Distributed MPC for Large Scale Systems using Agent-based Reinforcement Learning

Valeria Javalera*, Bernardo Morcego, Vicenç Puig***

** Institut de Rob¸tica i Inform¨tica Industrial, CSIC-UPC C/. Llorens i Artigas, 4-6, 08028 Barcelona, Spain (e-mail: vjavalera@iri.upc.edu)*
***Advanced Control Systems Group (SAC), Rambla Sant Nebridi, 10, 08222 Terrassa, Spain*

Abstract: In the present work, distributed control and artificial intelligence are combined in a control architecture for Large Scale Systems (LSS). The aim of this architecture is to provide a general structure and methodology to perform optimal control in networked distributed environments where multiple dependencies between sub-systems are found. Often these dependencies or connections represent control variables so the distributed control has to be consistent for both subsystems and the optimal value of these variables has to accomplish a common goal. The aim 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 to provide speed, scalability, and computational effort reduction. This approach is based on negotiation, cooperation and learning. Results of the application of this architecture 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.

Keywords: distributed control, distributed architectures, MPC, learning, Multi-agent systems

1. INTRODUCTION

Distributed and decentralized MPC (Model Predictive Control) schemes have been proposed over the last years in order to optimize complex LSS (Large Scale Systems). In opposite to decentralized systems, where the resulting subsystems are independent from each other, in distributed systems the resulting subsystems can have physical dependencies between them and therefore communication among them. One of the main problems of distributed control of LSS is to decide how those dependence relations between subsystems are preserved. Those 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 has to accomplish a common goal.

In order to do this, many negotiation techniques have been proposed (see for example, Camponogara, et al., (2002), Negenborn (2008), Venkat, et al., (2005), El Fawal, et al., (1998), Gómez, et al., (1998) and Rawlings & Stewart (2008)). Calculation time, problems handling multiple restrictions and multiple objectives and the impossibility to ensure convergence are the main problems of these approaches. Although there have been successful results there is still a need of a methodology that can be used for all kind of continuous LSS.

The authors believe that this open problem in control theory can be solved by the combination of adequate control and computer science techniques, more precisely, the combination of Model Predictive control (MPC), Multi-Agent Systems (MAS), and Reinforcement Learning (RL).

The problems treated by the Distributed Artificial Intelligence (DAI) and the Distributed Control (DC) communities are clearly similar. For this reason the authors propose to apply MAS techniques and technology to DC problems such as communication, coordination, need of adaptation (learning), autonomy and intelligence.

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. All this ideas are the basis of the proposed architecture. A methodology for the application of the proposed architecture is also provided.

2. 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. In particular, it is assumed 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 Siljack (1991). The set of partitions should be complete. This means that all system state and control variables should be included at least in one of the partitions.

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

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

where 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.

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.

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.

Each subsystem i is controlled by an MPC controller using:

- the model of the dynamics of subsystem i given by equation (1);
- 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.

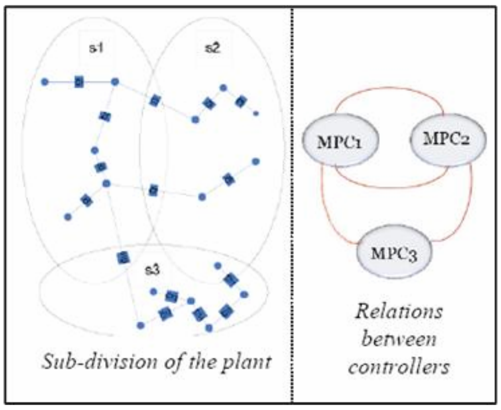

Figure 1: The problem of distributed control

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 fashion, i.e. with three local control agents that must preserve consistency between the shared variables.

3. REINFORCEMENT LEARNING AND MULTI AGENT SYSTEMS

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

Reinforcement learning is based on past experience, which is used to reduce the need of iterative methods, which facilitates that the system behaves almost like a reactive system with a very short time of response. RL 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 chances of convergence are high.

Another key feature of reinforcement learning is that it explicitly considers the whole problem of a goal-directed agent interacting with an uncertain environment. This is in contrast with many approaches that consider subproblems without addressing how they might fit into a larger picture. Sutton & Barto, (1998).

The use of RL in the negotiation process allows to: 1) make the process of negotiation adaptive; 2) learn from its own experience; 3) consider explicitly the whole problem of two goal-oriented agents; 3) deal with a dynamical and uncertain environment; 4) optimize with or without a model; 5) connect the process of negotiation with the process of MPC control.

