Towards the Group Formation through Social Norms

Daniel Villatoro and Jordi Sabater-Mir

Artificial Intelligence Research Institute (IIIA)
Spanish Scientific Research Council (CSIC)
Bellatera, Barcelona, Spain
{dvillatoro,jsabater}@iiia.csic.es

Abstract. This paper examines the decentralized formation of groups within a multiagent normative society. In our case, a group is defined based on the set of social norms used by its members: all the agents using the same set of norms belong to the same social group. In this paper we explore different mechanisms that allow agents to recognize the others as members of a certain social group. Considering as the basic mechanism the one that makes agents interact with other agents without considering the previous interactions and with no communication, three new algorithms have been developed and tested to improve the efficiency of the basic one. These algorithms are: (1) the *whitelisting* algorithm, which works as a recomender of trusted neighbours; (2) the *blacklisting* algorithm, whose basic functioning is based on defaming the non-related agents inside a certain social group; and (3) the *labelling* algorithm, which basically publishes information of the interactions with different agents allowing the rest to access that information. Simulation results are shown, confirming that these algorithms improve the efficiency of the basic one. Finally, we present and discuss some of the weak points of the algorithms presented as well as future improvements.

1 Introduction and Related Work

Social norms are part of our everyday life, and they have been of interest in several areas of research [3]. They help people self-organize in many situations where having an authority representative is not feasible. On the contrary to institutional rules, the responsibility to enforce social norms is not the task of a central authority but a task of each member of the society. From the book of Bicchieri [2], the following definition of social norms is extracted: "The social norms I am talking about are not the formal, prescriptive or proscriptive rules designed, imposed, and enforced by an exogenous authority through the administration of selective incentives. I rather discuss informal norms that emerge through the decentralized interaction of agents within a collective and are not imposed or designed by an authority". Social norms are used in human societies as a mechanism to improve the behavior of the individuals in those societies without relying on a centralized and omnipresent authority. In recent years, the

use of these kinds of norms has been considered also as a mechanism to regulate virtual societies and specifically societies formed by artificial agents ([8], [9], [4]). *The main objective of this research is to analyze the process of group formation around a common set of social norms and which algorithms make this process more efficient and faster.* Several researchers have already covered the problem of group formation. One seminal article is the work of Hales presenting the SLACER algorithm [5]. In this algorithm given a network of agents, when an agent finds another agent in a better situation than itself, it copies its strategy and neighbours. This rewiring algorithm produces an emergent behavior similar to the one we try to obtain by forming groups. Another interesting work is the one presented in [1], where the authors try to answer the question "to whom should agents connect to?", by experimenting on different agents' networks structures, to solve optimization problem. Finally, and also a technique used by Hales [6], is the "tagging mechanism", initially presented by Holland [7]. Tags are markings or social cues attached to individuals (agents) and observable by others. Agents maintain and modify tags on themselves and a team is formed by only collaborating with agents with the same tag.
Unlike these works on group formation, we are facing a decentralized environment where agents notice partially the characteristics that define the other agents group. This is the main reason for the necessity of the algorithms developed in this work.

2 Statement of the Problem

Social norms provide multiagent societies with a decentralized control mechanism. By giving agents with a set of social norms to use in the environment where they are located, they can self-regulate it and control undesired behaviors. But the definition of social norm that we have used in previous sections does not expressly include one vital aspect of the social norm, that is, the *coordinated reaction against outsiders*. The coordinated reaction from the group is twofold: (1) a *coordinated punishment* from the group against the outsider who explicitly does not abide by the same social norms; and (2) an *integration mechanism* to force this outsider to change its actual behavior to the one accepted by the group. Therefore agents need to have a notion of social group. The notion of belonging to a social group will be determined by the agent's behaviour. The behaviour of an agent is determined by the social norms it uses. In this way, a social group will be only formed by agents who share social norms [1]. Once agents possess this notion of social group they might be able to coordinate according to the situation in which they meet outsiders. Although we will still not cover the integration or punishment mechanisms, in this work we will *introduce different algorithms for the group formation process and contrast their efficiencies in different situations.* These algorithms will reduce the number of false positive members of the group by taking advantage of the social information.

