

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Programa de Doctorat:

AUTOMÀTICA, ROBÒTICA I VISIÒ

Proposta de Tesis Doctoral

NEGOTIATION IN DISTRIBUTED LARGE SCALE SYSTEMS:
A MULTI-AGENT MPC ARCHITECTURE

Valeria Javalera Rincón

Directors: Bernardo Morcego i Vicenç Puig

Juny 2009

NEGOTIATION IN DISTRIBUTED LARGE SCALE SYSTEMS: A MULTI-AGENT MPC ARCHITECTURE

ABSTRACT

In the present work, a distributed control architecture for large scale systems is proposed. This architecture is multi-agent based. The model plant is divided in several partitions and there is an MPC Agent in charge of each partition. MPC Agents interact over a platform that allows them to be located physically separated. One of the main new concepts of this architecture is the Negotiator Agent. Negotiator Agents interact with MPC Agents when they have common control variables. These shared variables represent physical connections between partitions that should be preserved in order to respect the topology of the network.

The case of study in which the proposal architecture will be applied and tested is the Barcelona water transport network. First results of using the proposed methodology are based on a small academical water network example.

List of figures

Figure 1: Diagram of the Barcelona water network.....	3
Figure 2: Example of conventional MPC.....	4
Figure 3: MPC Taxonomy.....	4
Figure 4: Distributed MPC Taxonomy.....	5
Figure 5: Distributed control system (Giovanini & Balderud).....	9
Figure 6: MAMPC system with three partitions	12
Figure 7: Structure of the plant partitioning.....	12
Figure 8: Agent interaction structure of MAMPC Architecture.	13
Figure 9 Assignment of agents, partitions and nodes	13
Figure 10: Internal architecture of the MPC Agent.....	13
Figure 11: Internal architecture of the Negotiation Agent	14
Figure 12: Distribution of the plant for the application problem	17
Figure 13: Demands of MPC Agent 1.....	19
Figure 14: Demands of MPC Agent 2.....	20
Figure 15: Q-learning algorithm	21
Figure 16: Communication protocol of the case of study.....	22
Figure 17: Outputs of MPC Agent 1	23
Figure 18: Centralized outputs corresponding to MPC Agent 1 outputs.....	23
Figure 19: Outputs of MPC Agent 2.....	23
Figure 20: Centralized outputs corresponding to MPC Agent 2 outputs.....	23
Figure 21: comparison between MAMPC and centralized MPC solutions of tank 1	23
Figure 22: comparison between MAMPC and centralized MPC solutions of tank 5.....	23
Figure 23: comparison between MAMPC and centralized MPC solutions of tank 7.....	23
Figure 24: comparison between MAMPC and centralized MPC solutions of tank 8.....	23

Acronyms

AGBAR	AGuas de BARcelona
AOP	Agent Oriented Paradigm
LSS	Large Scale Systems
MAS	Multi-Agent System
MPC	Model Predictive Control
PID	Proportional Integral Derivative

INDEX

Abstract.....	iii
List of figures.....	iv
1. Introduction	1
1.1. Case of study	1
2. State of art	3
2.1. Model Predictive control philosophy	3
2.2. <i>MPC</i> Taxonomy	4
2.3. Distributed MPC	5
2.1. Distributed MPC taxonomy	5
2.2. Cooperative vs independent algorithms.	5
2.3. Negotiation in cooperatives environments using MPC	6
2.4. Reinforcement Learning	6
2.5. Multi Agent Systems	7
2.5.1. Properties and Characteristics of the Agents	8

2.5.2. Potential advantages of Multi-Agent Systems.....	8
3. Definition of the problem.....	8
4. Objectives.....	9
5. Expected contributions	9
6. Proposed Architecture: Multi-Agent MPC.....	10
6.1. Partitioning of the plant.	10
6.2. Partitioning of the Optimization Problem.....	11
6.3. Elements of the MAMPC Architecture.....	11
6.4. Structure of the MAMPC Architecture.....	12
6.4.1. Internal description of the MPC Agent.....	13
6.4.2. Internal description of the Negotiation Agent	13
6.5. Dynamics of the MAMPC Architecture	14
6.6. Benefits of the proposed Architecture.....	14
7. Work plan.....	15
8. Preliminary results.....	17
8.1. Application of the MA-MPC Architecture	17
8.1.1. Analysis.....	17
8.1.2. Design.....	18
8.1.3. Training (exploration)	21
8.1.4. Exploitation	22
8.1.5. Implementation.....	22
8.1.6. Results	22
8.1.7. Conclusions	24
9. Material Resources.....	24
10. Bibliography	vi
Annex 1: Reinforcement Learning Elements and problem definition (Sutton & Barto, 1998).....	viii

1. 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 rivers networks for agriculture, etc. Other examples of applications of LSS but in discrete domain are: Traffic control, railways control, manufacturing industry, etc.

The quality of management and control of this kind of systems is crucial. Most of them are directly related with the quality of life of people in cities and have impact on the environment preservation. As for example: sewer networks, metropolitan water networks, canal and rivers networks for agriculture. If inefficient control strategies are used in these systems results might derive on: spills of contaminated water to the field, the sea or within the cities, floods, restrictions of water in the cities, bad quality of water, unsatisfied hydric needs in agriculture etc. In other types of LSS risks and consequences can be: pollution, traffic unsafety, blackouts, etc.

Model Predictive Control (MPC), also known as receding horizon control, is a control technique widely use in industry [see (Qin & Badwell, 2000) (Qin & Badwell, 2003) (Camacho & Bordons, Model Predictive Control in the process Industry, 1995)]. It has been also applied to LSS. Examples of applications in sewer networks can be found in : (Cembrano, Figueras, Quevedo, Puig, Salamero, & Martí, 2002) (Cembrano, Quevedo, Salamero, Puig, Figueras, & Martí, 2002) ; applications in water networks are: (Cembrano, Wells, Quevedo, Pérez, & Argelaguet, 2000) (Cembrano, Quevedo, Puig, Pérez, Figueras, & All, 2005) .

But due to the increase of automatization of LSS, complexity is also increasing. Such complexity is due to the need of many sensors and actuators in a dynamical non-linear environment. Additionally, LSS are composed of many interacting subsystems. Optimization of these systems requires restrictions to assure safety and guarantee operational limits satisfaction, cost reduction, etc. Finally, the increasing size of the systems is another important issue. All these problems are difficult to be overcome using a centralized control structure due to robustness and reliability problems and due to communications limitations. For all these reasons, many distributed MPC control have been developed and applied over the last forty years (Scattolini, 2009). In (Venkat,

Rawlings, & Wrieth, 2005) the authors consider that centralized MPC is widely used but unsuitable for LSS and talk about the need of a distributed control structure.

One of the main problems of distributed control of LSS is how relations between partitions are preserved. These relations could be pipes that connect two different control zones of a decentralized water transport network for example, or any 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. In the present work, a Multi-Agent MPC architecture is proposed to deal with the negotiation of these variables.

Although the proposed architecture is intended to be general enough to be applied in any continuous LSS (for discrete domain applications should be adapted), for validation purposes, a case study will be used based on the case of the Barcelona water transport network. In the next section this case of study will be presented.

The organization of this document is as follows:

Chapter 2 presents the state of art. In this chapter the background and the current state of the main topics related with this work will be explained. In particular MPC, distributed MPC, negotiation, Multi-Agent Systems, Reinforcement Learning and works that relate some of this areas will be discussed.

In Chapter 3, the description and formalization of the problem to be addressed are introduced. In Chapter 4 and 5 the objectives and contributions of the thesis, respectively, are presented. In Chapter 6, the proposed architecture is described. In Chapter 7, the work plan will show the work done and the planning of the next tasks. In Chapter 8, first results of using the methodology presented in this proposal are based on a small academic water network example.

1.1. Case of study

The objective of using a real case of study in this work is to validate results and technical viability of the proposed architecture. Although the solution obtained by the proposed architecture and methodology has to be efficient for the considered case, it has to be general enough for being applied in any kind of continuous LSS.

The case of study is the Barcelona water transport network. The network is managed by the company

Aguas de Barcelona (AGBAR), a partner of the European Project Decentralized and Wireless Control of Large Scale Systems, WIDE, of which this work is part of.

AGBAR not only supplies water to Barcelona city but also to the metropolitan area (see data in the table below). (WIDE - 224168 - FP7-ICT-2007-2, Decentralized Wireless Control of Large-Scale Systems)

Territorial extension	425 Km²
Drinkable water net	4.470 Km
Drinkable water production	237.7 hm³
Population	2.828.235
<i>Metropolitan area of Barcelona – Water net (2006)</i>	

The sources of water are the rivers Ter and Llobregat. Since 1976, the network has a centralized tele-control system, organized in a two-level architecture. At the upper level a supervisory control system installed in the control centre of AGBAR is in charge to optimally control the whole network by taking into account operational restrictions and consumer demands. This upper level provides the set-points for the lower-level control system. This optimizes the pressure profile to minimize losses by leakage and provide sufficient pressure, e.g. for high rise buildings. The system responds to changes in network topology (ruptures), typical daily/weekly profiles, as well as major changes in demand, etc. (WIDE - 224168 - FP7-ICT-2007-2, Decentralized Wireless Control of Large-Scale Systems)

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.

Sensors measurements are sent to the operational data base of the telecontrol information system via telephone XTC network or GSM radio using the ModBus communication protocol. (WIDE - 224168 - FP7-ICT-2007-2, Decentralized Wireless Control of Large-Scale Systems)

This water network, as any other, is composed by nodes, valves, pumps, tubes, and sources.

Figure 1 depicts the diagram of the Barcelona water transport network. Each tank supplies water a demand sector. The inputs/outputs of a tank are controlled using valves/pumps depending on the elevation. The state of the network are the tank levels/volumes of the tanks, which need to be optimally manage to satisfy the demands at the minimum cost without this disrupting the water service. From the control point of view, the control actions are the flow set points of valves and pumps. It is assumed that there is a local PID controller that controls the water pressure. Demands are considered as measured perturbations.

In the past, different versions of this network have been used as a case of study. First one is in the book (Brdys & Ulanicki, 1994) and latter in (Cembrano, Figueras, Quevedo, Puig, Salamero, & Martí, 2002) (Cembrano, Quevedo, Salamero, Puig, Figueras, & Martí, 2002) (Cembrano, Wells, Quevedo, Pérez, & Argelaguet, 2000).

Recent work on centralized MPC applied to the same network but with updated topology is (Caini, Puig Cayuela, & Cembrano, 2009). A decentralized approach of the same updated network is (Fambrini & Ocampo Martinez, 2009) and an optimal decomposition approach for the same net but in a distributed MPC fashion is studied in (Barcelli, 2008).

