Negotiation and Learning in Distributed MPC of Large Scale Systems

Valeria Javalera, Bernardo Morcego, Vicenç Puig

Abstract—**A key issue in distributed MPC control of Large Scale Systems (LSS) is how shared variables among the different MPC controller in charge of controlling each system partition (subsystems) are handled. When these connections represent control variables, the distributed control has to be consistent for both subsystems and the optimal value of these variables will have to accomplish a common goal.**

In order to achieve this, the present work combines ideas from Distributed Artificial Intelligence (DAI), Reinforcement Learning (RL) and Model Predictive Control (MPC) in order to provide an approach based on negotiation, cooperation and learning techniques.

Results of the application of this approach to a small drinking water network show that the resulting trajectories of the levels in tanks (control variables) can be acceptable compared to the centralized solution. The application to a real network (the Barcelona case) is currently under development.

Key Words: **Cooperative systems, Distributed control, Model Predictive Control, Multi agent Systems, Negotiation, Reinforcement Learning.**

I. INTRODUCTION

Large Scale Systems (LSS) are complex dynamical systems at service of everyone and in charge of industry, governments, and enterprises. The applications are wide. Examples of applications of LSS in continuous domains are: power networks, sewer networks, water networks, canal and river networks for agriculture, etc. Other examples of applications of LSS in discrete domain are traffic control, railway control, manufacturing industry, etc.

Model Predictive Control (MPC), also known as receding horizon control, is a control technique widely used in industry (see [1]) well suited for the control of continuous LSS. The theory of MPC is well developed; most aspects, such as stability, nonlinearity, and robustness, have been discussed in the literature (see [2]).

In MPC, the control input is obtained by solving a discrete-time optimal control problem over a given horizon, producing an optimal open-loop control input sequence. The first control in that sequence is applied. At the next sampling instant, a new optimal control problem is formulated and solved based on the new measurements.

Typically, MPC is implemented in a centralized way. The complete system is modeled, and all the control inputs are computed in one optimization problem. However, the increase of automation of LSS renders problems with a noticeable increase in complexity. Such complexity is due to the size of the system to be controlled and the huge number of sensors and actuators needed to carry out the control. Additionally, LSS are composed of many interacting subsystems. Thus, LSS control is difficult to be implemented using a centralized control structure because of robustness and reliability problems and due to communication limitations. For all these reasons, distributed and decentralized MPC control schemes have been developed and applied over the last years.

In decentralized systems the resulting subsystems are independent from each other. But the high level of connections and interdependence of LSS is the reason why, in most cases, they cannot be modeled as decentralized systems. In distributed systems, the resulting subsystems can have physical dependences between them and therefore communication among them.

One of the main problems of distributed control of LSS is how these dependence relations between subsystems are preserved. These relations could be, for example, pipes that connect two different control zones of a decentralized water transport network, or any other kind of connection between different control zones. When these connections represent control variables, the distributed control has to be consistent for both zones and the optimal value of these variables will have to accomplish a common goal.

The goal of the research described in this paper is to exploit the attractive features of MPC (meaningful objective functions and constraints) in a distributed implementation combining learning techniques to perform the negotiation of these variables in a cooperative Multi Agent environment and over a Multi Agent platform.

II. THE PROBLEM

In order to control an LSS in a distributed way, some assumptions have to be made on its dynamics, i.e. on the way the system behaves. Assume first that the system can be divided into n subsystems, where each subsystem consists of a set of nodes and the interconnections between them. The problem of determining the partitions of the network is not addressed in this paper; instead the reader is referred to [3]. The set of partitions should be complete. This means that all

This work was supported in part by the WIDE - 224168 - FP7-ICT-2007-2 project and by WATMAN ref. DPI2009-13744.

Valeria Javalera and Vicenç Puig are with the Institut de Robòtica I Informàtica Industrial (CSIC-UPC). Llorens i Artigas 4-6. 08028, Barcelona Spain. phone: 93 4015805; fax: 934015750; e-mail: vjavalera@iri.upc.edu. She is also supported by the Consejo Nacional de Ciencia y Tecnología (CoNACYT) México, D.F. and the Instituto Tecnológico Superior de Cajeme (ITESCA), Cd Obregón Son., México.

