Learning to Select a Coordination Mechanism

Cora B. Excelente-Toledo
University of Southampton
Dept. of Electronics and Computer Science
Southampton SO17 1BJ, UK.
cbet99r@ecs.soton.ac.uk

Nicholas R. Jennings
University of Southampton
Dept. of Electronics and Computer Science
Southampton SO17 1BJ, UK.
nrj@ecs.soton.ac.uk

ABSTRACT

This paper examines the potential and the impact of introducing learning capabilities into autonomous agents that make decisions at run-time about which mechanism to exploit in order to coordinate their activities. Specifically, the efficacy of learning is evaluated for making the decisions that are involved in determining when and how to coordinate. Our motivating hypothesis is that to deal with dynamic and unpredictable environments it is important to have agents that can learn the right situations in which to attempt to coordinate and the right method to use in those situations. This hypothesis is evaluated empirically, using reinforcement based algorithms, in a grid-world scenario in which a) an agent's prediction about the other agents in the environment is approximately correct and b) an agent can not correctly predict the others' behaviour. The results presented show when, where and why learning is effective when it comes to making a decision about selecting a coordination mechanism.

Categories and Subject Descriptors

I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Experimentation

1. INTRODUCTION

Effective coordination is essential if autonomous agents are to achieve their goals in a multiagent system. Such coordination is required to manage the various forms of dependency that naturally occur when the agents have inter-linked objectives, when they share a common environment, or when there are shared resources. To this end, a variety of mechanisms have been developed to address the coordination problem at different levels of abstraction [12, 13, 3].

All of these *coordination mechanisms* have different properties and characteristics and are suited to different types of tasks and environments. They vary in the degree to which coordination is prescribed at design time, the amount of time and effort they require to set up a given coordination episode at run-time, and the degree to which they are likely to be successful and produce coordinated behaviour in a given situation. In the majority of cases, these dimensions act as forces in opposing directions; coordination mechanisms that are guaranteed to succeed typically have high set up and maintenance costs, whereas mechanisms that have lower set up costs are also more likely to fail. In short, there is no universally best coordination mechanism.

In general, the choice of coordination mechanism is something that is imposed upon the system at design time. While this may be sufficient for predictable and stable environments, it is inappropriate in dynamic and open contexts because there is no scope for changing or modifying the mechanism to ensure there is a good fit with the prevailing circumstances [1]. In such environments, it is important that the agents have a variety of coordination mechanisms, with varying properties, at their disposal and that they can then select the mechanism which is most appropriate for the task at hand. Thus, for particularly important tasks, the agents may choose to adopt a coordination mechanism that is highly likely to succeed, but which will invariably have a correspondingly large set up cost. Whereas for less important tasks, a mechanism that is less likely to succeed, but which has lower set up costs, may be more appropriate. In short, this means that the coordination mechanism that is employed must be suited to the agents' prevailing circumstances.

To achieve the necessary degree of flexibility in coordination requires an agent to make decisions about when to coordinate and which coordination mechanism to use. To this end, previous work has developed a reasoning framework to achieve this [1, 4]. However, this work also highlighted the importance (as well as the difficulty) of making good approximations about the behaviour of other agents. This is especially true as the environment becomes more dynamic. Given this, a natural extension of the framework is to enable the agents to acquire knowledge through run-time adaptation. Thus, the agents need to be capable of learning to make the right decisions about their coordination problem.

Against this background, the aim of this research is to develop agents that can reason about the process of coordination and then select mechanisms that are appropriate to their current situation. More specifically, here, we deal with the problem of allowing agents to learn the right situation in which to apply the right coordination mechanism. This work advances the state of the art in the following ways.

AAMAS'02, July 15-19, 2002, Bologna, Italy.

Firstly, it introduces learning into that part of the agent's decision making process that is concerned with when and how to coordinate. Secondly, it empirically demonstrates where the benefits of learning can be obtained and where learning is not beneficial in this decision making context.

This paper is structured as follows. Section 2 presents our specific coordination scenario. Section 3 formalises the decision procedures of the agents. Section 4 explains how the decision making procedures are applied in a learning context. Section 5 reports on the experimental work to evaluate the effect of introducing the reinforcement based algorithms. Section 6 presents related work and Section 7 concludes and presents the areas of further work.

2. THE COORDINATION TESTBED

Our domain takes the form of a grid-world in which a number of autonomous *agents* (A_i) perform tasks for which they receive units of *reward* (R_i) (see [1] for a justification of the choice of this particular environment for studying coordination problems). Each agent has a *specific task* (ST_i) which only it can perform; there are other tasks which require several agents to perform them, called *cooperative tasks* (CTs). Each task has a reward associated with it, the rewards for the CTs are higher than those for STs since they must be divided among the coordinating agents.

The agents move around the grid one step at a time, up, down, left or right, or stay still. At any one time, each agent has a single *goal*, either its ST or a CT over which coordination needs to be achieved. On arrival at a square containing its goal, the agent receives the associated reward. In the case of STs, a new one appears, randomly, somewhere in the grid, visible only to the appropriate agent. In the case of CTs, a new one appears, randomly, somewhere in the grid, but this is only visible to an agent who subsequently arrives at that square. If an agent encounters a CT, while pursuing its current goal (i.e., its ST), it takes charge of the CT[1] and must decide on both whether to initiate coordination with other agents over this task, and which coordination mechanism (CM) it should use. In this context, each agent has a predefined range of CMs at its disposal. Each CM is parameterised by two key attributes: set up cost (in terms of time-steps) and chance of success. For example, a CM may take t time-steps to set up (modelled by the agent waiting that number of time-steps before requesting bids from other agents) and have a probability, p, of success (thus when the other agent(s) arrive at the CT square, the reward will be allocated with probability p, with zero reward otherwise). An agent may well decide that attempting to coordinate is not a viable option, in which case it adopts the null CM (i.e. the agent rejects adopting the CT as its goal).