[1] The configuration of social norms among agents is stationary. Emergence, adoption or retraction of norms is not covered in this work

In order to prove these algorithms we have developed a multi-agent based simulation that reproduces the following behavior: agents are distributed and initially do not know each other. Agents survive by consuming resources and these resources are obtained in two ways: finding them randomly in the environment, or, receiving them from other agents. When one agent decides to donate some of its resources (or energy) it means that this agent is loosing some of its own resources to give them to another agent. The set of social norms tell the agents under which situations (depending on both agents' states) they will donate energy to another agents and in which they will not. The task of the algorithms developed is detecting the norms that each agent follows, determining in this way to which social group each agent belongs to.

3 Simulation Model

The simulation algorithm is based on a discrete step timing model, where in each time step the algorithm observes the state and consequent actions of each agent before ticking another time step. Every time step, the simulation algorithm runs over every agent. The order in which the algorithm runs over the agents is randomly changed each time step.
Initially, in each time step, the algorithm evaluates (following a continuous uniform probability distibution) if every agent has to find resources by observing the agent *Resource Gathering Probability* ($P_{rg} \in [0, 1]$). P_{rg} specifies the probability an agent has to find resources each time step. In case the algorithm evaluates that an agent has to find resources, the agent will receive a large amount of resources that can either be used for its own consumption or for donating.
After that, the algorithm evaluates if an agent has to meet another agent by observing the agent *Interaction Probability* ($P_{int} \in [0, 1]$). P_{int} specifies the probability of an agent to meet another agent present in the simulation. As we will see, this process of mate selection will be slightly modified later in the presented algorithms.
Agents are initially loaded in the simulation platform with 100 resource units, and each time step, agents consume one resource unit and energy consumption. When an agent's resources are exhausted, the agent is not able to interact with any other agent and no agent can interact with it, remaining inactive until it finds resources.
The interactions among agents are done always in pairs, and both agents have to choose an action when interacting. This decision is taken following the *set of social norms* that each agent has internalized. The set of norms specifies if the agent has to give or not to give resources to the other agent, depending on both agent's internal resource levels. In order to formalize our concept of social norm, we first need to define several terms.
All agents can perceive a finite set of *observables* (*ob*) $\mathcal{O}$. Every agent also has a finite set of *actions* (*a*) A.
Every agent can find itself in a finite set of different *situations* (*sit*) $\mathcal{S}$. In other

words, a *situation* is a combination of different observables.
Given that, a **social norm** SN_i is a tuple formed by a situation and an action: $SN_i = \{\langle sit_g, a_h \rangle \mid sit_g \in \mathcal{S}, a_h \in A\}$.
In our scenario, the set of observables is formed by the following propositional terms: $\mathcal{O} = \{$ *Plenty(Me)*, *Plenty(You)*, *Normal(Me)*, *Normal(You)*, *Starving(Me)*, *Starving(You)*$\}$, where: *Plenty(X)* indicates that *Agent's X* resource level is over 100 units; *Normal(X)* indicates that *Agent's X* resource level is between 25 and 100 units; and, *Starving(X)* indicates that *Agent's X* resource level is below 25 units. The values that X can take are *Me* and *You*, representing the acting agent and the partner agent in the interaction. When two agents meet, each agent is able to observe its own level of resources and its partner level. The whole list of possible situations (formed by two observables) is detailed in Table 1. The set of possible actions is $A = \{\text{Give Resources}, \text{Do not Give Resources}\}$. The combination of all possible situations, each one associated to a concrete action, generates a **set of social norms**. The use of one set or another determines the behavior of an agent in the environment and its social identity. Therefore, in our scenario, two agents that use the same set of social norms (that is, behave exactly the same in front of each possible situation) are said to belong to the same social group.

