Inducing Desirable Behaviour through an Incentives Infrastructure*

Roberto Centeno, Holger Billhardt, and Sascha Ossowski
{roberto.centeno, holger.billhardt, sascha.ossowski}@urjc.es

Centre for Intelligent Information Technologies (CETINIA)
University Rey Juan Carlos - Spain

Abstract. In open multiagent systems, where agents may join/leave the system at runtime, participants can be heterogeneous, self-interested and may have been built with different architectures and languages. Therefore, in such a type of systems, we cannot assure that agents populating the system will behave according to the objectives of the system. To address this problem, organisational abstractions, such as roles, groups and norms, have been proposed as a promising solution. Norms are often coupled with penalties and rewards to deter agents from violating the rules of the system. But, what happens if a current population of agents does not care about these penalties/rewards. To deal with this problem, we propose an incentives infrastructure that allows to estimate agents' preferences, and can modify the consequences of actions in a way that agents have incentives to act in a certain manner. Employing this infrastructure, a desirable behaviour can be induced in the agents to fulfil the preferences of the system.

1 Introduction

A particular type of MultiAgent Systems (MAS from now on) are Open MAS. These systems are usually designed from a global perspective and with a general purpose in mind. However, at design time, the agents that will populate the system are unknown and the number of them may vary dynamically, due to they can join/leave the system at runtime. Besides, those agents might be heterogeneous, self-interested, built with different architectures and languages, and even built by different people/companies.

With this in mind, designers cannot assume agents will behave according to the preferences of the system. In order to address this problem, organisational structures have been proposed as a promising solution. In these approaches, authors propose the use of organisational abstractions such as roles, norms, groups, etc. [1–5] so as to regulate the activity of the participants. Therefore, the normative systems emerge as a key concept for regulating MAS [1, 2, 4, 5].

However, norms, from the point of view of agents, are just information that tell them what actions they are (not) allowed to perform in the system. Thus, in order to be effective, norms should be coupled with detection mechanisms – to detect when they are

* The present work has been partially funded by the Spanish Ministry of Education and Science under projects TIN2006-14630-C03-02 (FPI grants program) and "Agreement Technologies" (CONSOLIDER CSD2007-0022, INGENIO 2010)

not obeyed – and with penalties/rewards – to be applied when they are violated. In most cases, systems, as well as the norms and their penalties/rewards, are designed before knowing the agents that will populate them. In this sense, the question arises what happens if a current population of agents is not sensitive to these penalties/rewards. Then, norms can not be effectively enforced.

To deal with this problem, *hard norms* can be defined, which agents are not able to violate because the system relies on mechanisms to avoid such violations. For instance, in Electronic Institutions [1], by means of their infrastructure (Ameli [6]), agents are only able to perform actions which are acceptable in the current state. Nevertheless, in some domains the use of this kind of norms is not feasible due to their complexity and size, it could be impossible to take into account all possible exceptions. For example, in the traffic domain there exists a norm that says that it is forbidden to pass trough a solid line. However, it is possible and even desirable that sometimes such a norm could be violated, for instance if you can avoid an accident. Therefore, it is often easier and more efficient to define norms based on penalties/rewards, instead of using hard norms. However, as we said before, these penalties/rewards should be effective for the current population of the system.

Addressing this situation, we propose an incentive mechanism, following the work presented by Centeno et al. in [7] and based on the theory "Economic Analysis of Law", proposed by R.A. Posner in [8]. In this work the author analyses and checks how normative systems avoid the waste of resources and increase the efficiency of societies. Following these ideas we propose an incentives infrastructure that allows to estimate agents' preferences, and can modify the consequences of actions in a way that agents have incentives to act in a certain manner. Employing this infrastructure, a desirable behaviour can be induced in agents to fulfil the preferences of the system.

The rest of the paper is organised as follows, Section 2 provides a formalisation of the model on which we base our work and describes the problem we address. In Section 3 an incentives infrastructure is presented by describing its components. Section 4 shows the experimental results; finally Section 5 puts forward some related work and presents conclusions and some lines of future work.

