Using Incentive Mechanisms for an Adaptive Regulation of Open Multi-Agent Systems*

Roberto Centeno
Departamento LSI, UNED
Madrid, Spain
rcenteno@lsi.uned.es

Holger Billhardt
CETINIA, Universidad Rey Juan Carlos
Móstoles, Spain
holger.billhardt@urjc.es

Abstract

In this paper we propose a mechanism that encourages agents, participating in an open MAS, to follow a desirable behaviour, by introducing modifications in the environment. This mechanism is deployed by using an infrastructure based on institutional agents called *incentivators*. Each external agent is assigned to an incentivator that is able to discover its preferences, and to learn the suitable modifications in the environment, in order to improve the global utility of a system in response to inadequate design or changes in the population of participating agents. The mechanism is evaluated in a p2p scenario.

1 Introduction

Open MultiAgent Systems (OMAS) are systems designed with a general purpose in mind but with an unknown population of autonomous agents at design time. The agents that populate such systems may be heterogeneous (e.g., they may have different unknown preferences and behaviours) and their number may vary at runtime. Based on their open nature, the general problem when designing OMAS consists in assuring that agents will behave according to the system's objectives and preferences.

The research community has tackled this problem by defining organisational models that structure and regulate the agents' action space. Some approaches (e.g. Electronic Institutions [Esteva *et al.*, 2001]), define the actions that agents can take in each state of the system and rely on a certain infrastructure that assures that agents cannot violate the defined rules. In those proposals, the implemented rules – defined at design time – are fixed. They can be seen as heuristics that help to meet the system's global objectives. This approach has two disadvantages. First, the agents may still have a certain degree of freedom and this may still imply a more or less efficient completion of the systems overall objective. Second, the fixed nature of predefined rules may imply less flexibility in certain unforeseen situations – especially in highly dynamic and complex systems.

*This work was supported by the projects AT (CONSOLIDER CSD2007-0022, INGENIO 2010) and OVAMAH (TIN2009-13839-C03-02).

Other approaches (e.g., OMNI [V.Dignum *et al.*, 2004]) define the valid actions in terms of norms, but agents are able to violate such norms [V.Dignum *et al.*, 2004]. In order to avoid violations those approaches rely on penalties/rewards and implement violation detection mechanisms. However, the current population of the system may not be responsive to the defined penalties and rewards and, thus, the norms may not be effective. In this sense, approaches like the one proposed in [Cardoso and Oliveira, 2009] aim at adapting normative systems. However, such adaptations are applied to the whole population, that is, the penalties/rewards learnt are applied to all agents equally. Therefore, some agents may still not be responsive to the penalties/rewards.

In our opinion, in (norm based) OMAS it is hard to specify a good set of norms at design time. It may not be clear whether the proposed punishments/incentives have the desired influence on the agents nor whether the specified norms may actually effect the global utility in a positive way. Addressing this problem, we propose to endow OMAS with a mechanism that tries to induce agents at each moment to act in a way that is appropriate from the point of view of the global utility of the system. Our approach is inspired by the theory "Economic Analysis of Law", proposed by R.A. Posner in [R.A.Posner, 1977]. In this work the author analyses normative systems from an economic point of view. He focuses on the effects of norms in terms of outcomes on both, the behaviour of individuals and the society as a whole. Assuming that individuals are rational, norms are actually incentives that may induce agents to act in a certain way and this, in turn, may have a certain effect on the society. Following these ideas, we propose an adaptive incentive mechanism that: *i)* identifies the desirable actions the agents should perform in each state, *ii)* estimates the agents' preferences, and *iii)* induces agents to act in the desired way by modifying the consequences of their actions.

The paper is organized as follows, Section 2 provides basic definitions and assumptions. Our approach is presented in Section 3. Section 4 describes the experimental validation. Section 5 puts forward some related work. Finally, Section 6 points out some conclusion and future work.

2 Definitions and assumptions

Adapting the model presented in [Centeno *et al.*, 2009], we consider an agent participating in an OMAS to be a ratio-

nal utility maximizer defined as a tuple $\langle \mathcal{S},\ g,\ \mathcal{U},\ t,\ s_0 \rangle$; where $\mathcal{S}$ is the set of internal states of the agent (s_0 is the initial state); $g : \mathcal{X}' \times \mathcal{S} \rightarrow \mathcal{S}$ is the agent's state transition function that assigns a new internal state to the current state and a partial observation of an environmental state $x' \in \mathcal{X}'$; $\mathcal{U} : \mathcal{S} \rightarrow \mathbb{R}$ is the utility function that assigns a value to each possible internal state; and t is the agent's decision function such that $t : \mathcal{S} \rightarrow \mathcal{A}$ follows the principle of maximising the expected utility. 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 $\mathcal{A}$ is the set of possible actions; $eu(a, s)$ 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 a is executed in state s.

