Instituto Superior de Engenharia do Porto

ADAPTIVE LEARNING IN AGENTS BEHAVIOUR: A FRAMEWORK FOR ELECTRICITY MARKETS SIMULATION

Tiago Manuel Campelos Ferreira Pinto

Dissertation to obtain the Master of Science degree in
Computer Science

Specialization in
Knowledge-based and Decision Support Technologies

October, 2011

«À minha avó»

«Mas com buscar com o seu forçoso braço

As honras, que ele chame próprias suas;

Vigiando, e vestindo o forjado aço,

Sofrendo tempestades e ondas cruas;

Vencendo os torpes frios no regaço

Do Sul e regiões de abrigo nuas;

Engolindo o corrupto mantimento,

Temperado com um árduo sofrimento;»

Luís de Camões

Os Lusíadas VI-97

ACKNOWLEDGEMENTS

My first and most owed and sincere Thank You has to go to Dr. Zita Vale, which has been much more than a simple supervisor throughout this work. An enormous part of what I have accomplished is due to her. The lead example and posture even in the most little things that for many may pass unnoticed make her the person I most came to admire; and achieving her expectations has become one of my higher sources of motivation. This without even considering the direct contribution to the enrichment of this work with her recurrent wise advices, constant one-step-ahead ideas, and endless work endurance, unstoppable while the result is not perfect. All of this is just an infinitesimal part of what makes her the personification of excellence.

I would also like to acknowledge Dr. Fátima Rodrigues, which has contributed to success of this work with her large expertise in the area, and has always been present throughout the development of this work, following it closely, making sure the most appropriate decisions were taken.

An equally important thanks goes to the GECAD family, an incomparable workplace filled with friends rather than co-workers. Although I would not like to make particular distinctions, I must acknowledge the contribution and full availability of Tiago Miguel concerning the development of this work, with his fresh ideas, work spirit and personal friendship; and Tiago Sousa’s and Pedro Faria’s companionship and friendship throughout the many hours spent together working and helping each other in our particular dissertations. I would also like to thank Marco Silva for his availability and initiative in helping mine and others issues, while many of those times risking to compromise the finishing of his own work. To Hugo Morais goes my following acknowledgement, for his contribution with his incomparable knowledge concerning electricity markets. At last, the final persons from GECAD to whom I owe a personal thanks are Joana Neves, for her precious help in reviewing this dissertation document; and Joaquim Teixeira for his well known expertise in managing the GECAD’s computer science related technical issues.

Additionally, I would like to thank my closer family, particularly my grandmother Adélia, my brother João, and my mother Maria João, for their support and comprehension for the absent times spent working in this dissertation. The same applies to my friends, which had the good fortune of not having me around so often during the time of this work’s development. I would particularly like to thank Francisco for our discussions on many themes that revolve around this work’s scope, and that gave birth to some of the ideas that were applied here. I would also like to thank him and Cláudio for their big help in reviewing this dissertation. My final acknowledgement is for Mariana, which contributed with her expertise in design, making this document more appellative in terms of formatting.

ABSTRACT

Electricity markets are complex environments with very particular characteristics. A critical issue regarding these specific characteristics concerns the constant changes they are subject to. This is a result of the electricity markets' restructuring, which was performed so that the competitiveness could be increased, but it also had exponential implications in the increase of the complexity and unpredictability in those markets scope.

The constant growth in markets unpredictability resulted in an amplified need for market intervenient entities in foreseeing market behaviour. The need for understanding the market mechanisms and how the involved players' interaction affects the outcomes of the markets, contributed to the growth of usage of simulation tools. Multi-agent based software is particularly well fitted to analyze dynamic and adaptive systems with complex interactions among its constituents, such as electricity markets.

This dissertation presents ALBidS – Adaptive Learning strategic Bidding System, a multiagent system created to provide decision support to market negotiating players. This system is integrated with the MASCEM electricity market simulator, so that its advantage in supporting a market player can be tested using cases based on real markets' data.

ALBidS considers several different methodologies based on very distinct approaches, to provide alternative suggestions of which are the best actions for the supported player to perform. The approach chosen as the players' actual action is selected by the employment of reinforcement learning algorithms, which for each different situation, simulation circumstances and context, decides which proposed action is the one with higher possibility of achieving the most success.

Some of the considered approaches are supported by a mechanism that creates profiles of competitor players. These profiles are built accordingly to their observed past actions and reactions when faced with specific situations, such as success and failure.

The system's context awareness and simulation circumstances analysis, both in terms of results performance and execution time adaptation, are complementary mechanisms, which endow ALBidS with further adaptation and learning capabilities.

