Using Reinforcement Learning to Coordinate Better

Cora B. Excelente-Toledo [*]
National Laboratory of Advanced Computer Science, LANIA
Rébsamen No. 80, Col. Isleta, C.P. 91090
Xalapa, Veracruz, MEXICO
Tel. +52 228 8416100 ext. 5002
Fax. +52 228 8416101
`cora@lania.mx`

Nicholas R. Jennings
School of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK.
Tel. +44 23 8059 7681
Fax. +44 23 8059 3313
`nrj@ecs.soton.ac.uk`

April 20, 2005

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, our motivating hypothesis is that to deal with dynamic and unpredictable environments it is important to have agents that learn *the right situations in which to attempt coordination* and *the right coordination method to use in those situations.* In particular, the efficacy of learning is evaluated when agents have varying types and amounts of information when those coordinating decisions are taken. This hypothesis is evaluated empirically, 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 cannot 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.

Keywords: Coordination, agent interaction, collaborative agents, reinforcement learning.

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 they share resources. To this end, a variety of protocols and structures have been

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

developed to address the coordination problem. These range from long-term social laws (Shoham & Tennenholtz, 1992), through medium-term mechanisms such as Partial Global Planning (Durfee & Lesser, 1991), organizational structuring (Fox, 1981) and market protocols (Malone, 1987), to one-shot (short-term) mechanisms like the Contract-Net Protocol (Smith & Davis, 1981).

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 highly likely to succeed typically have high set up and maintenance costs, whereas mechanisms that have lower set up costs are also 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 (Galbraith, 1973).

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 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. This is especially the case in open and dynamic environments in which agent-based solutions are often deployed (Jennings, 2000).

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 (Bourne, Excelente-Toledo, & Jennings, 2000; Excelente-Toledo, 2003; Excelente-Toledo & Jennings, 2004). 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, we have previously developed, evaluated and shown the effectiveness of an agent reasoning framework to achieve this (Bourne *et al.*, 2000; Excelente-Toledo, 2003; Excelente-Toledo & Jennings, 2004). 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 an appropriate coordination mechanism (that has previously been effective in similar circumstances). In particular, we explore the use of a number of Q-learning algorithms in which the amount of information represented in the state varies. We show how this representation impacts upon the process of making run-time choices about the selection of coordination mechanisms in a number of different scenarios. The reason for this changing state representation is because it models the key determining factors of the agent's reasoning framework (i.e. the environmental factors and the responses of the other agents in the group).

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 (*agents learn the right situations in which to attempt coordination and the right coordination method to use in those situations*). Secondly, it explores different state representations in Q-learning implementation and, more importantly, it analyses the efficiency of each in different scenarios regarding coordination decisions. Finally, 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 introduces the decision procedures that enable autonomous agents to dynamically select coordination mechanisms. Section 4 explains how learning is applied to these decision making procedures. Section 5 introduces the experimental methodology used to perform a systematic empirical evaluation of the main hypotheses of this paper. Section 6 reports on the experimental work to evaluate the effect of introducing learning extensions into the agent's decision making and examines their impact. Section 7 deals with related work and, finally, section 8 concludes and presents the areas of further work.

2 The Coordination Testbed

The testbed domain is described in more detail in (Bourne *et al.*, 2000; Excelente-Toledo, 2003) (this includes a detailed justification for the choice of this scenario and the various design decisions within it). Here we just recount the basics that are necessary to understand the subsequent experiments. Our testbed consists of 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 or continue working on its ST. In this context, each agent has a predefined range of CMs at its disposal. Each CM is parameterised by two types of meta-data (see (Excelente-Toledo, 2003) for a mapping of several coordination mechanisms into this framework): 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

[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.

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 in its best interests, 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. If however, AiC initiates coordination but there is no AiCoops, then, we say that the AiC failed while attempting coordination. 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 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. The CT is achieved and the rewards are paid to AiCoops. Note that the reward is only given with probability p, the factor associated with the CM used to coordinate over the CT.

[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. If AiC finds a new CT, it ignores it.

[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. 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 (Smith & Davis, 1981) 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 can be relaxed (see (Excelente-Toledo, Bourne, & Jennings, 2001) for an example dealing with the dropping of contracts in order to better exploit new coordination opportunities). To model dynamism, unpredictability and open features in this grid world, the elements in the environment change their values at execution time. Relevant examples include the changing of the tasks' rewards (both for STs and CTs), the frequency with which tasks appear and disappear in the grid, the 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 make decisions based on factors that cannot be a priori predicted.

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 (Bourne *et al.*, 2000; Excelente-Toledo, 2003; Excelente-Toledo & Jennings, 2004). 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. 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 to 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. [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, $\text{CM}_j = (t_j, p_j)$, 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 ($\mathsf{ave\_dist}$) away of any randomly situated agent from the CT square as well as the likely average deviation ($\mathsf{ave\_dev}$) such agents would have to make to get there.

[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.

Based on these figures, the agent can assess the average surplus reward from coordinating over the CT at (x,y) using $\mathrm{CM}_j = (t_j, p_j)$. 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 S reward units per time-step, the average is calculated as $r = \frac{S}{\mathsf{ave\_dist}}$. The cost of CM_j is then 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. When AiC does not have any knowledge of the average reward of all the other agents in the environment, it uses its own (r) average reward as an approximation. [3]

$$\mathsf{ave\_bid}_j(x,y) = m \times \frac{r \times \mathsf{ave\_dev}(x,y)}{p_j} \tag{1}$$

Third, the AiC estimates the expected surplus ($\mathsf{ave\_payoff}$) of CM_j from adopting the CT by taking into account the probability of success of the task:

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

Using these estimates, the AiC can evaluate the expected surplus reward of adopting CM_j:

$$\mathsf{ave\_surplus}_j(x,y) = \mathsf{ave\_payoff}_j(x,y) - \mathsf{cost}_j(x,y) + \mathsf{ave\_bid}_j(x,y) \tag{2}$$

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).

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

Figure 2: Example of a coordination world grid.

To exemplify this decision procedure, consider the simple scenario of Figure 2 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 occupies a $[5 \times 5]$ grid and 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

[3] In order to estimate (1) it is assumed that the m is determined in advance or is part of the agent's knowledge. However, this assumption may not always be valid for cases in which the number of cooperative agents depends on the particulars of the coordination's objective. In such cases, the agents will need to predict this number based on previous experiences or some how estimate this information (e.g., the straightforward solution is that agents maintain an average of the number of helpers each time they accomplish coordination; more complex solutions might involve building a model for each agent each time there is an interaction).

