Multi-Agent Learning in both Cooperative and Competitive Environments

Francisca Teixeira, António J. M. Castro, Ana Paula Rocha, Eugénio Oliveira

LIACC, Department of Informatics Engineering,
Faculty of Engineering, University of Porto,
Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
{ei09139,antonio.castro,arocha,eco}@fe.up.pt

Abstract. Our intent is to present a mechanism suitable for agents that, immersed in an environment that is simultaneously cooperative and competitive, have to learn its own best behaviour not only from an individual point of view but also from a global perspective of the system. We consider the learning mechanism we propose to be a multi-agent learning mechanism not only because there are multiple agents learning concurrently in the same environment but also because it allows them to understand how to improve their performance and still not to damage the performance of the other agents. We tested our learning mechanism over the Disruption Management in Airline Operations Control Center application domain and the results show that it provides a good performance to the agents in cooperative as well as in competitive situations in the environment.

1 Introduction

The implementation of learning mechanisms in multi-agent systems has proved to be useful in both cooperative or competitive environments [8] [10]. Our intent is to present a mechanism suitable for agents that, immersed in an environment that is simultaneously cooperative and competitive, have to learn the best behaviour not only from an individual point of view but also from a global perspective of the system. The motivation for this research is a multi-agent system named MASDIMA (Multi-Agent System for Disruption Management) [2] [3], which is a system responsible for solving disruptions in airline operational plans and that is currently under development. Although this mechanism was designed and developed for this specific system, it has been designed in such a way that, in our opinion, it can also be applied successfully in other scenarios.

Consider a group of Respondent Agents (RA) that have been asked for potential solutions to a given problem. This problem can be decomposed on several different but interdependent dimensions and each of the RAs is responsible for only one of those dimensions. However, the problem must be solved as a whole. Therefore, the several RAs have to cooperate in order to solve the problem. This can be done through a sequential approach, where the first agent solves the problem for its dimension, imposes certain restrictions to the next agents (to

deal with the interdependencies between dimensions) and sends its solution as an input (and also a constraint) for the second agent, and so on along a chain of agents until the problem is solved. However, this approach will shorten the set of solutions, as the second agent has already certain restrictions it cannot violate. In our approach, we use a multi-round negotiation process between all RAs: each one of them will concurrently start the problem solving process by finding a solution for its own dimension and will ask to all other agents to complete it. After receiving all partial solutions, each one of the RAs will propose an integrated solution which includes its own solution on the dimension it is responsible for and, at the same time, includes possible solutions to the other dimensions. These proposals are then sent to the Organizer Agent (OA) which represents the global perspective of the system.

The OA evaluates the received proposals, chooses the best one and classifies the agent who proposed it as the winner of the round. It also provides some additional qualitative feedback regarding its own evaluation about all attributes of the proposal. This information is then sent by the OA to all RAs. The RA who won the previous round will maintain its proposal. The loser RAs shall adapt their proposal to the qualitative feedback received in order to propose a more suitable solution to the preferences of the OA in the future. Notice that the real values of those preferences are private to the group of RAs. The qualitative feedback received is the only information available for them to adapt their proposals.

By this description, it is clear that the RAs have two different roles in the negotiation. They need to cooperate among them in order to solve the given problem. At the same time, they are competing against each other as they all want to be the winner of the negotiation. The purpose of the learning mechanism is to endow the RAs with the capability of learning which proposals are seen as the best from the point of view of the OA and, at the same time, to maximize its own specific utility. We consider our learning mechanism a multi-agent learning topic not only because there are multiple agents learning concurrently but also because the agents have to understand what decisions (and actions) will help them to help the other agents. The agents have to learn how to improve its performance and still not to damage the performance of the other agents.

The rest of this paper is structured as it follows. In Section 2 we present the related work. Section 3 is the main one, in which we present our learning mechanism. The experiments made and results are presented in Section 4 and we conclude in Section 5.