The Agent-in-Charge (AiC) of the coordination selects a CM and, after waiting for the set up period, broadcasts a request for other agents to engage in coordination. The other agents respond with bids composed of the amount of reward they would require in order to participate in the CT and how many time-steps away from the CT square they are situated. If an agent's bid is successful, then it is termed Agent-in-Cooperation (AiCoop) to denote the fact that it is a participant (not AiC) for a CT task. The role Agent-in-ST (AiS) is used to denote the situation where an agent is working towards a ST.

Agents might receive more than one proposal at the same time step, in which case they reply with as many bids as the proposals they receive. However, they will only accept one CT contract at a time. Agreements between AiCs and AiCoops to achieve a particular CT are established via a contracting protocol. This contract-net-like protocol consists of three steps. In the first step, AiC broadcasts a proposal to all agents. It then waits for the bids. The second step involves selecting the bids and contracts from AiCs and AiCoops respectively (both of them have to consider refusals and denials of their corresponding offers). Finally, the third step consists of the commitment about the terms of the contract and the time step at which AiCoops will arrive at the CT square. Figure 1 gives the protocol the agents follow at each time-step.

[1] Agents arrive at a square. If AiS arrives at its ST cell, its goal is attained, it receives the reward and updates its goal. If AiCoop arrives at the CT cell, it notifies the AiC that it has arrived. It might have to wait in the cell until the remaining AiCoops arrive. If AiC receives confirmations from all AiCoops, the CT is achieved and the rewards are paid to AiCoops.

[2] If AiS finds a CT it must decide if it wants to become AiC and, if so, which CM= (t, p) it should use. If $t > 0$ it must wait t time-steps before broadcasting a request for coordination.

[3] If AiS receives a request for coordination, it decides whether and what to bid to participate in the CT. The AiC then evaluates all bids. If AiS's bid is accepted, it adopts CT as its new goal. Neither AiCoops nor AiC respond to requests for coordination.

[4] Each agent decides on its next move according to its current goal and all agents move simultaneously.

Figure 1: Basic protocol followed by agents

As more than two agents may be required to achieve a CT, it is necessary to deal with the fact that an AiCoop may have to wait in the CT cell while the remaining AiCoops arrive (because agents have to travel different distances). In such cases, the AiC pays an additional reward for the time elapsed—AiC knows the number of time steps that each AiCoop is likely to have to wait (specified in the bid) and the amount it will pay for waiting time at a specific predefined waiting rate (q). Thus when an AiCoop notifies the AiC of its arrival at the CT cell, it either receives its share of the CT reward or the waiting rate followed by its share of the CT reward.

To clarify the protocol and the previous description of the scenario, Figure 2 shows a 10x10 grid size at a specific time step with 5 agents in the grid and three CTs. The CT in position (1,9) requires 3 agents to be achieved, the one at (2,6) needs 4 agents and the one at (9,8) requires 2 agents. In the specific moment shown, the AiC-A_1 (at (1,9)) negotiated and is in agreement with two AiCoops (A_4 at (4,6) and A_3 at (4,8)) to achieve its CT at (1,9). A_0 and A_2 are AiSs (at (4,4) and (6,5) respectively) that are working towards their respective specific tasks at (2,2) and (6,5). No agents have found the CTs at (2,6) and (9,8).

3. DECISION MAKING PROCEDURES

Previous work has developed and evaluated a decision making framework for reasoning about whether and how to

[1]If several agents arrive at a CT square at the same time, one of them is arbitrarily deemed to be in charge and, if an agent finds more than one CT in a given cell, it randomly selects one of them for further analysis.

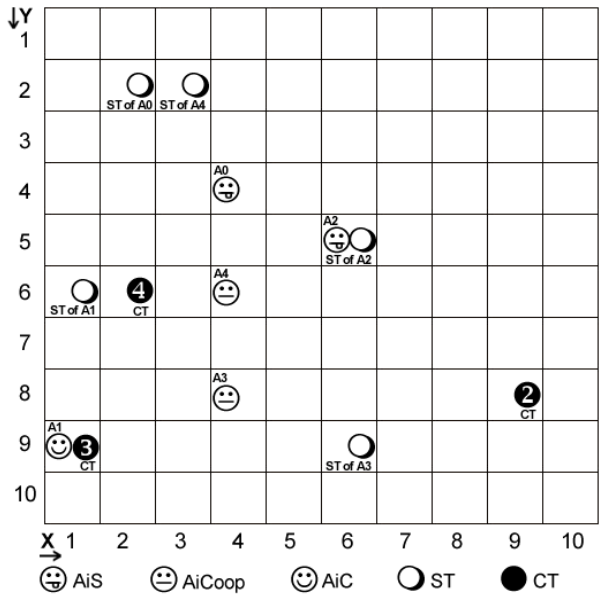

Figure 2: Scenario with agent roles

coordinate in this domain [1, 4]. Since the main focus in this paper is on the role and impact of learning on this framework, we do not discuss all the details of the model here. Rather we concentrate on the decisions where learning could have a role to play; i.e. in which CM to adopt, if any; how much to bid when a request for coordination is received; and how to determine which bid to accept, if any.