Agents act within an OMAS that defines their environment. We use $\mathcal{X}$ to denote the environmental state space. $\mathcal{U}_O : \mathcal{X} \rightarrow \mathbb{R}$ is the utility function of the system that assigns a value to each state. $\mathcal{U}_O$ encodes the design objectives of the system. From the point of view of the designer of the OMAS, the problem consists of how to assure the maximization of the system's utility assuming that agents will try to optimise their own individual utilities. In our approach we propose to use an incentive mechanism for this task [Centeno *et al.*, 2009].

Our notion of "incentive" is slightly different to the usual consideration of incentives to be something positive. In our work, we consider that incentives are any kind of modifications to the environment that have the aim to make a desirable action a_j more attractive than other alternatives $\{a_s, a_v, ...\}$ and such that a rational agent would decide to take a_j. Thus, an incentive is a modification introduced in the environment which can be interpreted in a different way by different agents. In this sense, the notion of positive and negative depend on the preferences of each agent. The rationale behind this is that changing the consequences of actions may produce variations in the expected utility of agents and, thus, may change their behaviour.

In addition to the rationality of agents, we make the following assumptions:

- **Assumption 1:** the action space in the system is finite.
- **Assumption 2:** the environment of a system can be discretized by a finite set of attributes: $\mathcal{X} = \{X_1, \ldots, X_n\}$. An environmental state $x_i \in \mathcal{X}$ can be modelled as a set of tuples $x_{i,j} = \langle attribute, value \rangle$ assigning a value to each attribute. The incentive mechanism has permission to modify the values of at least some of those attributes.
- **Assumption 3:** the OMAS has a multi-attribute utility function ([R.L.Keeney and H.Raiffa, 1993]) defined over some attributes of the environment. The attributes are additively independent. That is, the utility function can be expressed as: $\mathcal{U}_O(x_i) = \sum_{i=1}^{n} w_i \cdot u_{i,j}$ where $u_{i,j}$ is the utility of the attribute $x_{i,j}$ in the state x_i and w_i is the weight of such an attribute. Their preferences do not have to be aligned with the preferences of the system.
- **Assumption 4:** agents are sensitive to at least one of the environmental attributes incentivators are authorized to modify. That is, given such an attribute, they prefer certain values over others.

3 Incentive mechanism

The proposed incentive mechanism accomplishes two basic tasks: *i)* selecting the actions to be promoted in order to improve the global utility of the system; and *ii)* performing the required changes in the environment in order to make the desired actions more attractive for the agents.

Whereas in standard normative approaches both tasks are solved at design time (by specifying norms), we propose to solve them at runtime. We use learning algorithms that allow to adapt the incentive mechanism to the current agent population as well as to possible changes in the environment.

We propose to deploy the incentive mechanism as an infrastructure composed of a set of institutional agents called *incentivators*. Each agent is assigned to an incentivator that aims to discover its preferences and tries to learn how to stimulate the agent by modifying the consequences of its actions. That is, there is a 1-1 relation between external agents and incentivators. This means, incentives are provided on an individual basis and may be very different for different agents. Furthermore, incentivators can communicate with each other allowing them to coordinate their actions.

3.1 Discovering agents' preferences

The action selection mechanism of rational agents is based on an ordering over actions, in terms of expected utility. In order to induce such agents to perform a certain action, an incentive mechanism could either modify the consequences of that action in such a way that the resulting state is more attractive, or it could modify the consequences of the rest of the actions in order to make them less attractive. If the agent's preferences are known, this becomes a relatively easy task. However, in open systems, agents' preferences are unknown and, thus, have to be discovered.

One approach to preference elicitation is to enquire agents. Based on the responses, the utility functions can be estimated [C.Boutilier *et al.*, 2005]. However, this approach has obvious disadvantages, e.g. agents could lie. Alternatively, we propose to use a non-intrusive approach where the incentivators discover their agents' preferences by observing their behaviour in response to given incentives. The characteristics of this process are: *i)* it is a learning process; *ii)* it is independent, i. e., incentivators do not require coordination; and *iii)* an incentivator receives an immediate local reward (agent's reaction). With these characteristics in mind, we have chosen Q-learning with immediate rewards and ϵ-greedy action selection [C.Watkins, 1989] as a mechanism to carry out this task. For the sake of simplicity, we split the process into two different but related learning process. On one hand, the incentivator has to discover the attributes that affect an agent's utility function, and, on the other hand, it has to identify the values those attributes should take in order to make an agent change its behaviour[1].