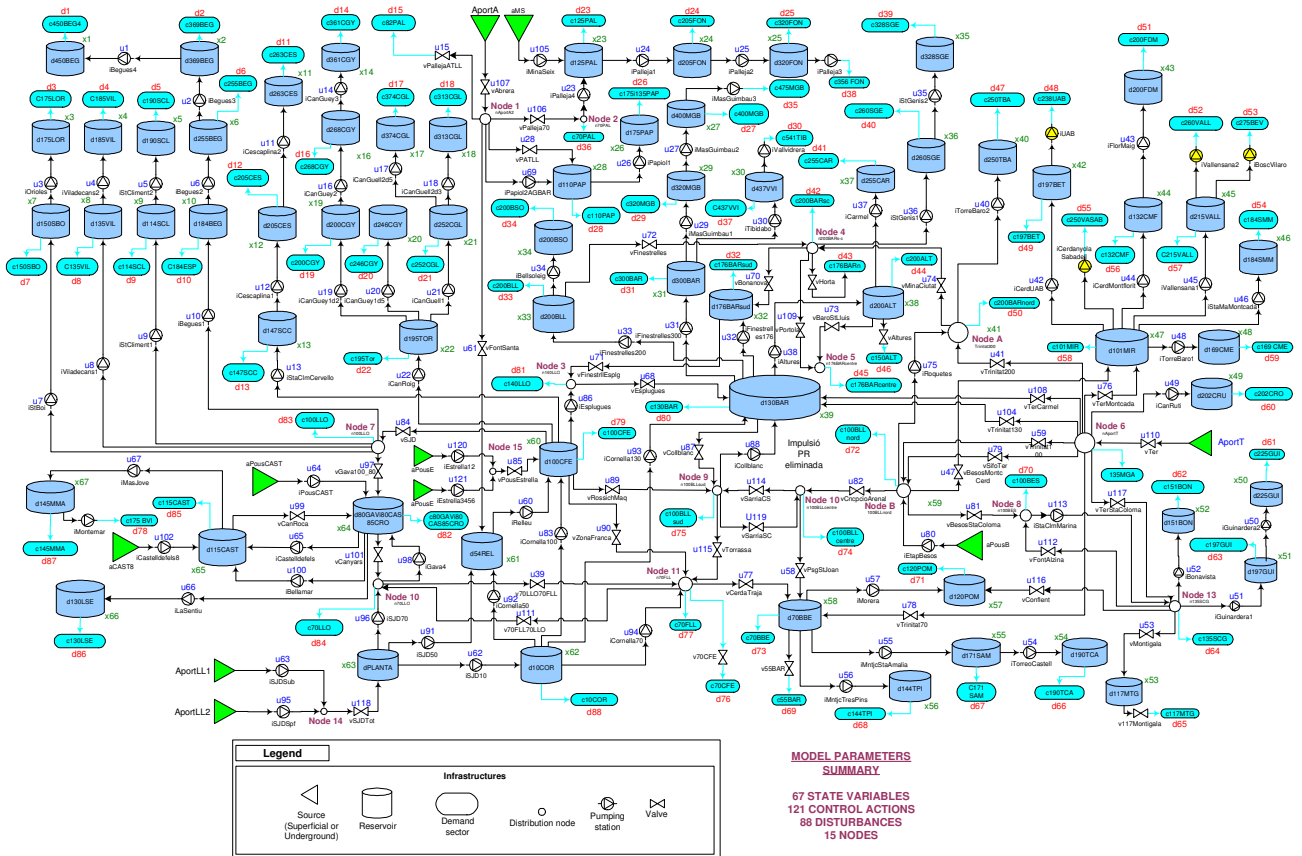


Figure 1: Diagram of the Barcelona water network.

2. STATE OF ART

2.1. Model Predictive control philosophy

As it was mentioned before, Model Predictive Control is a recognized powerful approach with proven capability to handle a large number of industrial control problems.

The philosophy of MPC is well resumed in (Scattolini, 2009) when it says that the main characteristic of MPC is to transform the control problem into an optimization one, so that at any sampling time instant, a sequence of futures control values is computed by solving a finite horizon optimal control problem. Then, only the first element of the computed control sequence is effectively used and the overall procedure is repeated at the next sampling time according to the so-called receding horizon principle. For a more detailed explanation about MPC see the text book (Camacho & Bordons, Model Predictive Control, 2004).

The main characteristics of centralized MPC are (Negenborn, De Shutter, & Hellendoorn, Multi-Agent model predictive control: A survey, 2004):

- The centralized system model is given by a (possibly time-varying) dynamic system of

difference or differential equations and constraints on inputs, states, and outputs.

- The goal of the control problem is to minimize a cost function. The control problem is stated as a multiple-objective optimization problem that is transformed to a single objective one using a weighted approach.
- The problem is solved by a single centralized agent, the information set of which consists of measurements of the physical system, and the control action set of which consists of all possible control actions. The agent solves the problem with a three-step procedure (see Figure 2):

1. It reformulates the problem of controlling the time-varying dynamic system using a time-invariant approximation of the system, with a control and a prediction horizon to make tractable the solution computation and a rolling horizon for robustness.

2. It solves the reformulated control problems, often using general, numerical solutions techniques, while taking into account constraints on control actions and states.

3. It combines the solutions to the approximations to obtain a solution to the overall problem. This typically involves implementing the control actions found from the beginning of

the time horizon of the current approximation, until the beginning of the next approximation.

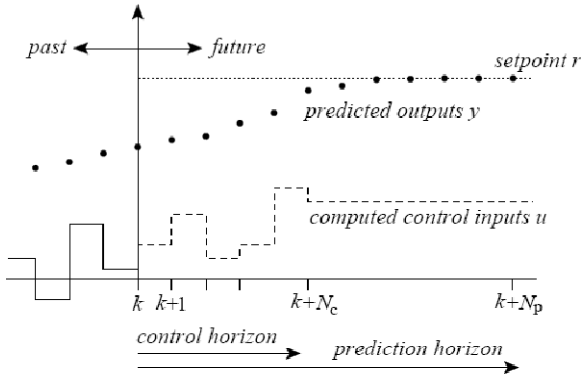


Figure 2: Example of conventional MPC.

In Figure 2, it can be noticed that the control problem is to find actions $u_k \dots u_{k+N_c}$, such that after N_p steps, the system behavior y approaches to the desired behavior y^* . In this example, y indeed reaches the desired set point y^* . (Negenborn, De Shutter, & Hellendoorn, Multi-Agent model predictive control: A survey, 2004)

In Negenborn, (De Shutter, & Hellendoorn 2004) one can find some advantage and disadvantages of the MPC framework:

Advantages

- The MPC framework handles input, state, and output constraints explicitly in a systematic way. This is due to the control problem formulation is based on the system model which includes the constraints.
- It can operate without intervention for long periods. This is due to the rolling horizon principle, which enables that the agent looks ahead to prevent the system from going in the wrong direction.
- It adapts easily to new contexts because of the rolling horizon use.

Disadvantages

- The approximation of the distributed control problem with static problems can be of large size. In particular, when the prediction horizon becomes large, the number of variables of which the agent has to find the value increases quickly.
- The resources needed for computation and memory may be high, increasing more when the time horizon increases. The amount of resources required also grows with increasing system complexity.

- The feasibility of the solution to distributed control problem is not guaranteed. Solutions to the distributed problem are not guaranteed to be solutions to the original control problem.

2.2. MPC Taxonomy

There are several applications and new approaches of MPC are continuously arising. Although it is a well known and accepted control strategy, it is still an open field of research.

Three main types of structures in which MPC is applied can be found in the literature (See figure 3)

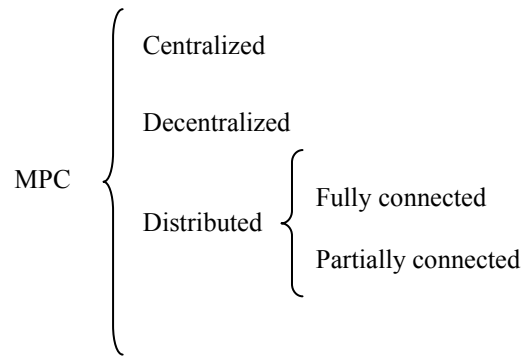


Figure 3: MPC Taxonomy

Centralized MPC is the classical way of implementing MPC strategy. In (Rawlings & Stewart, 2008) the authors state that the move from distributed PID to MPC of small systems was essentially a move towards centralized decision making. This technology gained support because the performance benefits were large.

But, as it was mention before, there are strong reasons that leads MPC to decentralized or distributed implementations.

According to (Scattolini, 2009), decentralized control is based on considering the control input (u) and the controlled output (y) variables are grouped into disjoint sets. These sets are then coupled to produce non-overlapping pairs from which local regulators can be single-input single-output or multivariable (locally centralized) depending on the cardinality of the selected input and output groups.

Many proposals has been suggested to define these sets disjoint where they are not naturally in that way. (Camponogara, Jia, Krogh, & Talukdar, 2002) (El Fawal, Georges, & Bornard, 1998) (Van Breemen & De Vries, 2001) (Barcelli, 2008) (Rawlings & Stewart, 2008). Other line of investigation is to communicate overlapping sets. This is called distributed control.

In distributed control structures, it is assumed that some information is transmitted among the local regulators, so that each one of them has some knowledge on the behavior of the others. In the next section distributed control implemented with MPC approach is discussed.

2.3. Distributed MPC

When the local regulators of a distributed control structure are designed with MPC, the information transmitted typically consists of the future predicted control or state variables computed locally. In this way, any local regulator can predict the interaction effects over the considered prediction horizon. If the information exchange among the local regulators concerns the predicted evolution of the systems states, any local regulator needs only to know the dynamics of the subsystem directly controlled. On the contrary, if the predictive control actions are transmitted, the local regulators must know the model of all subsystems. In any case, it is apparent that the transmission and the synchronization protocols have major impact on the achievable performance. (Scattolini, 2009)

Fully connected algorithms are the ones in which every regulator has bi-directional communication with all the other regulators. If any local regulator has communication just with a subset of the others, then is partially connected

2.4. Distributed MPC taxonomy

In (Scattolini, 2009), a taxonomy of MPC approaches based in the protocol used for exchanging information among local regulators is provided. This taxonomy is summarized in the following diagram:

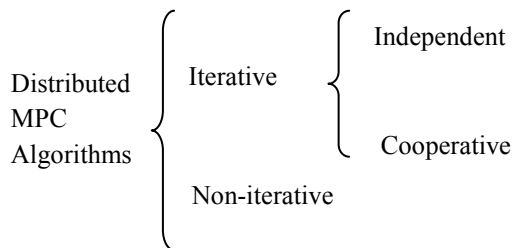


Figure 4: Distributed MPC Taxonomy

In iterative algorithms information is bi-directionally transmitted among local regulator many times within the sampling time. In non-iterative algorithms information is bi-directionally transmitted among the local regulators only once within each sampling time.

In iterative algorithms there is a sub- classification. When each local regulator minimizes a local performance index, it is said to be an independent algorithm, and when they minimize a global cost function it is called cooperative algorithm.