The agents' aim is to maximise their reward, in particular their average reward per unit time. To account for heterogeneity in the population, each agent keeps track of its own average reward, termed its *reward rate*, being its total cumulative reward divided by the total number of time-steps taken to obtain it. It uses this rate to decide how much to charge for its own services and occasionally to approximate the expected rates of other agents when it is not able to build up a picture them. Specifically, each agent uses its reward rate to evaluate and compare the different actions available to it; if it can maintain or improve this rate, it chooses to do so. Of course, this decision model approximates the true relative values of different actions.

3.1 Deciding which CM to select

An agent which, while pursuing its current goal, encounters a CT must decide whether to initiate coordination with other agents in order to perform it. To do this, the agent must determine whether there is any advantage in so doing. This depends not only on the reward that is being offered, but also on the CMs available, as well as on various environmental factors which affect the expected demands of the potential coordinating agents.

To model the *expected demands* of the other agents, the AiC assumes they are randomly distributed throughout the grid, and that their current goals are similarly distributed. Thus some agents may be near the CT while others may be far away; likewise, for some agents there would be a significant deviation from their ST to reach the CT, while others may be able to coordinate over the CT en route to their own goals. The agent assesses the possible CMs on the basis of how long before the task can be performed (including both the set up time and the average distance away each agent is situated) and of how much reward it is likely to obtain after deducting the expected reward requirement of the other agents (based on the amount of time they must spend deviating from their path and the CM's probability of success).

The agent uses all these factors to assess each CM in terms of the amount of surplus reward it can expect, over and above what it expects to obtain during its normal course of operation, i.e., its own average reward per time-step, r. The agent selects the CM which maximises this surplus[2].

To formalise this decision procedure, consider an $M \times N$ grid with reward size S for STs, and R for CTs, a coordination mechanism, CM=(t, p), which costs t time-steps to set up and has a probability of success p. In this grid-world of known size, the agent can calculate the expected average distance (`ave_dist`) away of any randomly situated agent from the CT square, as well as the likely average deviation (`ave_dev`) such agents would have to make to get there.

Based on these figures, the agent can assess the average surplus reward from coordinating over the CT at (x,y) using $\text{CM}_j = (t_j, p_j)$. First, it must estimate its own cost in terms of how long the CM will take to set up and how long it expects to wait for the other agents to arrive. Since the AiC would usually expect to receive r reward units per time-step[3], the cost of CM_j is given by:

$$\texttt{cost}_j(x, y) = r \times (t_j + \texttt{ave\_dist}(x, y))$$

Second, the AiC must estimate the average amount of reward the other m agents will require. To distinguish an agent's own average reward (r) from that of the others, r_AiCoop is used to refer to the average reward of all the other agents in the environment. When AiC does not have any knowledge of r_AiCoop it uses its own average reward as an approximation.

$$\texttt{ave\_bid}_j(x, y) = \frac{r\text{\_AiCoop} \times \texttt{ave\_dev}(x, y)}{p_j} \quad (1)$$

Third, the AiC estimates the expected surplus of CM_j from adopting the CT by taking into account the probability of success of the task:

$$\texttt{ave\_surplus}_j(x, y) = p_j \times R - (\texttt{cost}_j(x, y) + (m \times \texttt{ave\_bid}_j(x, y)) \quad (2)$$

When deciding which of its CMs to adopt, the agent computes its expected surplus reward from each of them and selects the one that maximises this value. If the surplus associated with all CMs is negative, the agent adopts the option of the null CM (which is defined to have zero surplus).

The formulation presented so far allows agents to take decisions about when and which CM to select in order to achieve a CT. Since this is the most important decision the agents face in this scenario [1], it is the one we concentrate on in terms of evaluating the role of learning[4].

[2]Though this may not be a globally optimal criterion for deciding which CM to use, it makes sense from a self-interested agent's point of view.

[3]The simple form of computing the average reward assumes that agents will obtain the reward of a ST per time step, $r = \frac{S}{\texttt{ave\_dist}}$.

[4]There are clearly other places where learning could play a

3.2 Deciding what to bid to become an AiCoop

When agents receive a request to participate in a CT they submit a bid based on the amount of reward that they would require to compensate them for deviating from their current goal. They also submit their current distance away from the CT square. The agents' bids are also affected by their social attitude factor; which determines whether they expect to receive more, less or the same average reward for social activities as they do for non-social ones. In this work, we only consider neutral agents that expect to receive the same from social actions as they do from non-social ones ([1] reports on the effect of this parameter). Other influences on their required reward are the amount of time spent in deviating from the CT square, an agent's average reward per time-step and the probability of success of the CM being proposed.

To formalise this, consider an agent, A_i with average reward per time-step r_i. The agent calculates its *deviation*, i.e., the number of extra time-steps it requires to reach its ST if it goes via the CT square. Note that if, for example, the CT square lies directly on a path to the ST, the agent's deviation would be zero.

To compute the reward A_i requires from engaging in coordination over the CT, it takes into account the compensation both for its deviation and for the possibility that the CM might fail. Thus, we have:

$$\texttt{bid}_{ij} = \frac{r_i \times deviation_i}{p_j} \quad (3)$$

The agent submits its bid to coordinate and its distance from the CT square. If an agent is selected to coordinate, it adopts the CT as its current goal. Its ST is only readopted after the CT has been accomplished; in particular, if it should arrive at the ST square en route for the CT, it does not receive its reward until it returns there.