The term agent is defined inconsistently in the three areas that this work combines (MPC, RL and DIA): In MPC, distributed and decentralized systems are usually called Multi-Agent Systems and their local controllers are called agents; In RL, the controller or the software entity that performs a RL algorithm is also called agent; In Distributed Artificial Intelligence (DAI), large and complex systems are solved in a distributed way through intelligent interacting entities named Agents.

As one can see, the terms behind DAI and Distributed MPC deal with the same or very similar concepts. Nevertheless, DAI is a more general area of research and years of work have developed the technology and techniques that led to a new programming paradigm: the Agent Oriented Paradigm (AOP). This paradigm provides tools (i.e. programming languages, methodologies, standards, communication platforms, etc.) that make the implementation of Multi Agent system feasible.

Many DAI researchers have defined the term Agent. This term is still a controversial issue. In Stan & Graesser (1996), the main agent definitions are presented and explained and their taxonomy is also provided. In the present work, the following definition of an agent is given, for unifying proposes:

"An agent is the basic entity of software that the AOP uses to describe an element that has some level of autonomy within a dynamic and complex system. Besides encapsulating its

characteristics and functionality, it implements processes of reaction and/or deliberation, as well as communication. It is represented, from its initial design, by means of a particular, proposed or experimental method of the AOP. The functionality of the Agent is given by its behaviors and its characteristics are represented in its internal state."

The use of the AOP in the proposed distributed control architecture allows to: 1) enjoy all the benefits of distributed systems, like speed-up of the system activity, since it allows parallel computation, scalability and flexibility, simplicity of design and maintenance of the system, robustness and reliability thanks to the possibility of implementing fault tolerance; 2) perform an appropriate coordination and synchronization of the agents; 3) provide a management and communication platform for the MAS; this allows one to allocate MPC Agents in different computers of a network with no added effort; 4) use appropriate development tools and standards; 5) use system analysis and design methods and tools in order to make an appropriate formalization and documentation of the system

3. MAMPC ARCHITECTURE

3.1 MAMPC Architecture

The proposed MAMPC distributed control architecture is defined as:

$$\gamma = \{A, N, P, W, V, U, b\} \qquad (2)$$

where:

- V is the set formed by all sets of *shared variables* and U is the set formed by all sets of *internal variables*.
- A is the set of MPC Agents. 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.
- N is the set of Negotiator Agents. The Negotiator Agent is the entity that is in charge of determining the value of one or more shared variables between two MPC Agents. In this negotiation, each MPC Agent is arranged to cooperate so that the negotiator agent solves the optimization of a common goal by means of an algorithm based on *Reinforcement Learning*. A negotiator Agent exists for every pair of MPC Agents that have one or more shared variables in common.
- W is the set of nodes. A node is the physical device (commonly a computer) in which the agents are located. There is a node for each MPC Agent. Nodes are communicated via some communication infrastructure (LAN, WAN or Internet).
- b is the agent platform, it works as a virtual machine providing the agents with a homogenous medium to communicate and providing the user a way to manage agents. This platform has to be installed and running in all nodes.

A methodology has been developed to assign all the elements of the MAMPC architecture given a system. This methodology is illustrated in the application section with an example.

3.2 Cooperation of MPC-Agents

The cooperative interaction of MPC agents is a basic issue in the proposed approach. Cooperation is carried out through three main actions:
1) Providing data (system states, errors, etc.) to the Negotiator Agent; 2) accepting the shared variable(s) provided by the Negotiator Agent; 3) solving the MPC control problem of its partition, adjusting the value(s) of its shared control variable(s) in order to coordinate the solution of the negotiation.

The Negotiator Agent determines the optimal value of the values in set V. This set contains the shared variables of two, and just two MPC Agents. The Negotiator Agent optimizes them through a Negotiation algorithm based on Reinforcement Learning (RL). Each shared variable is an optimization problem. This problem is solved as a whole looking for the optimal value of the relation. The method is based on the reinforcements given at each step and on the experience obtained. This experience is stored in a knowledge base, one for each negotiation variable.

In the distributed model of the system, shared variables appear in the local models of each MPC Agent involved in the relation, therefore they end up duplicated.

The Negotiator Agent restores the broken connections when the system was partitioned, unifiying this dupplicate variables in a single one, just as in the original model. Therefore, for the Negotiator Agent, this two control variables are taken as one.

The philosophy of the proposed negotiation algorithm is to consider the shared variables as belonging to a single problem with a single goal, instead of two different problems with conflicting goals. The Negotiator Agent solves the optimization problem for that variable and communicates the result to the MPC Agents at each sampling time. Then, MPC Agents set those values as a hard constraint in its respective internal control variables and recalculate the control law.