2.5. Cooperative vs independent algorithms.

Independent (non-cooperative) algorithms are widely studied in game theory (much more widely used in comparison with cooperative algorithms) and also applied in MPC distributed control strategies (see an example of application of min-max algorithm in (Jia & Krogh, 2002)). More details about cooperative games can be found in (Fernández García, 2000)

As discussed in (Venkat, Rawlings, & Wriugh, 2005) it is apparent that in iterative and independent algorithms each local regulator tends to move towards a Nash equilibrium, while iterative and cooperating methods seek to achieve the Pareto optimal solution provided by an ideal centralized control structure. However, Nash equilibrium can be even unstable and far from the Pareto optimal solution. So, specific constraints have to be included in the MPC problem formulation to guarantee closed-loop stability. (Scattolini, 2009)

As for the MPC algorithms published in the literature, the state feedback method described in (Camponogara, Jia, Krogh, & Talukdar, 2002) for discrete-time linear systems belongs to the set of independent, noniterative algorithms. A stability constraint is included in the problem formulation, although stability can be verified only a-posteriori with an analysis of the resulting closed-loop dynamics. Nash equilibrium solutions are searched in the independent, iterative and fully connected methods developed in (Du, Xi, & Li, 2001) for discrete-time unconstrained linear systems represented by input–output models. (Scattolini, 2009)

There is a analogous classification in game theory of distributed MPC taxonomy. Distributed algorithms correspond to algorithms where there is exchange of information between players. MPC iterative algorithms correspond to dynamic algorithms, whereas MPC non noniterative algorithms correspond to static algorithms. MPC independent algorithms correspond to non-cooperative algorithms and cooperative algorithms are called cooperative as well.

2.6. Negotiation in cooperative environments using MPC

The seminal Tamura coordination method was discussed in the book (Brdys & Ulanicki, 1994) even before MPC was first introduced. This method is based on using augmented Lagrangian to negotiate values on overlapping sub-networks in distributed large scale systems. Recent works have applied this method (El Fawal, Georges, & Bornard, 1998) (Gómez, Rodellar, Vea, Mantecon, & Cardona, 1998) (Negenborn et al, 2008).

An interesting approach is presented in (Venkat, Rawlings, & Wriqth, 2005), 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. (Scattolini, 2009)

In (Rawlings & Stewart, 2008), an alternative approach to solve the same problem was discussed. The novelty involves maintaining the distributed structure of all the local controllers, but changing the objective functions so that the local agents cooperate.

2.7. Reinforcement Learning.

Learning is the incorporation of knowledge and skills by an agent, leading to an improvement in the agent performance (Busonui, De Shutter, & Babuska, 2005). Learning is used mainly in systems where the environment is large, complex, open and time-varying. That is because designing an agent behavior that takes into consideration all the possible circumstances that the agent may encounter is a very difficult, if not impossible, task. Besides, openness and variation over time, implies that even if such a behavior were designed, it would quickly become obsolete as the environment changes. (Busonui, De Shutter, & Babuska, 2005)

Due to difficulties in dealing with open and time-varying environments, most multiagent learning algorithms are designed for unchanging environments. They typically involve some fixed learning structures that are updated by a set of rules involving some fixed or scheduled parameters. This kind of learning is called “static” learning (Busonui, De Shutter, & Babuska, 2005).

By allowing the learning parameters or structures of the static algorithms to adapt, the learning processes of the agents should be able to regain their ability of handling

open and time-varying environments (Busonui, De Shutter, & Babuska, 2005).

Note that adaptive learning is not a radically different process from learning. It can be viewed as a kind of “meta-learning” – that is, a special case of “learning how to learn” (Busonui, De Shutter, & Babuska, 2005).

In the book (Sutton & Barto, 1998) Reinforcement Learning (RL) is define as: learning what to do, how to map situations to actions, so as to maximize a numerical reward signal. 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. These two characteristics (trial-and-error search and delayed reward) are the two most important distinguishing features of reinforcement learning.

Reinforcement learning is defined not by characterizing learning methods, but by characterizing a learning problem. Any method that is well suited to solving that problem is considered to be a reinforcement learning method. The basic idea is to capture the most important aspects of the real problem facing a learning agent interacting with its environment to achieve a goal. Clearly, such an agent must be able to sense the state of the environment to some extent and must be able to take actions that affect the state. The agent also must have a goal or goals relating to the state of the environment. The formulation is intended to include just these three aspects (sensation, action, and goal) in their simplest possible forms without trivializing any of them (Sutton & Barto, 1998).

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)

A full specification of the reinforcement learning problem in terms of optimal control of Markov decision processes and a deeper explanation about the most important RL topics can be found in (Sutton & Barto, 1998).

Although the applications of RL are typically static, many control applications have been developed for dynamical environments (Agostini & Calaya) (Martinez & De Prada Moraga, 2003) (Tesauro, 2003). Even more, there are some works that relate MPC and RL. In (Ernst, Capitanescu, & Wehenkel, Reinforcement Learning Vs Model Predictive Control: A comparison on a power system problem, 2007) a comparison between both

approaches is made, and in (Ernst, Glavic, Capitanescu, & Wehenkel, 2006) they are seen as complementary frameworks.

An interesting paper about cooperative learning applying RL in control is (Bakhtiari, Araabi, & Nili AhmadAbadi, 2007). In the area of Distributed Artificial Intelligent, papers about learning in cooperative Multi-Agent systems with RL are (Lauer & Riedmiller, 2000) (Claus & Boutilier, 1998) (Kapetanakis & Kudenko, 2002). The last one also talks about coordination. Another application of RL for coordination in Multi-agents systems is (Boutilier, 1999). In all those papers, the term Multi-Agent is referring to agents in Distributed Artificial Intelligence terminology. In the next section a short description of these terms will be made.

2.8. Multi Agent Systems

The term agent has been used indiscriminately until now in this work. In control, 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.

There is a branch of Artificial Intelligence called Distributed Artificial Intelligence (IAD). This branch arises as a result of the natural evolution of the systems that could be found because they are more and more complex, large and often heterogeneous.

The solution of problems of this nature under a traditional scheme, involved the design of large and complex algorithms that consume a very high level of resources for calculation. It was about the 80's that it was thought that small and simple programs that interact with each other could considerably simplify the design and development of these systems reducing the necessary resources.

Many IAD 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 as well as a taxonomy of autonomous agents is provided. Next, some of these definitions are presented.

The Maes Agent: "Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed." (Stan & Graesser, 1996)

The IBM Agent: "Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program with some degree of independence or autonomy, and in doing so, employ some knowledge or representation of the user's goals or desires." (Stan & Graesser, 1996)

The Wooldridge and Jennings Agent: A hardware or (more usually) software-based computer system that enjoys the following properties:

- Autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
- Social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;
- Reactivity: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;
- Pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative." (Stan & Graesser, 1996)

As a result of many years of research in this area, important contributions have been made on theory, methodologies, communications protocols, standards and software tools that lead to the appearance of the Agent Oriented Paradigm (AOP).

The AOP is widely used in software applications and specially in Internet applications (for example e-commerce). Other interesting applications in robotics can be found in literature.

In control applications sometimes the terms of agent in IAD and in control are not consistent although there are some applications in terms of agents in the AOP way. Examples of these applications are mentioned next.

In the paper (Maturana, Staron, & Kenwood, 2005) new tools for developing MAS in distributed control applications are described and a case of study of a chilled-water system of a ship is presented.

Another, and more recent, application in distributed control network of interconnected chemical reactor is presented in (Tatara, Çinar, & Teymour, 2007)

2.8.1.Properties and Characteristics of the Agents

In (Stan & Graesser, 1996) we can find the following table that shows some properties of the agents.

Property	Other Names	Meaning
Reactive	(sensing and acting)	responds in a timely fashion to changes in the environment
Autonomous		exercises control over its own actions
Goal-oriented	pro-active purposeful	does not simply act in response to the environment
Temporally continuous		is a continuously running process
Communicative	socially able	communicates with other agents, perhaps including people
Learning	adaptive	changes its behavior based on its previous experience
Mobile		able to transport itself from one machine to another
Flexible		actions are not scripted
Character		believable "personality" and emotional state.

2.8.2.Potential advantages of Multi-Agent Systems

In (Busonui, De Shutter, & Babuska, 2005) one can find some of MAS principal potential advantages over centralized systems.

- Speed-up of the system activity, due to parallel computation.
- Robustness and reliability, when the capabilities of the agents overlap. The system is tolerant to failures in one or several agents, by having other agents take over the activity of the faulty ones.
- Scalability and flexibility. In principle, since MAS are inherently modular, adding and removing agents to the system should be easy. In this way, the system could adapt to a changing task on-the-fly, without ever needing to shutdown or to be redesigned.
- Ease of design, development, and maintenance. This also follows from the inherent modularity of the MAS.

The potential benefits described above should be carefully weighed with the simplicity of a centralized solution, considering the characteristics of the task.

3. DEFINITION OF THE PROBLEM

The standard MPC formulation is (Camponogara, Jia, Krogh, & Talukdar, 2002):

$$\min_{X(k), U(k)} J(X(k), U(k))$$

where

$$X(k) = \{x(k+1|k), \dots, x(k+N|k)\}$$

$$U(k) = \{u(k|k), \dots, u(k+N-1|k)\}$$

s.t.

$$x(k+i+1|k) = F(x(k+i|k), u(k+i|k)) \quad (i = 0, \dots, N-1)$$

$$G(X(k), U(k)) \leq 0$$

$$x(k|k) = x(k).$$

This formulation can be summarized as a series of static optimization problems:

$\{SP_k \mid k=0,1,2,\dots\}$, each of the form:

$$\begin{aligned} SP_k : \min_S J(S) \\ \text{s.t. } G(S) \leq 0 \\ H(S) = 0, \end{aligned}$$

where S is the vector of the decision variables, including state variables X and control variables U , over the prediction horizon. The equality constraint in the problem includes the prediction model and other equality operation constraints. (Camponogara, Jia, Krogh, & Talukdar, 2002)

Distributed MPC is a decomposition of SP_k into a set of M sub-problems, $\{SP_{ki} \mid i=1,2,\dots,M\}$, and each sub-problem is assigned to a different agent. (Camponogara, Jia, Krogh, & Talukdar, 2002)

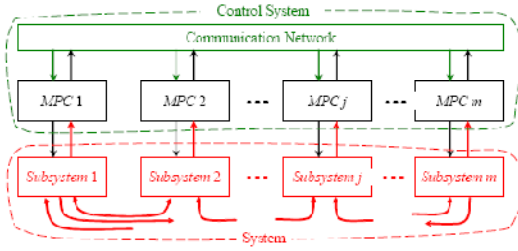


Figure 5: Distributed control system (Giovanini & Balderud, 2006)

Figure 5 shows the structure of the distributed control system problem. It can be seen how subsystems are connected (they can be fully connected or partially connected). These connections represent control variables shared among the objective functions of two agents.