Situation		**Action**
Starving(Me)	Starving(You)	To Give / Not To Give
Starving(Me)	Plenty(You)	To Give / Not To Give
Starving(Me)	Normal(You)	To Give / Not To Give
Plenty(Me)	Starving(You)	To Give / Not To Give
Plenty(Me)	Plenty(You)	To Give / Not To Give
Plenty(Me)	Normal(You)	To Give / Not To Give
Normal(Me)	Starving(You)	To Give / Not To Give
Normal(Me)	Plenty(You)	To Give / Not To Give
Normal(Me)	Normal(You)	To Give / Not To Give

Table 1. *Situations and Actions. Structure of a set of social norms.*

As we said before, in our scenario the situations are specified by the state of two interacting agents and therefore some situations are more likely to occur than others. For example, in a very rich resource environment, it will be highly unfrequent to find a couple of *Starving* agents. Due to this, a *Friendship Factor* is assigned to agents. This factor determines the minimum percentage of social norms that one agent has to share with another (before knowing how that agent behaves in the rest of situations) so the other agent can be considered as a candidate to exchange information with it. For example: a *Friendship Factor* of 0.2 means that one agent needs to share at least 20% of the norms, without taking into consideration the rest 80% of the norms, even if they do not know them, as long as those ones that they know are equal.

3.1 Basic Algorithm

The simplest mechanism of group formation in the previously described scenario is allowing agents to interact among them and make themselves keeping record of how socially related they are. This basic algorithm is represented in Algorithm 1.

```
repeat
    foreach Agent i do
        Randomly Meet Agent j;
        Interact following the set of social norms;
        Save information of partner behavior;
    end
until Exhausting Timesteps ;
```

Algorithm 1: Basic Group Formation Algorithm

This algorithm makes each agent interacting with the rest of the society without any preference or intelligence in the partner selection process. During the simulation and after each interaction, the agent observes the actions taken by its partners (that follow their own set of social norms) in different situations. This allows agents to determine if those partners can belong to its social group.
In the following sections we will define new algorithms and functionalities that improve the behavior of this basic algorithm. These new algorithms take advantage of the social information that agents are gathering during the process and that can share with other agents through a communication protocol.

3.2 Basic Functions

All the algorithms that will be presented from now on share some common functionalities that provide agents with more intelligence during the group formation process. In this section we describe these functionalities.

Similarity Evaluation Several times we have used the term "socially related agent", although it has not been explained yet how this relation is established and evaluated. In our scenario, agents evaluate their similarity based on the number of norms that they share. As it was explained in Section 2, the main problem that this approach presents is due to the nature of the social norms. The social norms will be used under determined situations that are defined by the state of both interacting agents (for example: *Plenty(Me)* and *Starving(You)*). Therefore there might be more or less frequent norms (in a resource-rich environment, it will be very rare to find the situation *Starving(Me)* and *Starving(You)*). As we explained previously, agents are given with a *Friendship Factor* and with a function to study the degree of similarity between two agents. Agents also possess a social memory, where they can remember the norms that other agents have used. Then, after each interaction with another agent, each agent will save in its

memory the other agent's norm and update the similarity with it. Therefore this algorithm basically counts the number of known norms that two agents share. In case only one norm is detected to be different, the similarity will be -1.

Similarity = 0; **foreach** *Known Situation of Agent j* **do**
  if *Action taken by Agent j == Action Agent i would have taken in Agent j Situation* **then**
    Similarity + = $\frac{1}{NumberofSituations}$;
  end
  else
    Similarity = −1 ;
  end
end

Algorithm 2: Similarity Evaluation

For example, if two agents have met previously four times, in four different situations (out of the 9 different possible situations that can occur in our scenario) each time, and they have used the same norms, these two agents will have a Similarity = $\frac{4}{9}$. On the other hand, if in a fifth interaction, in a different situation, they take different norms, the similarity will be updated to -1.

Intelligent Partner Selection The mechanism in charge of making a more intelligent partner salection for an interaction is shown in Algorithm 3. In section 3, it was introduced that agents have an *Interaction Probability* assigned to them to meet random agents present in the simulation. This algorithm modifies slightly that parameter, making that agents interact more frequently with another agents with a positive degree of similarity. In this way, agents will promote interactions with other agents socially closer to themselves.

Probability To Meet Random Agent = 1 - $\frac{NumberofKnownAgents}{TotalNumberofAgents}$;
if *Probability To Meet Random Agent ¿ Random Number* **then**
  Meet Known Agent;
end
else
  Meet Random Agent;
end

Algorithm 3: Intelligent Partner Selection.