2 Related Work

The learning process in multi-agent systems has been developed according to different methodologies: supervised learning, non supervised learning and reinforcement learning [11] [8]. Our intent is to present a mechanism that follows this last methodology and therefore we will not explore nor the concepts nor the systems using the other methodologies. We have chosen reinforcement learning

since it seems the one that best adapts to the context where we want to implement the learning mechanism: the environment is very dynamic. Note that it is either impractical or useless to obtain examples of what the agents should do because humans can not be aware of all constraints of the environment and the solutions proposed by humans are not always the best possible ones. Therefore, the agents should learn by interaction. We have used Q-Learning [13], which is a popular reinforcement learning algorithm whose performance has been studied in several application domains such as [7] or [12].

The studies [14], [9] and [6] are good examples on how to use learning in the context of a negotiation. However, the agents participating in the negotiation are either only competing or just collaborating, and they do not need to be concerned both with their own utility and the overall utility of the system. Nevertheless, these systems provided a good basis to the development of the mechanism we present in this paper.

According to Panait & Luke [5] cooperative agents try, through interaction and together, to perform a given task or maximize a global utility. The authors divide the several mechanisms by team learning or concurrent learning. In team learning the group of agents is seen as a team and there is only a single learning mechanism which is shared by all team members. Opposed to this approach is concurrent learning, where each member of the team has its own mechanism that is running concurrently to the others. We will not enumerate the advantages and disadvantages of each approach, which can be found at [5]. Concurrent learning seems to be the method more suitable for the problem we present in this paper, since it showed to be well suited for agents competing and, simultaneously, allows a decomposition of the problem in sub-problems. This last characteristic is specially important when the state space is larger, since it decreases significantly comparing with team learning, leading thus to a decrease of complexity.

In concurrent learning there are important issues to be considered [5], whether in a cooperative or competitive environment. One of them is the dynamics of the environment since there are multiple agents learning concurrently. The agents are likely to develop a cyclic adaptive behaviour to the other agents strategy. If the agents are cooperative, the responsibility to solve a given task should be shared by the entire team. The outcomes of performing an action, whether they are good or bad, should be shared and distributed among all agents. This is directly related to reward assignment, which is an important area discussed in researches such as [4] and [1].

Other important issue to consider in systems with cooperative agents is the development of a selfish behaviour. In other words, the agent is not wanted to have high utility values whereas the global utility of the system is low. Therefore their consciousness about what actions improve the overall utility must be strong and should be preferable to the actions that lead him to a selfish behaviour. An agent should try to learn about the decisions of the other agents in the environment so it can execute its own actions accordingly to the team behaviour.

3 Learning Mechanism

The learning mechanism we are proposing may be applied in different contexts. For better understanding the underlying concepts we will use the Disruption Management in Airline Operations Control Center (AOCC) application domain.

MASDIMA [2] [3] provides the AOCC with an efficient tool for solving disruptions through a process of negotiation. A disruption is an interruption to the regular flow of the airline operational plan that may rise a problem affecting three different and interdependent dimensions: aircraft, crew member and passenger. There are three respondent agents responsible for each one of these dimensions: Aircraft Manager, Crew Member Manager and Passenger Manager. These Managers[1] have to propose integrated solutions compliant with the dependency between dimensions to the organizer agent, which is called Supervisor. Each Manager does not achieve a solution by itself. Instead, it starts by solving the problem for its own dimension and then it asks the other Managers to complete its solution by complying with restrictions that are imposed to them. Consider that any solution is composed as described in Formula 1. There are two attributes by dimension (cost and delay). The Aircraft, Crew Member and Passenger dimensions are represented by A, C and P, respectively.

$$Solution =< cost_A, delay_A, cost_C, delay_C, cost_P, delay_P > \quad (1)$$

Each Manager sends its integrated solution to the Supervisor for evaluation. The Supervisor agent has a private (the Managers do not know them) utility function and preference values that are used for evaluating the proposals received. The Supervisor must choose a winner proposal in each round and communicate the result to the Manager agents. Each Manager receives feedback only about its own proposal. The feedback consists in information if its proposal won the last round and a qualitative value given to each attribute of its proposal. Note that these values do not reveal the exact value of the preferences of the Supervisor agent.