Bernardo Morcego and Vicenç Puig are with Universitat Politècnica de Catalunya. They belong to the Advanced Control Systems research group.; Rbla. Sant Nebridi 10, 08222 Terrassa (Barcelona), Spain; (e-mail: bernardo.morcego@upc.edu, vicenc.puig@upc.edu).

system state and control variables should be included at least in one of the partitions.

***Definition 1*. System partitions.** P is the set of system partitions and is denoted by

$$P = \{p_1, p_2, \ldots, p_{np}\} \quad (1)$$

where each system partition (subsystem) p_i is described by a deterministic linear time-invariant model that is expressed in discrete-time as follows

$$\begin{aligned} \mathbf{x}_i(k+1) &= \mathbf{A}_i\mathbf{x}_i(k) + \mathbf{B}_{u,i}\mathbf{u}_i(k) + \mathbf{B}_{d,i}\mathbf{d}_i(k) \\ \mathbf{y}_i(k) &= \mathbf{C}_i\mathbf{x}_i(k) + \mathbf{D}_{u,i}\mathbf{u}_i(k) + \mathbf{D}_{d,i}\mathbf{d}_i(k) \end{aligned} \quad (2)$$

Variables **x**, **y**, **u** and **d** are the state, output, input and disturbance vectors, respectively; **A**, **C**, **B** and **D** are the state, output, input and direct matrix, respectively. Subindices u and d refer to the type of inputs the matrix model, either control inputs or disturbances. Control variables are classified as internal or shared.

***Definition 2.* Internal Variables.** Internal variables are control variables that appear in the model of only one subsystem in the problem. The set of internal variables of one partition is defined by

$$U = \{u_1, u_2, \ldots, u_{nu}\} \quad (3)$$

***Definition 3*. Shared Variables.** Shared variables are control variables that appear in the model of at least two subsystems in the problem. Their values should be consistent in the subsystems they appear, so they are also called negotiated variables. V is the set of negotiated variables defined by

$$V = \{v_1, v_2, \ldots, v_{nv}\} \quad (4)$$

Each subsystem i is controlled by an MPC controller using:

- the model of the dynamics of subsystem i given by equation (2);
- the measured state $\mathbf{x}_i(k)$ of subsystem i;
- the exogenous inputs $\mathbf{d}_i(k+1)$ of subsystem i over a specific horizon of time;

As a result each MPC controller determines the values $\mathbf{u}_i(k)$ of subsystem i. The internal control variables are obtained directly by the MPC controller of this subsystem while the shared variables are proposed to be negotiated with the MPC controllers of the corresponding subsystem.

In *distributed control* the set of shared variables is not empty. The problem addressed in this paper is an agent based distributed control. There is one agent in charge of each system partition and its duties are to negotiate the shared variables with other agents and to calculate the control actions from the MPC formulation of its partition.

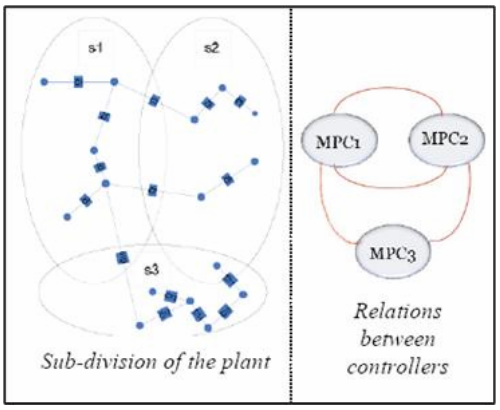

Figure 1: The problem of distributed control

Figure 1, on the left, shows a sample system divided into three partitions. There are three overlapping sets that contain four shared variables. The relations that represent those variables are shown on the right as lines. The problem consists in optimizing the manipulated variables of the global system in a distributed approach, i.e. with three local control agents that must preserve consistency between the shared variables.

III. Distributed MPC

In distributed control schemes, local control inputs are computed using local measurements and reduced-order models of the local dynamics [4].

