Learning When and How to Coordinate

Cora B. Excelente-Toledo *
National Laboratory of Advanced Computer Science, LANIA
Rébsamen No. 80, Col. Isleta, C.P. 91090
Xalapa, Veracruz, MEXICO
cora@lania.mx

Nicholas R. Jennings
School of Electronics and Computer Science
University of Southampton
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 predictions about the other agents in the environment are 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.

1 Introduction

Effective coordination is essential if autonomous agents are to achieve their goals in a multiagent system (MAS). 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 protocols and structures have been developed to address the coordination problem. These range from long-term social laws [26], through medium term mechanisms such as Partial Global Planning [7], organizational structuring [12] and market protocols [19] to one-shot (short-term) mechanisms like the Contract-Net Protocol [28].

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

*This work was done while the first author was a member of the Intelligence, Agents, Multimedia Group at the University of Southampton.

more likely to fail. Moreover, a coordination mechanism that works well in a reasonably static environment will often perform poorly in a dynamic and fast changing one. In short, there is no universally best coordination mechanism [13].

Given this situation, we believe it is important for the agents to have a variety of coordination mechanisms, with varying properties, at their disposal so that they can then select the particular mechanism that 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. However, to date, the choice of which coordination mechanism to use in a given situation is something that the designer typically imposes upon the system at design time (e.g. in a given application a particular social law will be used or it will be decided that all coordination activities will be handled by the contract net protocol). This means that in many cases the coordination mechanism that is employed is not ideally suited to the agents' prevailing circumstances. This inflexibility means that the performance of both individual agents and the overall system may be compromised (a fact exacerbated for the open and dynamic nature of the environments in which agent-based solutions are often deployed [16]).

To rectify this situation, our aim is to develop agents that can reason about the process of coordination and then select mechanisms that are appropriate to their current situation. That is, *the choice of coordination mechanism is made at run-time by the agents that need to coordinate.* We claim that fixing on a single coordination mechanism at design time is inappropriate, especially 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 [2, 8, 11]. To circumvent this problem and 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, our previous work has developed and evaluated a reasoning framework to achieve this [2, 9, 11]. 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.

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. 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.

The remainder of this paper is structured as follows. Section 2 details our specific coordination scenario. Section 3 formalises the decision procedures of the agents. Sections 4 and 5 present and evaluate the role of learning in this context. Section 6 deals with related work and, finally, section 7 concludes and presents the areas of further work.

2 The Coordination Testbed

The testbed domain follows the description of [2] and [8] contains a detailed justification for the choice of the scenario and the various design decisions within it. In summary, however, it is a grid world in which a number of autonomous *agents* (A_i) perform tasks for which they receive units of *reward* (R_i). 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 m 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 of the meta-data: set up cost (in terms of time-steps) and chance of success [2]. 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. Within this broad framework, Figure 1 highlights the specific decisions which have to be made (see Section 3 for more details) and gives the protocol the agents follow at each time-step.

[1] Agents arrive at a square. If the AiS arrives at its ST cell, its goal is attained, it receives the reward and updates its goal. If the AiCoop arrives at the CT cell, it notifies the AiC that it has arrived. The CT is achieved and the rewards are paid to the AiCoops.

[2] If the AiS finds a CT it must decide if it wants to become an 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. If the AiC finds a new CT, it ignores it.

[3] If the 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 the AiS's bid is accepted, it adopts CT as its new goal. The AiC does not 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.

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 AiS respectively (evaluation phase). 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.

This initial presentation involves several simplifying assumptions; in particular common knowledge, a deterministic environment and straightforward coordination mechanisms. However, the framework is also intended to be flexible so that these and other assumptions will be relaxed in future work (see [9], [10] and Appendix of [8]). To model dynamism, unpredictability and open

[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.

[2] Other attributes could undoubtedly be added to this list (such as quality of coordination, robustness, overhead limitations, communication cost and so on), but here we focus on those that we believe are necessary (if not sufficient) for our purposes (as advocated in [18]). See further discussion in [8].

features in this grid world, the elements in the environment change their values at execution time. Some examples are the changing of the tasks' rewards (both for STs and CTs); the frequency with which tasks appear and disappear in the grid; the changing number of agents in the environment; and the number of agents needed to achieve a CT. The main consequence of these variations is that they generate an environment in which agents face difficulty in estimating the decisions of other agents. Thus, agents have to take decisions based on factors that cannot be predetermined.

To clarify the protocols associated to each role and the previous description of the scenario, Figure 2 shows a $[10 \times 10]$ 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 it 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 $[9, 8]$.

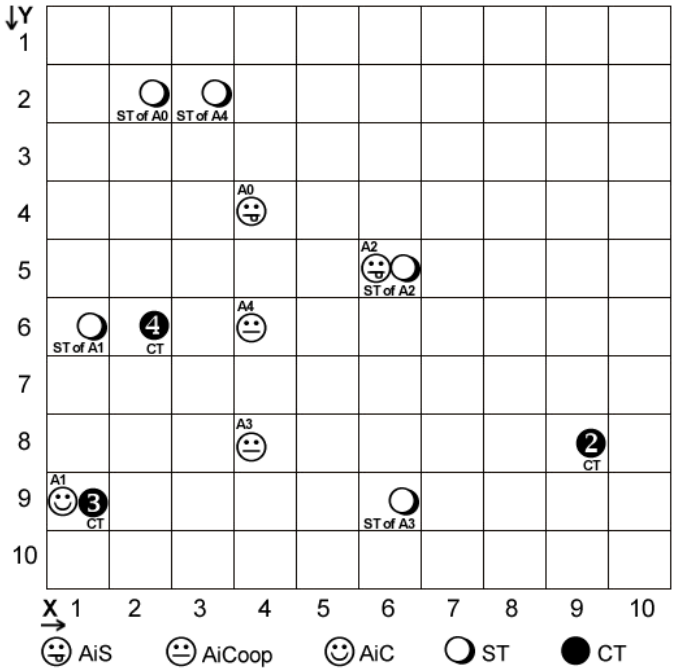

Figure 2: Scenario with agent roles.

3 The Agent's Decision Making Procedures

In our previous work we have developed and evaluated a decision making framework for reasoning about whether and how to coordinate in this domain [2, 9, 11]. 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.