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 in deviating to 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 2. 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:

$$\mathsf{bid}_{ij} = \frac{r_i \times deviation_i}{p_j} \tag{3}$$

[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 (1).

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 each subset 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:

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

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 $\mathsf{surplus}$ given by:

$$\mathsf{surplus}_j = p_j \times R - \mathsf{cost\_bid}_b - r \times (t_j + maxT_b) \tag{4}$$

Now, it may be the case that no bids are received that 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.

In this paper, the main focus is on giving 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 this section and, in particular, the decision procedure outlined in section 3.1 (equation (2)), allows agents to make decisions about when and which CM to select in order to achieve a CT. Thus, this is the major one with respect to reasoning about coordination mechanisms (Bourne *et al.*, 2000; Excelente-Toledo, 2003; Excelente-Toledo & Jennings, 2004) and it is, therefore, the one we concentrate on in terms of evaluating the role of learning as described in the next Section. [5]

4 The Role of Learning

Our investigation focuses on the use of reinforcement learning (RL) (Kaelbling, Littman, & Moore, 1996; Sutton & Barto, 1998) in coordination. 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 (Watkins & Dayan, 1992) 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 a 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 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:

$$\mathsf{Q}(\mathsf{s}, \mathsf{a}) \leftarrow \mathsf{Q}(\mathsf{s}, \mathsf{a}) + \alpha[\mathsf{r} + \gamma \times \max_{a'} \mathsf{Q}(\mathsf{s}', \mathsf{a}') - \mathsf{Q}(\mathsf{s}, \mathsf{a})] \quad (5)$$

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

[5] There are 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 (3)) or which bids to accept (equation (4)). However, these options are out of the scope of this paper.

about a positive reward, and an action that has not yet been explored and that consequently has an unknown reward ("exploitation versus exploration") (Sutton & Barto, 1998). In this work, we use a ϵ-greedy function that selects the action with the highest $\mathsf{Q(s,a)}$ value once all the actions have been explored a pre-determined number of times. In particular, we use $f(\mathsf{Q(s,a)}, n)$ (Russell & Norvig, 1995) which determines how greed (preference for high values of $\mathsf{Q(s,a)}$) is traded off against curiosity (preference for low values of n, namely, actions that have not been tried before). Formally speaking, the exploration function equates to:

$$f(\mathsf{Q(s,a)}, n) = \begin{cases} R^{+} & \text{if } n < N_e \\ \mathsf{Q(s,a)} & \text{otherwise} \end{cases} \tag{6}$$

where n is the number of times $\mathsf{Q(s,a)}$ 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 with 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 $\mathsf{Q(s,a)}$ value is visited $visits(s,a)$: $\alpha = \frac{1}{1+visits(s,a)}$; the $\mathsf{Q(s,a)}$ values are initialised with 0. And equation (6) is used as the exploration function. [6]

Thus, the main 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 surplus (equation (4)) is positive given the set of bids (equation (3)) it received.

This means then that when dealing with RL agents, 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.

Now, for the purposes of this analysis, NL and RL agents experience one of the the following outcomes: i) a successful coordination (i.e. the AiC selects a CM, finds AiCoops to coordinate with and the CT is successfully achieved using the CM selected); ii) the AiC initiates coordination with a selected CM but there are not enough successful bids to make agreements (this means that attempting coordination with the CM was a failure); and iii) the AiC does not select any of the CMs at its disposal (i.e. the null CM is selected and the agent continues with its individual problem solving activity).

On the basis of previous description, the learning agent's main objective is then to select the most suitable CM given its prevailing circumstances. To achieve this general goal, we explore the use of two Q-learning algorithms: RL1 and RL2. The main difference between them is in the way

[6]It is well known that the convergence time is determined by the exploitation and exploration function, the size of the look up table and the learning rate (Singh, Jaakkola, Littman, & Szepesvari, 2000). Here, 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 all Q-learning implementations.

they model the state representation at the moment when the decision about which CM to select is taken. In particular, our interest is in evaluating whether agents can improve their performance by explicitly representing one of the key components of the decision making of the CM, namely the ave_bid (equation (1)). [7] Thus, RL2 agents employ a Q-learning algorithm that does estimate and use ave_bid when selecting the CM, whereas RL1 does not model this information in its state representation.

Thus, in what follows, we first describe the general RL algorithm and then we detail the differences between RL1 and RL2. Finally, we discuss the algorithm followed by the NL agents.

RL: When a learning agent finds the CT, it performs the following basic cycle:

[1] It is in a particular state (represented by s)

[2] It makes a decision about the next action (a) to execute using the exploitation and exploration function of equation (6).

[3] The agent executes the selected action (a) which will be one of accomplishing a CT (if a CM is selected and successful), failing on a CT (if a CM is selected and unsuccessful), or selecting no CM (if the null CM is selected) and reaches a new state s'.

[4] It obtains a reinforcement reward as a result of the execution of action a. In particular, the reinforcement varies with the following outcomes:

- The CT is accomplished using the CM selected. In this case, a positive reinforcement is obtained that is based on the reward gained by achieving the CT after the payment to the AiCoops and the time invested in pursuing the task.
- The CT failed given the CM chosen. This situation occurs because no one replies to the request for coordination or because the reward requested to participate in the cooperative action by the AiCoops is too high for the AiC to accept it (i.e. the surplus (equation (4)) is negative). In either case, a negative reinforcement is calculated based on the average reward (r) lost in the time invested in the CM (t).
- The null CM is explored or exploited. Here, the reinforcement corresponds to the reward (CT reward) the agent might have obtained by investing an average time in a CT (modelled by average distance ave_dist).

[5] It updates the $Q(s, a)$ value (equation (5)).

[6] It goes to [1].

Turning now to the differences between the learning agents and taking previous descriptions of RL we have two variations: RL1 and RL2 agents. Each of them will be dealt in turn.