4. OBJECTIVES

- To develop a distributed control Architecture for Large Scale Systems based on three main concepts: Negotiation-Cooperation-Learning. As an answer to the distributed control problem defined in Chapter 3.
- To conjugate Distributed Model Predictive Control, Reinforcement Learning and the Agent Oriented Paradigm as the basis of the proposed approach.

- To prove technical feasibility of the proposed approach.
- To provide a general methodology for the application of the proposed architecture.
- To validate the proposed architecture applying it on the Barcelona water transportation network.
- To contrast results against the centralized and decentralized approaches applied to the same case of study.

5. EXPECTED CONTRIBUTIONS

As it was argued in the state of art (Chapter 2), as much MPC, RL and AOP are powerful tools widely studied and applied each one in their own area. Works have been made relating MPC and RL but not in cooperative or even negotiating environments. There is a very short intersection between AOP and control and no intersection at all of these three areas.

In the state of art, it was also shown how these three fields have many things in common. One of the contributions expected in this thesis is to select suitable algorithms and tools of these branches and adapt them to create an adaptive distributed control architecture capable of performing negotiation-cooperation-learning actions on an efficient communication platform.

In order to develop this efficient communication and coordination it will be necessary to exploit the protocols, standards and tools that the AOP offers.

Another expected contribution is to introduce the term agent in the control language as a basic element of the AOP and to combine suitable solutions between distributed control and distributed Artificial Intelligence.

For unifying purposes one will define Agent as:

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, implements processes of reaction and/or deliberation, as well as communication and 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 previous definition is analogous to the Object definition of the Object Oriented Paradigm (OOP), in the sense that, if we want to define what an object is without being placed in specific context, anything is an object. Nevertheless, within the computational and the OOP, the object concept is well defined and known. In the same way if we were not placed in any context, an agent can be almost anything. Since the theory of agents is part of Computer Science this is the suitable context to define the term. Placed in this context, this definition remarks the AOP and present it as a feasible solution for applications to the diverse areas of the sciences and engineering.

6. PROPOSED ARCHITECTURE: MULTI-AGENT MPC

In this section, the proposed architecture is presented. First, the concepts of partitioning of the plant and partitioning of the optimization problem will be defined for the aims of this work. Next, definitions of the elements of the architecture will be made as well as the structure of the architecture and its dynamics.

There have been detected two main problems in the design process of a distributed system: the partitioning of the plant and the partitioning of the optimization problem.

6.1. Partitioning of the plant.

Many proposals for partitioning the plant can be found in the literature (Camponogara, Jia, Krogh, & Talukdar, 2002) (El Fawal, Georges, & Bornard, 1998) (Van Breemen & De Vries, 2001) (Barcelli, 2008) (Rawlings & Stewart, 2008). For the implementation of the MAMPC (Multi-Agent MPC) Architecture an intuitive partitioning of the network can work. Nevertheless, if either a more rigorous method or an intuitive one is used, there are some considerations to take into account:

Plant partitioning considerations. The plant partitioning considerations represent all the

considerations to take in to account in the process of partitioning of the plant in order to apply the MAMPC Architecture. These considerations are:

- 1- The set of partitions must be a *complete set of partitions*.
- 2- The physical topology of the network must be respected. This means that the plant model represents a large infrastructure that will not be modified, not physically nor logically respecting its topology configuration.
- 3- Minimum relations between partitions are desirable (always fulfilling consideration 1).
- 4- Generalization and specification are allowed. That means that the model can be aggregated or disaggregated for simplification or specification without failing with that to consideration 1.
- 5- There is a compromise between the number of partitions and the number of relations between them. That means that it is not desirable to have too many partitions in order to have too little relations, nor to have few partitions with too many relations.
- 6- Economical, geographical, and management considerations have to be taken in to account and the result of the system partitioning must be approved by the final manager of the system.

Definition 1. Complete set of partitions. Given a model of the plant P , a complete set of partitions of P is:

$$P = s_1 \cup s_2 \cup \dots \cup s_n$$

where $s_1, s_2 \dots s_n$ are partitions (sub-networks) of the plant.

It is said to be complete because the sum of all partitions is equal to the original plant P . When the equation above is not satisfied the set of partitions will be not complete and therefore incorrect.

Definition 2. MAMPC Architecture. MAMPC is a distributed control architecture that can be defined as:

$$\gamma = \{A, N, P, W, V_{nn}, U_{na}, b\}$$

where:

A is the set of MPC Agents, N is the set of Negotiator Agents, P is the Complete Set of Partitions of the plant, W is the set of nodes, V_{nn} is the set formed by all sets of

Negotiation Variables where nn is the number of elements in N , U_{na} is the set formed by all sets of Internal Variables where na is the number of elements in A and b is the Agent platform.

6.2. Partitioning of the Optimization Problem.

In the state of the art it was mentioned that distributed control works propose to partition the system via the partitioning of the optimization problem. In this kind of approach the resulting partitions have no geographical or topological meaning and therefore for implementation the control has to be achieved remotely.

In the present approach, the partitioning of the plant is related to the partitioning of the optimization problem but it is not the same.

Definition 3. Partitioning of the Optimization Problem. The partitioning of the Optimization problem for the MAMPC Architecture represents the way in which the optimization problem is dealt with. It is divided in two parts. The *Agent Multivariable Problem* and the *Agent Negotiation Problem*.

Definition 4. The Agent Multivariable Problem. The Agent Multivariable problem is the control optimization of one partition of the system solved by an *MPC Agent* via MPC of all the *internal variables*.

Definition 5. The Agent Negotiator problem. The Agent Negotiator problem represents the optimal value of the *negotiation variables* that can exist between *MPC Agents*. This problem is solved by the Negotiator Agent and the result is the optimal value for the relation as a common goal. That means that agents cooperate in a way that this common goal has priority over the goal of each *MPC Agent*.

6.3. Elements of the MAMPC Architecture.

The main actors in MAMPC Architecture are *MPC Agents* and *Negotiators Agents*. They interact over an *Agent platform*. Other important entities are: *negotiation variables*, *internal variables* and *nodes*. Next a definition of each one is given.

Definition 6. MPC Agent. An MPC Agent is the entity of the MAMPC Architecture that is in charge of one specific partition of the system. There is one specific MPC Agent

for each partition of the problem. The MPC Agent solves the *agent multivariable problem* for all of its *internal variables* by means of MPC. Also an MPC Agent can have *negotiation variables*, in this case the MPC Agent will cooperate with one or more Negotiator Agents to determine the optimum value for these variables. An MPC Agent will cooperate with as many negotiator Agents as many MPC Agents with negotiation variables shares.

A is the set of MPC Agents defined by

$$A = \{a_1, a_2, \dots, a_{na}\}$$

where $\{n > 1\}$ na is the number of MPC Agents

Definition 7. Negotiator Agent. A negotiator Agent is the entity of the MAMPC Architecture capable to determine the value of one or more *negotiation variables* between two MPC Agents. In this negotiation, each MPC Agent is arranged to cooperate so that the negotiator agent solves an optimization of a common goal by means of an algorithm based on Reinforcement Learning. A negotiator Agent exists when two, and only two, MPC Agents have one or more *negotiation variables* in common.

N is the set of Negotiator Agents defined by

$$N = \{n_1, n_2, \dots, n_{nn}\}$$

where nn is the maximal possible value of the Negotiation Agents that satisfies:

$$nn \leq \sum_{i=0}^{nn-1} (nn - 1)$$

Definition 8. Nontrade variables. The nontrade variables are the independent control variables that each *MPC Agent* has. By *independent* it can be understood that a variable is not related to other *MPC Agent*.

U is the set of internal variables defined by

$$U = \{u_1, u_2, \dots, u_{nu}\} \quad \text{where } U_i \in a_i$$

Definition 9. Negotiation Variables. A negotiation variable between two *MPC Agents* exists when there is a physical connection preserved between two partitions of the system that represents a control variable. When this occurs each MPC Agent cooperates by means of a Negotiator Agent in order to determine a common optimum value for this variables. An MPC Agent can have many negotiation variables shared bilaterally with one or more MPC Agents.

V is the set of negotiation variables defined by

$$V = \{v_1, v_2, \dots, v_{nv}\}$$

Each Negotiator Agent deals with a subset of V .

Definition 10. Nodes. A node is the physical device (commonly a computer) in which the agents are located. In MAMPC Architecture, there is a node for each MPC Agent. Nodes are communicated via LAN, WAN or Internet.

W is the set of nodes defined by

$$W = \{w_1, w_2, \dots, w_{nw}\}$$

where there is a node for each MPC Agent.

Definition 11. Agent Platform. All agents in MAMPC Architecture, run over an agent platform. This platform has to be installed and running in all nodes. It works as a virtual machine providing the agents a homogenous medium to communicate and, the user, a way to manage agents. Agent platform is denoted by b .

6.4. Structure of the MAMPC Architecture

Once all the elements of the architecture were defined, the structure of the proposed MAMPC Architecture is presented.

It was mention before that the partitioning of the plant and the partitioning of the optimization problem (section 6.1 and 6.2 respectively) are two different things in MAMPC Architecture. The partitioning of the plant is related to the structure of the resulting MAMPC system and the partitioning of the optimization problem is related to the internal design of MPC Agents and Negotiation Agents. In order to explain how this and other new concepts are related Figure 6 is presented.

In Figure 6, a MAMPC system with three partitions is shown. This partitioning accomplishes the plant partitioning considerations defined before. The resulting structure of this MAMPC system is:

$$P = s_1 \cup s_2 \cup s_3$$

where P is a complete set of partitions

$$A = \{a_1, a_2, a_3\}$$

$$N = \{n_1, n_2, n_3\}$$

$$U_1 = \{c_1, c_2, c_6, c_9\}$$

$$U_2 = \{c_4, c_5, c_8\}$$

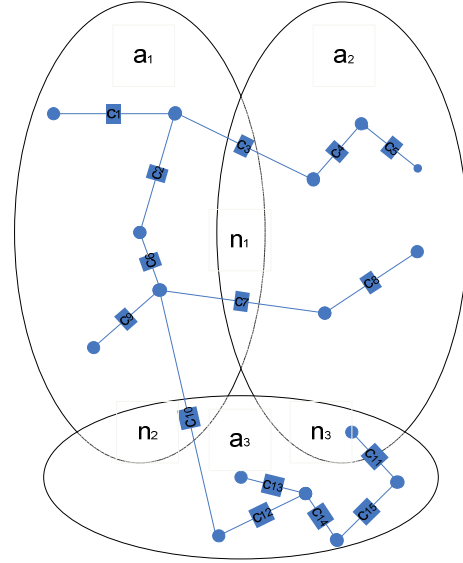


Figure 6: MAMPC system with three partitions

$$U_3 = \{c_{12}, c_{13}, c_{14}, c_{15}\}$$

$$V_{12} = \{c_3, c_7\}$$

$$V_{13} = \{c_{10}\}$$

$$V_{23} = \{c_{11}\}$$

$$W = \{w_1, w_2, w_3\}$$

As can be seen in Figure 6, a_1 is the MPC agent in charge of partition s_1 and its set of internal variables is U_1 . It has two sets of negotiation variables, V_{12} shared with a_2 and V_{13} shared with a_3 . So a_1 will be interacting with n_1 and n_2 , that are the negotiation agents in charge of V_{12} and V_{13} , respectively. a_1 will be allocated in w_1 , a_2 in w_2 and so on.