2 The Model

In our work we assume that agents are rational utility maximizers. Following the work presented by Centeno et al. in [7], a *rational agent* is modelled as a tuple $\langle \mathcal{S}, \mathcal{O}, g, per, \mathcal{U}, t, s_0 \rangle$; where $\mathcal{S}$ is the set of internal states of the agent; $\mathcal{O}$ is the observation space; $g : \mathcal{O} \times \mathcal{S} \rightarrow \mathcal{S}$ is the agent's state transition function; $per : \mathcal{X} \rightarrow \mathcal{O}$ is the perception function; $\mathcal{U} : \mathcal{S} \rightarrow \mathbb{R}$ is the utility function that assigns a value to each possible internal state; and t the agent's decision function such that $t : \mathcal{S} \rightarrow \mathcal{A}$ follows the principle of maximising the expected utility (MEU). That is, $t(s) = argmax_{a \in \mathcal{A}} eu(a, s) = argmax_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{U}(s') \cdot \overline{P_s}(s'|s, a)$, where $eu(a, s)$ is the expected utility of performing the action a in the state s; $\mathcal{U}(s')$ is the utility of the state s' estimated by the agent; and $\overline{P_s}(s'|s, a)$ is the agents' estimate, at state s, of the probability that state s' will occur when executing action a in state s. The utility function of an agent is defined over the possible internal states, so, it is *local* to the agent and,

thus, has to be defined with respect to what the agent observes from its environment. Therefore, the utility function is a means to solve the decision problem, from the agents own perspective, rather than as a measure of its performance in solving some given task as seen from the outside.

Agents are embedded in a MAS that specifies the environment in which they perform. We model a MAS as a tuple $\langle \mathcal{A}g, \mathcal{A}, \mathcal{X}, \Phi, \varphi, \mathcal{U}, x_0 \rangle$; where $\mathcal{A}g$ is a set of agents, $|\mathcal{A}g|$ denotes the number of agents; $\mathcal{A}$ is a possibly infinite action space that includes all possible actions that can be performed in the system; $\mathcal{X}$ is the environmental state space; $\Phi : \mathcal{X} \times \mathcal{A}^{|\mathcal{A}g|} \times \mathcal{X} \to [0..1]$ is the MAS transition probability distribution, describing how the environment evolves as a result of agents' actions; $\varphi : \mathcal{A}g \times \mathcal{X} \times \mathcal{A} \to \{0, 1\}$ is the agents' capability function describing the actions agents are able to perform in a given state of the environment (*physical* restrictions); $\mathcal{U} : \mathcal{X} \to \mathbb{R}$ is the global utility function of the system that assigns a value to each environmental state; and, finally, $x_0 \in \mathcal{X}$ stands for the initial state.

MAS are usually designed with a general purpose in mind – the global objective of the system. In this model, such an objective is represented by means of a set of preferences which are captured through the utility function $\mathcal{U}$. From the point of view of the designer, the problem consists of how to optimize the global utility of the system assuming that agents will try to optimize their own individual utilities. In [7] propose to introduce organisational mechanisms in the system, with the aim of influencing the behaviour of agents. In particular, we focus on *incentive mechanisms* that change the consequences agents' actions may have. The rationale behind this approach is that changing the consequences of actions may produce variations in the expected utility of agents, what, in fact, can be seen as the introduction of penalties or rewards. Accordingly, incentive mechanisms are formalised as a function $\Upsilon_{inc} : \mathcal{X}' \to [\mathcal{X} \times \mathcal{A}^{|\mathcal{A}g|} \times \mathcal{X} \to [0..1]]$ that changes the consequences of actions (i.e., the transition probability distribution Φ), taking into account the partial view the mechanism has about the environment ($\mathcal{X}'$).

Based on this model, we make the following assumptions:

Assumption 1 *The action space in the system is finite.*

Assumption 2 *Agents are utility maximisers. The utility functions of both, the global system and the agents, capture the utility at a long term, for instance by Bellman's Principle of Optimality [9]. That is, agents are able to calculate how good/bad is an action and they always choose the action that maximises their utility in the next state.*

Assumption 3 *The environment of a system can be discretized by a finite set of attributes: $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$. An environmental state $x_i \in \mathcal{X}$ can be modelled as a set of tuples $x_{i,j} = \langle attribute, value \rangle$ that assigns a value to each attribute.*

Assumption 4 *As consequence of assumption 3, the utility of an environmental state is the output of a multi-attribute utility function [10].*

Assumption 5 *The attributes are additively independent. That is, the utility function of the system can be expressed as: $\mathcal{U}(x_i) = \sum_{j=1}^{n} w_j \cdot u_{i,j}$ where $u_{i,j}$ is the utility of the attribute $x_{i,j}$ and w_j is the weight of attribute X_j in the global utility function.*