- **RL1**: An agent learns to select a CM by exactly following the RL algorithm and, in particular, it uses its role and position in the grid when the CT is found as the representation of s.
- **RL2**. An agent learns to select a CM using ave_bid. An agent of this type follows RL but has a modified state representation s and a different means of updating to s' in step [4]. Specifically,

[7]Recall that ave_bid represents the prediction of other agents' bids with which agents attempt coordination.

s is modelled with the agent's role, its position in the grid and the expected ave_bid. [8] The initial estimation of ave_bid in step [1] corresponds to:

$$\mathsf{ave\_bid}_j(x, y) = m \times r \times \mathsf{ave\_dev}(x, y)$$

Note that the only difference between this formulation and equation (1), is that p_j (the probability of success of a given CM) is unknown at this stage.

When RL2 agent reaches a new state s′, the state is updated with the ave_bid using equation (1) where p_j refers to the probability of success of the CM chosen.

NL: A non-learning agent makes decisions entirely based on the decision making procedures detailed in Section 3 and follows the protocol specified in Figure 1. Being precise, when an agent finds a CT, it calculates the expected average surplus (equation (2)) of each CM at its disposal. It then simply chooses the one with the best ave_surplus. For the next stages of the protocol, it uses equations (4) to decide which bids to accept and (3) to become AiCoop. Note that the NL agent uses equation (1) to calculate ave_bid in order to evaluate ave_surplus for each of the 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 (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 one in which predictions are inherently less accurate.

Bringing this all together, 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 calculate the ave_bid.

NL agents select a particular CM using equation (2).

5 Evaluation Methodology

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

[8]To simplify the state representation, ave_bid is in fact associated with a range of values. Given the values of the simulation variables (see Section 5.1), the ranges of the ave_bid are the following: $1 <$ave_bid< 1.84, $1.84 \geq$ave_bid< 2.64 and ave_bid≥ 2.64.

($p < 0.05$) or might be accepted in the contrary case (Cohen, 1995). 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.

ANOVA explains the relationships between groups by analysing all possible interactions among them. However, though it provides an answer to the hypothetical questions by indicating if the mean of the groups are equal or not, it does not indicate the relations between them (for example, which mean of which groups are the highest). Thus, in most cases, it is necessary to go a step further (post-analysis) to determine where the exact differences among the means occur between groups. This procedure consists of running a post-test to explore the data collected on a case by case basis (this is termed pairwise analysis) because it tests the difference between each pair of means. [9] This pairwise analysis is particularly important in those cases where more than two procedures are being tested (or one procedure is tested in more than two scenarios). In concrete terms, the post-test makes a comparison between the data collected and builds groups (as many as necessary) that have statistically homogeneous values. Each group is generated with an associated value (the p value) that indicates the degree of confidence from which each group was built.

For the purposes of the experimental evaluation presented in Section 6, all the hypotheses are evaluated with ANOVA and only the cases that involve more than two experimental variables are subject to a post evaluation. When we test two variables and ANOVA rejects the equality of means, the data collected is used to indicate the relationship between the two variables. We are also interested in determining which of the groups obtains the highest mean (from now on referred to as the *winner group*) because it represents the agents with the best performance.

5.1 Experimental and simulation variables

The following simulation variables were fixed for all learning experiments: size of the grid ($[10\times10]$), duration (500,000 time units) [10], 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 ($CM_1 = (1, 0.6)$, $CM_2 = (15, 0.7)$, $CM_3 = (30, 0.8)$, $CM_4 = (45, 0.9)$ and $CM_5 = (60, 1.0)$) [11].

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.

[9] Several statistical tests exist to perform this analysis. The one used here is called Tukey's honestly significant difference (HSD). This was chosen because it lies in the middle of the spectrum of alternatives; between LSD (which stands for *least significant differences*) and Scheffé tests which are the extreme in the conservative methods (Cohen, 1995; Lane, 2001).

[10] 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.

[11] These CMs were selected because previous results have indicated that these are the main ones that are selected by the agents in this setting (Bourne *et al.*, 2000; Excelente-Toledo & Jennings, 2002; Excelente-Toledo, 2003).

5.2 Evaluating hypotheses

To accept our main hypothesis, 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 AU obtained by performing a reinforcement based algorithm (RL) is the same as that obtained by agents which use the NL algorithm.

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

H3: the AU 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).

6 Experimental Evaluation

The experimental evaluation undertaken in this section follows the methodology described previously and is organised in the following way. Firstly, the four hypotheses of Section 5.2 are tested in a static environment (Section 6.1) and secondly the same evaluation procedure is followed but in a dynamic environment (Section 6.2).

Table 1: Agent's AU and TCT in scenario1.

Hypothesis to evaluate	p	Outcome	Winner
H1: $AU_{RL1}=AU_{RL2}=AU_{NL}$	0.000	Rejected	NL, RL2$^{(\cdot)}$
H2: $TCT_{RL1}=TCT_{RL2}=TCT_{NL}$	0.000	Rejected	RL2$^{(\cdot\cdot)}$
H3: $AU_{RL1}=AU_{RL2}$	0.000	Rejected	RL2
H4: $TCT_{RL1}=TCT_{RL2}$	0.000	Rejected	RL2

$^{(\cdot)}$ see Table 2 and $^{(\cdot\cdot)}$ Table 3 for details.

6.1 Selecting CMs in static environments (scenario1)

To start with, Table 1 and Figure 3 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. In order to understand the relationship between the algorithms a post-analysis of H1 is conducted (Table 2). The conclusion is that the performance of NL and RL2 is better by a statistically significant amount ($AU_{NL} = 88,018.64$, $AU_{RL2} = 87,570.24$) than RL1 ($AU_{RL1} = 81,064.28$). Furthermore, comparing the performance of RL1 and RL2 in H3 (Rejected), it is concluded that the different mechanisms used for learning do effect the AU obtained by agents. Figure 3 graphically shows that the reward gained by RL2 agents is better than that for RL1 agents. Moreover, it can also be concluded that an agent which learns to select the CM (RL2) performs the same as one which does not learn at all (NL).

