hübner. Jason, 6th of March 2007. http://jason.sourceforge.net/.

[3] 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: Autonomous Agents and Multiagent Sistems*. ACM, 2003.

[4] J. A. Torres, L. P. Nedel, and R. H. Bordini. Using the bdi architecture to produce autonomous characters in virtual worlds. In *IVA'03: International Conference on Intelligent Virtual Agents*. Springer, 2003.

[5] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of MAAMAW'96*, LNAI 1038, pages 42–55, 1996.

[6] M. Brenner. A multiagent planning language. In *Proceedings of ICAPS'03 Workshop on PDDL.*, 2003.

[7] K. S. Decker and V. R. Lesser. Designing a family of coordination algorithms. *Readings in Agents; Huhns and Singh editors*, 1997.

[8] J. A. Giampapa and K. Sycara. Team-Oriented Agent Coordination in the RETSINA Multi-Agent System. On *Tech. Report CMU-RI-TR-02-34*, Robotics Institute-Carnegie Mellon University, 2002.

[9] F. Grimaldo, M. Lozano, and F. Barber. Integrating social skills in task-oriented 3D IVA. In *IVA'05: International Conference on Intelligent Virtual Agents*. Springer, 2005.

[10] L. M. Hogg and N. Jennings. Socially intelligent reasoning for autonomous agents. *IEEE Transactions on System Man and Cybernetics*, 31(5), 2001.