In this context, the agents' aims are to maximise their reward; in particular their average reward per unit time. To this end, each agent keeps track of its own average reward, termed its *reward rate*, and 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 of 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 effect 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 then assesses the possible CMs on the basis of how long before the task can be performed and how much reward it is likely to obtain after deducting the expected reward requirement of the other agents. In the former case, it considers both the set up time and the average distance away each agent is situated, whereas the latter value is based on the amount of time agents must spend deviating from their path and the CM's probability of success. This assessment determines the amount of surplus reward the agent 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 then selects the CM that maximises this surplus [3].

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.

First, the average distance in each direction of a random square from a point $[x, y]$ is given by: $\texttt{x\_distance}(x) = \frac{2x(x-1)+M(M+1-2x)}{2M}$ and $\texttt{y\_distance}(y) = \frac{2y(y-1)+N(N+1-2y)}{2N}$.

Hence the ave_dist of any given agent from $[x, y]$ is: $\texttt{ave\_dist}(x, y) = \texttt{x\_distance}(x) + \texttt{y\_distance}(y)$. The average distance, ave_dist, of an agent from its ST is the average distance between two random points on the grid. This is given by averaging $\texttt{ave\_dist}(\texttt{x}, \texttt{y})$ over all x and y: $\texttt{ave\_dist} = \frac{\sum_{x=1}^{M}\sum_{y=1}^{N}\texttt{ave\_dist}(x,y)}{M \times N}$.

Finally, the average deviation of an agent to assist in a CT at square $[x, y]$ and then go on to its ST, as compared with going straight to its ST, is given by: $\texttt{ave\_dev}(x, y) = 2 \times \texttt{ave\_dist}(x, y) - \texttt{ave\_dist}$

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 ($r = \frac{S}{\texttt{ave\_dist}}$), the cost of CM_j is given by:

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

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:

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

[3] 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.

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

$$\mathsf{ave\_payoff}_j(x,y) = p_j \times R$$

Based on these figures, the AiC can assess the expected surplus reward from coordinating over the CT at (x,y) using $\mathrm{CM}_j = (t_j, p_j)$.

$$\begin{aligned}\mathsf{ave\_surplus}_j(x,y) = \mathsf{ave\_payoff}_j(x,y) - \\ (\mathsf{cost}_j(x,y) + (m \times \mathsf{ave\_bid}_j(x,y)))\end{aligned} \tag{1}$$

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).

$y\downarrow$					
1					
2			CT AiS_2		
3					AiS_1
4		ST_1			
5				ST_2	
$x\rightarrow$	1	2	3	4	5

Figure 3: Example of a coordination world grid.

To exemplify this decision procedure, consider the simple scenario (a grid size of $[5 \times 5]$) of Figure 3 at one instant in time with two agents (AiS_1 and AiS_2), two STs, one CT and two CMs: $\mathrm{CM}_1(3, 0.9)$ and $\mathrm{CM}_2(6, 1.0)$. AiS_2 finds a CT requiring one other agent with $R = 6$ at square $[3, 2]$. The average distance of other agents from $[3, 2]$ is 2.6. Since the average distance between two random squares is 3.2, the average deviation of any agent from $[3, 2]$ is 2. Assume that each ST has a reward $S = 2$, then the average reward per time-step of all agents is $\frac{2}{3.2} = 0.625$. The expected surplus reward of adopting each CM is given by:

$$\begin{aligned}\mathsf{cost}_1(3,2) &= (0.625 \times (3 + 2.6)) = 3.5 \\ \mathsf{ave\_bid}_1(3,2) &= \frac{(0.625 \times 1 \times 2)}{0.9} = 1.389 \\ \mathsf{ave\_payoff}_1(3,2) &= (0.9 \times 6) = 5.4 \\ \mathsf{ave\_surplus}_1(3,2) &= 0.511\end{aligned}$$

$$\begin{aligned}\mathsf{cost}_2(3,2) &= (0.625 \times (6 + 2.6)) = 5.375 \\ \mathsf{ave\_bid}_2(3,2) &= \frac{(0.625 \times 1 \times 2)}{1.0} = 1.25 \\ \mathsf{ave\_payoff}_2(3,2) &= (1.0 \times 6) = 6 \\ \mathsf{ave\_surplus}_2(3,2) &= -0.625\end{aligned}$$

Under these circumstances, AiS_2 decides to attempt coordination with CM_1 (becoming AiC) because it expects to obtain a profit. Note this is not the case with CM_2, where the negative result indicates there is not likely to be a surplus. Thus, in this case, if AiS_2 only had CM_2 at its disposal it would choose the null CM (expected surplus zero) and it would continue towards its ST.

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. Thus, an agent's required reward is determined by the amount of time spent on deviating from the CT square, its average reward per time-step and the probability of success of the CM being proposed [4].

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. Clearly, such an agent will be in a position to submit a very attractive bid, since the cost of coordinating is effectively zero.

Again by means of illustration consider the agents depicted in Figure 3. AiS_1 at $[5, 3]$ would take 4 time-steps to reach ST_1 at $[2, 4]$ directly, but 6 steps going via the CT at $[3, 2]$, a deviation of 2 time-steps. However, AiS_2 at $[1, 1]$ would take 7 time-steps to reach ST_2 at $[4, 5]$ directly, and also 7 steps going via the CT at $[3, 2]$; AiS_2 therefore has a deviation of 0.

To compute the reward AiS_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, the estimation of bid by agent i to participate in coordination is given by:

$$\mathsf{bid}_{ij} = \frac{r_i \times deviation_i}{p_j} \tag{2}$$

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 re-adopted after the CT has been accomplished.

3.3 Deciding which AiS 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 (bid_{ij}) and the time it will take to arrive (T_i).

The AiC's selection bid process is based on the calculation of the cost of each bid received. However, when more than two agents are 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). There are many ways of dealing with this situation (see discussion below). However to simplify the estimates of expected reward undertaken by the various agents, it is assumed the AiC pays an additional reward for the time elapsed. Thus, 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). The CT is achieved only when the AiC has received the confirmation of all m agents involved in the cooperation. 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.