[1]The focus is on how to persuade agents by learning which attributes have some influence on their behaviour. That is, incentivators do not actually model agents' utility functions.

Learning the attributes that influence agents' behaviour

In the scope of Q-learning, the action space Z_i of the incentivator for agent ag_i is composed of the attributes the incentivator is authorised to modify in the system. More formally: $Z_i \subseteq \{X_1, \ldots, X_n\}$; where X_j are attributes belonging to the environmental state of the system. Thus, when the incentivator takes the action $z_{i,j}$ this means that the attribute X_j will be modified. After that, it receives a reward that rates that action, and the action-value function estimation is updated with the typical update function of the Q-learning algorithm:

$$\mathcal{Q}_{t+1}(z_{i,j}) = \mathcal{Q}_t(z_{i,j}) + \alpha \cdot [\mathcal{R}_t(z_{i,j}) - \mathcal{Q}_t(z_{i,j})] \quad (1)$$

where α is the learning rate and $\mathcal{R}_t(z_{i,j})$ the reward. As we said before, the idea is to discover an agent's preferences by observing how it reacts to the selected modifications. In this sense we rate positively an action if the agent performs the action the incentivator intended, and negatively if not. This is captured with the following reward function:

$$\mathcal{R}_t(z_{i,j}) = \begin{cases} +1 & \text{if } ag_i \text{ performed the action} \\ -1 & \text{i.o.c.} \end{cases} \quad (2)$$

Besides, in order to explore new attributes, a random selection is made with probability ϵ, and the highest q-value attribute (*greedy* action) is exploited (chosen in the next step) with probability $(1 - \epsilon)$.

Learning the values of attributes

The next step in the learning process is to learn the most effective value of the selected attribute. The characteristics of this problem are the same as in the attribute learning process. Again, we use Q-learning with immediate rewards and ϵ-greedy action selection. In this case, the action space $Y_{i,j}$ of the incentivator for agent ag_i depends on the attribute X_j selected previously by the attribute learning algorithm. It is composed of the different values that X_j may take. Formally: $Y_{i,j} = \{value \in \left[value^{min}_{X_j}, value^{max}_{X_j}\right]\}$; where $value$ stands for the set of different values the attribute X_j may take[2]. As update and reward functions we use the same formulae as before (equations 1 and 2).

Combining both learning phases, in each step an incentivator proposes a modification – a new value for a pair x^*_i=$\langle attribute, value\rangle$. Over time, x^*_i eventually converges towards a pair that influences the behaviour of agent ag_i.

3.2 Identifying desirable actions

As we have introduced previously, the incentive mechanism has to decide which actions should be incentivized in order to improve the system's utility. In scenarios where the outcome of the action performed by an agent does not depend on the actions taken by others, these actions could be determined locally. However, in many common situations the outcome of an action depends on the joint action of all participating agents. In order to account for this fact, all incentivators should work as a team so as to coordinate the actions to be promoted. The main problem in order to carry out such a task is that incentivators have just a local view of the system – the result of the action performed by "their agents".

Therefore, this process should have the following capabilities: *i)* learning a joint action in a cooperative way; *ii)* dealing with the lack of information about the actions taken by other members of the team; and *iii)* dealing with immediate local rewards. With this in mind, incentivators are endowed with a reinforcement multiagent cooperative learning algorithm that updates the action-value function estimation with the typical Q-learning function (equation 1). As reward function we use the global utility, that is calculated by aggregating the local rewards through a gossip-based algorithm.

Incentivator's action space

The first issue that needs to be addressed, is determining the action set of an incentivator in this learning task. It is composed of the set of actions its agent (ag_i) can take in the current situation, combined with the attribute modification selected by the attribute learning algorithm described earlier. Formally: $V_i \subseteq \{\emptyset, \beta_{i,1}, \ldots, \beta_{i,n}\}$; where $\emptyset$ is the $skip$ action; and $\beta_{i,j} = \langle a_{i,j}, x^*_{i,j}\rangle$, where $a_{i,j}$ stands for an action agent ag_i can perform in the current situation; and $x^*_{i,j}$ is the attribute modification selected as result of the learning process explained in 3.1. When an incentivator takes the action $\beta_{i,j}$, this means that the action $a_{i,j}$ should be made more attractive by modifying its consequences through a change of the $\langle attribute,value\rangle$ pair $x^*_{i,j}$. The action $\emptyset$ means that none of the actions will be promoted (e.g. none of the parameters of the environment will be changed).