Our intent is to provide the Manager agents with learning capabilities that allow the agents to efficiently adapt their proposals to the Supervisor agent preferences, which they do not know. As we stated in Section 2, it is our opinion that the learning mechanism should use a reinforcement learning algorithm based on the feedback sent by the Supervisor about the Manager's last proposal.

In Q-Learning algorithm there is a value associated to each $state-action$ pair called $q-value$. The $state-action$ pair represents the execution of an $action$ when the agent is at a given $state$. The $q-value$ represents the estimated expected accumulated reward that is achieved by executing a given action in a given state. The $q-value$ will increase or decrease according to the quality of the new $state$ achieved. The greater the $q-value$, the better is the pair.

Each Manager agent has its own knowledge and is responsible only for part of the problem. Also, the Manager agents have a double role in the negotiation: they have to cooperate in order to build integrated solutions but they are also

[1] The Manager agents play the role of RAs described in section 1.

competing to be the winner of the negotiation. These two facts withdrew the idea of team learning. The agents do not share knowledge, as they are responsible for different dimensions. Yet, we must not forget that the Manager agents are competing against each other. By sharing the same learning process they would have access to the same strategy, which goes against the principles of competition. In our approach we used concurrent learning: each Manager agent has its own learning mechanism. This approach, besides solving the problems listed above, has the advantage of reducing the state space of the mechanism, which is very important for a fast convergence. When a Manager asks the other Managers to complete its partial solution, it is already imposing certain restrictions that those Managers can not violate, so that the dependencies between dimensions are guaranteed. They must complete the solution according to the plan already defined by the first Manager. In our opinion, they do not have the same responsibility than the first Manager in its proposal. If any reward, whether it is good or bad, is to be distributed by all Managers, we have to consider the weight of responsibility on that proposal of each Manager.

When modelling the learning mechanism, we tried to express what we had previously said by designing three different learning engines (algorithm running instances) for each Manager. One was for the negotiation with the Supervisor, while the other two were for requests between the Managers. Preliminary results showed that these engines were too restrictive when presenting proposals. We decided to modify certain aspects of the first mechanism that leads us to a new version with different preliminary results. We will explain both mechanisms and stress its main differences. The first version of the mechanism is named A while the final version is named B.

In A, the idea is to avoid mixing the different contexts of negotiation. The Manager is not in the same situation when he is dealing with the Supervisor or when he is responding to the other Manager requests. This was why we came with the idea of creating three different learning engines per Manager. The first agent should inform the other Managers about the feedback received, and it is up to them to update their learning engines and to improve their partial solutions, even with all restrictions that the first Manager imposed.

In B, we disabled the learning engines between Managers. Each Manager has only one engine that is used for the negotiation with the Supervisor. In this approach, we try to make the Manager that imposes restrictions to have more consciousness about his collaborators capabilities, and to act accordingly to them. This is done through the representation of the state and reward. We will describe these differences in the next subsection.

3.1 State, Action and Reward

The state is a knowledge representation of the present situation of the agent. The representation of the state in A and B can be stated as Formulas 2 and 3. In the application domain we are using as case study the solution changes according to the problem being solved which is characterized mainly by the cause and resources affected. We believe that the state must express this information. The

attribute *cause* may have 13 distinct values, which correspond to the main causes of disruptions we have identified. The attribute *resource* varies according to the dimension, but we have considered only 2 distinct values in each dimension. The values of these attributes are connected to the application domain and we believe not to be necessary their definition to the comprehension of the problem.