3.3 Deciding which AiCoop bids to accept

Once the AiC has received bids from all agents, it selects the set that maximises its surplus reward, given the new (definite) information it has received (cf. the approximation in section 3.1). For each agent, A_i, the AiC knows the amount of reward it will require ($\texttt{bid}_{ij}$) and the time it will take to arrive (T_i).

Since all AiCoops need to be in the cell at the same time to accomplish the CT, AiC needs to pay an additional award to those AiCoops that have to wait in the CT square. AiC calculates this reward by selecting the agent that will take the longest time to arrive from the set of bids received. From this, it can determine the maximum time, $\texttt{waiting\_time}_{ij}$, that each agent will spend in the cell. For this time, the waiting AiCoops are compensated at a pre-specified rate q.

Formally, AiC calculates the $\texttt{cost\_bid}_{ij}$ based on the reward each agent requires and the reward AiC has to pay for the $\texttt{waiting\_time}_{ij}$:

$$\texttt{cost\_bid}_{ij} = \texttt{bid}_{ij} + T_i \times r + \texttt{waiting\_time}_{ij} \times q$$

Next it selects the m bids with the minimum cost:

$$\mathcal{S} = \min[\texttt{cost\_bid}_{ij}]$$

From this $\mathcal{S}$, it selects the furthest bid (i.e., $maxT = \max_{i \in \mathcal{S}}[T_i]$) and calculates its expected surplus:

$$\texttt{surplus}_{ij} = p_j \times R - \texttt{bid}_{ij} - r \times (t_j + maxT) \quad (4)$$

Now, it may be the case that no bids are received which give a positive surplus. Even though the chosen CM had an expected surplus, by chance it may be that no agents are sufficiently near to provide reasonable bids. In such a situation, the AiC returns to step [2] of the protocol (Figure 1) since, although it has been unlucky, its state is essentially unchanged and attempting to coordinate again is still likely to produce a surplus.

4. THE ROLE OF LEARNING

The previous section highlighted the potential role of learning in deciding which CM to select. This section will now describe how learning was actually introduced. We decided to employ a reinforcement learning (RL) technique [6]. A reinforcement-based approach is appropriate because we are concerned with agents pursuing goals and obtaining rewards according to how effectively those goals are accomplished. Within this class, Q-learning [17] was chosen because it is an online algorithm which does not require a model of the environment and thus it is well suited to our dynamic and unpredictable scenario.

Here we employ the classical model of Q-learning which consists of:

- a finite set S of states s of the world ($\mathsf{s} \in \mathsf{S}$);
- a finite set A of actions a that can be performed ($\mathsf{a} \in \mathsf{A}$);
- a reward function $\mathsf{R} : \mathsf{S} \times \mathsf{A} \rightarrow \mathsf{r}$.

An agent's goal consists of learning a policy $\pi : \mathsf{S} \rightarrow \mathsf{A}$ that maximises the average sum of its rewards V:

$$V[\mathsf{r}_t + \gamma \mathsf{r}_{t+1} + \gamma^2 \mathsf{r}_{t+2} + ...] = V \sum_{i=0}^{\infty} \gamma^i \mathsf{r}_{t+i}$$

where $0 \leq \gamma < 1$ is the discount factor[5]. Thus, the agent's task is to learn the optimal policy π (i.e. $argmax_\pi V^\pi(\mathsf{s})$, $\forall(\mathsf{s})$).

Assume that agents always perform the cycle of being in particular state s, then they select and perform an action a, which causes that agent to enter a new state s' and receive an immediate payoff (reward $\mathsf{r}(\mathsf{s}, \mathsf{a})$). The Q-learning algorithm is based on the estimated values of the agent's state (s)-action (a) pairs, called $\mathsf{Q}(\mathsf{s}, \mathsf{a})$ values. Based on this experience, the agent updates its $\mathsf{Q}(\mathsf{s}, \mathsf{a})$ values using the formula:

$$\mathsf{Q}(\mathsf{s}, \mathsf{a}) = (1 - \alpha_n) \times \mathsf{Q}(\mathsf{s}, \mathsf{a}) + \alpha_n[\mathsf{r} + \gamma \times max_a \mathsf{Q}(\mathsf{s}', \mathsf{a}')]$$

where α is the learning rate (decreasing with time by calculating it with the number of times a $\mathsf{Q}(\mathsf{s}, \mathsf{a})$ value is visited $visits\_QValue$: $\alpha_n = \frac{1}{1+visits\_QValue}$). $\mathsf{Q}(\mathsf{s}', \mathsf{a}')$ represents the maximum $\mathsf{Q}(\mathsf{s}, \mathsf{a})$ value obtained by any previous action a' to arrive to state s'.

When agents select their next action to execute, they have to balance their decision between selecting an action

role, but here we concentrate on these decisions since they are the major ones with respect to reasoning about coordination mechanisms.

[5]The discount factor determines the value of future rewards. A reward r received t time steps in the future is worth only γ^{t-1} times what it would be worth if it were received immediately. As γ approximates 1, the function takes future rewards into account more strongly.

that, when performed in the past, brought about a positive reward, and an action that has not yet been performed, and therefore is associated with an uncertain reward ("exploitation versus exploration" [6]). For experimental evaluation purposes, we used $\gamma = 0.90$ (which means that the agent is reasonably farsighted) and as an exploration function $f(u, n)$ [9]:

$$f(u,n) = \begin{cases} R^+ & \text{if } n < N_e \\ u & \text{otherwise} \end{cases} \quad (5)$$

which returns the weight of a $\mathsf{Q}(\mathsf{s},\mathsf{a})$ value based on the number of times (n) this policy has been visited. The action associated with the best $f(u, n)$ value is selected to be performed. R^+ is the best possible reward that an agent can obtain in a given state, N_e corresponds to the number of times that agents should try a particular action-state pair and u represents the utility of a $\mathsf{Q}(\mathsf{s},\mathsf{a})$ value.