Thus, to decide which bids to accept, the general idea is that AiC selects the m proposals with least cost (from the total bids received $\mathcal{B}$). It does this by considering the reward requested in the bid and the waiting time cost ($\mathsf{cost\_bid}$) and then it estimates its expected reward given this cost and its investment. Formally, AiC calculates the cost of each subset b of $\mathcal{B}$ with m elements of the form (bid_{ij}, T_i). From b, AiC selects the agent that will take the longest time to arrive (i.e., $maxT_b = \max_{(\mathsf{bid}_{ij}, T_i) \in b}[T_i]$), then it can determine the maximum time that each agent will spend in the cell. Finally, it approximates the cost of each bid based on the reward and the waiting time an AiC has to pay:

[4] Note that the AiSs use the actual values of the concepts discussed, whereas the AiC's task is to make a good approximation of these components through equation $\mathsf{ave\_bid}_j(x, y)$.

$$\mathsf{cost\_bid}_b = \sum_{(\mathsf{bid}_{ij}, T_i)} (\mathsf{bid}_{ij} + (maxT_b - T_i) \times q)$$

Bringing all this together, AiC estimates the surplus it expects to obtain by taking into account the cost of the selected bids and its own investment to wait for the last AiCoop to arrive. The bids selected belong to the subset b of $\mathcal{B}$ that maximizes the surplus given by:

$$\mathsf{surplus}_j = p_j \times R - \mathsf{cost\_bid}_b - r \times (t_j + maxT_b) \quad (3)$$

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 abandons the CT and returns to its ST.

4 The Role of Learning

The main focus of this paper is to give the agents the capability of learning to make the right decisions about their coordination problem. That is, we wish to endow the agents with the capability of learning the right situation in which to apply the right coordination mechanism. Specifically, the agent's decision making framework presented in the previous section and, in particular, the decision procedure outlined in section 3.1, allows agents to take decisions about when and which CM to select in order to achieve a CT. Since this procedure is the major one with respect to reasoning about coordination mechanisms, it is the one we concentrate on in terms of evaluating the role of learning [5].

In this work, we decided to employ a reinforcement learning (RL) technique [17, 31]. 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 [34] was chosen because it is an online algorithm that does not require a model of the environment and thus it is well suited to our dynamic and unpredictable scenario.

In this study, each reinforcement learning agent uses a Q-learning algorithm. In general terms, an agent's objective is to learn a decision policy that is determined by the state/action value function. The classical model of Q-learning 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 expected sum of discounted 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. Formally speaking, the discount factor determines the value of future rewards in the following way: a reward r received t time steps in the future is worth only γ^t times what it would be worth if it were received immediately. As γ approximates 1, the function takes future rewards into account more strongly. Thus, the agent's task is to learn the optimal policy π (i.e. $\arg\max_\pi V^\pi(\mathsf{s})$, $\forall(\mathsf{s})$).

In more detail, assume that an agent always performs the cycle of being in particular state s, then selecting and performing an action a that causes the 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

[5] There are clearly other places where learning could play a role. For example, an agent might learn the decision about how much to bid to become an AiCoop (equation 2) and which bids to accept (equation 3).

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:

$$\mathsf{Q}(\mathsf{s},\mathsf{a}) \leftarrow \mathsf{Q}(\mathsf{s},\mathsf{a}) + \alpha[\mathsf{r} + \gamma \times \max_{a'} \mathsf{Q}(\mathsf{s}',\mathsf{a}') - \mathsf{Q}(\mathsf{s},\mathsf{a})]$$

where α is the learning rate which determines the rate of change of the estimation and $\max_{a'} \mathsf{Q}(\mathsf{s}',\mathsf{a}')$ is the value of the action that maximises the Q function at state s.

However, there is still the problem of how agents select their next action to execute. They have to balance their decision between selecting an action that, when exploited in the past, brought about a positive reward, and an action that has not yet been explored and that consequently has an unknown reward ("exploitation versus exploration")[31]. In this work, we use a function that selects the action with the highest $\mathsf{Q}(\mathsf{s},\mathsf{a})$ value once all the actions have been explored a pre-determined number of times. Formally, the exploration function $f(\mathsf{Q}(\mathsf{s},\mathsf{a}),n)$ [23] equates to:

$$f(\mathsf{Q}(\mathsf{s},\mathsf{a}),n) = \begin{cases} R^+ & \text{if } n < N_e \\ \mathsf{Q}(\mathsf{s},\mathsf{a}) & \text{otherwise} \end{cases} \quad (4)$$

where n is the number of times $\mathsf{Q}(\mathsf{s},\mathsf{a})$ has been visited, R^+ is the optimistic estimate of best possible reward that an agent can obtain in a given state and N_e corresponds to the number of times that agents should try a particular action-state pair.

In summary, for experimental evaluation purposes, agents use a Q-learning algorithm with the following values: γ is assigned value 0.01 which means that the agent is trying to maximize immediate reward; α is decreasing with time by calculating it with the number of times a $\mathsf{Q}(\mathsf{s},\mathsf{a})$ value is visited $visits(s,a)$: $\alpha = \frac{1}{1+visits(s,a)}$. And equation (4) is used as exploration function [6].

Thus, our objective is then to evaluate the effect of learning on the agents' decision making about CMs. To do this, we will compare the performance of agents that use a Q-learning algorithm (RL) with those that perform no learning (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 $\mathsf{surplus}$ (equation (3)) is positive given the set of bids (equation (2)) it received.

In more detail, when an NL agent finds a CT, it calculates the expected average surplus (equation (1)) of each CM at its disposal. It then simply chooses the one with the best bid. With RL it exploits-and-explores (equation (4)) 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[7] 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 and the position in the grid); 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 or not selecting a CM at all. Thus, the idea is that with Q-learning the agents will eventually learn the policy (after exploring sufficient situations) that allows them to know which CM to choose given a specific situation/state.

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

RL agents learn to select a particular CM according to the profit gained by accomplishing CTs with a particular CM.

NL agents do not engage in learning activities.

[6] It is well known that the convergence time for a learning algorithm is highly based on the exploration and exploitation function [27], the size of the look up tables and the learning rate. Here, it was not our objective to hand tune all these parameters to reduce the convergence time in particular cases, nor to provide theoretical results in such areas. Rather, we fixed the values of all parameters and kept them constant in all Q-learning implementations

[7] 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.

To finish the discussion on the role of learning in this 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 (2) and scenario2 in which AiSs calculate their bids in the same way but incorporate a degree of random noise over and above this base line figure. 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 one in which predictions are inherently less accurate.

5 Experimental Evaluation

The main hypothesis we seek to evaluate is whether agents coordinate more effectively in our scenario using the reinforcement based algorithms. To measure such benefits in our model, a set of experiments have been 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 [5]. In other words, ANOVA tests the significance of the observations by accepting or rejecting the hypothesis formulated. The observations are the set of values for the experimental variables as the result of the execution of a particular algorithm in a given environment.