Assumption 6 *All the participants in the system share the same ontology. This means that from the system level and from the perspective of the agents attributes refer to the same concepts. For example, the car agent a_i owns would be denoted in the same way from the agents and from the system as $car\_agent_{a_i}$.*

Following the model and taking into account assumption 3, changes in the MAS transition probability distribution (Φ) can be produced by changes in the probabilities of transiting to environmental states in which some of the attributes have been modified. Also, the agents' perception functions per become functions that given an environmental state – a set of attributes –, creates an observation consisting of a subset of that attributes. Similarly, an incentive mechanism Υ_{inc} modifies the system transition probability distribution taking into account the partial view the mechanism has about the system: the subset of attributes the mechanism is able to perceive.

2.1 The Problem

As we have said previously, an incentive mechanism is defined as a function Υ_{inc} that given a partial view of the system – a subset of attributes –, modifies the consequences of actions – the system transition probability distribution. So, the problem of designing an incentive mechanism boils down to address the following issues:

1. Deciding when the incentive mechanism should change the consequences of an action. In the model, this means which values of its the partial view of the world $\mathcal{X}'$ will fire the mechanism.
2. Selecting the action(s) whose consequences should be modified by the mechanism, that is, which actions in $\mathcal{A}$ should be incentivized/punished.
3. Deciding which agent(s) will be affected by the incentive.
4. Deciding the modification the mechanism should perform in order to influence the behaviour of an agent. In our model this corresponds to the environmental states that may be reached as consequences of actions in the transition probability distribution Φ.

Summarising, the problem of designing an incentive mechanism requires to learn which attributes should be modified, so as to make the consequences of a particular action more or less attractive for an agent. This includes the estimation of agents' preferences, as well as, to decide how the consequences of an action should be changed in order to incentivize it.

3 Incentives Infrastructure

The objective of an incentive mechanism is to induce an agent to perform an action(s), that maximises the utility of the system, by modifying the consequences of such actions. Since agents are autonomous and independent entities, we cannot assume the system will know agents' preferences. Thus, the objective of the incentive mechanism is twofold: *i)* discovering agents' preferences; and *ii)* selecting the appropriate incentive

to induce agents to behave in a certain manner. With this objective in mind, we propose an incentives infrastructure able to deal with those issues.

Similar to the use of governor agents in Electronic Institutions [6], we propose an incentives infrastructure where interactions between external agents and the system are mediated by institutional agents called *incentivators*. Each external agent participating in the system has assigned an incentivator and all actions selected by the agents will be performed in the system through their incentivators. Incentivator agents are in charge of both, discovering the preferences of their associated external agent, and modifying the consequences of certain actions with the aim to promote desired behaviour. Figure 1 shows the proposed architecture for the incentives infrastructure and a particular incentivator.

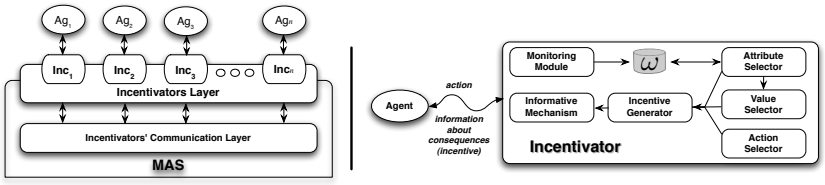


Fig. 1. Incentives Infrastructure and Incentivators architecture

Incentivators can communicate with each other through a communication layer. Each incentivator is responsible for one external agent and is composed of the following modules:

- *Action Selector*: this module has the objective of selecting the action that, if it would be performed by the agent in the current state, would be the best action for the system (e.g., regarding the systems global utility function).
- *Attribute and Value Selector*: these modules are in charge of determining the next incentive. That is, they decide which attribute will be modified and the new value of this attribute, as consequence of a particular action.
- *Incentive Generator*: this module receives a modification of an attribute from the attribute and value selector modules and a candidate action for modifying its consequences in the current environmental state. Taking into account this information it should decide if the proposed modification, applied as a consequence of the selected action, is still beneficial[1] for the system.
- *Monitoring Module*: it observes the actual activity of the agent and tends to model the agent's preferences.
- *Informative Mechanism*: the incentivator provides information to the agent about the potential consequences of its actions (e.g. the incentives associated to an action).

[1] With beneficial we mean that the system improves its utility with regard to the current state.