The Negotiator Agent optimization algorithm is based on the Q-learning algorithm that takes into account previous Agent experience and the reinforcements received at every action taken in the past on similar situations. Next, these elements are described in further detail.

3.2.1 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:

$$\varepsilon_i = g_i - y_i \qquad (3)$$

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:

$$s = \frac{|\varepsilon_{i1}| + |\varepsilon_{i2}|}{2} \qquad (4)$$

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 posible 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:

$$r = \rho - s \qquad (5)$$

where ρ is a value greater or equal than s.

3.2.2 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:

$$Q(s,a) = r + (\alpha \times Q(s,a)) \qquad (6)$$

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 actions are chosen according to

$$a' = \max_a(Q(s,a)) \qquad (7)$$

in order to select the value of the action (negotiated variable) with maximum reward for the next time instant. More details of this algorithm and about negotiation in the MAMPC architecture can be found in Javalera, Morcego & Puig, (2010).

4. APPLICATION EXAMPLE

4.1 Description

A small drinking water network is used to exemplify the proposed MAMPC architecture and its performance. The example was proposed in Barcelli (2008) 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.

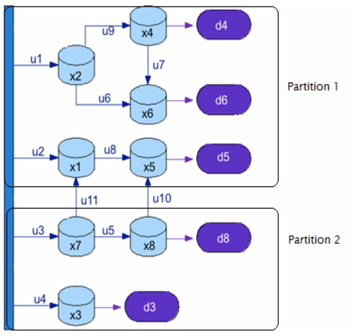

Figure 3: Case study and its partitioning

4.2 Analysis

In the analysis phase, the MAMPC Architecture is defined. This phase comprises the following tasks:

1) Definition of the optimization goals: the control goal of the application presented in Figure 2 is to keep a volume in tanks around 3m^3 Thus, the control objective if a centralized MPC was used can be formulated as follows:

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

2) Partitioning of the network: the system (plant to be controlled) presented in Figure 2 composed of the following states and control inputs

$$\begin{aligned} Plant = \{&x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, u_1, u_2, \\ &u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_{10}, u_{11}\} \end{aligned} \qquad (8)$$

is decomposed in two partitions using the epsilon-decomposition proposed by Siljack (1991):

$$p_1 = \{x_1, x_2, x_4, x_5, x_6\} \qquad (9)$$

$$p_2 = \{x_3, x_7, x_8\} \qquad (10)$$

$$V = \{u_{10}, u_{11}\} \qquad (11)$$

$$U_1 = \{u_1, u_2, u_6, u_7, u_8, u_9\} \qquad (12)$$

$$U_2 = \{u_3, u_4, u_5\} \qquad (13)$$

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:

$$Plant = P \cup U \cup V \qquad (14)$$

that can be easily verified in this example,

$$Plant = p_1 \cup p_2 \cup U_1 \cup U_2 \cup V \qquad (15)$$

Thus, the partition is a complete set of partitions. The control objective of each partition is the following:

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

3) Definition of the Architecture. In this step, the MAMPC Architecture is defined. Considering the definition of the architecture in (2), the remaining elements are defined as follows:

$$A = \{a_1, a_2\} \qquad (16)$$

$$N = \{n_1\} \qquad (17)$$

$$W = \{w_1, w_2\} \qquad (18)$$

4) Inclusion of restrictions and considerations: the maximum volume in tanks is 20 m^3, the control value of the messured 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 periodic behaviour that repeats every the day.

4.3 Design

In the design process, the subproblems of every MPC Agent and Negotiator Agent are formulated. This formulation is based on the information collected in the analysis phase.

The core of each MPC agent is an MPC controller. This controller solves the multivariable problem of one partition of the plant based on the models of each partition according to (9)-(12), all the MPC parameters and requierements have to be defined for both agents, such as:
1) The plant; 2) The measured, non-measured and manipulated variables; 3) Limits and constraints; 4) The negotiation variables, which are set as restrictions; 5) References (goals); 6) Prediction horizon; 7) Control horizon; 8) Initial state; 9) Perturbations models.

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. Figure 3 shows a sequence diagram of the communication protocol designed for this application.