In general, the structure of the plant partitioning can be seen as shown in Figure 7 where each partition can have one or more negotiation variables with other partitions or do not have any. If this is the case, it is said to be internal. The MAMPC Architecture can have partitions connected and internal mixed (partially connected).

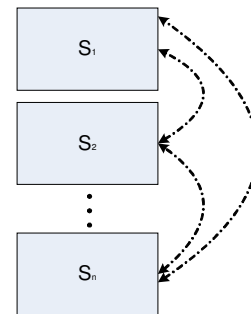


Figure 7: Structure of the plant partitioning

Another important thing of MAMPC Architecture

structure is how agents interact. The interaction is needed only for negotiation of control variables between MPC Agents. But this interaction can not be from MPC Agent to MPC Agent. It has to be through a Negotiation Agent. Figure 8 shows, in general, how this interaction is made.

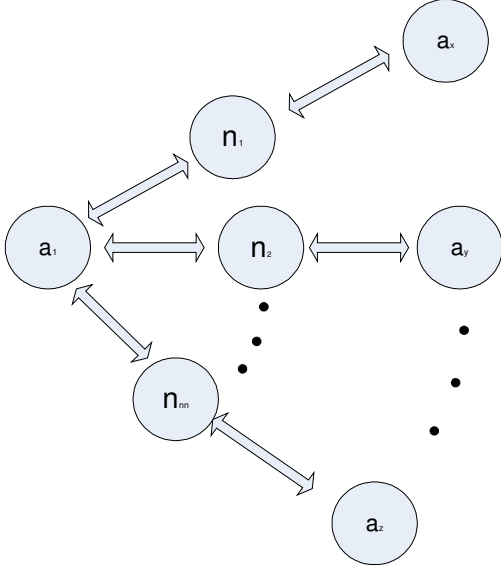


Figure 8: Agent interaction structure of MAMPC Architecture.

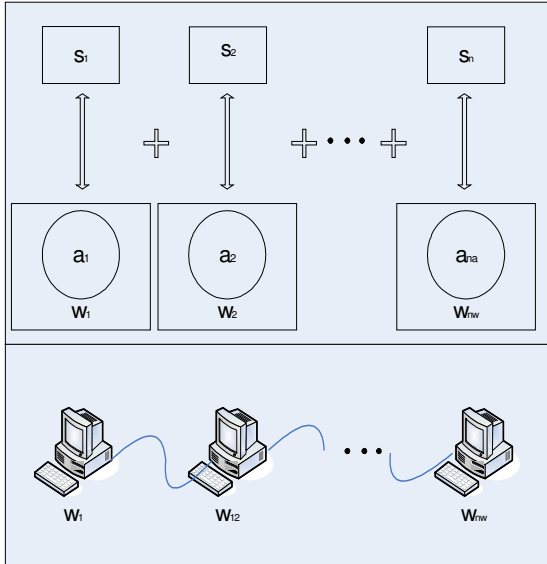


Figure 9 Assignment of agents, partitions and nodes

Figure 9 shows the general structure of the MAMPC Architecture. Each partition has sensors and actuators installed in the transport network. The local control is

achieved via PID. In a higher level, the Multi Agent system solves the distributed optimization problem. There is one MPC Agent for each partition of the plant and one node for each MPC Agent. Communication of agents is possible through the Agent Platform and custom physical connections between computers are needed.

6.4.1. INTERNAL DESCRIPTION OF THE MPC AGENT

The core of the MPC agent is a MPC controller. This controller solves the multivariable problem of one partition of the plant based in a model. This model contains the set U_x of the agent. Other important part of the MPC Agent is its 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 allowed good communication skills.

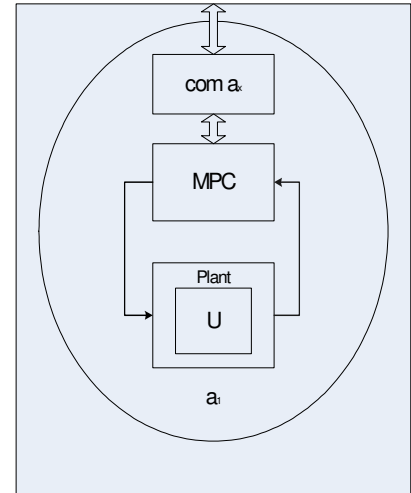


Figure 10: Internal architecture of the MPC Agent

6.4.2. INTERNAL DESCRIPTION OF THE NEGOTIATION AGENT

The Negotiation Agent determines the optimal value of a set V_x . This set contains the shared variables of two, and just two MPC Agents. The Negotiation Agent optimizes them through a Negotiation algorithm based on Reinforcement Learning. Each negotiation 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 reinforces given at each step and on the experience obtained. This experience is stored in a knowledge base, one for each negotiation variable. As MPC Agents, Negotiation Agents have a communication block that uses to communicate with two related MPC Agents.

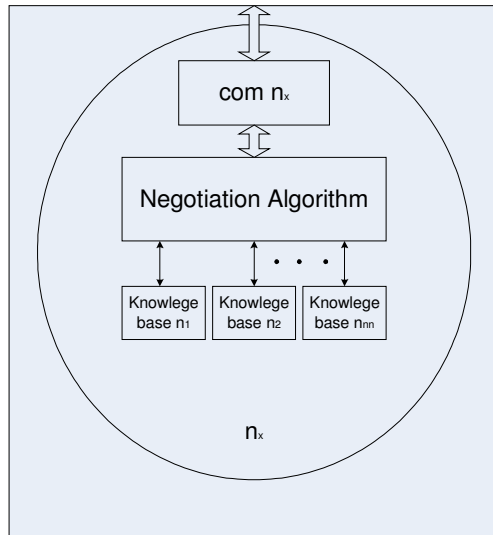


Figure 11: Internal architecture of the Negotiation Agent

6.5. DYNAMICS OF THE MAMPC ARCHITECTURE

The Dynamics of the MAMPC Architecture can be described in the following algorithm.

1. First Negotiation Agents applied a random action (that is the same to say that determine a random value for them negotiation variables)
2. The resulting values of the Negotiation Agents are sent to the respective MPC Agents
3. MPC Agents fix as restrictions in the manipulated variables all the received values and calculates the rest.
4. MPC Agents calculates the error obtained as a result of its negotiation variables.
5. MPC Agents send to the respective Negotiation Agents the result of the computed control action.
6. Negotiation Agents applied the negotiation algorithm and determine the new control action of all the shared variables.
7. Go to 2 until stop by the user
8. End

6.6. BENEFITS OF THE PROPOSED ARCHITECTURE

1. The use of MPC as principal control strategy will bring many benefits, in one hand all its technical benefits and in the other hand its acceptance in industry.
2. The flexibility of the partitioning technique. This

flexibility allows to conjugate administrative, geographic, topologic and economic criteria.

3. Negotiation is made from a cooperative point of view, optimizing the relation between the agents in order to obtain a global optimum for both agents.
4. In decentralization of highly connected large scale systems, negotiation is a promising solution.
5. This architecture makes a future fault tolerance implementation possible.
6. Plus, all other benefits of decentralization (speed, scalability, simplicity in the maintenance of the system, etc, etc)

The use of MAS will allow to:

1. Perform an appropriate coordination and synchronization of the agents
2. Provide a management and communication platform for the MAS. This will allow to allocate MPC Agents in different computers of a net.
3. To use appropriate tools of development and standards
4. To use methods and tools of Analysis and Design in order to make an appropriate formalization and documentation of the system

The use of RL in the negotiation process will allow to:

1. Make the process of negotiation adaptive
2. Learn from its own experience
3. Explicitly consider the whole problem of two goal-oriented agents
4. Deal with a dynamical and uncertain environment
5. Optimize whit or without a model
6. Connect the process of negotiation whit the one of the control MPC, this because of compatibilities found between them (see Chapter 2)

7. WORK PLAN

2008-2009

#	Task	Sep	Oct	Nov	Dec	Jen	Feb	Mar	Apr	May	Jun	Jul
1	Documentary Investigation (MPC, LSS, MAS)											
2	Searching of negotiation-cooperation algorithms											
3	Investigation on RL											
4	Choosing tools											
5	Connection and tool configuration											
6	Testing the connections											
7	Implementation of static Q-learning algorithm in Jade											
8	Design of the proposed Architecture											
9	Design of the negotiation algorithm											
10	Application of the proposed architecture in a toy problem											
11	Analysis of the results											
12	Writing and presentation of the thesis proposal											
13	Events ¹											

Work to do

Work done



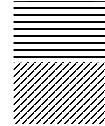
¹ Events like congresses, summer courses and visits to labs. Here are represented in two months but in fact they will be distributed in among summer of 2009 to summer of 2011. Events program so far: WIDE School (4-7 Jul, Siena Italy), NecSys09 (24-27 Sep, Venice, Italy)

2009-2010

#	Task	Sep	Oct	Nov	Dec	Jen	Feb	Mzo	Apr	May	Jun	Jul
14	Formalization of the framework MAS-MPC-RL											
15	Application of the architecture to the aggregated case of study											
16	Experiments and analysis of the results											
17	Investigation and testing of coordination techniques											

Work to do

Work done



2010-2011

#	Task	Sep	Oct	Nov	Dec	Jen	Feb	Mzo	Apr	May	Jun	Jul
18	Implementation of coordination in the case of study											
20	Learning to negotiate: refinement of the negotiation algorithm ²											
21	MAS coordination with multiples negotiators study.											
22	Application in case of study											
23	Writing of the thesis											
25	Finish											

² Learning to negotiate: refinement of the negotiation algorithm: In this task, different algorithms of RL and tests will be developed in order to refine the negotiation-learning algorithm. Effects of MPC and RL algorithm parameters will be studied and a criteria for their tuning will be provided in the proposed methodology.

8. PRELIMINARY RESULTS

So far, the work done can be summarized as:

Validation of the technical suitability of the connection of the selected tools. Connection, configuration and testing of MATLAB, Java and Jade have been done assuring with that the technical suitability of the proposal.

Research work on “Negotiation in distributed control systems by means of the application of Reinforcement Learning in Multi-Agent environments”. This work was presented as final project of the subject “Artificial Intelligence applied to Control and Identification”. It summarizes some parts of the state of art of RL and MAS. A static problem using the RL algorithm, Q-Learning, is also presented. This algorithm was implemented in Jade and a short description of this tool is also commented.

Development and application of the MA-MPC Architecture described in the next section.

8.1. Development and application of the MA-MPC Architecture

The proposed architecture was applied to a hypothetical water distribution network with 8 states (tanks) and 11 control variables. This system was divided in two cooperatives MPC Agents and a Negotiator Agent that determine the value of the overlapping variables subset V that contains two shared control variables (see Figure 12).