The following simulation variables were fixed for all learning experiments: size of the grid ($[10 \times 10]$), duration (500,000 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_1=(0,0.6), CM_2=(15,0.7), CM_3=(30,0.8), CM_4=(45,0.9) and CM_5=(60,1.0) [9]).

The experimental variables on which the analysis is based are: total agent reward obtained from its ST and CT tasks (AU), total agent reward obtained by agents in the Agent-in-ST role (AiS), total agent reward obtained by agents in the Agent-in-Charge role (AiC), total agent reward obtained by agents in the Agent-in-Cooperation role (AiCoop), and the total number of CTs accomplished (TCT). The experiments described collect the results of the experimental variables averaged over 10 simulation runs.

To accept our main hypothesis, the hypotheses presented below must be rejected, meaning that the values of the experimental variables of a particular learning algorithm should produce significantly better results than those obtained with no learning ones (the means are not equal). Therefore, the following hypotheses must be tested in scenario1 and scenario2:

H1: The AU obtained by performing a reinforcement based algorithm (RL) is the same as that obtained by agents that use the NL algorithm.

H2: The number of CTs (TCT) achieved by agents by means of a reinforcement based algorithm is identical to that of agents using NL.

To this end, we evaluate these hypothesis in the more static environment of scenario1 (Section 5.1) and the more dynamic environment of scenario2 (Section 5.2).

[8] We decided to evaluate over a fixed duration because in this scenario time counts and agents win reward at each time-step. 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 to optimal values.

[9] These CMs were selected because previous results indicated that these are the main ones that are selected by the agents in this setting [2, 10, 8, 11].

5.1 Learning to select a CM in a static environment

To start with, Table 1 and Figure 4 present 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. Here, the conclusion is that the performance of NL is better by a statistically significant amount ($AU_{NL} = 88,018.64$) than RL ($AU_{RL} = 81,064.28$); hence NL is noted as the winner. This can clearly be seen in Figure 4 (left side) in which the total reward gained by NL agents is higher than that for RL agents.

Hypothesis to evaluate	p	Outcome	Winner
H1: AU_{RL}=AU_{NL}	0.000	Rejected	NL
H2: TCT_{RL}=TCT_{NL}	0.000	Rejected	NL

Table 1: Agent's AU and TCT in scenario1: result of ANOVA

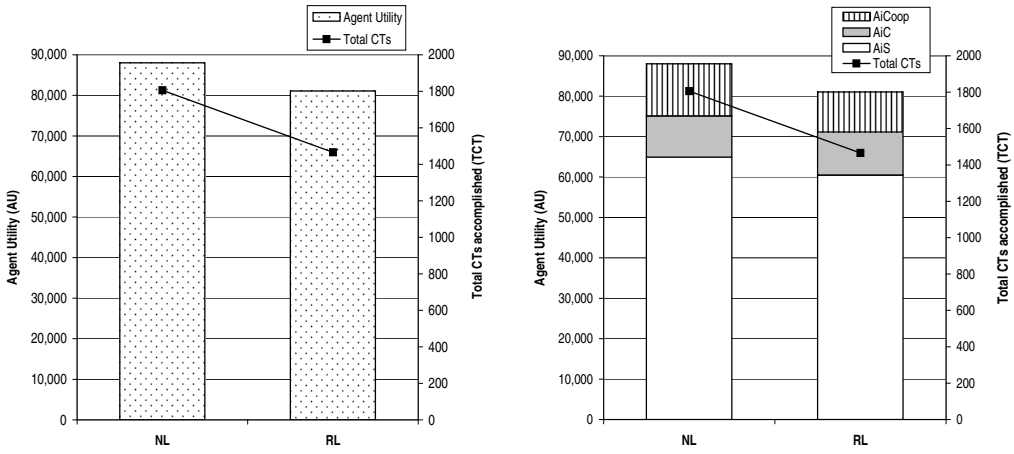

Figure 4: Contrasting agent's performance in scenario1.

H2 evaluates the effectiveness of achieving CTs. Again, this hypothesis is rejected which means that the TCT does depend on the algorithm executed. In line with the AU obtained by the agents, it can be seen that the more CTs that are achieved, the better the AU (H1 was rejected). However, this is an important result to analyse in detail, because a high number of CTs achieved need not necessarily lead to a better agent performance. In this scenario, agents firstly need to decide whether working cooperatively is more rewarding than working on their own and, secondly, and more importantly, they need to balance the time invested in setting up the CM with the final reward that it is likely to be achieved. Generally speaking, the agents that perform best are the ones that are effective at making this trade off and selecting the CMs in which the time to set up is well matched to the reward obtained with the CT. Thus, from the results obtained in this experiment, it seems that NL agents are not only accomplishing more CTs but are selecting the correct CM with which to maximise their AU. To verify this argument, Figure 4 (right side) and Table 2 explore effectiveness by agent role (again using ANOVA). Here, the results show that NL agents perform (statistically) significantly better in AiCoop and AiS roles than the corresponding RL agent roles (H3 and H5 are rejected). This good performance for the NL agents' roles is because they better balance their cooperative and individual behaviour than the RL agents do. Based on this, we would expect H4 to be rejected and have NL as the winner agent. In contrast, however, the AiCs do not make a significant difference to the reward obtained (H4 is accepted). The reason for this is because the same AiC reward might be obtained either by achieving a few number of profitable CTs or by accomplishing a higher number of less rewarding CTs. In terms of the selection of the CMs, this means that RL selected those CMs that have a high

probability of success and a corresponding high time to set up, leaving less time to achieve CTs. Meanwhile NL agents choose those CMs that take less time to set up and have lower probability of success. Thus, on the whole, NL performs better because it better balances its decision of when to go for the CT with the CM that allows it to gain more reward.