H2 and H4 evaluate the effectiveness of achieving CTs. Both hypotheses are rejected which means that the total of CTs achieved does depend on the algorithm executed. In this case, the

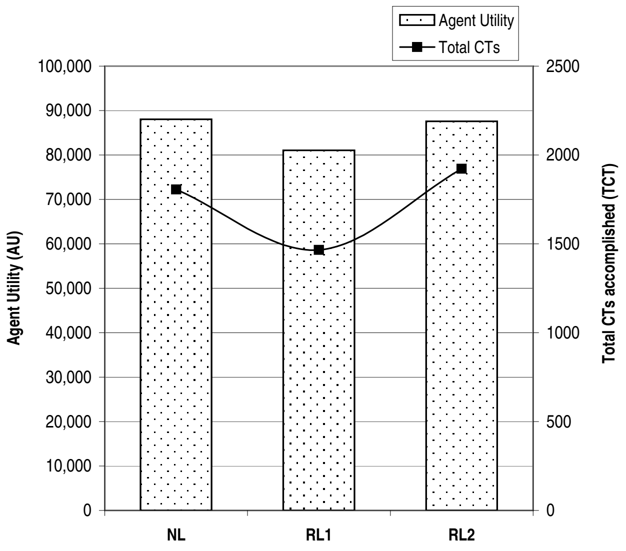

Figure 3: Contrasting agent's abilities in scenario1.

more CTs that are accomplished, the better the AU that is obtained. Here NL and RL2 agents accomplish more CTs than RL1 and, consequently, they gain more reward (see axis Y of Figure 3 and the result of the post-analysis regarding TCT in Table 3). However, this is an important result to analyse in detail, because a high number of CTs achieved, does not necessarily mean that an agent performs better. This argument can be seen by observing Tables 2 and 3 (Figure 3 graphically shows the same information). Here, despite the fact that RL2 agents achieved statistically the highest number of TCT (three groups were formed in the post-analysis), their AU gained is not higher than that obtained by NL agents (RL2 and NL agents belong to the same group in Table 2). Actually, the results of applying ANOVA to the TCT achieved and AU gained by NL and RL2 corroborate this explanation and are shown in Table 4. Specifically, H5 shows that the total reward obtained by RL2 is the same as that obtained by agents using the NL algorithm (H5 is accepted, there is no statistically significant effect on the AU), whereas H6 demonstrates that the total number of CTs achieved by NL agents is identical to the TCTs accomplished by RL2 agents (H6 is rejected, the TCT obtained by each algorithm is significantly different).

From the results obtained in the previous experiment it can be seen that in this scenario, the agent's optimal behaviour is achieved by firstly taking the correct decision about when to attempt coordination. And, secondly, by selecting the CM whose time to set up is balanced by the amount of reward obtained. Moreover, RL2 and NL are the ones that make decisions which maximise their AU. Thus, to better understand the differences or similarities between RL2 and NL, Table 5 tests ANOVA by agent role with the following hypotheses:

H7: the AU obtained by Agents-in-ST role which perform a reinforcement based algorithm (RL2) is the same as that obtained by AiS agents which use the NL algorithm.

H8: the total reward obtained by RL2 agents in the Agent-in-Charge role is the same as that obtained by NL AiC agents.

H9: the AU obtained by RL2 Agents-in-Cooperation role (AiCoop) is identical to the total reward obtained by NL agents in the AiCoop role.

Table 2: H1 in scenario1: post-analysis.

Agent	**AU**	
	1	2
RL1	81,064.28	
RL2		87,570.24
NL		88,018.64
p	1.000	0.791

Table 3: H2 in scenario1: post-analysis.

Agent	**TCT**		
	1	2	3
RL1	1,465.62		
NL		1,805.90	
RL2			1,922.20
p	1.000	1.000	1.000

Table 4: Contrasting RL2 and NL agents in scenario1.

Hypothesis to evaluate	**p**	**Outcome**	**Winner**
H5: $\mathsf{AU}_{\mathsf{RL2}}=\mathsf{AU}_{\mathsf{NL}}$	0.594	Accepted	none
H6: $\mathsf{TCT}_{\mathsf{RL2}}=\mathsf{TCT}_{\mathsf{NL}}$	0.000	Rejected	RL2

Figure 4 illustrates these results. They indicates that while RL2 AiC is significantly better than the corresponding NL agent role (H8 rejected), NL AiS is better than RL2 AiS (H7 rejected). H9 is accepted which means that the two agents accomplish similar rewards in their AiCoop role. As can be seen, both agent types achieve (statistically speaking) the same AU (recall that H5 was accepted) but from different sources. In this experiment, the best AiC (as a result of accomplishing more CTs) does not mean more reward is obtained. This is because high AU might have originated from fewer and more profitable CTs (in the case of NL) or a high number of less rewarding tasks (in the case of RL2). In terms of selection of the CMs, this means that NL selected those with higher probability of success and higher time to set up (less time to achieve CTs), while RL2 agents choose those CMs with less time to set up but less probability of success.

Table 5: Contrasting RL2 and NL agent's role AU in scenario1.

Hypothesis to evaluate	**p**	**Outcome**	**Winner**
H7: $\mathsf{AiS}_{\mathsf{RL2}}=\mathsf{AiS}_{\mathsf{NL}}$	0.009	Rejected	NL
H8: $\mathsf{AiC}_{\mathsf{RL2}}=\mathsf{AiC}_{\mathsf{NL}}$	0.000	Rejected	RL2
H9: $\mathsf{AiCoop}_{\mathsf{RL2}}=\mathsf{AiCoop}_{\mathsf{NL}}$	0.546	Accepted	none

Whereas the previous discussion only compared RL2 and NL agents, in what follows, we undertake a similar analysis but considering all three algorithms. To this end, Table 6 and Figure 4 show the result of the data collected and Tables 7, 8 and 9 present the post analysis performed to H10, H11 and H12 respectively.

H10: the AU obtained by Agents-in-ST role which perform a reinforcement based algorithm (either RL1 or RL2) is the same as that obtained by AiS agents which use the NL algorithm.

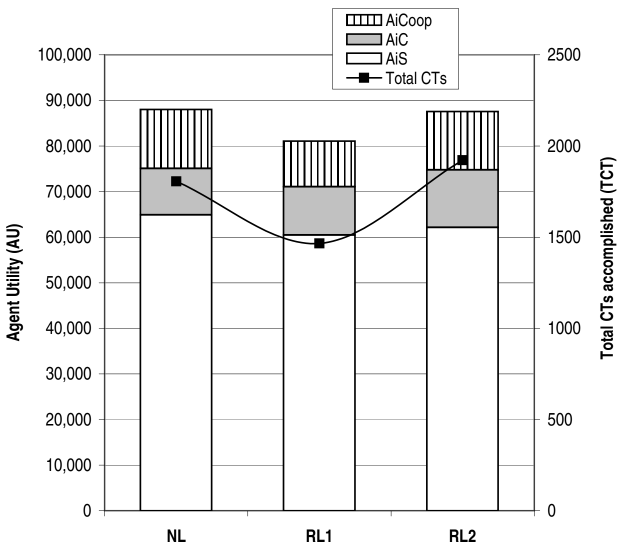

Figure 4: Contrasting agent's roles abilities in scenario1.