3.1 Discovering Agent's Preferences

An incentivator has to know or at least has to estimate an agent's preferences in order to be able to incentivize certain actions. Here we assume that agents may have different and a priori unknown preferences.

A possible approach for identifying agents' preferences is either to ask agents a set of questions or to give them a questionnaire asking directly about their preferences. Taking into account the answers given, their utility functions are estimated [11]. However, this approach might be difficult to apply in an open multiagent system where agents are autonomous and the system is not able to impose any actions. We propose a mechanism that learns an agent's preferences from monitoring its behaviour in response to given incentives.

An incentivator is endowed with a structure where an agent's preferences are estimated. More formally, we define a preferences vector as: $\omega_{ag_i} = [\omega_1, \ldots, \omega_n]$, where:

- n is the number of attributes in the system, such that it corresponds with the attributes that compounds an environmental state;
- ω_j is the weight agent ag_i has estimated on the utility of attribute X_j; we assume $\sum_{j=1}^{n} \omega_j = 1$;
- ω_{ag_i} is the estimated preferences vector for agent ag_i.

For instance a vector $\omega_{ag_1} = [0.1, 0.2, 0.7]$ means that the environmental state is composed of three attributes and that agent ag_1 is interested 0.1 in the attribute X_1, 0.2 in the attribute X_2 and 0.7 in the X_3.

The task of the incentivator is to estimate the preferences vector of its assigned external agent. In order to do that, we use Q-learning with immediate rewards and ϵ-greedy action selection [12]. In the scope of this learning method, the action space Z_i of the incentivator for agent ag_i is composed of the attributes it can modify as consequence of its action. More formally: $Z_i \subseteq \{X_1, \ldots, X_n\}$, where X_j are the attributes that compose an environmental state.

The idea behind this approach is that the incentivator estimates the preferences of an agent by modifying an attribute as consequence of an action and observing if the agent is induced to perform (or not) such an action. Thus, when an incentivator takes the action z_j, that means to modify the attribute X_j, during the learning process, it receives a reward that rates that action, and it updates its action-value function estimation as follows:

$$\mathcal{Q}_{t+1}(z_j) = \mathcal{Q}_t(z_j) + \alpha \cdot [\mathcal{R}_t(z_j) - \mathcal{Q}_t(z_j)] \tag{1}$$

where α is the learning rate and $\mathcal{R}_t(z_j)$ is the reward. As reward the incentivator focuses on how the agent reacts to the modification proposed. Thus, it rates the action positively when the agent performs the action the incentivator wanted to, and negatively in other case:

$$\mathcal{R}_t(z_j) = \begin{cases} \text{+1 if agent performed the action} \\ \text{-1 i.o.c.} \end{cases} \tag{2}$$

In order to discover new attributes in which the agent could be interested in, incentivators select a random attribute modification with small probability ϵ, and exploit the

attribute with a the highest Q-value, the attribute in which the agent is most interested in (*greedy* action), with probability $(1 - \epsilon)$.

The next step in the learning process is to learn the most appropriate value of a selected attribute, in order to incentivize the agent. This task is carried out by the *Value Selector Module*. Similarly to the attribute selection process, this task is developed by using Q-learning with immediate rewards and ϵ-greedy action selection. In this case, the action space Y_i of the value selector module is composed of the different values the attribute proposed by the attribute selector may take. Formally, $Y_i = \{value_j \in \left[value_{X_i}^{min}, value_{X_i}^{max}\right]\}$, where $value_j$ stands for the set of different values the attribute X_i may take. The update and reward functions are the same as before (formulas 1 and 2). So, observing how the agent reacts to the new consequences the incentivator is able to estimate the attributes and their appropriate values that affect the utility of an agent. Obviously, the incentivator can only modify those attributes it is capable and has enough permission to change.

Summarising, as a result of the tasks carried out by those modules, a modification of an attribute of the environmental state is proposed. Formally, $x_{i,j}^{*} = \langle attribute, value \rangle$, where $attribute$ is selected by the attribute selector, and $value$ by the value selector. The next step is to select the action to incentivize.

3.2 Selecting the Action to Incentivize

At the same time that an incentivator is selecting the next incentive – an attribute to modify and its new value – it has to select the action to incentivize. This task is carried out by the *Action Selector*. This module follows an intuitive approach, trying to induce the action that gives the highest utility for the system. The incentivator, on behalf of the system, wants the agent to perform the action that would lead to the state with the best utility from a system perspective.