Calculating the reward

After taking an action ($v_{i,j} \in V_i$), the incentivator i receives a reward that rates such an action, and the action-value function estimation is updated by using the corresponding formula (equation 1). This reward is calculated as follows: $\mathcal{R}_t(v_{i,j}) = \bar{\mathcal{U}}_i(x)$; where $\bar{\mathcal{U}}_i(x)$ is the incentivator's estimation of the global utility in the environmental state x reached after the last step. Since an incentivator has only a local view of the system, it can calculate $\bar{\mathcal{U}}_i(x)$ only based on its local perception of the environment[3]. In order to take into account the actions taken by other incentivators, it should transform its local estimation into an estimation of the global utility. Based on the assumption of additive independence of the attributes in the system's utility function (Assumption 3), the global utility can be estimated by aggregating the local utility estimations of all incentivators. In order to perform this task, each incentivator is endowed with the gossip-based aggregation algorithm presented in [M.Jelasity *et al.*, 2005].

a) active thread	**b)** passive thread
do once each δ time steps	**loop**
$j \leftarrow getIncentivator()$	$\bar{\mathcal{U}}_j(x) \leftarrow receive(*)$
$send\,\bar{\mathcal{U}}_i(x)$ to j	$send\,\bar{\mathcal{U}}_i(x)$ to $sender(\bar{\mathcal{U}}_j(x))$
$\bar{\mathcal{U}}_j(x) \leftarrow receive(j)$	$\bar{\mathcal{U}}_i(x) \leftarrow update(\bar{\mathcal{U}}_i(x), \bar{\mathcal{U}}_j(x))$
$\bar{\mathcal{U}}_i(x) \leftarrow update(\bar{\mathcal{U}}_i(x), \bar{\mathcal{U}}_j(x))$	**end loop**

Table 1: Gossip-based algorithm executed by an incentivator

The idea is that each incentivator holds a local value, and

[2]This approach requires the set of values to be discrete. In case it is continuous, it should be discretized previously.

[3]We assume that incentivators know the system's utility function.

by exchanging messages with its neighbours the local values are aggregated by using some aggregation function. Two different threads are executed (see table 1). The active thread periodically initiates an information exchange with a random incentivator j by sending a message containing the local utility estimation $\bar{\mathcal{U}}_i(x)$ and waits for a response with the utility estimation $\bar{\mathcal{U}}_j(x)$ from incentivator j. On the other hand, the passive thread waits for messages sent by other incentivators and replies with the local utility estimate. The $update$ method updates the local utility estimation by aggregating the current value and the received value. In our particular use case (see Section 4), we have chosen the average. Therefore, $update(\bar{\mathcal{U}}_i(x), \bar{\mathcal{U}}_j(x))$ returns $(\bar{\mathcal{U}}_i(x) + \bar{\mathcal{U}}_j(x))/2$. This function decreases the variance over the set of all local estimates of the global utility.

3.3 Interaction between agents and incentivators

Two different types of interactions could be performed between an incentivator and its agent. On one hand, in order to enable agents to reason about incentives, the incentivator informs its agents about the consequences of the desirable actions. Before an agent selects a new action, it will query the incentivator asking for the possible new consequences of the actions the agent considers[4].

On the other hand, each incentivator observes the reaction of "its" agents to the proposed incentives. This information provides feedback for the preference learning algorithm described earlier. It should be noted that it is possible that an agent may perform an action because of its own interests (and not because of the proposed incentive). We do not have any mean to distinguish such situations. We assume that the exploration/exploitation process will detect such situations and converge to an estimation of the agent's preferences.

4 Case study: a P2P system

We have chosen a peer-to-peer (p2p) file sharing scenario for evaluating our approach. Such scenarios are clear examples of open systems where the objectives of individuals may not coincide with the objectives of the system.

In such systems normally only few peers (*seeders*) have the whole information; and the rest of peers (*leechers*) download pieces by using a particular protocol. We focus on peers sharing a file with the BitTorrent protocol [B.Cohen, 2008]. Following this protocol, a file is split in pieces of $256KB$ each and every piece is split in 16 sub-pieces called *blocks*. Peers exchange blocks and each block can be downloaded from different peers. For the sake of simplicity we leave out the phases in which peers and data are identified and peers get a list of neighbours to communicate with. We just focus on the phase carried out to get each block. In this phase, each peer sends a *bitfield* message to its neighbours asking for their file status. After that, the peer has to decide which neighbours will be asked for the next block to download. A *request* message is sent to the selected peers. When a peer receives a request message it has to decide whether the requested block will be uploaded or not. Once a peer accepts the request, it sends a *piece* message containing the requested block. Immediately, the receiver of the piece sends a *cancel* message to the other neighbours it asked for the same block. When the download is finished, a *have* message is sent to the agent's neighbours in order to update their file status.