Distributed MPC algorithms are classified into *iterative* and *non-iterative* and further sub classified into *independent* or *cooperative* algorithms.

In iterative algorithms information is bi-directionally transmitted among local regulators many times within the sampling time. In non-iterative algorithms information is bi-directionally transmitted among local regulators only once within each sampling time. [4] gives a review of distributed control architectures for LSS.

The aim of independent (non-cooperative) algorithms is to get better results than the other controllers, which are seen as opponents. They have also been applied in MPC distributed control strategies (see, e.g., [5]).

Contrarily, cooperative algorithms intend to find a compromise for shared variables in order to maximize the performance of the complete system, worsening if necessary the performance of individual partitions. [6] and [7] are two recent examples of the application of cooperative algorithms, the first one is non-iterative and the second one is iterative.

The proposed distributed architecture is non-iterative and cooperative and it uses learning techniques to provide a more accurate result. It is cooperative because although every subsystem has its local goal they all share a common one, so they need to cooperate to achieve it. The solution sought here is to achieve the Pareto optimal solution (like cooperative algorithms) provided by an ideal centralized control structure rather than letting each local regulator tend towards a Nash equilibrium (like independent algorithms).

Therefore, there is a compromise between local goals and the common goal. The proposed Architecture uses reinforcement learned negotiation to balance this situation.

IV. Reinforcement Learning.

Learning techniques are powerful tools used mainly in large and complex systems in dynamical environments. For the problem of negotiation in cooperative environments described above the application of RL is a good option.

In [8] the two most important distinguishing features of RL are given: the trial-and-error search and delayed reward. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards.

Reinforcement learning is based on past experience, which is used to reduce the need of iterative methods. It is a well known and formally studied family of learning techniques. Moreover, depending on the formulation of the problem and the richness of experience data, the possibilities of convergence are high.

Although the applications of RL are typically static, many control applications have been developed for dynamical environments [4], [10]. Even more, there are some works that relate MPC and RL, as in [11], where they are seen as complementary frameworks.

V. Negotiation in cooperative environments using MPC

Negotiation between distributed controllers in LSS is an open issue. Conventional negotiation techniques are not suitable for many reasons: calculation time, problems handling multiple restrictions and multiple objectives and the impossibility to ensure convergence are the most common reasons. Although there are successful results, there is a need of a methodology that can be used for all kinds of continuous LSS.

One of the most accepted techniques is the augmented Lagrangian method. The seminal Tamura coordination method was discussed in the book [12] even before MPC was first introduced. This method is based on using augmented Lagrangian technique to negotiate values on overlapping sub-networks in distributed large scale systems. Recent works have applied this method [13], [14], [6]. The main problems with this method are calculation time and the impossibility to ensure convergence.

Another negotiation approach is presented in [7], where an iterative, cooperating method for linear discrete-time systems is presented. In particular, the proposed approach guarantees the attainment of the global (Pareto) optimum when the iterative procedure converges, but still ensures closed-loop stability and feasibility if the procedure is stopped at any intermediate iteration. [4]. The main problem of this approach is again the calculation time.

In [15], an alternative approach to solve the same problem was discussed. The novelty of that approach involves maintaining the distributed structure of all the local controllers, but changing the objective functions so that the local agents cooperate. The main problem with this approach is that there is no systematic method to adjust the objective functions.

VI. Philosophy of the Negotiatior Agent algorithm

All the ideas and the proposed algorithm presented here are integrated in the MAMPC Architecture described in [16]. This architecture also provides a methodology that helps in the development process of the distributed MPC controller.

The main elements of the MAMPC architecture are *MPC Agents* and *Negotiator Agents*.

Definition 4. **MPC Agent.** An MPC Agent is the entity that is in charge of controlling one specific partition of the system.

There is one MPC Agent for each system partition. The MPC Agent solves an MPC control problem considering the *internal variables* of the partition and cooperating with one or more Negotiator Agents to determine the optimum value of the *shared variables*.