H11: the total reward obtained by RL1 and RL2 agents in AiC role is the same as that obtained by NL AiC agents.

H12: the AU obtained by RLs Agents-in-Cooperation role (AiCoop) is identical to the total reward obtained by NL agents in the AiCoop role.

Table 6: Agent's role AU in scenario1.

Hypothesis to evaluate	**p**	**Outcome**	**Winner**
H10: $AiS_{RL1}=AiS_{RL2}=AiS_{NL}$	0.000	Rejected	NL$^{(\cdot)}$
H11: $AiC_{RL1}=AiC_{RL2}=AiC_{NL}$	0.000	Rejected	RL2$^{(\cdot\cdot)}$
H12: $AiCoop_{RL1}=AiCoop_{RL2}=AiCoop_{NL}$	0.000	Rejected	NL, RL2$^{(\cdots)}$

$^{(\cdot)}$ see Table 7, $^{(\cdot\cdot)}$ Table 8 and $^{(\cdots)}$ Table 9 for details.

As expected, the first thing to notice is that the results are consistent with those described in Table 5. This is because NL and RL2 agents have superior performance in all three roles when compared against RL1 (H10, H11 and H12 are rejected). This means, the performance of RL1 agents do not have a significant effect on the experimental variables. NL still performs better than the others in terms of taking advantage of pursuing STs (i.e. by not attempting a cooperative activity (H10 rejected)). Turning to the analysis of the cooperative behaviour, here the results of Table 5 show that RL2 is the agent which performs the best as an AiC (H11 is rejected) and (RL2 and NL) obtain the same as an AiCoop (H12 is rejected as both agents have statistically speaking the same results). Generally speaking, it can be seen that RL2 and NL better balance their decision of when to go for the CT with the CM which allows them to gain more reward.

When considering the two learning based algorithms, it is clear that those agents that take into account the ave_bid of other agents (RL2) perform better than those that do not (RL1). Thus H3 and H4 are rejected in Table 1 and RL2 is the winner in both cases. Moreover, the information provided by the policies learnt by RL2 is more informative. For example, both agents learn that the optimal action when agents are in the corner of the grid (position $[1, 10]$) is to select CM_1.

However, with RL2, agents additionally learn that this is the action to execute if and only if the ave_bid is higher than 2.64. Generally speaking, the optimal policy with RL2 might vary for each of the different values of the ave_bid, whereas with RL1 this information cannot be extracted from the policies because it is not explicitly represented.

Table 7: H10 in scenario1: post-analysis.
Role to analyse: AiS

Agent	**AU**	
	1	2
RL1	60,475.28	
RL2	62,136.72	
NL		64,899.00
p	0.124	1.000

Table 8: H11 in scenario1: post-analysis.
Role to analyse: AiC

Agent	**AU**	
	1	2
NL	10,192.50	
RL1	10,615.21	
RL2		12,641.40
p	0.346	1.000

Table 9: H12 in scenario1: post-analysis.
Role to analyse: AiCoop

Agent	**AU**	
	1	2
RL1	9,973.79	
RL2		12,792.12
NL		12,927.13
p	1.000	0.746

Contrary to our intuitions, neither of the learning agents perform better than NL. One reason for this conclusion is that an NL agent performs well because it uses a decision making process that models reasonably well the various actions that take place in a static environment. In other words, NL agents use equation (1) to predict the possible bids the other agents will make to participate in a cooperative task. And, as it turns out, this estimate is not far from the real bid they eventually make. The second reason is that the results for the learning based algorithms were obtained considering a fixed period that included the exploitation and exploration phase, not once the learning agents had converged to the optimal policy (equation (6)). 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. Actually, it turns out to be the case (we have verified that when agents are initiated with the policies they have learnt from the past experiment, they perform significantly better than the corresponding NL agents (these issues have been thoroughly explored in (Excelente-Toledo & Jennings, 2003))). Speaking more generally, NL agents require knowing a priori what the regularities of the domain will be, whereas the learning agents can perform successfully as long as regularity exists. However, despite this improvement, we consider it very unlikely that agents will be able to take advantage of the knowledge learnt when the environment is in a constant state of flux. In such cases, it is not valid to assume that agents will continue behaving in similar ways or that the interaction will take place with the same agents they encountered before. In such scenarios, it will be more difficult to find NL agents that can accurately predict the other's behaviour and it is very unlikely that agents can successfully use the same decision making framework. Thus, for these reasons, we believe our agents should be capable of adapting their decision making as a result of what is occurring in the environment, i.e. during acting in more demanding scenarios. Therefore, our objective is to design agents that inhabit open and dynamic environments and we require the learning agents to show a superior performance when considering both the exploring and the exploiting phases. Thus, in what follows, the use of learning is further explored in situations in which one of the fundamental actions associated with cooperative activity is more challenging to predict (scenario2).

6.2 Selecting CMs in dynamic environments (scenario2)

Turning now to the more dynamic environment of scenario2. We tested the same basic set of hypotheses (H1, H2, H3 and H4) and the results are summarised in Table 10 and Figure 5. First, we analyzed the hypotheses related with AU. Similar to the results obtained in Table 1, we conclude that applying RLs and NL produces distinctive results. But, conversely to Table 1, RL2 agents get significantly better results ($\mathsf{AU}_{\mathsf{RL2}} = 75,952.66$) than RL1 agents ($\mathsf{AU}_{\mathsf{RL1}} = 73,690.06$) and RL1 agents get a significantly higher AU than NL agents ($\mathsf{AU}_{\mathsf{NL}} = 68,333.94$). These results are clear in Figure 5 and the relationship between the groups is shown in Table 11. What is more, regarding the differences in the agents' performances shown by the learning algorithms (H3), the results are once again that the AU of RL2 agents are significantly higher than those accomplished by RL1 agents.

Table 10: Agent's AU and TCT in scenario2.