Hypothesis to evaluate	p	Outcome	Winner
H3: AiS_{RL}=AiS_{NL}	0.000	Rejected	NL
H4: AiC_{RL}=AiC_{NL}	0.247	Accepted	None
H5: $AiCoop_{RL}$=$AiCoop_{NL}$	0.000	Rejected	NL

Table 2: AU per agent's role in scenario1: result of ANOVA

One reason for this apparently poor performance of RL agents is that the results were obtained for a fixed period that included the exploration and exploitation phase, not once the learning agents had converged to the optimal policy (equation (4)). Recall that agents in the learning process need to try the CMs (exploration) even though some of them are not necessarily worth it (since this is a necessary step in the task of learning). To this end, it could be argued that if we allow RL agents to compete with NL agents once they know in which situation they should select which CMs, their performance would improve. To verify this, in the following experiment, the agents are initiated with the policies they have learnt from the past experiment and the simulation is run again. To clearly distinguish between both phases, in what follows we define a *training* phase as covering both the phases when exploration is occurring and after convergence has occurred, and the *acting* phase as when the RL agents only exploit the optimal policies (i.e. after convergence has occurred).

Hypothesis to evaluate	p	Outcome	Winner
H1: AU_{RL}=AU_{NL}	0.000	Rejected	RL
H2: TCT_{RL}=TCT_{NL}	0.000	Rejected	RL

Table 3: Agent's AU and TCT in scenario1 (acting phase): result of ANOVA

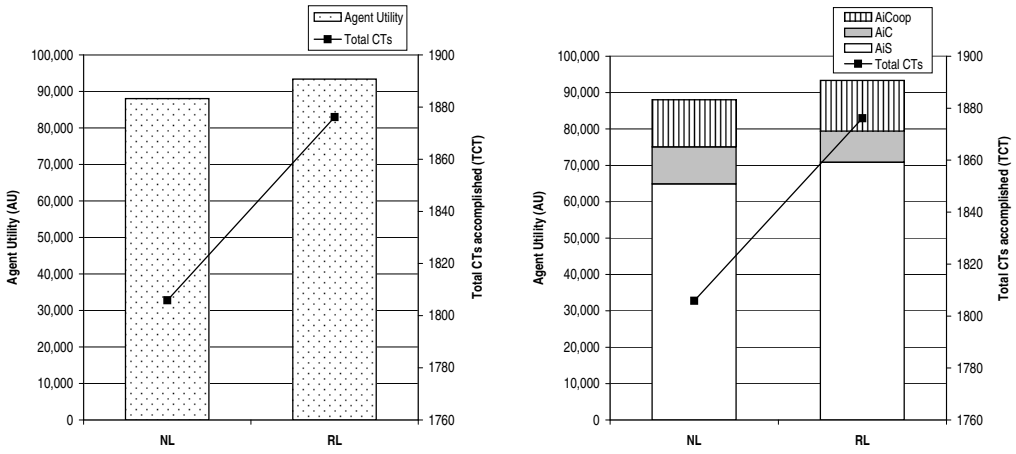

Figure 5: Contrasting agent's performance in scenario1 (acting phase).

For this new situation, tables 3 and 4 show the results of testing the same hypotheses as before with ANOVA and Figure 5 plots the data collected. The first thing to notice is that the RL agents perform (statistically) significantly better than the NL ones (see left side of Figure 5). RLs improve their decision making by selecting the optimal policy and consequently increasing their AU by 15% (H1 is rejected). The second aspect, and the main reason for the improvement, is

that RL agents now accomplish more CTs. Thus, the TCT of the RL agent is now 1,876.14 against the 1,465.62 obtained in the past experiment. The third aspect is to analyse the performance of the various agent roles for both algorithms. Here, in contrast to the previous results, the NL's AiC is significantly better than RL's AiC (H4 is rejected). But the price paid for this is that the NL agents are more likely to select the wrong CMs (i.e. the non-optimal ones in a specific situation). This is because NL agents could have accomplished as many CTs as RL agents if they selected other CMs. Regarding the other roles, the RL's AiCoop and AiS roles show that these agents choose the CM that enables them to accomplish more STs and to gain a good reward working in cooperation. Finally, as before, the more CTs that an agents accomplishes, the better its performance.

Hypothesis to evaluate	**p**	**Outcome**	**Winner**
H3: $\mathsf{AiS}_{\mathsf{RL}}$=$\mathsf{AiS}_{\mathsf{NL}}$	0.000	Rejected	RL
H4: $\mathsf{AiC}_{\mathsf{RL}}$=$\mathsf{AiC}_{\mathsf{NL}}$	0.000	Rejected	NL
H5: $\mathsf{AiCoop}_{\mathsf{RL}}$=$\mathsf{AiCoop}_{\mathsf{NL}}$	0.000	Rejected	RL

Table 4: AU per agent' role in scenario1 (acting phase): result of ANOVA

From the above, we have shown that NL agents perform well when they use a decision making process that precisely models the various actions that take place in the environment. However, they did not do well when compared with RL agents whose learning has converged. But, the next question is how likely it will be that the situation is sufficiently static for the agent's learning to converge and then exploit the converged policies?. From our perspective, this is very unlikely; our agents must be capable of adapting their decision making when and during *acting* and as a result of what is occurring in the environment. Therefore, we want RL agents to perform in a superior fashion when considering both the training and the acting phases. Thus, in order to address this problem, in what follows, the use of learning is further explored when one of previous assumptions is no longer guaranteed. In particular, we explore whether learning agents can still find and exploit optimal policies in situations in which one of the fundamental actions associated with cooperative activity is more challenging to predict (scenario2).

5.2 Learning to select a CM in a dynamic environment

In this set of experiments we tested the same set of hypotheses as in Section 5.1. Let us first consider the case where measurements are taken including the exploitation and exploration phase (namely, during the training phase). The initial results are summarised in Table 5 and Figure 6. First, we analyzed the hypotheses related with AU. Similarly to the results obtained in Table 1, we conclude that applying RL and NL produces distinctive results. But, conversely to Table 1, RL agents get significantly better results ($\mathsf{AU}_{\mathsf{RL}} = 73,690.06$) than NL ($\mathsf{AU}_{\mathsf{NL}} = 68,333.94$) (as can be seen in Figure 6).

Hypothesis to evaluate	**p**	**Outcome**	**Winner**
H1: $\mathsf{AU}_{\mathsf{RL}}$=$\mathsf{AU}_{\mathsf{NL}}$	0.000	Rejected	RL
H2: $\overline{\mathsf{TCT}}_{\mathsf{RL}}$=$\overline{\mathsf{TCT}}_{\mathsf{NL}}$	0.000	Rejected	NL

Table 5: Agent's AU and TCT in scenario2: result of ANOVA.