4.1 P2P system model

Regarding the most common problems in p2p systems (e.g. non-cooperation of peers, flooding of the network, etc. [D.Hales, 2004]), the objectives of the system could be specified as follows: *i)* peers should download/upload as many files/blocks as possible in order to increase the number of available files; *ii)* the usage of the network should be as low as possible; and *iii)* the time spent on downloading files should be as short as possible in order to avoid an overload of the network. These objectives are captured by the following multi-attribute utility function: $\mathcal{U}_O(x) = \mathcal{U}_{files} \cdot w_0 + \mathcal{U}_{blocks}(x) \cdot w_1 + \mathcal{U}_{C_n}(x) \cdot w_2 + \mathcal{U}_{C_t}(x) \cdot w_3$. $\mathcal{U}_{files}(x)$, is the utility of the number of already downloaded files. $\mathcal{U}_{blocks}(x)$ represents the utility of the number of downloading/uploading blocks in the state x (the greater the number of downloading/uploading blocks, the greater is the utility). $\mathcal{U}_{C_n}(x)$ is the utility of the usage of the network in state x. Following the work presented in [J.Campos *et al.*, 2009], this parameter is defined as *network cost* and represents the sum of the network usage of each message (c_{m_i}) sent in the network. It is calculated as follows: $C_n = \sum_{i=0}^{\#msgs} c_{m_i}$, such that, $c_{m_i} = m_{length} \cdot Lat(m_{org}, m_{dst})$; where m_{length} is the length of a message, and $Lat(m_{org}, m_{dst})$ is the latency between the origin and destination[5]. The lower the network cost, the greater is the utility. Finally, $\mathcal{U}_{C_t}(x_j)$ is the utility of the time spent on downloading a file (*time cost*). The shorter the time cost, the greater is the utility. The partial utilities $\mathcal{U}_x$ are calculated, in case of maximization, as the ratio between the actual value of the parameter x divided by its maximum possible value in the current state, (1 minus the ratio in case of minimization). w_0, w_1, w_2, w_3 allow us to weight the attributes.

4.2 Peer agent model

Peers are modelled as rational agents. We focus on two decisions peers have to take: *i)* to decide to how many neighbours they will send a request asking for the next block to download; and *ii)* to decide how many requests should be accepted (i.e., how many piece messages are sent). Accordingly, the action space of a peer is: $\mathcal{A} = \{send_{Piece}(n), send_{Request}(n), skip\}$, where n is the number of neighbours it sends a piece/request message.

We assume that peers have to pay a regular fee in order to connect to the network by using a certain bandwidth. Besides, a peer has a file it is sharing and whose status can be partially or completely downloaded. We assume that a peer knows the latency with all its neighbours; and its file status is updated when *have* messages are received.

The attributes that may have some influence on a peer's preferences are bandwidth, fee, number of download-

[4] We suppose rational agents will always use this "service" since it is in their own interest.

[5] Latency is assumed to be constant and symmetric.

files to share	files=1; size=15 blocks	latencies randomly in	$[10, 400]$ ms
bandwidths randomly in	$\{640, 1024, 2048, 4096\}$ Mb/s	agent's fee randomly in	$[10, 50]$ with steps=5
messages length	piece=128Bytes request,cancel=1Byte	incentivators messages	length=1Byte
		time steps equivalence	1 timeStep = $10ms$
messages ttl	ttl=6 time steps	system utility function	w_0=0.5 w_1=0.2 w_2=0.125 w_3=0.175
number of peers (agents)	50		
learning alg. parameters	α=0.9 ϵ=0.1	q-values initialisation	1 (maximum value)
neighbours per peer	$[2, 4]$	percentage of seeder	$\leq 35\%$

Table 2: Experimental setup

ing/uploading blocks and time spent on downloading a file. They are captured by using the following multi-attribute utility function: $\mathcal{U}(x) = \mathcal{U}_{bw}(x) \cdot w_4 + \mathcal{U}_{fee}(x) \cdot w_5 + \mathcal{U}_{down}(x) \cdot w_6 + \mathcal{U}_{up}(x) \cdot w_7 + \mathcal{U}_t(x) \cdot w_8$; where $\mathcal{U}_{bw}(x)$ is the utility of the bandwidth rate - the bandwidth regarding to the available bandwidth; $\mathcal{U}_{fee}(x)$ represents the utility of the fee that a peer is paying; $\mathcal{U}_{down}(x)$ stands for the utility of the number of downloading blocks; $\mathcal{U}_{up}(x)$ is the utility of the number of uploading blocks to other peers; $\mathcal{U}_t(x)$ is the utility of the time spent on downloading a file; and w_4, w_5, w_6, w_7, w_8 are the weights assigned to each attribute.