In this work, our objective is to evaluate the effect of learning on the agents' decision making about CMs. To do this, we will compare the performance of agents which use a Q-learning algorithm (RL) with those that do not (NL). Here the key difference is how the agents select the CM with which they will attempt coordination (step [2] in the protocol specified in Figure 1). For the remaining steps of the protocol, both RL and NL agents employ the decision making procedures outlined in Section 3 to make agreements when `surplus` (equation (4)) is positive given the set of `bid`s (equation (3)) it received.

In more detail, when an agent finds a CT, it calculates the expected surplus (equation (2)) of each CM at its disposal. With NL the agent simply chooses the one with the best `ave_surplus`. With RL it exploits-and-explores (equation (5)) the set of CMs. When using RL, the reinforcement is used to measure the benefit of having selected a particular CM which corresponds to the surplus gained by achieving a CT using the CM chosen[6] after paying the AiCoops. This means the agent-state corresponds to the abstraction of the particular situation that agents experience when a CT is found (for example, the agent role, position in the grid and so on); the agent-action represents the set of options an agent has at its disposal (i.e. the set of coordination mechanisms it can select, including the null CM) and the reinforcement is modelled as the reward obtained by selecting the particular CM. Thus, the idea is that with Q-learning the agents will eventually learn the policy (after exploring sufficient situations) which allows them to know which CM to choose given a specific situation/state.

It is clear that the reinforcement is a central element in the process of learning because it is the mechanism to praise or blame if a good or bad action is performed. Thus, we decided to consider alternative values and moments to provide the reinforcement. The role of the reinforcement is to assess the evaluation performed on the choice of CM. To this end, the `ave_surplus` corresponds to the predicted value that an AiC expects to obtain by selecting a particular CM. When the negotiation phase is finished, the AiC receives firm information from the other agents and it is in position to evaluate this prediction. Thus, the AiC compares its predicted `ave_surplus` with the firm value negotiated (i.e. the `surplus`, equation (4)). If the prediction is close enough ($\pm$ 25%) to the real information, a strong reinforcement is made; but if it is not that close, a negative reinforcement is made[7].

In addition to this basic reinforcement method, we wanted to see whether the coordination decision making could be improved if the AiC builds a model of the other agents in the environment. That is, can RL agents improve their prediction of `ave_surplus` if they have a model of the other agents? To evaluate this, we do not need to construct a complex representation of the other agents but, rather, we can simply record the key variables that are crucial to coordination decisions. In particular, we decided to explore $r_$AiCoop in equation (1) and we let AiC calculate the value of $r_$AiCoop by averaging the bids it receives from the other agents (in contrast to using the AiC's own average reward).

In summary, the agents' performance will be analysed using the following algorithms:

- RL1 agents learn to select a particular CM according to the profit gained by accomplishing CTs with a particular CM.
- RL2 agents learn to select a particular CM according to the accuracy with which they predict the `ave_surplus` (which is based on the tailoring of $r_$AiCoop to their prevailing circumstances).
- NL agents do not engage in learning activities.

To finish the discussion on the role of learning in our model, it is necessary to specify the features of the environment in which the algorithms will be tested. Two scenarios have been designed: scenario1 in which all AiSs in the environment become AiCoop by submitting a bid which is calculated by equation (3) and scenario2 in which AiSs calculate their bids in the same way but they vary the result by a random factor. The reason for this change is that in the general case AiCs face a great deal of uncertainty in predicting this value. Thus the random element mirrors environments in which predictions are less accurate. Together, these two scenarios constitute a reasonably static environment in which good predictions can be made and a more dynamic environment in which predictions are inherently less accurate.

5. EXPERIMENTAL EVALUATION

The main hypothesis we seek to evaluate in this section is whether agents coordinate more effectively in our scenario using the reinforcement based algorithms. To measure the benefits of introducing or refining an agent's abilities in our model, a set of experiments were designed as a formal methodology to provide information about the experimental variables. In order to test and to verify the hypothesis questions we employ statistical inference methods, in particular analysis of variance (ANOVA) is used to test hypotheses about differences between the means collected. The null hypothesis (H0) of equal means can be rejected when the procedure reveals for all experiments that the differences among means are significant ($p < 0.05$) or might be accepted in the contrary case. That is, the execution of an algorithm in a specific environment generates a set of values for the experimental variables that can be analysed under the same

[6]Actually, accomplishing CTs is the only case considered. Even though agents achieve ST tasks, this information is not considered as reinforcement since it is not relevant to the agent's decisions about CMs.

[7]The absolute value of `ave_surplus` is used to provide the reinforcement in either the positive or the negative direction.

circumstances and situations in order to probe hypotheses using ANOVA.

In this case, the experimental variables were: Total Agent Reward obtained from its ST and CT tasks, which is normally called agent utility (AU), and the Total number of CTs accomplished (TCT). The following simulation variables were fixed for all the experiments: size of the grid (10x10), duration (50,000 time units)[8], number of CTs in the grid at any one time (3), number of agents in the environment (5), ST reward (1), CT reward (20), maximum number of agents needed to achieve a CT (3), coordination mechanisms considered by an agent (CM(0,0.6), CM(15,0.7), CM (30,0.8), CM(45,0.9) and CM(60,1.0)[9]). The experiments described collect the results of the experimental variables averaged over 10 simulation runs.