Hypothesis to evaluate	p	Outcome	Winner
H1: $\mathsf{AU}_{\mathsf{RL1}}=\mathsf{AU}_{\mathsf{RL2}}=\mathsf{AU}_{\mathsf{NL}}$	0.000	Rejected	RL2$^{(\cdot)}$
H2: $\mathsf{TCT}_{\mathsf{RL1}}=\mathsf{TCT}_{\mathsf{RL2}}=\mathsf{TCT}_{\mathsf{NL}}$	0.000	Rejected	NL$^{(\cdot\cdot)}$
H3: $\mathsf{AU}_{\mathsf{RL1}}=\mathsf{AU}_{\mathsf{RL2}}$	0.000	Rejected	RL2
H4: $\mathsf{TCT}_{\mathsf{RL1}}=\mathsf{TCT}_{\mathsf{RL2}}$	0.200	Accepted	none

$^{(\cdot)}$ see Table 11 and $^{(\cdot\cdot)}$ Table 12 for details.

With reference to the TCT accomplished, the hypotheses of equal means of H2 and H4 are rejected. Figure 5 shows on its Y axis the total CTs accomplished by agent type. As can be seen, there is a significant impact on the TCT achieved when performing RL or NL; the results obtained by RLs are in the range of 30% less than those obtained by NL (Table 12 illustrates that after the post-analysis, RL agents accomplish (statistically speaking) the same number of CTs). The relevant aspect to discuss now, though, is that in contrast to previous experiments, NL obtains a lower AU despite achieving more CTs. This corroborates our previous explanation about the correct selection of the CM and its repercussions for the agent's performance. To this end, Table 13 and Figure 6 show the results of testing ANOVA for the next hypotheses, as we did in scenario1:

H10: the total reward obtained by Agents-in-ST role which perform a RL algorithm (either RL1 or RL2) is the same as that obtained by AiS agents which use the NL algorithm.

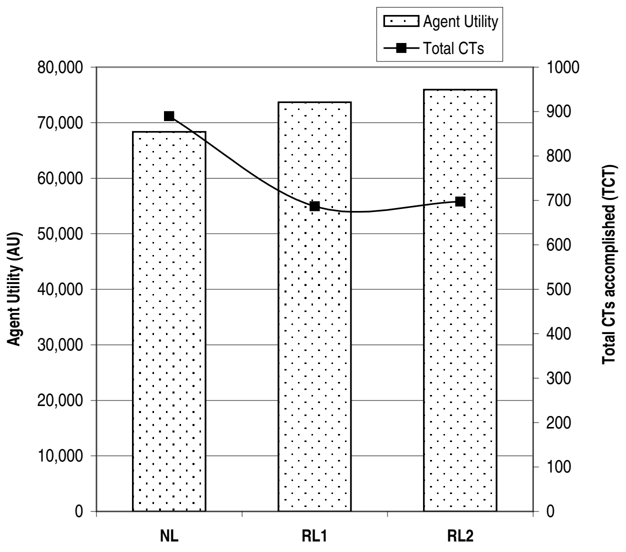

Figure 5: Contrasting agent's abilities in scenario2.

H11: the AU obtained by RL1 and RL2 agents in AiC role is the same as that obtained by NL AiC agents.

H12: the AU obtained by RLs Agents-in-Cooperation role (AiCoop) is similat to the total reward obtained by NL agents in the AiCoop role.

Table 11: H1 in scenario2: post-analysis.

Agent	**AU**		
	1	2	3
NL	68,333.94		
RL1		73,690.06	
RL2			75,952.66
p	1.000	1.000	1.000

Table 12: H2 in scenario2: post-analysis.

Agent	**TCT**	
	1	2
RL1	686.54	
RL2	697.46	
NL		889.46
p	0.392	1.000

The results are as follows. The reward gained by achieving CTs by the cooperative roles NL-AiC and NL-AiCoop are higher than those gained by the corresponding RL roles because they achieve more CTs (H11 and H12 are rejected and Tables 16 and 17 show how the agents compare with one other). However, the time invested in the CTs and the reward gained from them was not sufficient to match the reward gained by RL AiSs (RL-AiSs obtained approximately 88% of the total reward by accomplishing STs and NL-AiSs achieved 74%). Accordingly, the reward gained by achieving ST tasks is the largest part of the total reward and NL accomplishes much fewer ST tasks than

RLs. Thus, regarding the test of individual performance, H10 is rejected and RL2 is the agent type winner (Table 15 presents the distribution of the data collected regarding AiS roles). The reason for this result is that agents 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). With NL, it seems that the AiCs cannot make good enough predictions of ave_bid. Therefore they attempt coordination (or the AiC might even fail after the evaluation phase) even though the profit obtained after achieving the CT was not as high as the reward that was being gained by RL-AiSs. However, it is important to observe that the solution is not to avoid the CT tasks and only pursue STs in scenario2. Rather, the answer is to find the right balance between the two because in this scenario CTs always provide better rewards than STs. 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.

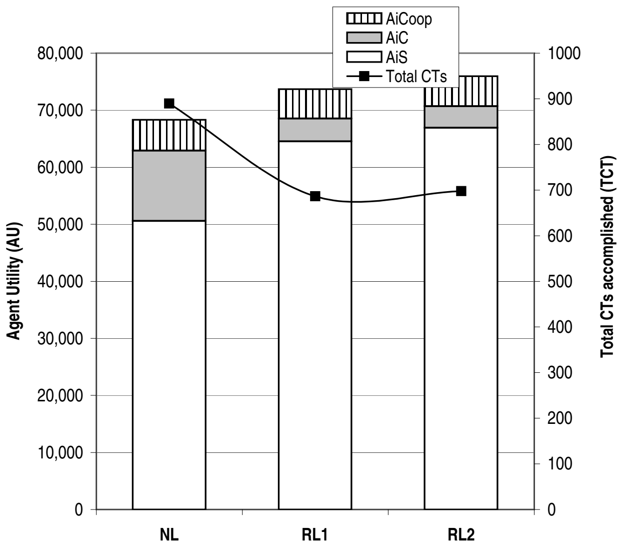

Figure 6: Contrasting agent's roles abilities in scenario2.

Table 13: Agent's role AU in scenario2.

Hypothesis to evaluate	p	Outcome	Winner
H10: $AiS_{RL1}=AiS_{RL2}=AiS_{NL}$	0.000	Rejected	RL2$^{(\cdot)}$
H11: $AiC_{RL1}=AiC_{RL2}=AiC_{NL}$	0.000	Rejected	NL$^{(\cdot)}$
H12: $AiCoop_{RL1}=AiCoop_{RL2}=AiCoop_{NL}$	0.000	Rejected	NL$^{(\cdots)}$

$^{(\cdot)}$ see Table 15, $^{(\cdot\cdot)}$ see Table 16 and $^{(\cdots)}$ Table 17 for details.