4.3 Experimental setup

We have carried out experiments that compare two different types of regulation mechanisms: a standard normative system, and the proposed incentive mechanism. The aim of both mechanisms is to improve the system's global utility. In addition, both mechanisms are compared to the case where no regulation takes place. The system has been instantiated with the parameters shown in table 2.

The normative system is based on a set of norms coupled with penalties that are applied when norms are violated. In particular, three norms have been designed at design time, that is, before knowing the population of the system: **N1**: "It is prohibited to use more bandwidth than 85%"; **N2**: "A peer is obliged to upload a block when at least 25% of the bandwidth is available"; and **N3**: "It is prohibited to request a block to more than the 85% of neighbours". The set of norms is designed according to the global objective of the system. Norm violations are penalised with an increase of the fee in 5 units. Such violations are detected with a 100% efficiency.

Regarding the incentive mechanism, it is deployed by taking advantages of the own nature of p2p systems. That is, incentivators are located at network service providers. Thus, the communication among them will be fast. Incentivators are authorized to modify the bandwidths and the connection fees of peers. So, the action space of agent ag_i's incentivator is $Z_i = \{fee_{ag_i}, bw_{ag_i}\}$ and $Y_{i,fee} = \{value_{fee_{ag_i}} \in [value^{min}_{fee_{ag_i}}, value^{max}_{fee_{ag_i}}]\}$ in case the attribute selected is the fee; or $Y_{i,bw} = \{value_{bw_{ag_i}} \in [value^{min}_{bw_{ag_i}}, value^{max}_{bw_{ag_i}}]\}$ in case it is bandwidth. The actions that can be incentivized are $send_{Piece}(n)$, $send_{Request}(n)$ and $skip$, where n can be instantiated by a number of neighbours.

4.4 Experimental results

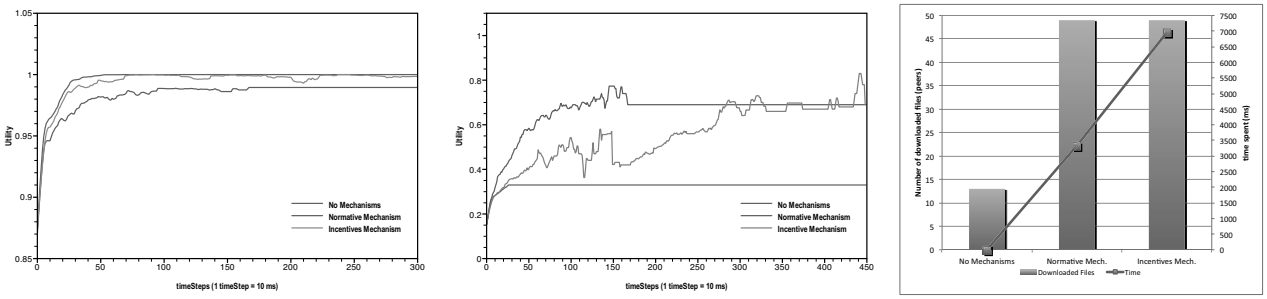
Using the parameters described above, we have conducted two experiments. In the first one, agents are sensitive to the fee (they prefer lower fees and, thus, the penalties for norm violations are effective). In the second experiment, agents are rather insensitive to changes in the fee and, thus, the effectiveness of the normative system is not assured. In both experiments the system is populated with non-collaborative agents. That is, they are interested in downloading blocks, but not in uploading blocks.

Experiment 1: Non-collaborative agents, sensitive to connection fee

In this experiment, the fact that agents are none-collaborative but sensitive to changes in the fee they are paying for connecting to the network is captured by the following weights in their utility function: w_4=0.2, w_5=0.399, w_6=0.3, w_7=0.001 and w_8=0.1.

Figure 1(a) plots the average utility obtained by all peers participating in the system. As we can see, agents obtain the highest utility when the system is executed without any regulation mechanisms. This seems obvious, because nothing restricts the freedom of agents to act in the way they want to. The second best performance is presented when the incentive mechanism is used. The reason is that agents are regulated by giving incentives, that is, they are incentivized to perform in a certain manner by paying a lower fee, in this particular case. The normative system works quite "bad" regarding the agents utility. That happens because norms restrict agents' behaviour, so, their utility is lower because of the punishments. Regarding the system, figure 1(b) plots the evolution of the utility for each configuration. It shows clearly that the system's utility is low if no regulation takes place. This was expected because the analysed population of peers does not behave according to the system's preferences. On the other hand, the normative and incentive mechanisms work similarly well. The incentive mechanism is a bit "slower" at the beginning due to the learning algorithm, that is, incentivators have to discover that the connection fee is the attribute that has the highest influence on the peers. This is also shown in figure 1(c), which compares the mechanisms regarding to the number of peers that are able to download the whole file and the time spent on it. By using the normative, as well as the incentive mechanism, 49 out of 50 are able to download the whole file. However, the time spent is higher in the case of the incentive mechanism, due to the learning process, as it has been pointed out previously.