3.3 Whitelisting Algorithm

The *Whitelisting Algorithm* is based on the idea of recommending known trusted partners to your friends. When an agent discovers a new trusted partner (that is, the *degree of similarity* with that agent has gone above the *friendship factor*), it will recommend this agent to its other friends as a possible partner to interact with. These other agents, if they do not know this recommended agent, will add it to their preference list of agents to interact with in order to confirm this similarity. The algorithm is shown in Algorithm 4.

```
repeat
    foreach Agent i do
        Receive Recommendations;
        if Any Recommendation then
            Agent j = Any of the Recommended;
        end
        else
            Agent j = Intelligent Partner Selection;
        end
        if Similarity with Agent j > 0 OR Agent Unknown then
            Interact following the set of social norms;
            Save information of partner behavior;
            Recalculate Similarity with Agent j;
            if Similarity with Agent j > FRIENDSHIP FACTOR then
                Select M Known Agents with Similarity > FRIENDSHIP FACTOR ;
                Inform Agent j about these M Known Agents ;
            end
        end
        else
            Inform about the action the norms dictate, but Do Not Give;
        end
    end
until Exhausting Timesteps ;
```

Algorithm 4: Whitelisting Group Formation Algorithm

3.4 Blacklisting Algorithm

In this case, unlike the whitelisting algorithm, the idea is to inform about unsatisfactory interactions the rest of agents in the social group. Once an agent is detected as a non-member of the social group, this information is transmitted to the other members of the group so they can avoid that agent in the future. The algorithm is shown in Algorithm 5.

```
repeat
    foreach Agent i do
        Agent j = Intelligent Partner Selection;
        if Similarity with Agent j > 0 OR Agent Unknown then
            Interact following the set of social norms;
            Save information of partner behavior;
            Recalculate Similarity with Agent j;
            if Similarity with Agent j == −1 then
                Select M Known Agents with Similarity > FRIENDSHIP FACTOR ;
                Inform these M Known Agents about Agent j;
            end
        end
        else
            Inform about the action the norms dictate, but Do Not Give;
        end
    end
until Exhausting Timesteps ;
```

Algorithm 5: Blacklisting Group Formation Algorithm

3.5 Labelling Algorithm

Another algorithm implemented to solve the problem of group recognition is the *Labelling Algorithm*. The main idea here is that agents are able to assign "labels" to other agents that can be accessed by a partner when two agents interact. Each agent carries the labels that others assign to it. The content of these labels is: (1) the identity of the agent with whom it interacted, (2) the situation in which they interacted and (3) the result of the interaction (*true* value in case the norm followed was the same for both agents, and *false* value otherwise). It is straightforward to see that the power of this algorithm is based on publishing and making accesible to all agents previous interactions with different agents.

Thus, when one agent interacts with another, this agent is not only provided with the information it obtained from direct experiences with the other one or communicated experiences from members of its group, but also indirectly (and through the labels) from all the agents that have interacted with other agent. Consequently, after evaluating the result of the interaction, one agent can also check other agent with similar experiences to that other agent, and use that information. The algorithm can be seen in Algorithm 6.

```
repeat
    foreach Agent i do
        Agent j = Intelligent Partner Selection;
        if Similarity with Agent j > 0 OR Agent Unknown then
            Interact following the set of social norms;
            Save information of partner behavior;
            if Action Agent j == Action Agent i would have taken in Agent j Situation then
                Add label(Agent i ID, Agent j Situation, true) to Agent j ;
                IDs of Agents to Veto = Obtain IDs from Agent's j Labels with (Agent j Situation, false);
            end
            if Action Agent j != Action Agent i would have taken in Agent j Situation then
                Add label(Agent i ID, Agent j Situation, false) to Agent j ;
                IDs of Agents to Veto = Obtain IDs from Agent's j Labels with (Agent j Situation, true);
            end
            foreach Agents to Veto do
                Save information from the label;
                Recalculate Similarity with another Agent;
            end
        end
        else
            Inform about the action the norms dictate, but Do Not Give;
        end
    end
until Exhausting Timesteps ;
```

Algorithm 6: Labelling Group Formation Algorithm

4 Experiments and Results

The purpose of the following experiments is to analyse to which extend the different presented algorithms improve the behavior of the basic one, allowing agents to recognize members of their respective groups. Each experiment will vary by a combination of: (1) the number of groups present in the society, (2) the number of norms differing from each group, (3) the interaction probability, and, (4) the friendship factor (these last two presented on Sec. 3). To observe the efficiency of the algorithms during the process of group formation, we count the number of false positives agents that each agent has. The definition of a false positive agent is an agent that another agent believes to be in the same social group that it is (due to the incomplete information) but it actually is not. For example: two agents sharing all the norms except one, if these two agents have met in the other similar situations and not in the different one, they will believe that the other agent is in its social group while it is not, therefore, they will be false positives agents one to each other.