The learning mechanism should allow the Manager to learn how to improve its own proposals. The Manager must have knowledge about its current situation in the negotiation. The qualitative feedback received is included in the representation of the state through $costFeedback$ and $delayFeedback$ attributes.

$$A : state =< cause, resource, costFeedback, delayFeedback > \quad (2)$$

$$B : state =< cause, resource, costFeedback, delayFeedback, agents > \quad (3)$$

$$costFeedback, delayFeedback \in \{veryhigh, high, ok, low\}$$

$$agents \in \{good, bad, verybad\}$$

The difference between the two representations is the attribute *agents*, which tries to express the other two Managers situation and exists only in the representation of the mechanism B. Its value is calculated according to the quality of the feedback given to the attributes of the other dimensions. To calculate this quality we assigned a punctuation to each value of the feedback (*veryhigh* is 2.0, *high* is 1.0, *ok* is 0.0 and *low* is 0.5) and, after counting all points, a final value is given according to the Formula 4.

$$agents = \begin{cases} \text{if } 0 \geq \text{points} \leq 2, & good \\ \text{if } 2 > \text{points} \leq 4, & bad \\ \text{if points} > 4, & very\ bad. \end{cases} \quad (4)$$

The *agents* attribute will allow the Manager to have more consciousness about the other Managers situation, since its own state depends on the feedback assigned to the attributes of the other Managers dimensions. In mechanism A the state expresses only the individual situation of each Manager.

The representation of an action is the same for both mechanisms A and B and is presented in Formula 5. Both attributes $costAction$ and $delayAction$ refer to an action to apply on the value of the correspondent attributes of the partial solution proposal. Their value shall increase, decrease or be the same, according to the action *inc*, *dec* or *keep*, respectively.

$$action =< costAction, delayAction > \quad (5)$$

$$costAction, delayAction \in \{inc, dec, keep\}$$

The reward received by each Manager depends on whether it is the winner of the round or not. If so, it will receive a value that will increase the $q - value$ of the pair state-action selected. Otherwise, it will receive a penalization according to the feedback received (grater values to worst classifications).

$$A : R = \begin{cases} \text{if winner}, & m \\ \text{if loser}, & \frac{m}{2} - \sum_{i=1}^{m} penalization_i. \end{cases} \quad (6)$$

$$B : R = \begin{cases} \text{if winner,} & m \\ \text{if loser,} & \frac{m}{2} - \sum_{i=1}^{m} penalization_i \times discount_i. \end{cases} \quad (7)$$

In both Equations 6 and 7 m is the number of attributes considered. In A (Equation 6) the Manager does not use the feedback given to the attributes of the other dimensions ($m = 2$). Its penalization is calculated only according to the feedback given to the attributes of its own dimension. In B (Equation 7), the penalization received depends on all attributes ($m = 6$). We have applied a discount factor to the attributes of the other dimensions. If the attribute belongs to the dimension *discount* is 1.0, if not is 0.4. With this discount factor we tried to take into account the degree of responsibility of each Manager in the solution proposal, which we think may be different.

3.2 Workflow

We will give a practical example of the workflow. Imagine that the Supervisor has given the feedback represented at Formula 8 to the Aircraft Manager proposal which was not the winner of this round.

$$AircraftProposalFeedback =< ok, low, high, ok, veryhigh, low > \quad (8)$$

In mechanism A, the Aircraft Manager must communicate the partial feedback to the respective manager. The Crew Member Manager will receive only its part of the feedback, which are the third and forth attributes, as well as the Passenger Manager, which will receive the fifth and sixth attributes. Each Manager will then update its learning mechanism engine. While the Aircraft Manager is updating the engine for direct negotiation with the Supervisor, the other two are updating an engine for the request of the Aircraft Manager. These engines will not influence the Managers on their negotiation with the Supervisor. They will influence their behaviour only when the Aircraft Manager asks them to complete its solution.

In mechanism B the Aircraft Manager will not communicate the feedback to any Manager. Its part of the feedback will be used to build the state. The other attributes will be used to calculate the fifth attribute of the state, as represented in the Formula 9.

$$B : AircraftState =< cause, resource, ok, low, bad > \quad (9)$$

In mechanism A, the responsibility of improving the Crew Member and Passenger Manager proposals only belongs to these agents. They know what was their evaluation and they must adapt their partial solutions to the feedback sent by the Aircraft Manager. However, they already have the set of restrictions imposed by the Aircraft Manager. From the preliminary results, it is our opinion that the learning mechanism resulted in more restrictions hindering the Managers in presenting more valuable proposals.