Concluding, in this case the "design-time" norms are effective and both, the normative and the incentive mechanism obtain similar results. The normative mechanism performs slightly better which is basically due to the required learning



(a) Agents utility (b) System utility (c) Downloaded files and time

Figure 1: Experimental results: experiment 1

process in the incentive mechanism.

Experiment 2: Non-collaborative agents, insensitive to the connection fee

In this experiment we have specifically chosen a peer population that is not sensitive to changes in the fee they are paying, e.g., they do not care about increasing fees. This is reflected in the agents' utility function (w_4=0.2, w_5=0.001, w_6=0.399, w_7=0.001 and w_8=0.399; in particular w_5=0.001). This case describes a scenario where the designed norms (all based on an increase in the fee as a punishment) will not be very effective for the given population of agents. This result can be clearly observed in Figure 2. Figure 2(a) plots the average utility obtained by all peers. The results are clearly better when the incentive mechanism is employed. Implicitly, this mechanisms is able to identify that instead of the fee, the bandwidth has an influence on peers' utility. It uses changes in the available bandwidth to make the upload of blocks to other peers attractive. Figure 2(b) plots the utility of the system when it is regulated by the different mechanisms. As it was expected, the system improves its performance when it is regulated by the incentive mechanism because it is able to influence agents' behaviour regarding the sending of their blocks. Finally, in figure 2(c) we can see the number of peers that are able to download the whole file. In the case of the normative and no mechanisms, none of the peers are able to download the files, only the initial seeders have the whole file, that is why the time spent on downloading the files is zero. With the incentive mechanism 48 out of 50 peers download the whole file, spending 7640ms.

It is important to note that the overhead introduced as a consequence of the gossip algorithm is taken into consideration in the *network cost* attribute calculated in the system utility function. That means, the incentive mechanism is beneficial even if the overhead is taken into account. The experiment makes clear that a normative system does not fulfil its goal if the agents are not sensitive to the applied punishments. In contrast, the proposed incentive mechanism is able to adapt to such situations.

5 Related work

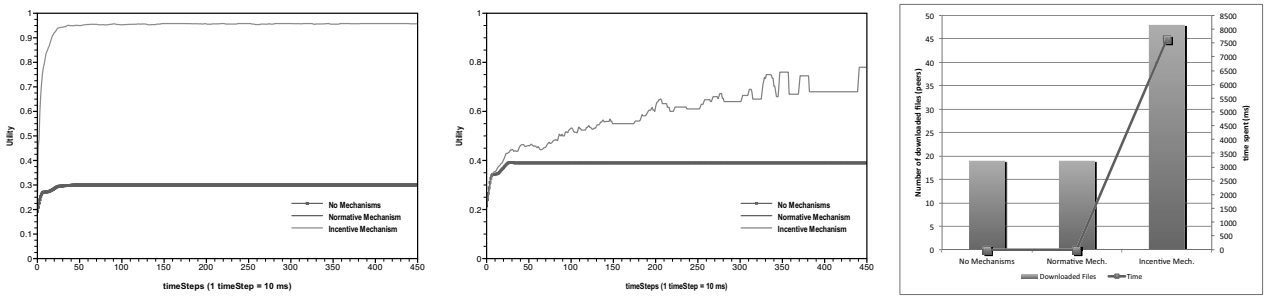
Many approaches to regulate open MAS rely on the concept of *norm*. As mentioned before, in some approaches (e.g., [Esteva *et al.*, 2001]), norms define the set of allowed actions in each possible state of a system and mechanisms are employed to assure that agents only perform those actions. We think our approach is compatible with such approaches. Our work can be used for inducing a particular action, among all possible valid actions, optimising in that way the system's efficiency. Other approaches, like [V.Dignum *et al.*, 2004], propose to use norms with penalties/rewards and allow agents to violate a norm. In these approaches norms are usually defined at designed time, what implies that some assumptions are made with regard to the parameters that have an influence on agents' preferences. In our work, we do not assume a priori knowledge about agents' preferences; we try to learn such preferences for each agent and, thus, provide a basis for an individualised incentivation/punishment.