4.1 Experiment Design

We load the simulation of a society with the following characteristics:

- Number of agents = 100.
- An agent can interact with all of the rest.
- All agents have the same Interaction Probability that will be fixed depending on the experiment.
- All agents have the same Resource Gathering Probability fixed to ($P_{RG}(Agent_i) = 0.0025$). The amount of resources found is also fixed to 250 units. With these values we simulate an environment where resources are difficult to find although when they are found, they appear in a huge amount.
- Agents are not able to change their set of social norms.

Each simulation is run for 10 times, during 10000 steps, and then averaged the results.

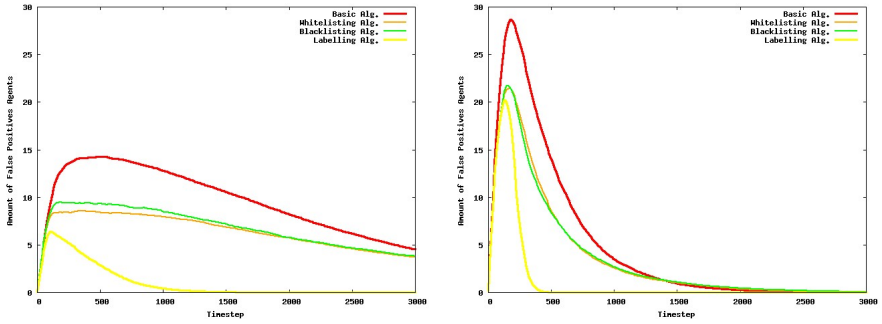
4.2 Two Heterogeneous Groups

This experimental setting has been designed to prove the correctness of the experimental platform. The society is divided into two groups, and the number of norms that both groups share is zero. Therefore, after a single interaction, two agents can detect the other as a non-member of their group.
As expected, the simulation results show that with the four algorithms, no agent ever has a false positive friend.
One remarkable point in the efficiency of our algorithms (that is common also for the rest of experiments) corresponds to the nature of the norms. The social norms in our scenario indicates whom *To Give* against whom *Not To Give*, and agents always abide by their social norms. Therefore agents initially loaded with altruist norms (where most of the norms indicate *To Give*) are in a clear disadvantage respect to the others. As they will donate more resources, they are prone to "die", and consequently loosing ability of interaction until they find resources again. So, the results on the group formation algorithms are biased by this effect of the nature of social norms, slowering its efficiency. In case these effects of the social norms would not take place, more interactions would be occuring, and group formation would happen faster.

4.3 Two Homogeneous Groups

In order to increase the difficulty of the scenario for the algorithm to solve the group formation problem, the homogeneity between groups is increased. In this way we can prove that the algorithms work in this kind of incomplete information scenarios, where the recognition of an agent from another group might not depend only on one interaction.
In this experimental setting we still split the society into two equally populated groups. This time the two groups differ in 3 norms (out of a total of 9). Therefore agents will need a certain number of interactions (using the basic algorithm) until reducing the number of false positives. The results for the simulations with an Interaction Probability of 0.1 and 0.3 are shown in Fig. 1.

We observe the same pattern in both experiments. As we can see, the labelling algorithm outperforms the other three both in number of false positives and in



(a) 0.1 Interaction Prob. (b) 0.3 Interaction Prob.