The example choosen was the presented in (Barcelli, 2008) where a centralized and a decentralized solution was proposed.

The objectives of this case of study are:

- To refine the proposed architecture through the identification of problems in the development process.
- To validate the connection between MPC and RL frameworks.
- To validate the technical feasibility of connection of the chosen tools.
- To detect relevant design issues to be considered in the proposed architecture.

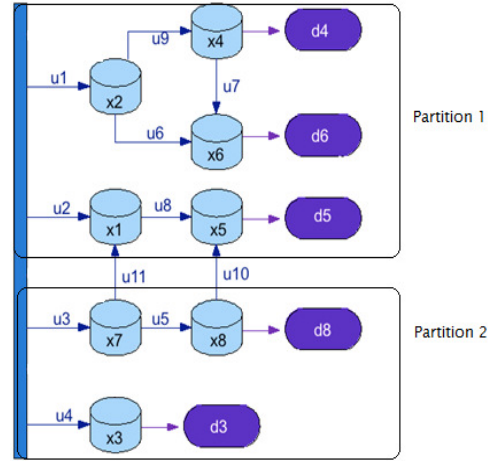


Figure 12: Academic case of study and its partitioning

The proposed methodology can be divided in for phases

- Analysis
- Design
- Training
- Exploitation

Next each phase will be detailed.

8.1.1. Analysis

In the analysis phase, the MA-MPC Architecture is defined for this case of study. The main tasks to develop in this phase are:

- Definition of the optimization goals
- Definition of the partition of the network
- Definition of the architecture
- Definition of restrictions and other considerations

Defining Goals

The control goal of the application presented in Figure 12 is to keep a volume in tanks around 3m^3

Defining the network partitions

The system is decomposed in two partitions:

$$S_1 = \{x_1, x_2, x_3, x_4, x_5, x_6\}$$

$$S_2 = \{x_3, x_7, x_8\}$$

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

$$P = S_1 \cup S_2 \cup U_1 \cup U_2$$

Thus, the partition is a complete set of partitions.

Defining the Architecture

In this step, the MA-MPC Architecture is defined for the problem.

Starting from the definition of the architecture, every element is defined as follows:

$$\gamma = \{A, N, P, W, V_{nn}, U_{na}, b\}$$

where:

$$A = \{a_1, a_2\}$$

$$N = \{n_1\}$$

$$E = \{w_1, w_2\}$$

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

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

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

Defining restrictions and considerations

The maximum volume in tanks is $20m^3$, the control value of the measured variables is from 0.0 to 0.4 except for the 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 throughout the day.

8.1.2. Design

In the design process the sub-problems of every MPC-Agent and Negotiator Agent are formulated. This

formulation is based in the information collected in the analysis phase.

Formulation of the MPC problem

In this step all the MPC parameters and requirements have to be defined for both agents, such as:

- The plant
- The measured, non-measured and manipulated variables
- Limits and constraints
- References (goals)
- Prediction horizon
- Control Horizon
- Initial state
- Perturbations models

All these data have to be set in all MPC-Agents. The prediction and control horizon should be the same for all MPC-Agents.

The definition of these parameters for this problem are:

Agent 1

Plant

$$A_1 = I_5$$

$$B_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

$$C_1 = I_5$$

$$D_1 = \mathbf{0}$$

States						
MPC Agent 1:	x_1	x_2	x_3	x_4	x_5	
Corresponding state in the centralized plant	x_1	x_2	x_4	x_5	x_6	
Outputs						
MPC Agent 1:	y_1	y_2	y_3	y_4	y_5	
Corresponding output in the	y_1	y_2	y_4	y_5	y_6	

centralized plant										
y_x Max value=20										
y_x Mix value=0										
Perturbation variables (demands)										
	MPC Agent 1:	u_9	u_{10}	u_{11}						
	Corresponding variable in the centralized plant	d_4	d_6	d_5						
	u Min value	0.0	0.0	0.0						
	u Max value	0.4	0.4	0.4						
Manipulated variables (Internal variables)										
MPC Agent 1:	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8		
Corresponding variable in the centralized plant	u_1	u_2	u_6	u_7	u_8	u_9	u_{10}	u_{11}		
u Min value	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
u Max value	0.4	0.1	0.4	0.4	0.4	0.4	0.4	0.4	0.4	
Shared variables										
Goal: 3										
Prediction horizon (p):24										
Control horizon (m):1										
Initial State:										
MPC Agent 1:		x_1	x_2	x_3	x_4	x_5				
Initial value		0	3	5	0	5				

Perturbations (demands)

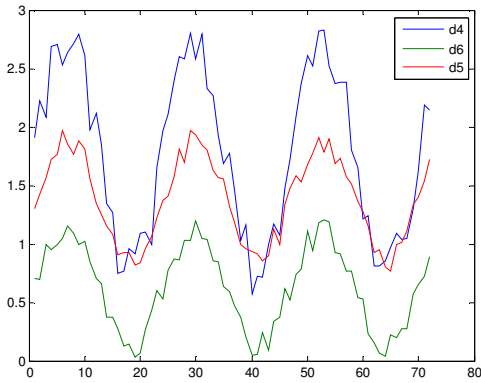


Figure 13: Demands of MPC Agent 1

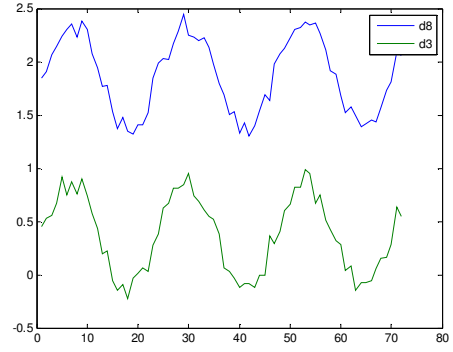


Figure 14: Demands of MPC Agent 2

Agent 2

$$\begin{aligned}
 A_2 &= I_3 \\
 B_2 &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 \end{bmatrix} \\
 C_2 &= I_3 \\
 D_2 &= \mathbf{0}
 \end{aligned}$$

States						
	MPC Agent 2:	x_1	x_2	x_3		
	Corresponding state in the centralized plant	x_3	x_7	x_8		
Outputs						
	MPC Agent 2:	y_1	y_2	y_3		
	Corresponding output in the centralized plant	y_3	y_7	y_8		
y_x Max value=20						
y_x Mix value=0						
Perturbation variables (demands)						
	MPC Agent 2:	u_6	u_7			
	Corresponding variable in the centralized plant	d_8	d_3			
	u Min value	0.0	0.0			
	u Max value	0.4	0.4			
Manipulated variables (Internal variables)						
	MPC Agent 2:	u_1	u_2	u_3	u_4	u_5
	Corresponding state in the centralized plant	u_3	u_4	u_5	u_{10}	u_{11}
	u Min value	0.0	0.0	0.0	0.0	0.0
	u Max value	0.4	0.4	0.4	0.4	0.4
Shared variables						

Goal: 3				
Prediction horizon (p):24				
Control horizon (m):1				
Initial State:				
MPC Agent 2:	x_1	x_2	x_3	
Initial value	5	10	20	

Cooperation of MPC-Agents

The cooperative interaction of MPC agents is a basic issue in the proposed approach. Three main actions are necessary to perform this cooperation:

- To perform actions and provide data required by the Negotiator Agent
- To accept the value determined by the Negotiator Agent of its internal shared variable.
- To adjust the value of its control variables in order to coordinate the solution of the negotiation and solve with this the multivariable problem.

Philosophy of the Negotiator Agent algorithm

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 unifying this duplicate 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 constraint 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 algorithm is based on Q-learning algorithm (see Figure 15), and adapted to be applied in dynamical environments. Next, the formulation of the algorithm is detailed.

Formulation of the negotiation-learning problem

- A set of Knowledge bases (Q-tables)
- A communication protocol that allows him to have bi-directional communication with two MPC-Agents with shared control variables.
- A negotiation algorithm

Input: learning rate α , discount factor γ

- 1: $Q(x, u) \leftarrow 0, \forall x \in X, u \in U$
- 2: observe initial state x
- 3: loop
- 4: $u \leftarrow h(x)$ where h is a policy derived from Q (e.g., ϵ -greedy)
- 5: apply u , observe r and x'
- 6: $Q(x, u) \leftarrow Q(x, u) + \alpha [r + \gamma \max_{u' \in U} Q(x', u') - Q(x, u)]$
- 7: $x \leftarrow x'$
- 8: end loop

Figure 15: Q-learning algorithm

The goal of the Negotiation agent is to determine the optimal value of the set of shared variables V . Each element of the set V is an optimization problem addressed individually by the Negotiator Agent and there is a Knowledge base for each one.

The formulation described below was applied to shared variables of the problem³.

Q-Table

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

For this problem two bidimensional Q-matrices were built. One dimation is for state and the other for the control action $Q(s, a)$.

³ A definition of the RL elements mentioned in this formulation can be found in Annex 1.

States in the Q-table

The states (s) were defined based on the philosophy of the proposed algorithm. Therefore the state represents the global state of each sub-problem. This state is established in terms of the error of the output with respect to the goal. To determine the state MPC-Agents have to cooperate agreeing the internal value of Negotiated Variables.

The definition of the error that MPC Agents use is:

$$\varepsilon_i = g_i - y_i$$

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

The state is determined by:

$$s = \frac{1}{2} (|\varepsilon_{i1}| + |\varepsilon_{i2}|)$$

ε_{i1} is the error of the variable i of agent one, and ε_{i2} of the correspondent variable in agent two. This state is updated every sampling time.

Since the states are continuous, they have to be discretized for the application on the RL algorithm. In this case 100 discrete states, from 0 to 10 were defined.

Actions in the Q-table

In this approach, actions (a) are all the possible values that the shared variable can take. Since all the manipulated variables are set between 0 and 0.4 the determined discretization is 40 states from 0.0 to 0.4. So, the resulting table Q (s, a) is Q(100,40)

Reward function

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

$$r = \rho - s \quad \text{where } \rho \text{ is a value } \geq \text{ than } s$$

Communication protocol

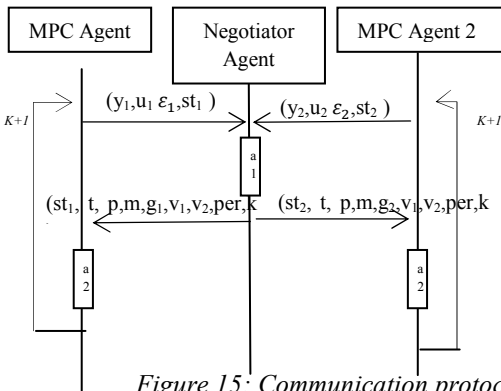


Figure 15: Communication protocol

Figure 15 shows a sequence diagram of the communication protocol designed for this application

where:

y_x is the output signal of the MPC Agent in the instant k
 u_x in the control vector of all the measured variables of the MPC Agent in the instant k
 ε_x is the error with respect to the goal of the output y_i , where y_i is the output of the internal variable.
 st_x in the state of the plant in the k instant
 t is the sampling time
 p is the prediction horizon
 m is the control horizon
 g_x is the goal
 v_1 is the shared variable 1
 v_2 is the shared variable 2
 per are the Perturbations
 k is the current time instant
 x can be 1 for Agent 1 or 2 for agent 2
 $a1$ is the process of the negotiation algorithm
 $a2$ is the process of the MPC Agent algorithm

The negotiation algorithm has two modes, the training (exploration) mode and the exploitation mode

8.1.3. Training (exploration)

As in any RL algorithm, the proposed architecture is based on the agent experience and the expected reinforcements. As richer the agent experience has been, as efficient the optimization algorithm will be.