Definition 5. **Negotiator Agent.** A Negotiator Agent is the entity that is in charge of determining the value of one or more shared variables between two MPC Agents.

A negotiator Agent exists for every pair of MPC Agents that have one or more *shared variables* in common. Each MPC Agent is arranged to cooperate so that the negotiator agent solves the optimization of a common goal by means of a *Reinforcement Learning* algorithm. This algoritm is based on the Q-learning algorithm, and adapted to be applied in dynamical environments.

For the partitioning of the network purposes, in the distributed model the shared control variables have to be duplicated. This is done in order to provide each MPC-Agent involved in the relation with an internal representation of the shared variable.

The Negotiator Agent seeks to restore the connections broken in the distribution problem, connecting what was divided unifiying this dupplicate variables in just one as in the original model. Therefore, for the Negotiator Agent, this two control variables are taken as just one.

The philosophy of the negotiation algorithm proposed is to consider the shared variables not has a two diferent problems with conflicting goals but as one problem with just one goal, like in the centralized approach. The Negotiator Agent solves the optimization problem for these variable and communicate the result to the MPC-Agents at each sampling time. Since the MPC-Agents are able to cooperate, the MPC- Agents will take the value, set it as a hard contraint in its respective internal control variables and recalculate the multivariable control problem.

The optimization of the Negotiator Agent algorithm is based on its experience and in maximizing the reinforcements received of every action taken in the past on similar situations.

This algoritm is based on Q-learning algorithm, and adapted to be applied in dynamical environments. Next, the formulation of the algorithm is detailed.

A. Formulation of the negotiation-learning problem

Each shared variable constitutes an optimization problem that is assigned to a Negotiator Agent The internal architecture of the Negotiator comprises the following elements: Q-tables, a communication protocol and a negotiation algorithm. Next, these elements are described in further detail.

Q-table: the Q-table represents the knowledge base of the agent, which has a Q-table for each shared variable because each one can have diferent behaviour and even different goals.

Q-tables maintain the reinforcement gained for each possible state and action. A state represents the global state of each sub-problem, which is established in terms of the error of the output with respect to the goal. The definition of the error that MPC Agents use is:

$$\varepsilon_i = g_i - y_i \tag{5}$$

where ε_i is the error, g_i is the goal and y_i is the output of variable i.

The state value is determined by:

$$s = \frac{|\varepsilon_{i1}| + |\varepsilon_{i2}|}{2} \tag{6}$$

where ε_{i1} is the error of the variable i of first agent, and ε_{i2} of the corresponding variable in the second agent. This state is updated every sampling time.

Actions (a) are all the possible values that the shared variable can take.

Since states and actions are continuous, they have to be discretized for the application of the RL algorithm.

The reward function determines the reward of every action taken by the agent. In this case, the reward function is:

$$r = \rho - s \tag{7}$$

where ρ is a positive constant offset larger than the largest s.

Communication protocol: MPC Agents start the comunication by transmitting to the Negotiator Agent the relevant information of the current step: output vector, $y_i(k)$, control actions vector, $u_i(k)$, the absolute error with respect to the goal of the shared variable, $\varepsilon_i(k)$, and the sampling time k. Then, the algorithm of the Negotiator Agent is executed. When it finishes, it communicates the resulting value of the shared variable to the MPC Agents. In order to solve their MPC control problem they take those values as restrictions. After that, the procces starts again.

Negotiation algorithm: This algorithm is divided in two phases: the training phase and the exploitation phase. In both cases, the rule for updating Q-table values is:

$$Q(s,a) = r + (\alpha \times Q(s,a)) \tag{8}$$

Where $\alpha \in [0,1]$ weighs past experience importance.

The training phase creates a new Q-table off-line using stored data obtained, for instance, from the control actions determined by the centralized approach.

Once the Q-table is initialized, the exploitation phase can start. The main difference here is that next step actions are chosen according to

$$a' = \max_a(Q(s,a)) \tag{9}$$

in order to select the value of the action (negotiated variable) with maximum reward for the next time instant.