With mechanism B, we tried to make the Aircraft Manager more cooperative and flexible to its collaborators. We wanted the Aircraft Manager not to select only the actions that are better for itself, but those that are more useful for all Manager agents and thus for the overall system.

4 Experimentation, Results and Discussion

We have performed experiments with three different approaches: the system without the learning mechanisms (NL), the system using the learning mechanism A (LA) and the system using the learning mechanism B (LB) as presented in the previous section. It is important to point out that the performance of the system without any learning mechanism has already been compared to the performance of the traditional sequential approach used in the AOCC in previous researches [2]. Therefore, we will not focus on the improvements over this traditional approach but we only compare the three versions (NL, LA and LB) regarding the new MAS-based approach.

Our system was developed and tested using real data from TAP Portugal airline company. For the experiments we used the records of a whole month of airline operations which contained 49 case problems affecting the plan. The data presented for each version is an average of n sequential executions of the system. In each execution the system solves and presents a solution for the 49 existent case problems. The results obtained and presented bellow are extracted from the achieved solutions in those three executions. The metrics we use to compare the different versions of the system are defined as it follows:

- **Average Global Utility** This value is calculated according to the global utility of each winner solution, which is a value that expresses the cost and delays implied by all dimensions of a specific solution, according to the preferences of the Supervisor[2]. The less are the costs and delays the higher is the utility of the solution. Higher utilities are best. The value varies in the range [0,1] and is calculated as it follows:

$$\overline{U_{global}} = \frac{\sum_{e=1}^{n}(\overline{U_{global}}_e)}{n} \tag{10}$$

  $\overline{U_{global}}_e$ is the arithmetic average of the global utility of each winner solution for the case problems included in one execution of the system.

- **Average Aircraft Manager Utility** This value is calculated according to the individual Aircraft Manager utility ($\overline{U_A}$) of each winner solution, which is a value that expresses the cost and delays of its dimension for a specific solution, according to the preferences of the Aircraft Manager [2]. The less are the costs and delays of its dimension the higher is the utility of the solution. Higher utilities are best. The value varies in the range [0,1] and is calculated as it follows:

$$\overline{U_A} = \frac{\sum_{e=1}^{n}(\overline{U_A}_e)}{n} \tag{11}$$

  $\overline{U_A}_e$ is the arithmetic average of the utility of the aircraft partial-solution included in the negotiation winner solution for the Aircraft Manager, for the case problems included in one execution of the system.

- **Average Crew Member Manager Utility** This value is calculated similarly to Average Aircraft Manager Utility. However, once we are calculating the individual Crew Member Manager utility ($\overline{U_C}$) [2], we will have to replace $\overline{U_{Ae}}$ for $\overline{U_{Ce}}$ and $\overline{U_A}$ for $\overline{U_C}$ in Equation 11.

- **Average Passenger Manager Utility** This value is calculated similarly to Average Aircraft Manager Utility. However, once we are calculating the individual Passenger Manager utility ($\overline{U_P}$) [2], we will have to replace $\overline{U_{Ae}}$ for $\overline{U_{Pe}}$ and $\overline{U_A}$ for $\overline{U_P}$ in Equation 11.

- **Average Δ to the Domain Optimal Solution** Taking the initial operational plan as the optimal solution, one can compare the difference between the utility of the optimal solution ($Uopt_i$) and the utility of the solution achieved by the system ($Uglobal_i$), for each case problem i.

$$\overline{\Delta\left(optimal\right)}_i = Uopt_i - Uglobal_i \tag{12}$$

$$\overline{\Delta\left(optimal\right)} = \frac{\sum_{e=1}^{n}(\overline{\Delta\left(optimal\right)}_e)}{n} \tag{13}$$

  $\overline{\Delta\left(optimal\right)}_e$ is the arithmetic average of the Δ to the domain optimal solution of all case problems included in one execution of the system.