Therefore, what the incentivator has to do is to estimate the result of each possible action the agent is able to perform, and calculate the utility of the system in each resulting state. The function that allows the incentivator to simulate the result of an action is domain-dependent, so it depends on the particular system we are dealing with. Note that in the current work we focus on how to induce a desirable behaviour in an agent. For this reason, when the incentivator estimates the result of an action performed by its agent, it will not take into account possible conflicts among other actions performed by other agents in the same state.

The domain-dependent algorithm executed by the action selector module returns a list of actions ranked by an estimation over the utility the system would get in case the agent performs each possible action. Formally, this list is represented by $\bigtriangledown_{x_j}^{ag_i} = \langle a_1, \ldots, a_n \rangle \mid eu(\overrightarrow{x_j, ag_i, a_1}) \geq \ldots \geq eu(\overrightarrow{x_j, ag_i, a_n})$, where:

- $\bigtriangledown_{x_j}^{ag_i}$ stands for the list of actions agent ag_i is able to perform in the environmental state x_j, sorted by the expected utility of the system;
- $eu(\overrightarrow{x_j, ag_i, a_k})$ is the expected utility of the system in the environmental state reached as consequence of the action a_k, performed by the agent ag_i, in the environmental state x_j.

As soon as this list is calculated, and the incentive ($x^*_{i,j}$) is proposed, both parameters are introduced in the incentive generator in order to decide whether this combination is still beneficial for the system.

3.3 Generating the Incentive

When a new incentive is proposed it could be necessary to assure that such an incentive is not damaging the objective of the system. That is, the changes in the environment proposed as an incentive could produce undesirable states for the global system. To evaluate whether or not the new consequences of the action are still the best option (not only for the agent but also for the system) is the task of the *Incentive Generator Module*. The result of this process may be the rejection of the proposed incentive ($x^*_{i,j}$) if the result would be even worse for the system than if the agent performs the worst action (from the point of view of the system).

The decision is taken by the following algorithm:

Algorithm 1: deciding if the proposed incentive is given or not

Input: $\bigtriangledown_{x_j}^{ag_i}, x^*_{i,k}$
Output: $a_s \in \bigtriangledown_{x_j}^{ag_i}$ such that a_s is the best action to incentivize

1 **for** $s = 1$ *to* n **do**
2 $\quad a_s \leftarrow \bigtriangledown_{x_j}^{ag_i}[s]$;
3 $\quad a_{s+1} \leftarrow \bigtriangledown_{x_j}^{ag_i}[s+1]$;
4 $\quad$ **if** $(eu(\overrightarrow{(x_j x^*_{i,k}), ag_i, a_s}) \geq eu(\overrightarrow{x_j, ag_i, a_s}))\vee$
$\quad (eu(\overrightarrow{(x_j x^*_{i,k}) ag_i, a_s}) \geq eu(\overrightarrow{x_j, ag_i, a_{s+1}}))$ **then**
5 $\quad\quad$ **return** a_s;
6 $\quad$ **end if**
7 **end for**
8 **return** $a_s \leftarrow \emptyset$;

Summarizing, the algorithm focuses on finding an action to incentivize, such that if the agent performs this action, and the proposed incentives (changes in the environment) are applied in the resulting state, the expected utility of the system is greater or equal than if the agent performs the same action (without the changes in the environment). Another case when the action should be incentivized, is when the expected utility of the system with the new consequences is greater or equal than if the agent performs the following best action, with regard to the utility of the system. The possible solutions of the algorithm are either the best action to incentivize by using the incentive proposed, or no action that means that is better not to give the incentive proposed to the agent.

3.4 Monitoring and Informing the Agent

As we said in previous sections, in order to discover the agent's preferences the incentivator observes the reaction of the agent regarding a proposed incentive for a given

action. In particular, the incentivator monitories the action the agent actually performs and evaluates if this action is the same as the action selected by the incentivator to incentivize. It then uses this information to provide the required feedback for the Q-learning algorithms presented before. It could be possible that an agent actually performed an action because its own interests (and not because of the incentives). However, the incentivator does not have any way to distinguish such a situation. We assume that the exploration/exploitation process in the Q-learning algorithms will detect such cases and converges to an estimation of the agent's correct preferences.

In order to enable agents to reason about incentives, the incentivator informs the agents about the consequences of their actions (the incentives that may apply). Before an agent selects its new action to perform, it can query the informative mechanism about the incentives that apply to which action.