VII. Application Example

A small drinking water network is used to exemplify the proposed algorithm and its integration in the MAMPC architecture. The example was proposed in [17] where a centralized and a decentralized solution was studied and compared. This hypothetical water distribution network has 8 states (tanks) and 11 control variables (valves), see Figure 2. It can be divided into two subsystems. Two MPC Agents are used to determine the internal control variables of each subsystem. On the other hand, one Negotiator Agent is responsible of negotiating the values of the two shared control variables between the two MPC agents.

The control goal of the application presented in Figure 2 is to keep a volume in tanks around $3m^3$. The control objective of the centralized MPC is formulated as follows:

$$J = \min \sum_{k=1}^{H_p} \left(\sum_{n_x=1}^{8} (x_{n_x}(k) - x_{ref})^2 \right) \tag{10}$$

The system is decomposed in two partitions:

$$p_1 = \{x_1, x_2, x_4, x_5, x_6\} \tag{11}$$

$$p_2 = \{x_3, x_7, x_8\} \tag{12}$$

$$V = \{u_{10}, u_{11}\} \tag{13}$$

$$U_1 = \{u_1, u_2, u_6, u_7, u_8, u_9\} \tag{14}$$

$$U_2 = \{u_3, u_4, u_5\} \tag{15}$$

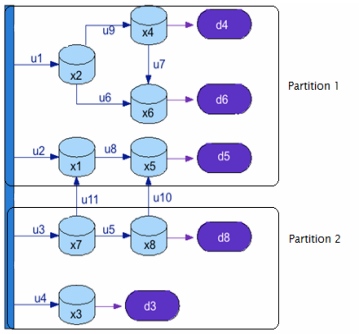

Figure 2: Case study and its partitioning

The plant is defined by all its state and input variables

$$Plant = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_{10}, u_{11}\} \quad (16)$$

An important step is to check that the partinioning of the plant leads to a complete set of partitions. This is accomplished verifying the following relation:

$$Plant = P \cup U \cup V \quad (17)$$

which can be easily checked here. Thus, the partition is a complete set of partitions. The control objective of each partition is the following:

$$J_i = \min \sum_{k=1}^{H_p} \left(\sum_{n_x \in p_i} (x_{nx}(k) - x_{ref})^2 \right) \quad (18)$$

The MAMPC Architecture for this problem is comprised of two MPC Agents, one for each partition, and one Negotiator Agent.

The maximum volume in tanks is 20m^3, the control value of the measured variables is from 0.0 to 0.4 except for u_2 that is from 0.0 to 0.1. The sampling time is 1 hour and the prediction horizon is 24 hours. The demands are considered as measured perturbations. They typically present a sinusoidal-like behaviour throghout the day.

The core of the MPC agent is an MPC controller. This controller solves the multivariable problem of one partition of the plant based on a model. This model contains the set u_x of the agent. Another important part of the MPC Agent is the communication block. MPC Agents can communicate in a sophisticated way because they are implemented using the Agent Oriented Paradigm. This paradigm provides methods, standards and tools that allow good communication skills.

An off-line training using the RL algorithm presented in Section VI was carried out in order to provide this experience to the Negotiatior Agent. As in any RL algorithm, the proposed architecture is based on the agent experience and the expected reinforcements. The richer the agent experience has been, the more efficient the optimization algorithm will be. Thus, as a good starting point for the agent training process, control actions determined from a 48 hours scenario of a centralized MPC were used as initialization values. From this point, the training continued taking random actions. The reward was calculated for all actions.

In the RL exploitation phase, the knowledge acquired in the training phase is used to solve the MPC distributed problem through the MAS.