The Utility Functions for each agent are different, including the weights and preferred values. We start our analysis by comparing the average of the utilities of the solutions achieved by each approach. As one can see in Figure 1, the system already had a very good average on the global perspective when working without any mechanism of learning. Nevertheless, both the approaches LA and LB were capable of increasing this value, specially the approach LB, which increased the global utility in 3.0% when compared with NL. The most interesting results were those related to the individual utility of each Manager. Although the approach LA registered some improvements over NL, it was with the approach LB that the agents increased their individual utility in a significant way.

This results are, in our opinion, the confirmation of our hypothesis over the approaches LA and LB. In LA we tried to make the Managers that were requested more responsible for the proposals, by implementing a learning engine between Managers. This method lead to too much restrictions, since they had to calculate the partial solution taking into account the restrictions of the Manager that made the request (due to the fact of having interdependent dimensions). The Manager who requested the partial solution was not taking into account that its solution could affect the others, given the restrictions associated: it took the selfish position by proposing what was individually more useful for itself. This was also true for the Managers who were requested. They were trying to reach a global solution only by measuring their own utility.

Unlike the approach LA, in LB the Managers requesting the partial-solutions are even more responsible. They have a broader perspective of the integrated

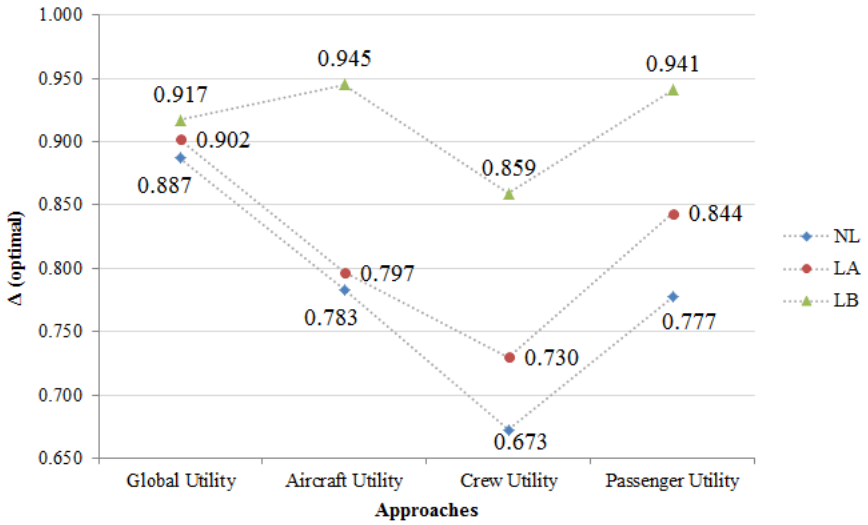

Fig. 1. Average of utilities for each Manager and for the global system by approach.

solution and they can adapt their partial solutions to the other Manager's performance. Without the restrictive engines of the mechanism A, and together with the first Manager strategy of taking into consideration their capabilities, the other Managers were capable of completing the solution, complying with the dependencies between dimensions and with more utility values for themselves. This resulted in solutions with increased global utility. We consider to have achieved a state of mutual cooperation between the agents: the agents try to help the others as much as possible.

Once the environment where the agents are is also competitive, we consider that each agent has individually taken advantage of the approach LB. Comparing LB to NL (see Figure 1), the lowest increment in the individual utility of all Managers was of 16.2% which is a considerable progress. Even when compared to LA, the lowest increment was of 9.7% which is also a significant increase.

Looking at the results in Figure 2, we can verify that the LB approach reflects improvements over both approaches LA and NL, for its solution are closer, in terms of utility, to the initial planning and scheduling of the operational plan. Given these results, we think we have achieved, with the approach LB, a good multi-agent learning mechanism for both competitive and cooperative environments, once the results show improvements in the individual utility of the respondent agents without having a negative impact over the global utility of the system.