With reference to the TCT accomplished, the hypothesis of equal means of H2 is rejected. There is a significant impact on the TCT achieved when performing RL or NL (the results are 686.54 to RL and 889.46 to NL). The relevant aspect to discuss now, though, is that in contrast to the previous experiments, NL obtains a lower AU despite achieving more CTs. This corroborates the explanation about the importance of deciding when to pursue a CT with the correct selection of the CM and its repercussion on the agent's performance. In other words, the major decision an agent faces is to decide whether to attempt coordination or not. This is important because an agent might gain more long term reward accomplishing STs rather than investing time in

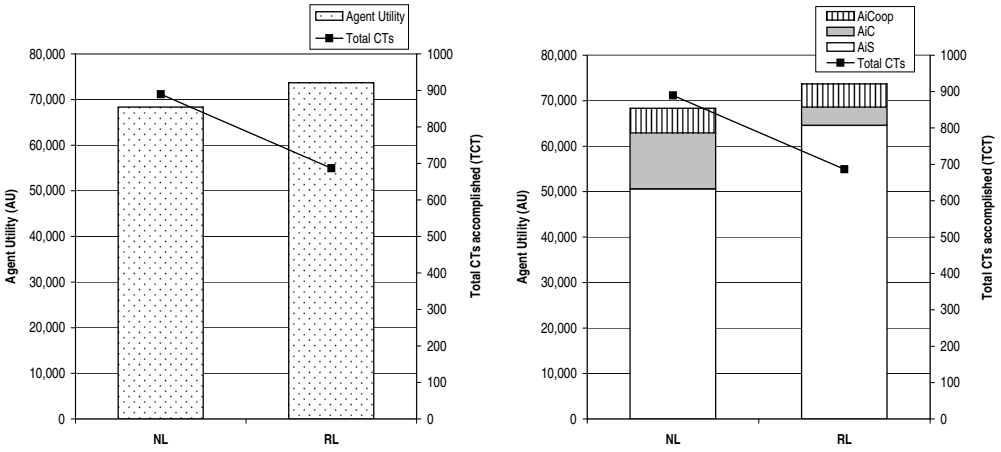

Figure 6: Contrasting agent's performance in scenario2.

unsuccessful CTs (which is particular frequent in scenario2). To this end, Figure 6 (right section) and Table 6 show that the reward gained by achieving CTs by the cooperative roles NL-AiCs and NL-AiCoop are higher than those gained by the corresponding RL ones. This is because, on the one hand, they achieve more CTs, and, on the other hand, although the AiCoop bids might be higher than in scenario1, when they are accepted and become agreements agents gain a good profit from them (H4 and H5 are rejected). However, despite the fact that this cooperative behaviour could be seen as a good decision to make, the time invested on these endeavours was not sufficient to recover the reward that was being gained by RL AiSs (RL AiSs obtained in total approximately 88% of the total reward by accomplishing STs and NL-AiSs achieved 74%). Thus, regarding the hypothesis that evaluates AiSs roles, H3 is rejected and RL agents are the winners. An additional justification of the results of Table 6 is that when agents decide to attempt coordination, they might invest a significant amount of time on the CM and, in the end, the AiCoops often request higher bids than those in scenario1 (meaning the AiCs' profit is reduced). Thus, RL agents perform better because they are more certain about when to invest time in a CT with the correct CM 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.

Hypothesis to evaluate	**p**	**Outcome**	**Winner**
H3: $\mathsf{AiS}_{\mathsf{RL}}$=$\mathsf{AiS}_{\mathsf{NL}}$	0.000	Rejected	RL
H4: $\mathsf{AiC}_{\mathsf{RL}}$=$\mathsf{AiC}_{\mathsf{NL}}$	0.000	Rejected	NL
H5: $\mathsf{AiCoop}_{\mathsf{RL}}$=$\mathsf{AiCoop}_{\mathsf{NL}}$	0.000	Rejected	NL

Table 6: AU per agent' role in scenario2: result of ANOVA

It is not difficult to see that while in scenario1 the NL agents could accurately predict the amount requested from others for engaging in a CT. However, this is not the case for the more unpredictable environment of scenario2. As a result, agents might not only select the wrong CM, but they may also attempt coordination when this was not the best thing to do. Therefore, the optimal policy varies from attempting coordination less frequently than in a static environment to not attempting coordination at all. This is supported by the fact that the TCT gained by both agent types in scenario2 is considerably lower (TCT has a mean of 788.00) than the amount accomplished in scenario1 (TCT mean of 1,635.76). Being more concrete, if the NL agents' predictions of bid are too low (being optimistic about the possible future cooperative agents), they will always initiate coordination even in situations where it is 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 that explore and exploit the CMs is the most reasonable

thing to do in dynamic environments because agents cannot be certain about the others' actions.

The final experiment analyses what happens for the various performance measurements after finding the optimal policies (as explored in Section 5.1 with the acting phase). In particular, we examine whether learning agents can still take advantage of knowing which CM to apply in this more demanding environment.

Hypothesis to evaluate	p	Outcome	Winner
H1: AU_{RL}=AU_{NL}	0.000	Rejected	RL
H2: TCT_{RL}=TCT_{NL}	0.000	Rejected	NL

Table 7: Agent's AU and TCT in scenario2 (acting phase): result of ANOVA

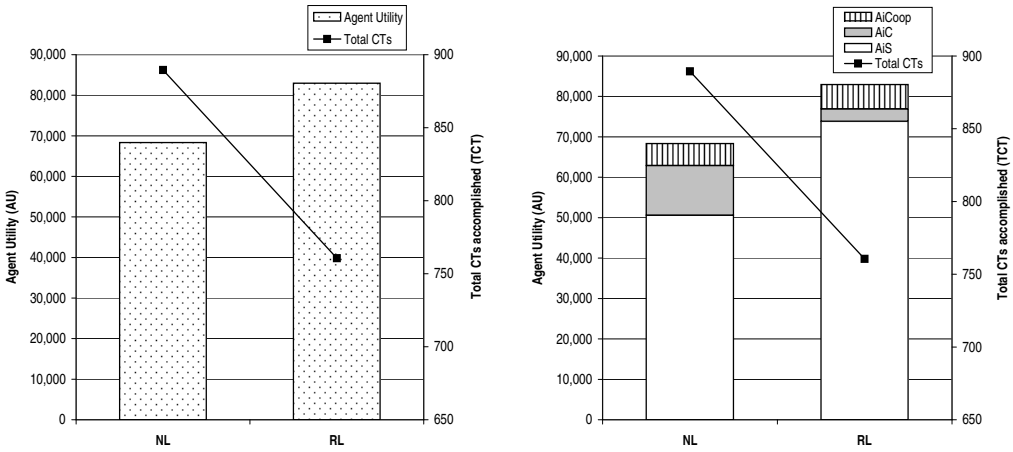

Figure 7: Contrasting agent's performance in scenario2 (acting phase).