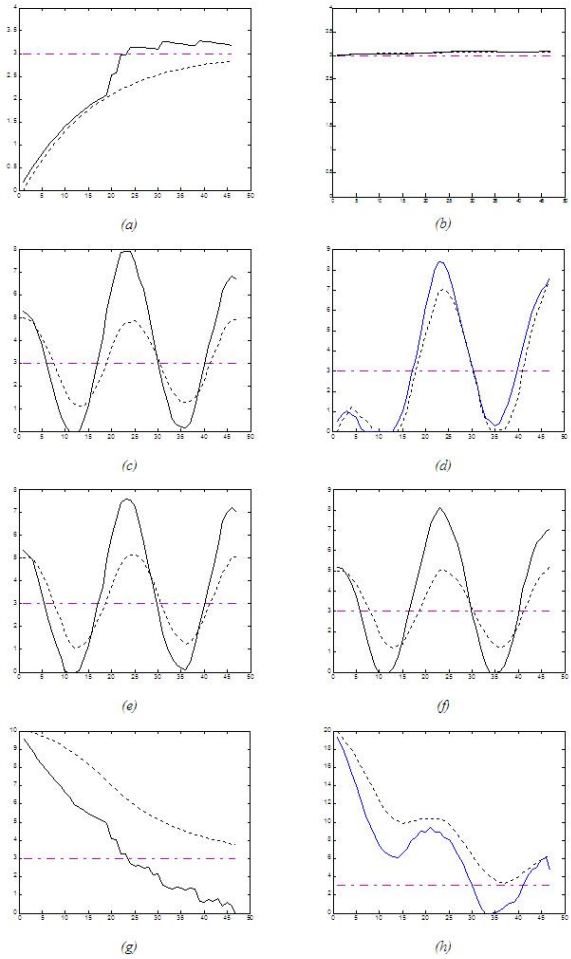

Figure 3: Distributed MAMPC solution (solid -) against centralized MPC solution (dashed --). Reference (-.-). (a) Tank 1; (b) tank 2; (c) tank 3; (d) tank 4; (e) tank 5 ; (f) tank 6; (g) tank 7; (h) tank 8.

The results obtained using the proposed MAMPC Architecture are shown in Figure 3. Each graph presents a 48 hour scenario, showing the trajectory of the output (water volumes in tanks). The results are contrasted with the centralized MPC solution (dashed line) for each tank. The following table presents the optimal objective function value obtained using the proposed distributed MPC solution against the centralized.

$J_{centralized}$	13.3712
$J_{distributed}$	14.7201

The objective function for the distributed solution is larger than the centralized one. However, the graphs show

that, in some cases (tanks 1, 2 and 8, Figure 3a, 3b and 3h, respectively), the error of the distributed MAMPC Architecture solution is lower almost during all the scenario. It is important to note that the volume of tanks 1, and 8 depends directly on the value of the negotiated variables (u_{10} and u_{11}).

VIII. Conclusions

The results obtained suggest that the use of MAMPC architecture based on RL negotiation can converge to the centralized MPC solution with an acceptable degree of approximation but taking advantage from the MAS properties and the tools that the Agent Oriented Paradigm (AOP) provides for development and implementation. Even more, the application of learning techniques can provide the Negotiator Agent the ability of prediction. Training of the negotiator can be made directly from a centralized MPC or from human operator driven control. In order to achieve optimization, no model is needed by the negotiator. Data from centralized MPC is advisable but non essential. The type and quality of the training is a very important issue in order to obtain an efficient optimization. Also the compromise between exploration and exploitation can be implemented on-line to enable the system not just adaptation to the problem but adaptation to changes in time. In this paper, this capability is not addressed in training but in exploring during the optimization. Communication protocols and coordination methods for MAS have to be studied and tested in a more complex case of study in which many agents interact.

IX. Further Research

The MAMPC architecture presented in this work is currently being tested on the Barcelona water transport network in the context of the European Project Decentralized and Wireless Control of Large Scale Systems, WIDE. The Barcelona water network is comprised of 200 sectors with approximately 400 control points. At present, the Barcelona information system receives, in real time, data from 200 control points, mainly through flow meters and a few pressure sensors. This network has been used as a LSS case of study to test several LSS control approaches, see [12] and [18], [19]. As starting point for the application of the MAMPC Architecture, recent work on centralized [20] and decentralized MPC [21] applied to the Barcelona network is being used, as well as, the partitioning algorithm developed by [17].

References

[1] Qin, S. J., & Badwell, T. A. (2003). A survey of industrial Model Predictive Control Technology. Control Engineering Practice, 11, pp. 733–764.