An off-line training was done in order to provide this experience to the Negotiator Agent. First exhaustive methods were applied, but the matrices obtained let many states without being visited. So, control actions determined using the centralized approach were used as initialization values for the agent training process. The following training algorithm summarizes the agent training problem developed:

1. Set up the parameters and data of the MAMPC Architecture.
 - a. Set the perturbations (demands) vector
 - b. Set up a vector of the control actions taken by the shared variable in the centralized case.
 - c. Set up the initial state of the plant
2. Create MPCAgent1 and MPCAgent2
3. $i=1$
4. do
 - a. Select i action for the shared variable of the action vector
 - b. Send parameters to MPCAgent1 and MPCAgent2
 - c. Receive ε_{i1} and ε_{i2}
 - d. Calculate state (s)
 - e. Calculate reward (s)
 - f. Update Q-matrix
$$Q(s, a) = r + (\alpha \times (Q(s, a)))$$
5. While $i \leq$ vector of control actions length
6. $j=i$
7. Define number of iterations for training (iterations)
8. do
 - a. Select a random action
 - b. Send parameters to MPCAgent1 and MPCAgent2
 - c. Receive ε_{i1} and ε_{i2}
 - d. Calculate state (s)
 - e. Calculate reward (s)
 - f. Update Q-matrix
$$Q(s, a) = r + (\alpha \times (Q(s, a)))$$
9. While $i \leq$ iterations
10. End

8.1.4. Exploitation

In RL exploitation phase the knowledge acquired in the exploration (and training) phase is used. (More details about exploration-exploitation are given in Annex 1)

Exploitation phase use the knowledge acquired in order to solve the MPC distributed problem through the MA system.

The exploitation algorithm is the following:

1. Load Q-matrices
2. Load perturbations
3. Load initial values of the plant of MPCAgent1 and MPCAgent2.
4. Time (simulation time)
5. Select a random action (a)
6. do
 - a. Send parameters to MPCAgent1 and MPCAgent2
 - b. Receive ε_{i1} and ε_{i2}
 - c. Calculate state (s)
 - d. Calculate reward (r)
 - e. Select next action
 - f. Update Q-matrix
$$a = \max_{a'} (Q(s', a'))$$

$$Q(s, a) = r + (\alpha \times (Q(s', a')))$$
- g. $s=s'$
- h. $a=a'$
7. While $i \leq$ time

8.1.5. Implementation

The MPC Agents were implemented in MATLAB and exported to Java with MATLAB Builder toolbox. The Negotiator Agent and the support classes were implemented in Java.

8.1.6. Results

The results obtained using the decentralized MPC using MA are compared with the corresponding outputs of the centralized solution. Figure 16 and 18 presents the outputs of Agent 1 and 2, while Figure 17 and 19 present the same outputs obtained by the centralized MPC controller.

A more detailed comparison of the outputs of the shared variables is shown in Figures 20,21,22,23. In this figures outputs (levels) of the tanks related to shared variables (tanks 1, 5, 7 and 9 of Figure 12 respectively) are compare whit the same outputs of the centralized case solve whit MPC.

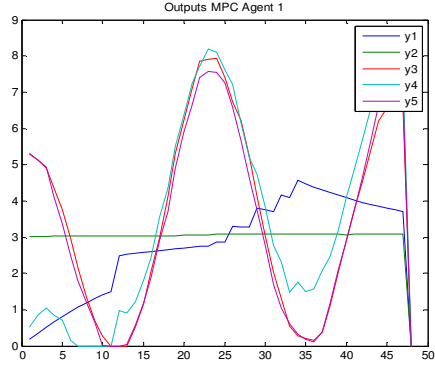


Figure 16: Outputs of MPC Agent 1

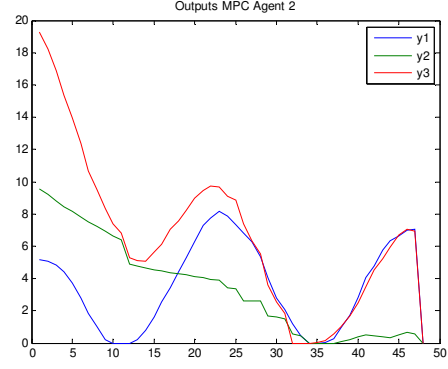


Figure 17: Outputs of MPC Agent 2

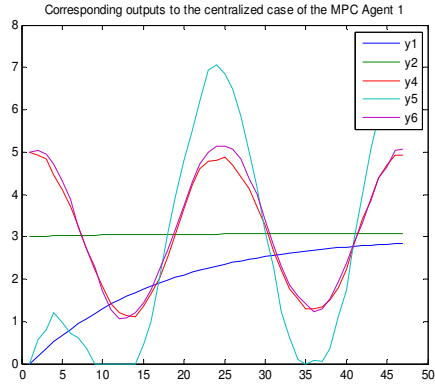


Figure 18: Centralized outputs of the corresponding outputs of MPC Agent 1

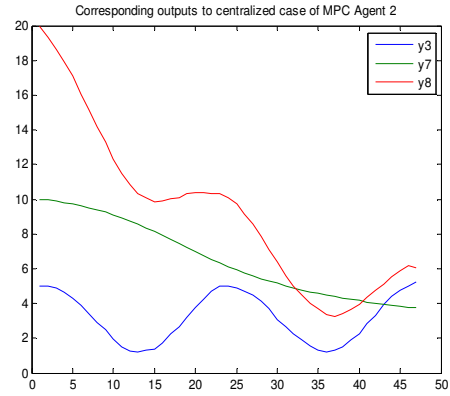


Figure 19: Centralized outputs of the corresponding outputs of MPC Agent 2

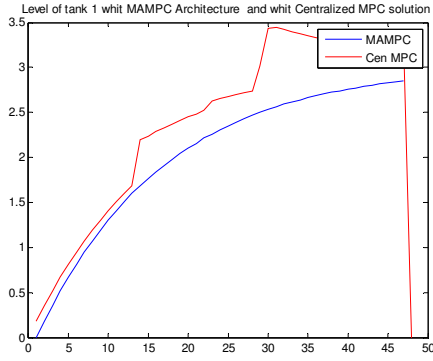


Figure 20: comparison between MAMPC and centralized MPC solutions of tank 1

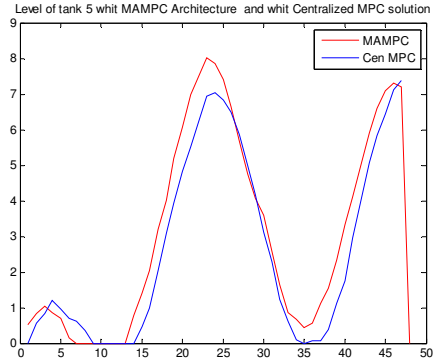


Figure 21: comparison between MAMPC and centralized MPC solutions of tank 5

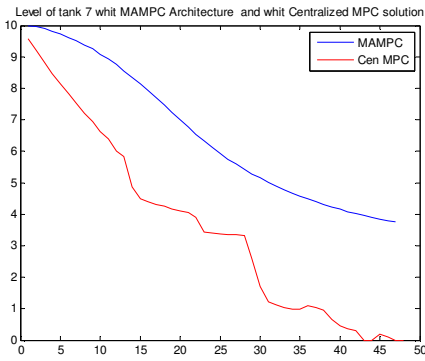


Figure 22: comparison between MAMPC and centralized MPC solutions of tank 7

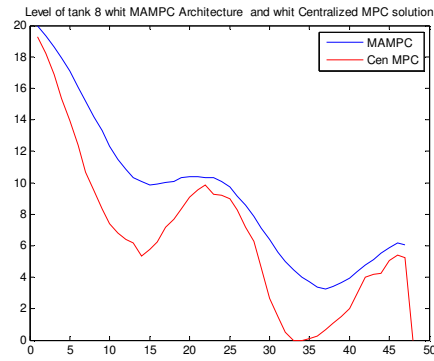


Figure 23: comparison between MAMPC and centralized MPC solutions of tank 8

8.1.7. Conclusions

The results obtained suggest that the use of a distributed control architecture based on negotiation can converge to the centralized solution with an acceptable degree of approximation but benefiting from the decentralization properties. 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. In order to achieve optimization and even prediction abilities, no model is needed by the negotiator. Data from centralized MPC is advisable but non essential.

Interaction between classes exported from MATLAB to Java were very efficient in this application. MATLAB allows to profit from the MPC toolbox and all the facilities to manage systems and matrices.

The use of Java will allow to use Jade for the Multi-agent implementation since Jade works in Java. Jade is Multi-agent language that provides support to manage agents and its interactions through a special platform called Jade RMA (Remote Agent Management) GUI.

The type and quality of the training is a very important issue in order to obtain an efficient optimization. Some relevant things to consider in order to improve the performance of the algorithm are:

- The number of iterations in the off-line training is important to make more efficient the algorithm
- To consider more control actions obtained from different operating scenarios in the training phase will provide more significant experience.
- The effects of the variation of the parameter α have to be studied.
- The interrelation of the parameters in the prediction process for MPC Agents and Negotiator Agents have to be studied.
- The discretization and the values of the actions and states that can be chosen from it are a key issue.
- There is a compromise between exploration and exploitation that can be implemented on line to enable the system adaptation capability taken that does not come from training but from exploring during the optimization. This compromise have to be studied.
- The computational training cost is high (for one computer), so the parallel implementation in a MAS have to be studied.
- Communication protocols for MAS have to be studied and tested

- Additional coordination methods for MAS have to be studied and tested.
- Other RL algorithms (like Sarsa) can be adapted and proved for this architecture.
- The effects of increasing the number and size of the partitions needs to be studied
- The effects of increasing the number of shared variables needs to be studied.

9. MATERIAL RESOURCES

This work is part of the European project: Decentralized and wireless control of Large Scale Systems, WIDE (TREP FP-7INF-SO-ICT-224168) and it is developed at the Institut de Robòtica i Informàtica Industrial (IRI). Data of the case of study is provided by AGBAR who is also partner of the WIDE project. Other collaborators of the project will provide some models and control data of the centralized approach. Expenses of computer, publications and trips to related events will be cover by the funds of the project. No special requirements are needed.