Fig. 1. Two Homogeneous Groups.

speed of convergence (anulling the number of false positives). We can also observe that both the *Whitelisting* and the *Blacklisting* algorithms show a similar performance between them. The similarity in the efficiency of these two algorithms is due to the *Friendship Factor*. For a deeper analysis of the influence of the *Friendship Factor* in these two algorithms see Sec. 5.
Another point to remark is the effect of the *Interaction Probability*. In this experiment we observe only a difference in the speed of convergence, making a society with members that they interact more frequently, converges in a faster way. We can also observe that a society with a higher *Interaction Probability* factor presents a higher number of false positive friends, and this is due to the number of interactions. The more an agent interacts, the more agents it meets, and the possibility of having a higher number of false positive friends increases. Depending on the type of scenario, a higher interaction probability can be a drawback. In this case it is due to the nature of the scenario, and its resource exchanging procedures. The fact of interacting more, implies for some agents (those whose norms specify so), donating more energy, which eventually implies the "death" of the agent. As we explained in Sec. 3, when an agent "dies", it remains inactive in the simulation (it is not possible to interact with it) until it finds resources.

4.4 Four Heterogeneous Groups

After proving the efficiency of the algorithms splitting two groups, their behavior will be tested in a more difficult scenario. We will introduce now four different groups to recognize. In this experimental setting we split the society into 4 equally populated groups.

1. The social norms of the first group imply *Not To Give* in all the situations.
2. The social norms of the second group imply *To Give* in all the situations.

3. The social norms of the third group imply *Not To Give* in all the situations, except 4.
4. The social norms of the third group imply *To Give* in all the situations, except 4.

Therefore, agents will find very easy to identify the agents from the other groups. In Figure 2(a) we can observe the same pattern of results obtained in the previous experiments. Although convergence is slower due to the higher number of groups, all the algorithms perform in a similar way as in the previous experiments.

4.5 Four Homogeneous Groups

Finally, and in order to prove the efficiency of the algorithms, a more difficult scenario is designed. This time the society will be split into 4 groups, but more homogeneous than in the previous experiment. We split the society into 4 equally populated groups:

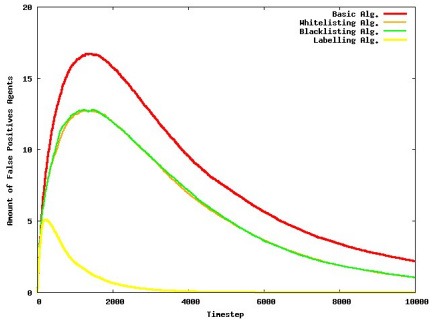
1. The social norms of the first group imply *Not To Give* in all the situations.
2. The social norms of the second group imply *To Give* in all the situations.
3. The social norms of the third group imply *Not To Give* in all the situations, except 2.
4. The social norms of the third group imply *To Give* in all the situations, except 2.

We consider this scenario a very difficult situation where to detect false positives, because the two norms where they differ are very infrequent.

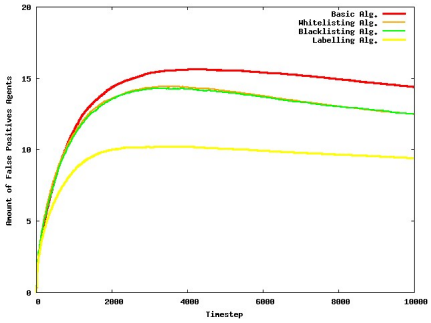
Simulation results shown in Fig. 2(b) and Fig. 2(c) confirm the difficulty of this scenario. We can see that agents (in both situations) still have a relatively high number of false positives (meaning that almost a whole group has not been detected as a false positive). Notwithstanding the slow convergence of the algorithms, we still observe the same pattern of results. We would like to remark again that the slow convergence of all the algorithms is due to the low number of different norms between groups, and the number of times that these norms are used (due to the low probability of that situation to take place).

5 Discussion on the Whitelisting Algorithm

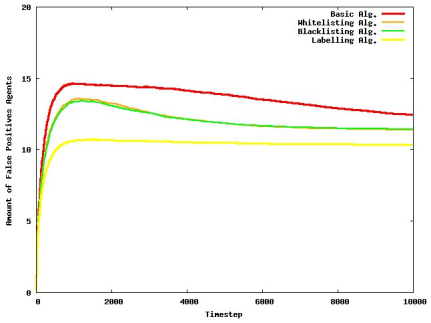
The efficiency of this algorithm depends on two factors: (1) the *Friendship factor*, and (2) the current cohesion of the group. Using the whitelisting algorithm with a low *Friendship Factor* implies a bad performance of it. For example, one agent can find another agent (socially different but with some common norms) and think that they belong to the same social group. Thus we will have here a false positive that it will also be recommended to other agents considered friends (due to the *friendship factor*). Consequently a low *Friendship Factor* might produce agents to send false information to wrong agents. On the other hand, a high *Friendship Factor* will make that the algorithm behave as the basic one (improved with the *Intelligent Partner Selection*) and



(a) Heterogeneous Groups. 0.1 Interaction Prob.