[2] Bemporad, A. and Morari, M. "Robust model predictive control: A survey," in Robustness in Identification and Control (Lecture Notes in Control and Information Sciences), vol. 245. New York: Springer-Verlag, 1999, pp. 207-226.

[3] Siljack, D.D. (1991). Decentralized Control of Complex Systems, Academic Press, New York.

[4] Scattolini, R. (2009). Architectures for distributed and hiearical Model Predictive Control- A Review. Journal of Process Control , 723-731.

[5] Camponogara, E., Jia, D., Krogh, B. H., & Talukdar, S. (2002, Feb). Distributed Model Predictive Control. IEEE Control Systems Megazine , 44-52

[6] Negenborn, R. R. (2008). Multi-Agent Model Predictive Control with applications to power networks. Engineering Applications of Artificial Intelligence , 21, 353-366.

[7] Venkat, A. N., Rawlings, J. B., & Wrigth, S. J. (2005). Stability and Optimality of distributed Model Predictive Control. IEEE Conference on Decision and Control / IEE European.

[8] Sutton, & Barto. (1998). Reinforcement Learning, An introduction. London, England: MIT Press Cambridge Massachussetts.

[9] Agostini, A., & Celaya, E. Feasible control of complex systems using automatic learning. in Proc. ICINCO (Barcelona) 2005.

[10] Tesauro, G. (2003). Extending Q- Learning to General Adaptive Multi-gent System. In Advances in Neural Information Processing Systems . MIT Press.

[11] Ernst, D., Glavic, M., Capitanescu, F., & Wehenkel, L. (2006). Model Predictive Control and Reinforcement Learning as a two complementary frameworks.vii Proceedings of the 13th IFAC Workshop on Control Applications of Optimisation, (p. 6).

[12] Brdys, M. A., & Ulanicki, B. (1994). Operational control of water systems, Structures, Algorithms and Applications. Great Britain: Prentice Hall International.

[13] El Fawal, H., Georges, D., & Bornard, G. (1998). Optimal control of complex irrigation systems via descomposition-coordination and the use of augmented lagrangian. In IEEE (Ed.), in Proc. IEEE Int. conference Systems, man and cybernetics, 4, pp. 3874-3879. San Diego. CA.

[14] Gómez, M., Rodellar, J., Vea, F., Mantecon, J., & Cardona, J. (1998). Decentralized adaptive control for water distribution. Proceedings of the 1998 IEEE International on systems, man and cybernetics, (pp. 1411-1416). San diego Califoirnia. USA.

[15] Rawlings, J. B., & Stewart, B. (2008). Coordinating multiple optimization-Based controllers: New opportunities and challenges. Journal of process control (18), 839-845.

[16] Javalera, V., Morcego, B., Puig, V., (2010). Distributed MPC for Large Scale Systems using Agent-Based Reinforcement Learning. Proceedings of the IFAC 12th LSS Symposium Large Scale Systems: Theory and Applications.

[17] Barcelli, D. (2008). Optimal decomposition of Barcelona´s water distribution network system for applying dsitribuited Model Predictive Control. Master thesis . Universitat Politècnica de Cataluña-IRI-Universitá degli Study di Siena.

[18] Cembrano, G., Quevedo, J., Salamero, M., Puig, V., Figueras, J., & Martí, J. (2004). Optimal control of urban drainage systems. A case of study. Control Engineering Practice (12), 1-9.

[19] Cembrano, G., Wells, G., Quevedo, J., Pérez, R., & Argelaguet, R. (2000). Optimal Control of a water distribution network in a supervisory control system. Control of Engineering Practice (8), 1177-1188.

[20] Caini, E., Puig, V., & Cembrano, G. (2009). Development of a simulation environmet for water drinking networks: Application to the validation of a centralized MPC controller for the Barcelona Case of study. Barcelona, Spain: IRI-CSIC-UPC.

[21] Fambrini, V., & Ocampo Martinez, C. (2009). Modelling a decentralized Model Predictive Control of drinking water network. Barcelona, Spain: IRI-CSIC-UPC.