As this set of experiments shows, learning agents can take advantage of knowing which CM to apply in this more demanding environment. However, it is also important to evaluate how agents which employ the various reinforcement based algorithms compare with one another. To start with, the type of learning algorithms followed by the agents do not have a significant effect on the TCT (i.e. no matter how they learn to select the CM, agents still accomplish the same number of CTs (H4 is accepted and both algorithms formed one group in the post-analysis of Table 12)). However, RL2 performs better when evaluating AU (H3 is rejected). This is because the reward obtained per agent role indicates that RL2 agents perform better than RL1 in this environment. To this end, Table 14 shows the results of testing H13, H14 and H15 which read as follows:

H13: the total reward obtained by RL1 AiS agents is the same as that obtained by AiS agents which use the RL2 algorithm.

H14: the AU obtained by RL1 and RL2 agents in AiC role is the same.

H15: the total reward obtained by RLs Agents-in-Cooperation role (AiCoop) is identical.

From the results of Table 14 it can be seen that the roles that have a significant effect on the AU are AiC and AiS (H13 and H14 are rejected), whereas the AiCoop role does not make a significant difference to the final AU obtained (H15 is accepted). The explanation for this result is that RL2 agents are able to better balance their decision making about when to attempt coordination even though there is a significant degree of uncertainty about the outcome. This is achieved when an agent makes decisions about the CM based on the others' cooperative behaviour (which is exactly what is being modelled by RL2). Thus, although at first it might seem a bad performance to accomplish more STs than any other agent type, when these results are combined with the results of the other roles, it is clear that RL2 agents show a better performance than RL1. This is supported with the evidence that the biggest reward is gained by AiS agents and the reward gained by cooperative roles is the smallest amount.

Table 14: Agent's role AU in scenario2.

Hypothesis to evaluate	**p**	**Outcome**	**Winner**
H13: $\text{AiS}_{\text{RL1}}=\text{AiS}_{\text{RL2}}$	0.000	Rejected	RL2
H14: $\text{AiC}_{\text{RL1}}=\text{AiC}_{\text{RL2}}$	0.000	Rejected	RL1
H15: $\text{AiCoop}_{\text{RL1}}=\text{AiCoop}_{\text{RL2}}$	0.400	Accepted	None

Table 15: H10 in scenario2: post-analysis.
Role to analyse: AiS

Agent	**AU**		
	1	2	3
NL	50,599.18		
RL1		64,547.90	
RL2			66,911.98
p	1.000	1.000	1.000

Table 16: H11 in scenario2: post-analysis.
Role to analyse: AiC

Agent	**AU**		
	1	2	3
RL2	3,796.33		
RL1		3,999.72	
NL			12,295.16
p	1.000	1.000	1.000

In conclusion, 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, this was not the case for the more unpredictable environment of scenario2. As a result, agents might not only select an inappropriate CM, but they may also attempt coordination when this is 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 all agent types in scenario2 is considerably lower (TCT has a mean of 757.82) than the amount accomplished in scenario1 (TCT mean of 1,738.24). Being more concrete, 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 that exploit and explore the CMs is the most reasonable thing to do in dynamic environments because agents cannot be certain about the others' actions.

Generally speaking, 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. Moreover, incorporating the ave_bid in the learning process helps RL2 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.

Table 17: H12 in scenario2: post-analysis.

Role to analyse: AiCoop

Agent	**AU**	
	1	2
RL1	5,142.44	
RL2	5,244.35	
NL		5,439.59
p	0.060	1.000

7 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., (Shoham & Tennenholtz, 1992; Smith & Davis, 1981; Durfee & Lesser, 1991; Rosenschein & Zlotkin, 1994)). 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 (Durfee, 1999) 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 (Jennings, 1993). 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. (Barber, Han, & Liu, 2000) 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 (Boutilier, 1999) 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 (Sen & Weiss, 1999; Stone & Veloso, 2000). 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 (Nagayuki, Ishii, & Doya, 2000; Hu & Wellman, 1998; Claus & Boutilier, 1998). Generally speaking, this strand of work deals with creating an explicit representation of other agents in order to predict their actions so that an agent can take more informed decisions in the future. In the second case, Q-learning has been applied to learn how to coordinate or cooperate to achieve common goals by using specific strategies (Tan, 1993; Sen, Sekaran, & Hale, 1994). 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 -regarding coordinarion is the COLLAGE (Prasad & Lesser, 1999) and LODES (Sugawara & Lesser, 1998) 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 make decisions about coordination, rather than dealing with all problems might occur among the interactions specific interaction. 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.

In more recent research, Garland and Alterman (Garland & Alterman, 2001, 2004) propose the use case-based planning and learning probability estimates to allow agents to better coordinate. In particular, agents do learn on-line from past experience so that they take more informed decisions about which plan is "the" appropriate to execute in a particular coordination problem. In this work however, the learning outcome is to improve the decision making about planning, communication and adaptation of plans. This point of view is different from ours where it is assumed that planning is one instance of a CM and then the question to answer is whether planning should be selected in a specific circumstance.

In our previous reasearch work we have shown that autonomous agents that make run-time decisions about the most appropriate mechanism to coordinate their activities exhibit better performance than those that do not (Excelente-Toledo, 2003; Excelente-Toledo & Jennings, 2004). However, although the agents' coordination decisions are more effective and efficient, as the environment becomes more dynamic and unpredictable, there is a greater need to exhibit behaviour that is tailored to the agents' prevailing problem solving context. Thus, (Excelente-Toledo & Jennings, 2002) and (Excelente-Toledo & Jennings, 2003) present a preliminary evaluation of how such flexibility can be achieved through learning and adaptation (specifically using Q-learning). However this work does not deal with modelling others' bids in the state representation and, moreover, it does not explore the effects of taking into account this key component of the agent's decision making.

8 Conclusions and Future Work

This paper analysed the use and the efficacy of agents learning to make better decisions about how to coordinate more effectively. 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 build upon (Excelente-Toledo & Jennings, 2002) to demonstrate that the more informed the decision making about the possible agents to coordinate with, the better the cooperative outcome. We also showed that learning was less effective when agents operate in more static environments in which they can make reasonably accurate predictions about their environment and other agents.