As expected, and similarly to the results of scenario1, RL agents improve their performance by exploiting the policies they have learnt. The AU and TCT gained by RL agents in Figure 7 are higher than the corresponding AU and TCT of Figure 6 and, consequently, higher than the NL agents. Furthermore, given the more challenging task of engaging in coordination activities, here again the TCT accomplished by RL agents increased (though not considerably) by approximately 10%. This gives evidence to the claim that the optimal policy was not necessarily to attempt the CT every time it is found. To confirm this claim, Table 8 shows that the reward gained by RL AiSs increased in comparison with the AU gained by this role in the training phase. Moreover, this value is statistically higher than the ones gained by NLs (H3 is rejected). Regarding the cooperative roles (AiC and AiCoop), it is clear from Figure 7 (right side) that NL AiC is the winner (H4 is rejected) but once again this does not ensure a overall superiority because the other roles are better played by the RL agents (H5 is rejected).

Hypothesis to evaluate	p	Outcome	Winner
H3: AiS_{RL}=AiS_{NL}	0.000	Rejected	RL
H4: AiC_{RL}=AiC_{NL}	0.000	Rejected	NL
H5: $AiCoop_{RL}$=$AiCoop_{NL}$	0.000	Rejected	RL

Table 8: AU per agent's role in scenario2 (acting phase): result of ANOVA

Finally, to better understand where the differences occur when dealing with RL agents in scenario2, Table 9 shows the hypothesis evaluation for all the experimental variables. However, in order to distinguish between the various aspects of learning, RL1 stands for the ones that take into consideration the training phase whereas RL2 refers to the agents that perform solely in the acting

phase. Thus, the first thing to see is that RL2 has a better performance than RL1 in almost all the evaluations (H1, H2, H3 and H5 were rejected). The only hypothesis in which RL1 was superior is the one associated with the AiC role (which unfortunately is not the most profitable action to take in this scenario). The second aspect is that the optimal solution is *not to avoid engaging in cooperative action altogether, but rather not to spend much time on it.* Thus, RL agents learnt to decide to attempt coordination with CM_1 in most situations, which means that the agents minimise the amount of time they spend setting up the CM.

Hypothesis to evaluate	**p**	**Outcome**	**Winner**
H1: AU_{RL1}=AU_{RL2}	0.000	Rejected	RL2
H2: TCT_{RL1}=TCT_{NL2}	0.000	Rejected	RL2
H3: AiS_{RL1}=AiS_{RL2}	0.000	Rejected	RL2
H4: AiC_{RL1}=AiC_{RL2}	0.000	Rejected	RL1
H5: $AiCoop_{RL1}$=$AiCoop_{RL2}$	0.000	Rejected	RL2

Table 9: Contrasting RL agent's roles in scenario2 (training and acting phase): result of ANOVA

In summary, in dynamic and unpredictable environments RL agents perform better than NL agents 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). RL agents then use this time to take advantage of pursuing STs. Learning about the coordination decision helps RL agents to have a more precise model of what is occurring in the environment and consequently their decision making is improved. This, in turn, means the agents are more effective at maximising their profits. Unlike RL agents, the poor performance of NL agents is because they did not change their way of reasoning about their environment and they could not detect and adapt their decision making to the changes in the other agents' behaviours.

One aspect that needs further discussion is the apparently better performance of RL agents in the acting phase compared with that achieved in the training phase alone (in both scenario1 and scenario2). We believe that this observation needs to be taken with caution for a number of reasons. Firstly, we believe that it is very unlikely that agents will be able to use the same optimal policies if the environment is in a constant state of flux. In both experiments the environment did not actually change after the training phase and the specific policies exploited were consequently effective also in the acting phase. Thus, if the environment changes (i.e. the agents leave and enter the system or agents alter their behaviour), it cannot be guaranteed that the policies being exploited are still the optimal ones in the transformed environment. Secondly, despite the improvement shown by RL agents acting using the learnt optimal policies, it is important to recall that our objective is to design agents that can operate effectively in open and dynamic environments. Hence, the exploration and exploitation is fundamental to the convergence time of learning algorithms and needs to be associated with the changes that are occurring in the environment. To this end, we still believe that it is necessary investigation more fully about what to learn and when to stop learning given environmental changes. However, this is the subject of further work.

6 Related Work

There are two broad strands of work that are primarily related to our model and each will now be dealt with in turn.

- work on reasoning about coordination and
- work on multiagent learning.

In terms of coordination, most existing work assumes it is a design time problem (e.g., [26, 28, 7, 22]). Thus, comparatively little work addresses run-time reasoning about the selection of particular coordination protocols. Among those that do deal with this issue, a variety of research

positions have been investigated related to how flexibility can be introduced in different aspects and at different levels of coordination. However, from the perspective of this work, these can all be classified as introducing flexibility into particular cases of coordination mechanisms or in a somewhat restricted manner. In more detail, Durfee [6] has argued that agents need the flexibility to coordinate at different levels of abstraction, depending upon their particular needs at a given moment in time. To date, however, this work has focused on building such flexibility into the basic planning mechanisms of the individual agents. As yet, there are no mechanisms for explicitly reasoning about which level to coordinate at in a given situation. Such flexibility was also built into cooperative problem solving agents by Jennings [15]. Here, agents could choose to cooperate according to various conventions which dictated how they should behave in a particular team problem solving context. These conventions varied in terms of the time they took to establish and the communication overhead they imposed upon the agents. However, again, there was no reasoning mechanism for determining which convention was appropriate for a given situation. Barber et al. [1] present a software engineering framework that enables agents to vary their coordination mechanisms according to their prevailing circumstances. They also identify criteria for determining when particular mechanisms are appropriate. However, the decision procedures for actually trading-off these criteria are not well developed. Boutilier [3] presents a decision making framework, based on multi-agent Markov decision processes, that does reason about the state of a coordination mechanism. However, his work is concerned with optimal reasoning within the context of a given coordination mechanism, rather than actually reasoning about which mechanism to employ in a particular situation.