To probe the acceptance of the main hypothesis, all the hypotheses presented below must be rejected (meaning that the hypothesis of equal means is false) and the values of the experimental variables of a particular learning algorithm should produce significantly better results than those obtained with NL. Therefore, the following hypotheses must be tested in scenario1 and scenario2:

H1: the agent utility obtained by performing a reinforcement based algorithm is the same as that obtained by agents which use the NL algorithm.

H2: the number of CTs achieved by agents by means of either reinforcement learning algorithm is identical to that of agents using NL.

H3: the agent utility obtained by RL1 is the same as that of RL2 (evaluated in the case where H1 rejected).

H4: the number of CTs accomplished by RL1 is identical to that of RL2 (evaluated in the case where H2 is rejected).

Table 1 presents a summary of the results obtained by performing ANOVA on the data collected by each of the algorithms in scenario1. Let's first analyse the agent utility hypothesis. H1 is rejected, meaning that the performance of the algorithms does have a significant effect on the AU obtained. To understand this result, a post-analysis of the AU values obtained by each algorithm was necessary. Here, the interesting conclusion is that the performance of NL is better by a statistically significant amount ($AU_{NL} = 10,138.30$) than RL1 and RL2 ($AU_{RL1} = 9,413.58$, $AU_{RL2} = 9,399.00$). Furthermore, comparing the performance of RL1 and RL2 in H3 (Accepted), it is concluded that the value and the moment of praising or blaming agents does not have any effect on the AU obtained.

H2 evaluates the effectiveness of achieving CTs. This hypothesis is accepted which means that the total of CTs achieved does not depend on the algorithm executed. This is an important result to analyse in detail. In particular, why does NL perform better if all the algorithms achieve the same number of CTs? One reason is that NL accomplishes more ST tasks than the other algorithms. Figure 3 (left part) shows that the reward obtained by achieving STs is the biggest part of the total reward. Another, and more important reason, is that in this scenario it is highly expensive (due to the set-up cost of the CMs) to invest in a CT when there is some uncertainty about achieving it. With NL, it seems that AiC can make a good prediction of the `ave_surplus`. Meanwhile with RLs, the agents evaluate and explore the CMs and when they learn what is and what is not a good decision, the time has elapsed[10].

Hypothesis to evaluate	p	Outcome
H1: AU_{RL1}=AU_{RL2}=AU_{NL}	0.000	Rejected
H2: TCT_{RL1}=TCT_{RL2}=TCT_{NL}	0.876	Accepted
H3: AU_{RL1}=AU_{RL2}	0.590	Accepted
H4: Not evaluated		

Table 1: scenario1, result of ANOVA

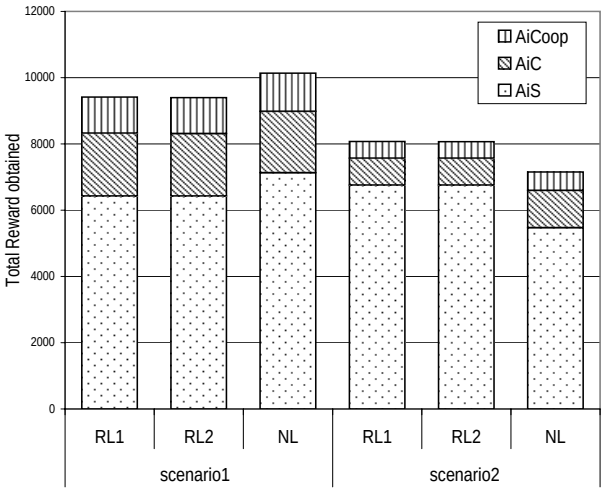

Figure 3: Reward obtained by agent role

Contrary to our intuition, the two versions of Q-learning (RL1, RL2) do not have any effect on the agents' performance. Analysing in detail, we find that this is because there is no change of information between the time agents make agreements and the time when they achieve the task. That is, no events occur which allow agents to change the reinforcement. Examples of events that could make a difference are if there are any decommitments or delays from AiCoops. The behaviour of the RLs is exploring and exploiting the actions and receiving a reinforcement which praises or blames the actions performed. Thus the two versions of RLs are simply reinforcing the same actions but with different values.

Turning now to the more dynamic environment of scenario2. We tested the same set of hypotheses and the results are summarised in Table 2. First, we analyzed the hypothe-

[8]We decided to evaluate for a fixed duration because in this scenario time counts and agents win reward at each timestep. Thus, it is reasonable to compare the behaviour of all algorithms under the same parameters. The duration selected is sufficient for the learning algorithms to converge.

[9]These CMs were selected because previous results have indicated that these are the main ones that are selected by the agents in this setting [1].

[10]The convergence time for the RLs is a combination of the learning rate, the exploration and exploitation function, the state representations and so on. It was not our objective to hand tune all these parameters to reduce the convergence time in particular cases. Rather, we fixed the values of all parameters and kept them constant in both Q-learning implementations.

ses related with agent utility (H1, H3). Similarly to the results obtained in Table 1, we conclude that applying RL and NL produces distinctive results (H1 is rejected). But, conversely to Table 1, in this point RL1 and RL2 get significantly better results ($AU_{RL1} = 8,063.30$ and $AU_{RL2} = 8,067.98$) than NL ($AU_{NL} = 7,151.12$). Additionally, observing the ANOVA result of both RL algorithms, we make the same conclusion as we did in scenario1; namely, giving the reinforcement before and after the negotiation phase makes no difference to the final reward obtained by agents (H4 is accepted).