4 Experimental Results

To evaluate the incentives infrastructure proposed in this work we have designed and implemented a small "toy" example. The system is composed of a grid of size NxN, where each column is filled with a different colour: red, black or blue (see figure 2).

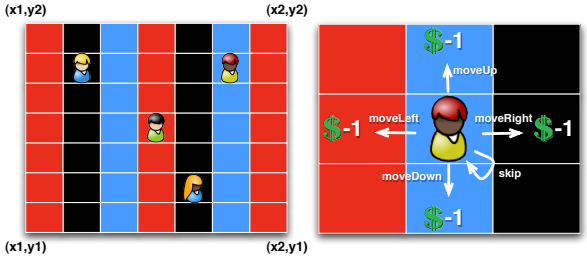


Fig. 2. Grid representation and action space of an agent

Rational agents are situated in such a grid, being able to move around it, moving a position up, down, left or right. That is, the action space of the system is: $\mathcal{A} = \{moveUp, moveDown, moveLeft, moveRight, skip\}$. An agent's position changes when it performs one of these action (except for skip). Furthermore, each agent participating in the system has assigned an amount of money that is reduced by one unit when it changes its position. On the other hand, the global system has also an amount of money that is increased by the money that agents spend on moving around the grid. An environmental state is composed of the following attributes: $\mathcal{X} = \{agent_1Position, \ldots, agent_1Money, \ldots, gridSize, squaresColours_1, \ldots, systemMoney\}$ where their possible values are *i)* the position, in terms of coordinates, of each agent; *ii)* the current money each agent owns; *iii)* the size of the grid (the parameter N); *iv)* the current

colour of each square (red, black or blue); and *iv)* the amount of money owned by the system. The environment is designed in a deterministic way, i. e. two o more agents can be in the same position. The objective of the system is that agents are in the central position of the grid as well as to get as much money as possible. These preferences are expressed by means of the following utility function: $\mathcal{U}(x_j) = \mathcal{U}_{systemMoney}(x_j) \cdot w_0 + \sum_{k=1}^{|\mathcal{A}g|} \mathcal{U}_{agent_k Position}(x_j) \cdot w_k$ where the utility over the position of each agent is measured as how far they are from the central position – by using the Manhattan distance –; and the money of the system is measured such that the more money the system gets the more utility it obtains. On the other hand, the objective of the agents is threefold, to reach a corner, to stay in a particular colour and to save as much money as possible. It is expressed by the general utility function: $\mathcal{U}_{a_k}(x_j) = \mathcal{U}_{agent_k Position}(x_j) \cdot w_1 + \mathcal{U}_{squaresColours_{agent_k Position}}(x_j) \cdot w_2 + \mathcal{U}_{agent_k Money}(x_j) \cdot w_3$, where the utility of their position is measured as how far they are from the corner they want to reach; the utility of the colour is 1 when they are on a square with the preferred colour (0 in other case); and the utility of the money is measured like in the case of the system.

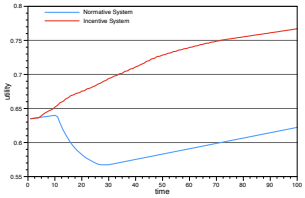
	Exp1	**Exp2**
Grid/Agents/Steps	$200/30/100$	$200/30/100$
$\mathcal{U}_{Ag}$	$Random(w_1, w_2, w_3)$	$w_1, w_2 = 0.45\ w_3 = 0.1$
$\mathcal{U}_{MAS}$	$Random(w_0, w_k)$	$Random(w_0, w_k)$
Norm Limit/Penalty	$10/-50$	$10/-50$
$agent_i Money^*$	$\pm 5\%$	$\pm 5\%$
$agent_i Money_0$	$Random(1.000)$	$Random(1.000)$
$agent_i Position_0$	$(100, 100)$	$(100, 100)$
corner to reach	$Random(4)$	$Random(4)$

In order to evaluate and compare our approach, we design a normative system to regulate the system described before. Such a system defines a global norm that is known by all participants and says that *"it is prohibited to go beyond a established area from the central point of the grid"*. This norm is coupled with penalties that reduce the agents' money when they violate the norm. We assume perfect (100%) detection of norm violations and fines are applied automatically. Thus, the system fulfils its original objectives: agents should stay as close as possible to the central position, and on the other hand, the system gets money if agents violate the norm. In comparison to the normative approach, we endowed the same system with the incentives infrastructure and where incentivators have enough permissions to modify the attributes $squaresColours_j$ (changing the colour of a particular square to black, red or blue) and $agent_k Money$ (increasing/decreasing the money of agents). Obviously, when an incentivator gives/takes money to/from an agent, this money is taken/given from/to the system.