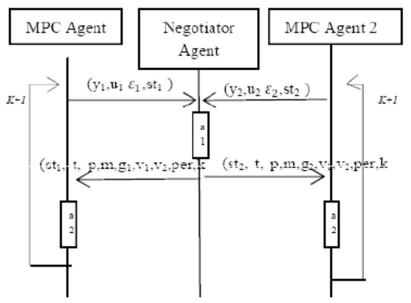
MPC Agent
Negotiator Agent
MPC Agent 2
$(y_1, u_1\ \varepsilon_1, st_1)$
$(y_2, u_2\ \varepsilon_2, st_2)$
K+1
K+1
a 1
$(st_1, t, p, m, g_1, v_1, v_2, per, k$
$(st_2, t, p, m, g_2, v_1, v_2, per, k$
a 2
a 2

Figure 3: Communication protocol

The diagram shows how MPC Agents start the comunication by interchanging the resulting output of the control applied ($y_i(k)$), the vector of controls applied ($u_i(k)$), 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 result of the optimization and the parameter needed by de MPC Agents to solve its multivariable problem taking as restrictions the values given by the negotiator. After that, the procces starts again.

4.4 Training

An off-line training using the RL 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.

4.5 Exploitation

In the RL exploitation phase, the knowledge adquired in the training phase is used to solve the MPC distributed problem through the MA system.

The results obtained using the proposed MAMPC Architecture are shown in Figure 4. 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

Thus distributed solution is not as good as the centralized one. However, the graphs show that, in some cases (tanks 1, 2 and 8, Figure 4a, 4b and 4h, respectively), the distributed MAMPC Architecture solution is better. 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}).

5. 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.

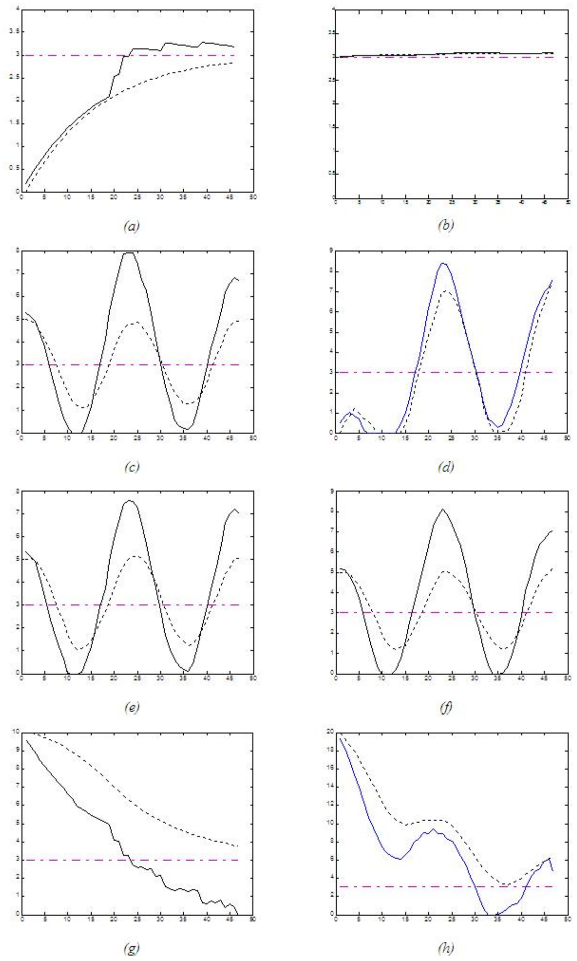
Figure 4: Distributed MAMPC solution (solid -) against centralized MPC solution (dashed --). Reference (-.-). (a) Tank 1; (b) tank 2; (c) tank 3; (d) tank 4; (e) tank 5 ; (f) tank 6; (g) tank 7; (h) tank 8.

6. 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 Brdys & Ulanicki (1994) and Cembrano, et al., (2000). As starting point for the application of the MAMPC Architecture, recent work on centralized Caini, et al., (2009) and decentralized MPC Fambrini & Ocampo (2009) applied to the Barcelona network is being used, as well as, the partitioning algorithm developed by Barcelli (2008).

ACKNOWLEDGMENT

The authors wish to thank the support received by WIDE-224168-FP7-ICT-2007-2 project and WATMAN ref. DPI2009-13744 of Spanish Ministry of Education.of Spanish Ministry of Education. First author is supported by CONACYT and partially supported by ITESCA

REFERENCES

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

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

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.

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

Cembrano, G., et al., (2000). Optimal Control of a water distribution network in a supervisory control system. *Control of Engineering Practice* (8), 1177-1188.

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.

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

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.

Javalera, V., Morcego, B., & Puig, V. (2010). Negotiation and Learning in Distributed MPC of Large Scale Systems. Proceedings of the 2010 IFAC American Control Conference. Baltimore, USA.

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

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

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

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

Stan, F., & Graesser, A. (1996). Is it an agent or just a program?: A taxonomy of autonomous agents*. Proc. of the third International workshop on Agent theories, architectures and lenguages* . Springer-Verlag.

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.