In terms of the work on learning, a vast literature has been produced in recent years concerning the use of learning techniques (particularly Q-learning) in multiagent systems [25, 29]. The focus has been mainly on two aspects. In the first one, an agent's goal is to learn about the other agents or their environment in order to predict their behaviour or to produce a model of them [20, 14, 4]. In the second case, Q-learning has been applied to learn how to coordinate or cooperate to achieve common goals by using specific strategies [32, 24]. 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. To date, however, there has been comparatively little work concerned with learning which CM to select in a given context.

The most relevant work to our own is the COLLAGE [21] and LODES [30] 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 does indeed improve the decision making when agents are uncertain about the other agents' actions. This improvement occurs because the agents learn to recognise 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 compete against agents that can make reasonably 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 the decision procedures are specific to our grid-world scenario, we believe that the general processes and structures we developed are suitable for reasoning about coordination mechanisms in more general domains (see [8] for several examples of how the scenario can be mapped into a variety of real world problems). 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 an important first step in that direction.

For the future, the aim is to extend the use of learning to cover other aspects of the agent's decision framework; such as, to learn the decision about how much to bid in a request for coordination (Section 3.2), when to become an AiCoop (Section 3.1) and which bids to accept (Section 3.3). Learning could be also introduced in areas other than learning to make decisions, for example agents could learn the constituent factors of the decisions themselves [8]. It is also intended to allow agents to construct models of one another and to have the ability to vary the details of this modelling according to the agent's coordination context. In particular, it is believed that in order to accomplish more effective learning objectives, agents should model the others as *1-level agents* (using the terminology of [33]) by explicitly representing knowledge about others. This is because most of the agent's decisions take into consideration predictions about the other agents and to refine these predictions an agent needs to represent in a more precise way the behaviour of the others in the scenario. In a broader context, a final aspect to discuss is that learning can also be employed to learn the meta-data parameters of the CM (i.e. the CM's parameters could be modified given the efficiency of its actual execution). However, this aspect was not addressed in this paper because it is directly related with the specifics of how the different CMs operate to achieve coordination.

8 Acknowledges

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

References

[1] K. S. Barber, D. C. Han, and T. H. Liu. Coordinating distributed decision making using reusable interaction specifications. In *Design and Applications of Intelligent Agents: Third Pacific Rim International Workshop on Multi-Agents (PRIMA 2000)*, pages 1–15, Melbourne, Australia, August 2000.

[2] R. A. Bourne, C. B. Excelente-Toledo, and N. R. Jennings. Run-time selection of coordination mechanisms in multi-agent systems. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000)*, pages 348–352, August 2000.

[3] C. Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 478–485, Stockholm, Sweden, July-August 1999.

[4] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 746–752, Madison, MI, July 26-30 1998.

[5] P. R. Cohen. *Empirical Methods for Artificial Intelligence*. The MIT Press: Cambridge, MA, 1995.

[6] E. H. Durfee. Practically coordinating. *AI Magazine*, 20(1):99–116, Spring 1999.

[7] 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, September 1991.

[8] C. B. Excelente Toledo. *The Dynamic Selection of Coordination Mechanisms*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, February 2003.

[9] C. B. Excelente-Toledo, R. A. Bourne, and N. R. Jennings. Reasoning about commitments and penalties for coordination between autonomous agents. In *Proceedings of the Fifth International Conference on Autonomous Agents (AGENTS'01)*, pages 131–138, Montreal, Quebec, Canada, May-June 2001.

[10] C. B. Excelente-Toledo and N. R. Jennings. Learning to select a coordination mechanism. In *Proceedings of the First International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS'02)*, pages 1106–1113, Bologna, Italy, July 15-19 2002.

[11] C. B. Excelente-Toledo and N. R. Jennings. The dynamic selection of coordination mechanisms. *Journal of Autonomous Agents and Multi-Agent Systems*, To appear. Kluwer Academic Publishers, 2003.

[12] M. S. Fox. An organizational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):70–80, January 1981.

[13] J. Galbraith. *Designing Complex Organizations*. Addison-Wesley Publishing Company, Inc. Reading, MA, 1973.

[14] J. Hu and M. P. Wellman. Online learning about other agents in a dynamic multiagent system. In *Proceedings of Second International Conference on Autonomous Agents (AGENTS'98)*, pages 239–246, Minneapolis, MN, May 9-13 1998.

[15] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.

[16] N. R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296, 2000.

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

[18] V. R. Lesser. Reflections of the nature of multi-agent coordination and its implications for an agent architecture. *Autonomous Agents and Multi-Agent Systems*, 1(1):89–111, July 1998.

[19] T. W. Malone. Modeling coordination in organizations and markets. *Management Science*, 33(10):1317–1332, October 1987.

[20] Y. Nagayuki, S. Ishii, and K. Doya. Multi-agent reinforcement learning: An approach based on the other agent's internal model. In *Proceedings on the Fourth International Conference on Multi-Agent Systems (ICMAS-00)*, pages 215–221, Boston, MA, July 2000.

[21] 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.

[22] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. The MIT Press: Cambridge, MA, 1994.

[23] S. J. Russell and P. Norvig. Reinforcement learning. In *Artificial Intelligence: A Modern Approach*, volume Learning, chapter 20, pages 598–624. Prentice Hall: Upple Saddle River, NJ, 1995.

[24] S. Sen, M. Sekaran, and J. Hale. Learning to cooperate without sharing information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 426–431, Amherst, MA, July 1994.

[25] S. Sen and G. Weiss. Learning in multiagent systems. In G. Weiss, editor, *Multiagent Systems: A Modern Approach To Distributed Artificial Intelligence*, chapter 6, pages 259–298. The MIT Press: Cambridge, MA, 1999.

[26] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 276–281, San Jose, California, July 1992.

[27] S. P. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvari. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, March 2000.

[28] R. G. Smith and R. Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):61–70, January 1981.

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

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

[31] R. S. Sutton and G. A. Barto. *Reinforcement Learning: an introduction*. The MIT Press: Cambridge, MA, 1998.

[32] M. Tan. Multi-agent reinforcement learning: Independent vs cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, Amherst, MA, 1993.

[33] J. M. Vidal and E. H. Durfee. Agents learning about agents: A framework and analysis. In *Collected papers from the AAAI-97 workshop on Multiagent Learning*, Providence, Rhode Island, 1997.

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