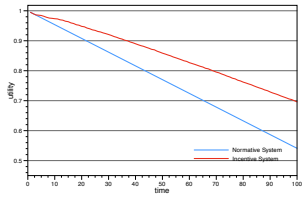
We have set up the system with the parameters specified in the table 4, carrying out two different experiments: *exp1* and *exp2*. In the first case (exp1), the weights of the parameters of the utility functions of agents and of the system are selected randomly. When the system is executed with the normative system, the permitted area is estab-

lished to 10 squares from the central point and a fine of 50 units of money is applied when agents go out of this area.

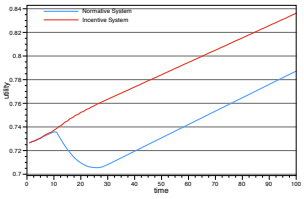
Graphics 3(a) and 3(b) plot the utility of both the system and the agents in experiment **Exp1**. The blue line represents the case when the system is regulated by the norm while the red line represents the results when using the incentives infrastructure. Both the system and agents get better results, with regard to their utility in the case in which the system is regulated by the incentive infrastructure. In case of agents' utility we can observe clearly, how their utility increases in both situations, however when the normative system is working and the system is in the time-step 10, agents' utility decrease constantly. It happens due to some agents at this time are in the limit of going out of the established area (10 squares from the central point) and they decide to go beyond it violating the norm; and the system penalizes them decreasing their money. After that, their utility starts rising up again due to they are closer to the corner they want to reach. Regarding the system's utility, we can observe that in both situations the utility starts in the maximum possible. This is because the agents' initial position is in the central point of the grid. As agents are moving around the grid, the system looses utility, but this loss is lower when the incentive infrastructure is working. This is because incentivators are able to incentivize their agents to stay closer to the central position.



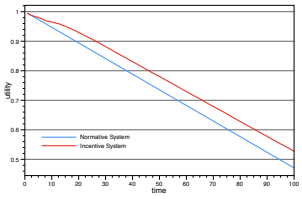
(a) Fig. 4: Agents' Utility Exp1



(b) Fig. 4: System's Utility Exp1



(c) Fig. 4: Agents' Utility Exp2



(d) Fig. 4: System's Utility Exp2

Graphics 3(c) and 3(d) plot the results obtained after executing the system with the parameters defined in experiment **Exp2**. In this case, the system is populated with agents that do not care so much about the money (parameter $w_3 = 0.1$ in agents' utility function). The system's utility is quite similar with both mechanisms, norms and incentives, even so, the system looses less utility when it is regulated by the incentives infrastructure.

Comparing the two experiments, we can conclude that the approach presented in this work, based on an incentives infrastructure works better to regulate a system when

it is populated by agents unknown at a design time. Besides, not just the system performs better (regarding its utility); also the agents obtain more utility because the incentives infrastructure is able to discover their preferences and applies personalised incentives/penalties.

5 Conclusion

In this paper we have presented an incentives model and infrastructure that is able to *i)* discover the agents' preferences that might be based on different attributes of the environment; and *ii)* provide the suitable incentive by modifying the consequences of a particular action, such that desirable behaviours are induced. The incentives infrastructure is designed by using institutional agents called *incentivators*. Each incentivator is in charge of one external agent. In particular, it tries to discover the preferences of its agent, decides which would be the most desired action regarding the systems objectives, and incentivizes its agent to do that desired action. Incentivators use reinforcement learning techniques (Q-learning with immediate rewards and ϵ-greedy action selection) to learn the best incentivation policy for the agents they are in charge of.