Hypothesis to evaluate	p	Outcome
H1: AU_{RL1}=AU_{RL2}=AU_{NL}	0.000	Rejected
H2: TCT_{RL1}=TCT_{RL2}=TCT_{NL}	0.000	Rejected
H3: AU_{RL1}=AU_{RL2}	0.835	Accepted
H4: TCT_{RL1}=TCT_{RL2}	0.940	Accepted

Table 2: scenario2, result of ANOVA

With reference to the values of TCT (H2 and H4). The hypothesis of equal means of H2 is rejected and H4 is accepted. There is a significant impact on the TCT achieved when performing RLs or NL, where the results are 65 to RLs and 84 to NL. The relevant aspect to discuss now, though, is why NL obtains a lower AU despite achieving more CTs? Being consistent with the previous explanation, and observing Figure 3 (right section), it can be seen that the reward gained by achieving CTs for NL-AiCs is bigger than that gained by RL-AiCs because they achieve more CTs. However, the time invested on them was not sufficient to recover the reward that was being gained by RL-AiSs (RL-AiSs obtained in total approximately 84% of the total reward by accomplishing STs and NL-AiSs achieved 76%). The reason for this result is that agents invest a significant amount of time to set up the CM and, in the end, the AiCoops often request higher bids than those in scenario1 (meaning the AiCs' profit is reduced). Thus, RLs perform better because they are more certain about when to invest time in a CT and, more importantly, when not to do it (because it is not worth it). They then use this time to take advantage of pursuing STs.

Before making a general conclusion about the irrelevance of the different options of reinforcement and modelling other agents, Figure 4 shows the performance of the algorithms from a different perspective. Here, we analyze the performance of agents by evaluating the average surplus reward expected (equation 2). We undertake this analysis because is important to know if it really worthwhile having adaptive agents instead of just hand tuning the agents' decision making procedures about which CM to select. To be precise, NL agents select the CM based on `ave_surplus`, but RLs agents do not (RL2 uses the result of this evaluation just as a reinforcement). Thus, the idea is to count the number of cases in which the action selected by either RL algorithm coincides with the outcome indicated by equation 2. In other words, given a specific situation, RL agents exploit the CMs and evaluate this formulation, then they analyze the `ave_surplus` of that CM. If it is positive, then a case of TT is encountered (because the CM was exploited and it coincided with the decision based on `ave_surplus`). On the other hand, if it is negative, the CM was exploited but this did not correspond with the evaluation of equation 2 (case TF). Additionally, we have the cases in which the CM was not exploited and this corresponds (FF) or not (FT) with the decision based on `ave_surplus`. It is clear that with NL all the cases have to be consistent because they attempt coordination based on this evaluation. As for the RLs, it will depend on how this equation is calculated. Both NL and RL1 agents employ equation 2 using their own r values for r_AiCoop, whereas RL2 uses the value that it has learned based on previous encounters. Figure 4 shows only the TT and FF cases since the others (TF and FT) are not relevant in this discussion. The first thing to notice is the clear difference between RLs, which is due to the different ways in which the `ave_surplus` is calculated. The second observation is that RL2 has more FF cases, meaning that most of the time the action it performs is to not attempt coordination and in these cases this decision is consistent with that based on the `ave_surplus`. The reason for this result is that RL2 is modelling others using r_AiCoop which helps it to make a good prediction of the other agents. Given this, the obvious question to ask is: could NL perfom better by modelling others in the same way as RL2? Here, the answer is no. The system utility obtained by NL agents in scenario1 degrades considerably when agents use r_AiCoop ($AU_{NL} = 7,575.28$) and in scenario2 it raises slightly ($AU_{NL} = 7,578.42$). Despite the improvement in scenario2, it is not sufficient to have H1 accepted. If the NL agents' predictions of `ave_surplus` are too low (being optimistic about the possible future cooperative agents), they will always initiate coordination even in situations where it not the best decision to make. However, if their predictions are too high (being pessimist) they will never attempt coordination. Thus, we can conclude that having learning agents which explore and exploit actions, taking advantage of long-term profits, is the more reasonable thing to do in dynamic environments because agents can not be certain about the others' actions.

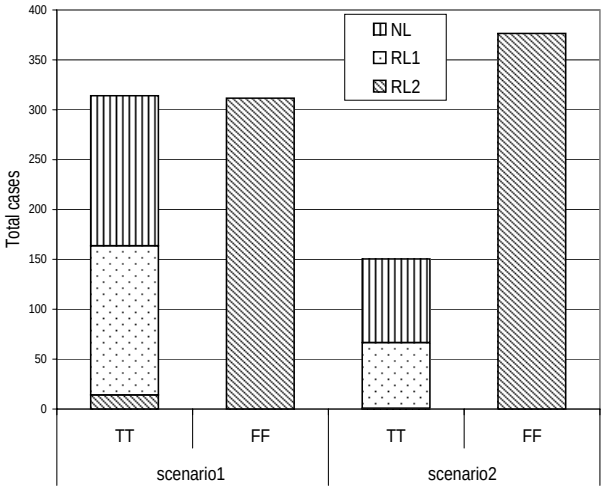

Figure 4: RL and `ave_surplus` action selection

6. RELATED WORK

A vast literature has been produced in recent years concerning the use of learning techniques (particularly Q-learning) in multiagent systems [11, 14]. The focus has been mainly on two aspects. In the first one, an agent's goal is to learn about the other agents and their environment by observation in order to predict their behaviour or to produce a model of them [7, 5, 2]. In the second case, Q-learning has been