(b) Homogeneous Groups. 0.1 Interaction Prob.



(c) Homogeneous Groups. 0.3 Interaction Prob.

Fig. 2. Four Groups.

agents will not start sending recommendations (which is the main power of this algorithm) until the *Degree of Similarity* equals at least to the *Friendship Factor*.

Closely related to this, we have the hypotheses that this algorithm will show a good performance when used in an environment where groups are already formed (the situation where newcomers want to join a group). Even in cases where the cohesion of the group is very low, we suspect that we need a more flexible *Similarity function* than the one we currently use. The current similarity function only considers two agents to be equal if all the social norms they use are exactly equal, but we believe that more flexible functions would make the algorithms to behave better. As noticed beforehand, there are some norms that are used more frequently than others. For example, it might happen than some norms (and therefore the situations that the norms regulate) only happen once in 1000 situations. Therefore, expecting every agent in the society to detect all the other agents in the society in that rare situation seems unfeasible. Hence, a

more flexible similarity function would be the one that evaluates two agents to be equal if 80% of the most used norms are the same.

6 Discussion on the Blacklisting Algorithm

Similarly to the Whitelisting algorithm, this algorithm presents symmetric problems. In this algorithm, when the *Friendship Factor* is very low, agents will send information possibly to wrong agents. Agents using the blacklisting algorithm will not send the information until the similarity with the other agents is above the *Friendship Factor*. On the other hand, when the *Friendship Factor* is very high, agents will not take advantage of the algorithm until the similarity is reached.
We suspect that in the situations where there is a large cohesion between the members of the group the performance of the algorithm would be accetable, because the degree of similarity between agents of the group will already be very high. On the other hand, on scenarios with low cohesion, we need, as we said in Sec. 5, a more flexible *Similarity function* than the one we currently use.

7 Discussion on the Labelling Algorithm

After showing a good performance in the experimental section, this algorithm obtains its efficiency at the high cost of making a lot of information public and available to others. Unlike the other algorithms (where the information was always coming from a trusted third party), in this algorithm we take advantage not only of the information from the agent's social group, but, from past interactions with other agents. This implies a cost for the agents, that is a loss in their privacy of the interactions. We have made an analysis of the possible "attacks" this algorithm could suffer:

- **Delete all labels**: In a situation in which all the agents would decide to delete all their labels, the labelling algorithm would still behave better than the random one due to the *Intelligent Partner Selection*.
- **Self-adding false labels**: Agents can decide to assign false labels to themselves, so they can create the image of belonging to a group different than the one they actually belong to. But, we can still distingish two situations:

  1. **Self-adding false labels from an existing agent in the society, but with a different value than the real one**: The objective of doing this would be to make others believe that this agent belongs (or not) to the same group than the one whose identity has been impersonated. Cryptography provide us with solutions (such as digital signatures) to avoid this problem.
  2. **Self-adding false labels from an invented agent in the society**: This attack would be done to create a sense of "popularity", although in the algorithms we are using now, this societal aspect is not taken into account. Therefore this kind of attacks in the current algorithms would reduce the

speed of convergence of this algorithm as this useless information would be taking the space of real information in the labelling area of the agents.

– **Delete selected labels**: This might be the most common attack. Certain agents might decide to delete information of the group they do (or do not) belong to. Therefore we analyze these two simmetric situations:
  1. **Delete *False* labels**: By doing that an agent will maintain public the labels from members that belong to its group, deleting those of who do not belong to their group. Under this attack, the algorithm would still work better than the random one.
  2. **Delete *True* labels**: It is the simmetric situation with respect to the previous one; but less precise information will be given. In this attack agents will maintain public the information of those who are not in the same group as they are, but leaving still unrecognizable the group they belong to.