In order to focus on the learning issue, some knowledge is assumed in the framework about the agents and the scenario. However, one of the assumptions in this work is that the environment in which the decision making takes place is dynamic, open, and heterogeneous and agents face great difficulty when taking coordinating decisions. This is because in such environments it is impossible to enumerate in advance the wide variety of contexts in which coordination is likely to be needed. In fact, agents face a significant challenge even to know what agents are present at any given moment; because agents can enter and leave the system at any time (openness), because the system and its components are in a continuous state of change (dynamism), or because the agents themselves exhibit different behaviour, have different capabilities and have their own agenda (heterogeneity). In these cases, it is especially important to have agents that are capable of automatically tailoring their coordination decisions to respond to the prevailing context. Examples of such systems are Web applications, e-commerce systems and grid computing application.

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 (Excelente-Toledo, 2003) for several examples of how the scenario can be mapped into a variety of real world problems including transportation problems and coordinated information retrieval) [12]. In particular the issues of how to exploit learning techniques to allow agents to make decisions based on their experience is a key aspect that needs broader investigation. We believe that the results presented here among others 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). 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 (Vidal & Durfee, 1997)) by explicitly representing knowledge about others or about the effect of their interactions. In this paper, we are addressing only part of the problem by assuming that the actions performed by other agents alter the environment that the agent is perceiving and sensing. Thus, agents do not explicitly model the behaviour of others. It is important to address this aspect 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 refined given the efficiency of the CMs actual execution).

[12]In (Excelente-Toledo, 2003) we mapped our grid world scenario into a package delivery problem (Rosenschein & Zlotkin, 1994) where trucks are the agents that move around the grid with parcels to deliver at specific locations. The final destinations for their parcels correspond to the agents' specific tasks. There are special packages (a package being a group of parcels) that have to be delivered by more than one truck (because of their size) and these correspond to CTs. The truck's goal is to deliver a number of parcels to specific locations. The more parcels delivered by a truck, the more profit it receives. Similarly, the coordinated information retrieval domain consists of having a number of agents with the task of downloading documents from specific locations in the Internet (Huhns & Stephens, 1999). The action of downloading has an associated cost that represents the price paid for the use of the server. The agent's objective is to reduce, as far as possible, the cost of downloading. Each time an agent has a document to retrieve, it might download it by itself or it could minimise the cost by coordinating its activities with those of other agents that are also interested in the same document.

In both exemplar applications, all the concepts and constituent factors of the decision making framework discussed in Section 3 were mapped into the formulations with only minor changes. This endeavour highlights the fact that the scenario and framework portrays and describes the key coordinating processes that can be found in concrete applications domains (as well as the more generic testbed) meaning that the framework does indeed have a broader applicability. And most probably, depending on the domain, some additional concepts (the truck space for example in the transportation domain) will need to be incorporated or modelled more precisely (in the coordinated information retrieval domain, the use of real time could have been employed instead of the routing connection). However, those concepts that have already introduced represent the major ones in which many examples can be mapped and tested and are not specific to the grid world scenario outlined in this work.

Acknowledgment

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 A. C.

References

Barber, K. S., Han, D. C., & Liu, T. H. (2000). 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)*, pp. 1–15 Melbourne, Australia.

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

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

Claus, C., & Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pp. 746–752 Madison, MI.

Cohen, P. R. (1995). *Empirical Methods for Artificial Intelligence.* The MIT Press: Cambridge, MA.

Durfee, E. H. (1999). Practically coordinating. *AI Magazine*, *20*(1), 99–116.

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

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

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

Excelente-Toledo, C. B., & Jennings, N. R. (2003). Learning when and how to coordinate. *Web Intelligence and Agent Systems: An International Journal*, *1*(3-4), 203–218.

Excelente-Toledo, C. B., & Jennings, N. R. (2004). The dynamic selection of coordination mechanisms. *Journal of Autonomous Agents and Multi-Agent Systems*, *9*(1-2), 55–85.

Excelente-Toledo, C. B. (2003). *The Dynamic Selection of Coordination Mechanisms.* Ph.D. thesis, Department of Electronics and Computer Science, University of Southampton. http://eprints.ecs.soton.ac.uk/7814/.

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

Galbraith, J. (1973). *Designing Complex Organizations.* Addison-Wesley Publishing Company, Inc. Reading, MA.

Garland, A., & Alterman, R. (2001). Learning procedural knowledge to better coordinate. In Nebel, B. (Ed.), *Proceedings of Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pp. 1073–1083 Seattle, Washington. Menlo Park, CA: AAAI Press.

Garland, A., & Alterman, R. (2004). Autonomous agents that learn to better coordinate. *Autonomous Agents and Multi-Agent Systems*, *8*(3), 267–301.

Hu, J., & Wellman, M. P. (1998). Online learning about other agents in a dynamic multi-agent system. In *Proceedings of Second International Conference on Autonomous Agents (AGENTS'98)*, pp. 239–246 Minneapolis, MN.

Huhns, M. N., & Stephens, L. M. (1999). Multiagent systems and society of agents. In Weiss, G. (Ed.), *Multiagent Systems: A Modern Approach To Distributed Artificial Intelligence*, Chapter 2, pp. 79–120. The MIT Press, Cambridge, MA.

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

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

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

Lane, D. M. (2001). Hyperstat online textbook. http://davidmlane.com/ hyperstat/. January/2003.

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

Nagayuki, Y., Ishii, S., & Doya, K. (2000). 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)*, pp. 215–221 Boston, MA.

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

Rosenschein, J. S., & Zlotkin, G. (1994). *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers.* The MIT Press: Cambridge, MA.

Russell, S. J., & Norvig, P. (1995). Reinforcement learning. In *Artificial Intelligence: A Modern Approach*, Vol. Learning, Chapter 20, pp. 598–624. Prentice Hall: Upple Saddle River, NJ.

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

Sen, S., & Weiss, G. (1999). Learning in multiagent systems. In Weiss, G. (Ed.), *Multiagent Systems: A Modern Approach To Distributed Artificial Intelligence*, Chapter 6, pp. 259–298. The MIT Press: Cambridge, MA.

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

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

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

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

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

Sutton, R. S., & Barto, G. A. (1998). *Reinforcement Learning: an introduction.* The MIT Press: Cambridge, MA.

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

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

Watkins, C. J. C. H., & Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, *8*, 279–292.