applied to learn how to coordinate or cooperate to achieve common goals by using specific strategies [16, 10]. The success in these two lines of research has mainly been to improve the cooperation or coordination between the agents in the environment. While this is clearly an important issue to address, we are more concerned with learning to select particular coordination mechanisms. Much less work has been done in this area. The most relevant work to our own is the COLLAGE [8] and LODES [15] systems. The objective in both systems is to improve coordination by learning to select a coordination strategy in appropriate situations. However the aspects each system addresses are different and their findings are complementary. LODES is more interested in having agents capable of learning the key information that is necessary to improve coordination in specific situations. In COLLAGE agents learn how to choose the most appropriate coordination strategy given a particular situation. Thus, LODES focuses on "what information to learn" and COLLAGE on "learning the situation where to use a coordination strategy". It is important to notice that both systems are concerned with the detailed activities of coordination as part of the learning process. For agents to solve a particular coordination problem, they have to solve all the interrelations and dependencies between their actions. Thus agents first plan the actions to perform and then execute them. To solve this, both systems have to handle deep knowledge: about the domain in the case of LODES and about coordination strategies with COLLAGE. In our case, however, the research aim is broadly similar, but our assumptions are different and we deal with the problem using alternative solutions. In our framework, agents are endowed with a set of decision making procedures to select adequate coordination mechanisms. By dealing with an abstract set of such mechanisms, we consider it more important to have agents that have the capacity to take decisions about coordination, rather than dealing with all the interactions between them. We leave the latter to the details of the subsequent tasks of the associated protocol. Furthermore, we believe that as agents are increasingly being required to deal with more dynamic issues then online learning will become more important. COLLAGE, by contrast, uses instance based learning techniques in which there is a phase of recovery of examples and one of training. Consequently, the system has well defined moments in which these phases are performed which gives the additional problem of determining when each phase should finish.

7. CONCLUSIONS AND FUTURE WORK

This paper analysed the use and the efficacy of agents learning about making decisions about when and how to coordinate. We showed that learning improves the decision making when agents are uncertain about the other agents' actions. This improvement occurs because agents learn to recognize the situations where the most profitable actions must be selected. We also showed that learning was ineffective when agents operate in more static environments in which they can make accurate predictions about their environment and other agents.

Speaking more generally, we believe it is important to develop techniques that enable agents to coordinate flexibly in dynamic and unpredictable environments. Although several of the detailed aspects of our decision procedures are specific to our grid-world scenario, we believe that the general processes and structures are suitable for reasoning about coordination mechanisms in more general domains. In particular, the issues of when and how to exploit learning techniques to allow agents to take decisions based on their experience is a key aspect that needs broader investigation. To this end, the results presented here can be viewed as a first step in that direction.

For the future, we aim to extend the use of learning to cover other aspects of the agent's decision framework. In particular, to learn the decision about how much to bid in a request for coordination, when to become an AiCoop (equation 3) and which bids to accept (equation 4). We also intend to allow agents to construct more sophisticated models of one another and to have the ability to vary the details of this modelling according to the agent's coordination context.

8. ACKNOWLEDGMENTS

The first author acknowledges the funding of Mexico's National Council of Science and Technology, CONACyT.

9. REFERENCES

[1] R. A. Bourne, C. B. Excelente-Toledo, and N. R. Jennings. Run-time selection of coordination mechanisms in multi-agent systems. In *Proc. of the 14th European Conf. on AI*, 2000.

[2] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proc. of 15th Nat. Conf. on AI*, 1998.

[3] E. H. Durfee and V. R. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167–1183, 1991.

[4] C. B. Excelente-Toledo, R. A. Bourne, and N. R. Jennings. Reasoning about commitments and penalties for coordination between autonomous agents. In *Proc. of the 5th Intl. Conf. on Autonomous Agents*, 2001.

[5] J. Hu and M. P. Wellman. Online learning about other agents in a dynamic multiagent system. In *Proc. of 2nd Intl. Conf. on Autonomous Agents*, 1998.

[6] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of AI Research*, 4(4):237–285, 1996.

[7] Y. Nagayuki, S. Ishii, and K. Doya. Multi-agent reinforcement learning: An approach based on the other agent's internal model. In *Proc. of the 4th Intl. Conf. on MultiAgent Systems*, 2000.

[8] M. V. N. Prasad and V. R. Lesser. Learning situation-specific coordination in cooperative multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 2(2):173–207, 1999.

[9] S. J. Russell and P. Norvig. Reinforcement learning. In *AI: A Modern Approach*, chapter 20, pages 598–624. Prentice Hall, 1995.

[10] S. Sen, M. Sekaran, and J. Hale. Learning to cooperate without sharing information. In *Proc. of the 12th Nat. Conf. on AI*, 1994.

[11] S. Sen and G. Weiss. Learning in multiagent systems. In G. Weiss, editor, *Multiagent Systems*, chapter 6, pages 259–298. The MIT Press, 1999.

[12] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proc. of the 10th Nat. Conf. on AI*, pages 276–281, 1992.

[13] R. G. Smith and R. Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):61–70, 1980.

[14] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 3(8):345–383, June 2000.

[15] T. Sugawara and V. Lesser. Learning to improve coordinated actions in cooperative distributed problem-solving environments. *Machine Learning*, 33(2/3):129–153, 1998.

[16] M. Tan. Multi-agent reinforcement learning: Independent vs cooperative agents. In *Proc. of the 10th Intl. Conf. on Machine Learning*, 1993.

[17] C. J. C. H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 1992.