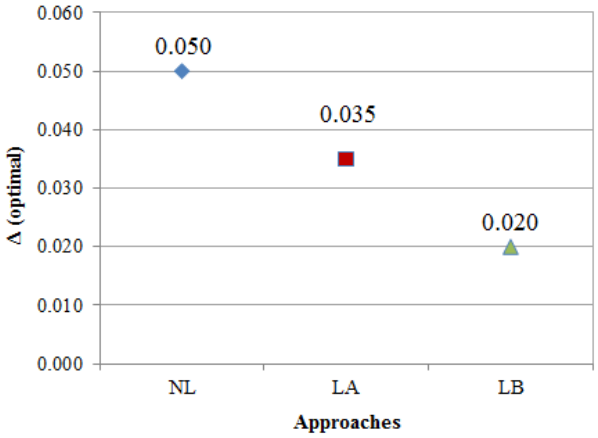

Fig. 2. Average Δ to the Domain Optimal Solution (lower values are better).

5 Summary

Our main goal was to present a learning mechanism suitable for a group of agents that need to collaborate among them even though they are in a negotiation environment and, therefore, competing against each other.

The learning mechanism should allow them to understand what proposals were more useful not only from their point of view but also from a global perspective of the system. We used reinforcement learning concepts for the implementation of the mechanism.

First we introduced the problem from a broad perspective and then we used the application domain of Disruption Management in Airline Operations Control Center as the context to our hypothesis and experiments.

Two learning mechanisms with different architectures and concepts were presented. In the first learning mechanism (A) we proposed a more individual point of view for each agent, although with a certain concern from the agents in helping each other. Yet, this concern was still a little bit selfish as the agent was measuring only its own individual utility even when helping the others. The second learning mechanism we proposed came out as an evolution of the first. In the learning mechanism B, we tried to make the agent conscious about the situation of the other agents. We modelled all other agents current situation into the agent's own state so that the agent could be more reasonable in its proposals formulation taking into account other agents capabilities.

Our experiments were done using different versions of MASDIMA, which is a multi-agent system for solving disruptions in previous plans: without any learning mechanism, with the version A of the learning mechanism and with the final and improved version B of the learning mechanism. The results presented

show that learning mechanism B had the best performance, providing an efficient learning mechanism for these kind of environments.

It is our opinion that the final version of the learning mechanism presented can be applied to other contexts involving both cooperative and competitive environments. However, since we did not tested the mechanism outside the Disruption Management application domain, further research is needed in order to be more conclusive.

References

1. Tucker Balch. Behavioral diversity in learning robot teams. Technical report, 1998.
2. Antonio J. M. Castro and Eugenio Oliveira. A new concept for disruption management in airline operations control. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 3(3):269–290, March 2011.
3. Antonio J. M. Castro, Ana Paula Rocha, and Eugenio Oliveira. Towards an autonomous and intelligent airline operations control. In *Proceedings of the 2012 15th IEEE Conference on Intelligent Transportation Systems (ITSC 2012)*, pages 1429–1434, Anchorage, Alaska, USA, September 16-19 2012.
4. H. Wolpert David and Kagan Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(02n03):265–279, 2001.
5. Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11:387–434, 2005.
6. Ana Paula Rocha. *Metodologias de Negociação em Sistemas Multi-Agentes para Empresas Virtuais*. Phd thesis, Faculty of Engineering, University of Porto, December 2001.
7. Eduardo Rodrigues Gomes and Ryszard Kowalczyk. Dynamic analysis of multiagent q-learning with ε-greedy exploration. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 369–376. ACM, 2009.
8. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, December 2009.
9. Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for robocup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.
10. Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.
11. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998.
12. Keiki Takadama and Hironori Fujita. Toward guidelines for modeling learning agents in multiagent-based simulation: Implications from q-learning and sarsa agents. In Paul Davidsson, Brian Logan, and Keiki Takadama, editors, *Multi-Agent and Multi-Agent-Based Simulation*, volume 3415 of *Lecture Notes in Computer Science*, pages 159–172. Springer Berlin Heidelberg, 2005.
13. C. J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.
14. N.P. Ziogos, A.C. Tellidou, V.P. Gountis, and A.G. Bakirtzis. A reinforcement learning algorithm for market participants in ftr auctions. In *Power Tech, 2007 IEEE Lausanne*, pages 943–948, 2007.