These series of attacks have not been tested yet in our simulation platform, but they will be added as one of the points in the future work section.

8 Conclusions and Future Work

These preliminary results give us some initial insights about the efficiency of the algorithms solving the group formation problem. As we can see, the algorithms behave similarly in different situations.
The *friendship factor*, together with the rigid evaluation of the similarity between agents, are the parameters that control the behavior of the whitelisting and the blacklisting algorithms. The *friendship factor* determines when the whitelisting and blacklisting algorithms start working. While this threshold is not reached, the behavior showed is that of the basic algorithm. Once the threshold is overpassed both algorithms start working and improving the basic one by initiating the transmission of information to other agents. Another important problem is that our example brings some consequences due to the nature of the norms. The norms used indicate whom to give and not to give, therefore, some groups will have a preference over the others; for example, in a scarce-resource environment, altruist agents (with a set of norms that force them to always share) mixed with a population of totally selfish agents will be in a harder situation. The fact of donating implies altruist agents loosing energy, and eventually "dying". Consequently loosing oportunities to detect the people in their group, and the false positives. One of the main conclussions that we can extract from the simulations is that the similarity evaluation function used is too rigid and strict and somehow not very realistic. Norms are attached to situations, and situations depend on the enviromental condition (in our case the resource gathering probability will determine the most frequent situations), and it is reasonable to think that agents can determine which norms are the most used or more important for themselves. Therefore we will introduce a more dynamic concept of similarity; for example, based on the number of times that a situation occurs in a given simulation. Adding this dinamicity we will allow agents to control in a more flexible

way the concept of social group, even though not sharing exactly the same set of social norms. Till now we have been always considered societies of strangers that initially have no information about the other members and have to start forming the groups. We want to study also how the algorithms behave in societies where the groups are already formed and use the algorithms for introducing newcomers into the group. We suspect that algorithms like whitelisting or blacklisting will considerably outperform the basic one, because the *friendship factor* will have been already reached with those that belong to your group, consequently avoiding the basic execution part of the algorithm, and anulling the transmission of false positives.

9 Acknowledgments

This work was supported by the European Community under the FP6 programme [eRep project CIT5-028575, OpenKnowledge project FP6-027253]; the Spanish Education and Science Ministry [AEI project TIN2006-15662-C02-01, AT project CONSOLIDER CSD2007-0022, INGENIO 2010]; Proyecto Intramural de Frontera MacNorms [PIFCOO-08-00017] and the Generalitat de Catalunya [2005-SGR-00093]. This work was partially supported by the Santa Fe Institute. Daniel Villatoro is supported by a CSIC predoctoral fellowship under JAE program.

References

1. Ricardo Araujo and Luis Lamb. Memetic networks: Analyzing the effects of network propoerties in multi-agent performance. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008.
2. Cristina Bicchieri. *The Grammar of Society: The nature and Dynamics of Social Norms*. Cambridge University Press, 2006.
3. Jon Elster. Social norms and economic theory. *Journal of Economic Perspectives 3*, 4:99–117, 1989.
4. Amandine Grizard, Laurent Vercouter, Tiberiu Stratulat, and Guillaume Muller. A peer-to-peer normative system to achieve social order. In *Workshop on COIN @ AAMAS' 06*, 2006.
5. David Hales and Stefano Arteconi. Slacer: A self-organizing protocol for coordination in p2p networks. *IEEE Intelligent Systems*, 21:29,35, 2006.
6. David Hales and Bruce Edmonds. Applying a socially-inspired technique (tags) to improve cooperation in p2p networks. *EEE Transactions in Systems, Man and Cybernetics - Part A: Systems and Humans, 35*, 3:385–395, 2005.
7. John Holland. The effects of labels (tags) on social interactions. *Working Paper Santa Fe Institute*, 93-10-064, 1993.
8. Nicole J. Saam and Andreas Harrer. Simulating norms, social inequality, and functional change in artificial societies. *Journal of Artificial Societies and Social Simulation*, 2(1), 1999.
9. Yoav Shoham and Moshe Tenneholtz. On the synthesis of useful social laws for artificial agent societies (preliminary report). In *Proceedings of the AAAI Conference*, pages 276–281, 1992.