Many authors have proposed the use of social concepts such as roles, norms, groups, and so on to regulate the activities of agents in a MAS if there is no control over the components [1–5]. The concept of *norm* appears as a main piece in most of these approaches. Norms are mechanisms that define control policies to establish and reinforce agents to accomplish the objective of the system – *"the rules of the game"*. So, the main objective of normative systems is to restrict the action spaces of agents such that they behave how the system (or its designer) wants them to behave. Some approaches, like [1], focus on defining the set of allowed actions in each possible state of the system, such that agents are just allowed to perform valid actions, from the point of view of the whole system. However, there exist domains (e.g. traffic, e-commerce, etc.) where due to their complexity and size, it is very difficult to define all possible valid actions for each state, or even it could be counter-productive. For this reason, other approaches, like [2], propose to couple norms with penalties/rewards which gives agents the possibility to violate a norm. These penalties/rewards are usually designed at design time, where participants of the system are still unknown. Therefore, designers have to assume two main assumptions: *i)* agents, that will populate the system, are interested in a particular attribute (e.g. the money); and *ii)* the grade that a modification in this attribute affects agents such that they are deterred to perform certain actions. For instance, in the traffic domain, it is usually assumed that all drivers, modelled as agents, are interested in saving money, and the penalties for violating norms are set as an obligation of paying an amount of money, that is usually the same for all drivers. In order to avoid these assumptions, other approaches, like [13, 14], are emerged for basing on incentives to induce desirable behaviours to agents participating in a system. One difference between these approaches and the one presented in this paper, is that the authors in [13, 14] focus on how to discover an appropriate reward to teach a particular policy to an agent. In our work, we consider incentives as something that could be positive or negative, that is, it is a modification in the consequences of an action. In fact, the valuation of such modifications depend on the agents. Some agents may consider as a positive

modification what others consider to be negative. Another difference is that in [13, 14] the authors assume that all agents might be incentivized by adding a reward; that is only one possible attribute is considered. In comparison, we try to discover which attributes affect each individual agent in order to provide a personalised incentive.

In this work, we have only considered scenarios where the actions performed by agents do not influence the actions or the utility of other agents at the same time. However, this is a simplifying assumption. In our future work, we want to deal with cases where agents do influence each other. In particular, we will use multiagent learning techniques, to coordinate incentivators in order to be able to induce a desired joint action of the agents in the system. That is why the infrastructure architecture allows incentivators to communicate each others. Furthermore, we would like to apply the approach in a real world domain, for instance e-commerce or peer-to-peer applications.

References

1. Esteva, M., Rodriguez, J., Sierra, C., Garcia, P., Arcos, J.: On the formal specification of electronic institutions. Agent Mediated Electronic Commerce **LNAI 1991** (2001) 126–147
2. Dignum, V., Vazquez-Salceda, J., Dignum, F.: OMNI: Introducing social structure, norms and ontologies into agent organizations. In: Proc. Programming multi-agent systems: 2nd Int. Workshop. Volume LNCS 3346. (2004) 181–198
3. Hubner, J., Sichman, J., Boissier, O.: MOISE+: Towards a Structural, Functional and Deontic Model for MAS Organizations. In: Proc. AAMAS. (2002) 501–502
4. DeLoach, S., Oyenan, W., Matson, E.: A capabilities-based theory of artificial organizations. J. Autonomous Agents and Multiagent Systems **16** (2008) 13–56
5. Boella, G., Hulstijn, J., van der Torre, L.W.N.: Virtual organizations as normative multiagent systems. In: HICSS, IEEE Computer Society (2005)
6. Esteva, M., Rosell, B., Rodríguez-Aguilar, J., Arcos, J.: AMELI: An agent-based middleware for electronic institutions. In: Proc. of the 3rd Int. Joint Conference on Autonomous Agents and Multiagent Systems. Volume 1. (2004) 236–243
7. Centeno, R., Billhardt, H., Hermoso, R., Ossowski, S.: Organising mas: A formal model based on organisational mechanisms. In: SAC: 24th Annual ACM Symposium on Applied Computing. (2009) 740–746
8. Posner, R.A.: Economic analysis of law. Little Brown (1977)
9. Bellman, R.: Dynamic Programming. Princeton University Press (1957)
10. Keeney, R., Raiffa, H.: Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Cambridge University Press (1993)
11. Boutilier, C., Patrascu, R., Poupart, P., Schuurmans, D.: Regret-based utility elicitation in constraint-based decision problems. In: IJCAI: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence. (2005) 929–934
12. Watkins, C.: Learning from Delayed Rewards. PhD thesis, King's College, Cambridge, UK (1989)
13. Zhang, H., Parkes, D.: Value-based policy teaching with active indirect elicitation. In: AAAI: Proc. of the 23rd Int. Conference on Artificial intelligence, AAAI Press (2008) 208–214
14. L. Dufton, K.L.: Multiagent policy teaching. In: AAMAS: Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems. (2009)