In a way, our work is related to mechanism design as it aims at influencing the behaviour of rational agents in a desired direction. However, instead of directly defining the interaction rules of the system, we try to achieve desired behaviour of agents by modifying their environment at runtime. Furthermore, we do not assume that the agents' pay-off functions are known. Instead, we intend to learn which of the environment attributes are relevant for an agent's pay-off, so as to dynamically adjust incentives for it at runtime.

Recently, some papers have been published with a similar focus as ours [Z.Rabinovich *et al.*, 2010; L.Dufton and K.Larson, 2009]. All of them address the problem of how to influence agents' behaviour in order to induce some desirable behaviour. They focus on formal approaches based on environment design techniques, incentive based approaches and behaviour cultivation. Similar to our case, the authors show through experimental evaluations that their approaches effectively induce desirable behaviour.

6 Conclusion

In this paper we have put forward an incentive mechanism that is able to induce desirable behaviour by modifying the consequences of actions in open MAS. The mechanism tries to discover which attributes have some influence on agents' preferences and learns how agents can be incentivized. It is deployed by using an infrastructure based on institutional agents called *incentivators*. The incentivators use the Q-learning algorithm to discover agents' preferences by observing how they react to modifications in the environment.



(a) Agents utility (b) System utility (c) Downloaded files and time

Figure 2: Experimental results: experiment 2

Moreover, they learn – in a cooperative way, using Q-learning and a gossip-based reward calculation algorithm – which joint actions should be incentivized in order to increase the utility of the system. Finally, the proposed mechanism has been tested in a p2p file sharing scenario, showing that it is a valid alternative to standard normative systems. In particular it outperforms regulation mechanisms based on fixed norms if the design assumptions of such norms are not fulfilled.

As future work, we plan to explore other learning techniques that may be more suitable in scenarios where agents preferences may vary over time. In principle, Q-learning can deal with such situations, but it may be too slow to obtain the desired adaptation of the system. Furthermore, the mechanism is currently able to perform the modification of just one attribute at the time. In this sense, we will explore the possibility of modifying a set of attributes jointly.

References

[B.Cohen, 2008] B.Cohen. The bittorrent protocol specification, 2008.

[Cardoso and Oliveira, 2009] Henrique Lopes Cardoso and Eugénio Oliveira. Adaptive deterrence sanctions in a normative framework. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 02*, WI-IAT '09, pages 36–43, Washington, DC, USA, 2009. IEEE Computer Society.

[C.Boutilier *et al.*, 2005] C.Boutilier, R.Patrascu, P.Poupart, and D.Schuurmans. Regret-based utility elicitation in constraint-based decision problems. In *IJCAI'09*, pages 929–934, 2005.

[Centeno *et al.*, 2009] R. Centeno, H. Billhardt, R. Hermoso, and S. Ossowski. Organising mas: A formal model based on organisational mechanisms. In *SAC: 24th Annual ACM Symposium on Applied Computing*, pages 740–746, 2009.

[C.Watkins, 1989] C.Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

[D.Hales, 2004] D.Hales. From selfish nodes to cooperative networks " emergent link-based incentives in peer-to-peer networks. In *Proc. of the 4th Int. Conf. on Peer-to-Peer Computing*, pages 151–158, 2004.

[Esteva *et al.*, 2001] M. Esteva, J.A. Rodriguez, C. Sierra, P. Garcia, and J.L. Arcos. On the formal specification of electronic institutions. *Agent Mediated Electronic Commerce*, LNAI 1991:126–147, 2001.

[J.Campos *et al.*, 2009] J.Campos, M.López-Sánchez, and M.Esteva. Multi-agent system adaptation in a peer-to-peer scenario. In *SAC'09*, pages 735–739. ACM, 2009.

[L.Dufton and K.Larson, 2009] L.Dufton and K.Larson. Multiagent policy teaching. In *AAMAS'09*, 2009.

[M.Jelasity *et al.*, 2005] M.Jelasity, A.Montresor, and O.Babaoglu. Gossip based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.

[R.A.Posner, 1977] R.A.Posner. *Economic analysis of law*. Little Brown, 1977.

[R.L.Keeney and H.Raiffa, 1993] R.L.Keeney and H.Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Cambridge University Press, 1993.

[V.Dignum *et al.*, 2004] V.Dignum, J.Vazquez-Salceda, and F.Dignum. OMNI: Introducing social structure, norms and ontologies into agent organizations. In *PROMAS'04*, volume LNCS 3346, pages 181–198, 2004.

[Z.Rabinovich *et al.*, 2010] Z.Rabinovich, L.Dufton, K.Larson, and N.Jennings. Cultivating desired behaviour: Policy teaching via environment-dynamics tweaks. In *AAMAS'10*, pages 1097–1104, 2010.