10. BIBLIOGRAPHY

Agostini, A., & Calaya, E. Feasible control of complex systems using automatic learning. *in Proc. ICINCO (Barcelona) 2005*.

Bakhtiari, R., Araabi, B. N., & Nili AhmadAbadi, M. (2007). A cooperative learning approach to mixed performance controller design: A behavioural viewpoint. *Int. J. Intelligent Systems Technologies and applications* , 2, 137-160.

Barcelli, D. (2008). Optimal decomposition of Barcelona's water distribution network system for applying distributed Model Predictive Control. *Master thesis* . Universitat Politècnica de Catalunya-IRI-Università degli Studi di Siena.

Boutillier, C. (1999). Sequential Optimality and coordination in Multi-Agent Systems. *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* .

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

Busonui, L., De Shutter, B., & Babuska, R. (2005). Learning and coordination in dynamic Multiagent Systems. *Technical report, Delf center for Systems and control* .

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

Camacho, E. F., & Bordons, C. (2004). *Model Predictive Control*. Springer.

Camacho, E. F., & Bordons, C. (1995). *Model Predictive Control in the process Industry*.

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

Cembrano, G., Figueras, J., Quevedo, J., Puig, V., Salamero, M., & Martí, J. (2002). Global control of the Barcelona Sewerage system for environment protection. *IFAC*.

Cembrano, G., Quevedo, J., Puig, V., Pérez, R., Figueras, J., & All. (2005). First results of predictive control applications on water supply and distribution in Santiago-Chile. *Proceedings of the IFAC world conference*.

Cembrano, G., Quevedo, J., Salamero, M., Puig, V., Figueras, J., & Martí, J. (2002). Optimal control of urban drainage systems. A case of study. *Control Engineering Practice* (12), 1-9.

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.

Claus, C., & Boutillier, C. (1998). The dynamics of Reinforcement Learning in cooperative multiagent systems. *In Proceedings of the Fifteenth National Conference on Artificial Intelligence* .

Du, X., Xi, Y., & Li, S. (2001). Distributed Model Predictive Control for Large Scale Systems. *Proceedings of the IEEE American Control Conference*, (pp. 3142-3143). Arlington, VA, USA.

El Fawal, H., Georges, D., & Bornard, G. (1998). Optimal control of complex irrigation systems via decomposition-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.

Ernst, D., Capitanescu, F., & Wehenkel, L. (2007). Reinforcement Learning Vs Model Predictive Control: A comparison on a power system problem. *IEEE Transactions on Power Systems* , 22.

Ernst, D., Glavic, M., Capitanescu, F., & Wehenkel, L. (2006). Model Predictive Control and Reinforcement Learning as a two complementary frameworks.

Proceedings of the 13th IFAC Workshop on Control Applications of Optimisation, (p. 6).

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

Fernández García, J. R. (2000). Complejidad y algoritmos en juegos cooperativos . *Tesis Doctoral, departamento de matemática aplicada, Universidad de Sevilla* .

Giovanini, L., & Balderud, J. (2006). Game approach to distributed MPC. *Proceedings of the International Control Conference*. Glasgow.

Gómez, M., Rodellar, J., Veá, 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 California. USA.

Jia, D., & Krogh, B. (2002). Min-max feedback model predictive control for distributed control with communications. *Proceedings of the IEEE American Control Conference*, (pp. 4507-4512). Anchorage, AK. USA.

Kapentanakis, S., & Kudenko, D. (2002). Reinforcement Learning of coordination in cooperative multi-agents systems. *Eighteenth national conference on Artificial intelligence*, (pp. 326-331). Edmonton, Alberta, Canada .

Lauer, M., & Riedmiller, M. (2000). An algorithm for distributed Reinforcement Learning in cooperative Multi-Agent system. *Proceedings of the Seventeenth International Conference on Machine Learning* , (pp. 535-542).

Martinez, E., & De Prada Moraga, C. (2003). Control Inteligente de Procesos usando aprendizaje por interacción. *XXIV Jornadas de Automática*. León, Spain: Universidad de León.

Maturana, P., Staron, R., & Kenwood, H. (2005, Ene-Feb). Metodologies and tools for intelligent agents in distributed control. *IEEE Intelligent Systems* , 42-49.

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

Negenborn, R. R., De Shutter, B., & Hellendoorn, J. (2004). *Multi-Agent model predictive control: A survey*. Technical report, Delf University of Technology, Delf center for systems and control.

Qin, S. J., & Badwell, T. A. (2003). A survey of industrial Model Predictive Control Technology. 11.

Qin, S. J., & Badwell, T. A. (2000). An overview of non linear Model Predictive Control Applications. In *Nonlinear Predictive Control* (pp. 369-392). Verlag.

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

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

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 languages* . Springer-Verlag.

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

Tatara, E., Çinar, A., & Teymour, F. (2007). Control of complex distributed systems with distributed intelligent agents. *Journal of process control* (17), 415-427.

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

Van Breemen, A. N., & De Vryes, T. A. (2001). Design and implementation of a room thermostat using an agent based approach. *Control Eng. Practice* , 9 (3), 233-248.

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

WIDE - 224168 - FP7-ICT-2007-2, *Decentralized Wireless Control of Large-Scale Systems*. (n.d.). Retrieved May 29, 2009, from <http://ist-wide.dii.unisi.it/>

ANNEX 1: REINFORCEMENT LEARNING ELEMENTS AND PROBLEM DEFINITION (SUTTON & BARTO, 1998)

Elements of Reinforcement Learning

Beyond the agent and the environment, one can identify four main subelements of a reinforcement learning system: a *policy*, a *reward function*, a *value function*, and, optionally, a *model* of the environment.

A *policy* defines the learning agent's way of behaving at a given time. Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states. It corresponds to what in psychology would be called a set of stimulus-response rules or associations. In some cases the policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as a search process. The policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine behavior. In general, policies may be stochastic.

A **reward function** defines the goal in a reinforcement learning problem. Roughly speaking, it maps each perceived state (or state-action pair) of the environment to a single number, a *reward*, indicating the intrinsic desirability of that state. A reinforcement learning agent's sole objective is to maximize the total reward it receives in the long run. The reward function must necessarily be unalterable by the agent. It may, however, serve as a basis for altering the policy. For example, if an action selected by the policy is followed by low reward, then the policy may be changed to select some other action in that situation in the future. In general, reward functions may be stochastic.

Whereas a reward function indicates what is good in an immediate sense, a *value function* specifies what is good in the long run. Roughly speaking, the *value* of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state. Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the *long-term* desirability of states after taking into account the states that are likely to follow, and the rewards available in those states. For example, a state might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards. Or the reverse could be true.

Rewards are in a sense primary, whereas values, as predictions of rewards, are secondary. Without rewards there could be no values, and the only purpose of estimating

values is to achieve more reward. Nevertheless, it is values with which we are most concerned when making and evaluating decisions. Action choices are made based on value judgments. We seek actions that bring about states of highest value, not highest reward, because these actions obtain the greatest amount of reward for us over the long run. In decision-making and planning, the derived quantity called value is the one with which we are most concerned. Unfortunately, it is much harder to determine values than it is to determine rewards. Rewards are basically given directly by the environment, but values must be estimated and reestimated from the sequences of observations an agent makes over its entire lifetime. In fact, the most important component of almost all reinforcement learning algorithms is a method for efficiently estimating values. The central role of value estimation is arguably the most important thing we have learned about reinforcement learning over the last few decades.

The fourth and final element of some reinforcement learning systems is a *model* of the environment. This is something that mimics the behavior of the environment. For example, given a state and action, the model might predict the resultant next state and next reward. Models are used for *planning*, by which we mean any way of deciding on a course of action by considering possible future situations before they are actually experienced. The incorporation of models and planning into reinforcement learning systems is a relatively new development. Early reinforcement learning systems were explicitly trial-and-error learners; what they did was viewed as almost the *opposite* of planning. Nevertheless, it gradually became clear that reinforcement learning methods are closely related to dynamic programming methods, which do use models, and that they in turn are closely related to state-space planning methods.

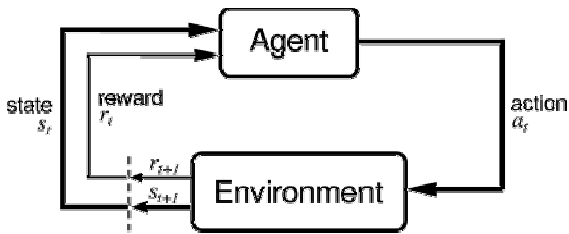
The reinforcement learning problem

The Agent-Environment Interface

The reinforcement learning problem is meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The learner and decision-maker is called the *agent*. The thing it interacts with, comprising everything outside the agent, is called the *environment*. These interact continually, the agent selecting actions and the environment responding to those actions and

presenting new situations to the agent. The environment also gives rise to rewards, special numerical values that the agent tries to maximize over time. A complete specification of an environment defines a *task*, one instance of the reinforcement learning problem.

More specifically, the agent and environment interact at each of a sequence of discrete time steps, $t=1,2,\dots,n$. At each time step t , the agent receives some representation of the environment's *state*, $s_t \in \mathcal{S}$, where $\mathcal{S}$ is the set of possible states, and on that basis selects an *action*, $a_t \in \mathcal{A}(s_t)$, where $\mathcal{A}(s_t)$ is the set of actions available in state s_t . One time step later, in part as a consequence of its action, the agent receives a numerical *reward*, $r_{t+1} \in \mathcal{R}$, and finds itself in a new state, s_{t+1} . Figure below diagrams the agent-environment interaction.



The agent-environment interaction in reinforcement learning.

At each time step, the agent implements a mapping from states to probabilities of selecting each possible action. This mapping is called the agent's *policy* and is denoted π_t , where $\pi_t(s, a)$ is the probability that $U_t = a$ if $s_t = s$. Reinforcement learning methods specify how the agent changes its policy as a result of its experience. The agent's goal, roughly speaking, is to maximize the total amount of reward it receives over the long run.

This framework is abstract and flexible and can be applied to many different problems in many different ways. For example, the time steps need not refer to fixed intervals of real time; they can refer to arbitrary successive stages of decision-making and acting. The actions can be low-level controls, such as the voltages applied to the motors of a robot arm, or high-level decisions, such as whether or not to have lunch or to go to graduate school. Similarly, the states can take a wide variety of forms. They can be completely determined by low-level sensations, such as direct sensor readings, or they can be more high-level and abstract, such as symbolic descriptions of objects in a room. Some of what makes up a state could be based on memory of past sensations or even be entirely mental or subjective. For example, an agent could be in "the state" of not being sure where an object is, or of having just been "surprised" in some clearly defined sense. Similarly, some actions might be totally mental or computational. For example, some actions might control what an agent chooses to think about, or where it focuses its attention. In general, actions can be any decisions we want to learn how to make, and the states can be anything we can know that might be useful in making them.