Keywords:

Adaptive Learning; Artificial Intelligence; Decision Support Systems; Electricity Markets; Multiagent Simulation

RESUMO

Os mercados de electricidade sofreram um processo de reestruturação que originou um aumento considerável da competitividade neste sector e, consequentemente, criou novos desafios na operação das entidades nele envolvidas. De forma a ultrapassar estes desafios é essencial para os profissionais uma compreensão detalhada dos princípios destes mercados e de como gerir os seus investimentos num ambiente tão dinâmico e competitivo. A crescente necessidade de entender estes mecanismos e a forma como a interacção das entidades envolvidas afecta os resultados destes mercados levou a uma grande procura de ferramentas de software, nomeadamente simulação, para analisar possíveis resultados de cada contexto de mercado para as várias entidades participantes. Os sistemas multi-agente são adequados à análise de sistemas dinâmicos e adaptativos com interacções complexas entre os seus constituintes, e portanto, várias ferramentas de modelação dirigidas para o estudo dos mercados reestruturados de electricidade usam este tipo de técnicas.

Tirando partido destes simuladores, é possível estudar vários tipos de mercados e a interacção entre as entidades neles envolvidas. No entanto, todos estes simuladores apresentam lacunas no que diz respeito ao apoio à decisão a essas entidades, nomeadamente na gestão dos seus investimentos. Um aspecto tão relevante como é a utilização de todo este suporte de simulação para permitir aos agentes de mercado realmente aprenderem com a experiência de mercado e desenvolveram capacidades para analisar contextos de negociação e adaptar automaticamente os seus comportamentos estratégicos de acordo com as circunstâncias, não é considerado na amplitude que é requerida. É neste âmbito que esta dissertação contribui, utilizando técnicas de inteligência artificial para oferecer um apoio relevante e eficaz às decisões estratégicas das empresas envolvidas nestes tipos de negociação.

O principal objectivo deste trabalho é dotar essas entidades de capacidades que lhes permitam apresentar comportamentos inteligentes e adaptativos na sua actuação nos mercados de electricidade de forma a serem capazes de atingir os seus objectivos da melhor forma possível, sendo capazes de reconhecer e actuar em conformidade com os contextos em que estão inseridas.

De forma a atingir este objectivo, foi desenvolvido o sistema ALBidS – *Adaptive Learning strategic Bidding System* (sistema de aprendizagem adaptativa para licitações estratégias). Este sistema está implementado como um sistema multi-agente independente, em que cada agente é responsável pela execução de uma abordagem estratégica diferente. Este sistema está integrado com o simulador MASCEM, para que seja possível testar e validar as contribuições dadas num contexto de simulação de mercados já implementado e consolidado. Sendo este simulador uma ferramenta que simula mercados de electricidade permitindo a utilização de informação obtida a partir de mercados de electricidade reais, garante-se, assim, também que as conclusões retiradas deste trabalho são apoiadas por experimentação baseada em casos reais ou quase reais.

A definição das estratégias de oferta dos agentes de mercado é baseada na aprendizagem adaptativa por parte das entidades, considerando o histórico do sistema, através da informação disponível, incluindo informação recolhida durante a utilização do próprio sistema multi-agente. Para isso são propostos e testados vários algoritmos e metodologias de aprendizagem e análise de dados, para que conjuntamente contribuam para que os agentes possam tomar as melhores decisões em cada momento de acordo com o contexto identificado.

Um contributo importante do trabalho está na proposta destes algoritmos, na sua combinação e na obtenção de conhecimento relativo à utilização criteriosa dos algoritmos considerados em função do contexto, utilizando o conceito de *context awareness*. A análise destes contextos é efectuada por um mecanismo desenvolvido para esse efeito, analisando as características específicas de cada dia e período de negociação.

São estudados e analisados vários algoritmos baseados em abordagens diversas, para que seja possível contemplar formas distintas de resolver problemas, dependendo de circunstâncias concretas. Entre estas abordagens, podem referir-se: redes neuronais artificiais dinâmicas; teoria de jogos; médias/regressões lineares; abordagens económicas, tendo em conta a análise macroeconómica e sectorial, e também a análise interna das empresas no que diz respeito aos seus investimentos e perspectivas de crescimento; algoritmos de Inteligência Artificial (IA), como os algoritmos *Roth-Erev* e o *Q-Learning* de aprendizagem por reforço; uma abordagem baseada na teoria do determinismo, em que são analisadas todas as variáveis intervenientes na obtenção dos resultados pelo simulador; e outras propostas de algoritmos de aprendizagem e análise de dados específicos para determinadas situações, bem como a combinação de algoritmos de tipos diversos.

Numa camada superior aos algoritmos mencionados foi implementado um mecanismo de aprendizagem por reforço, baseado em estatísticas e em probabilidades, que é responsável por escolher em cada altura a proposta de licitação que dá mais garantias de sucesso. Com o passar do tempo, vão sendo actualizadas as estatísticas, através da análise dos resultados de cada proposta. Este mecanismo permite que em cada momento sejam escolhidos os algoritmos que estão a ter os melhores resultados para cada situação e contexto. Ao serem considerados vários algoritmos, de naturezas completamente distintas, consegue-se uma maior probabilidade de haver sempre algum a oferecer bons resultados.

Existe também a possibilidade de se definir as preferências e parametrizações relativas a cada algoritmo individualmente, e também de se definirem preferências relativas ao desempenho dos algoritmos no que diz respeito à eficiência computacional, permitindo que o utilizador escolha a relação eficiência/probabilidade de sucesso, de acordo com as suas preferências. O sistema excluirá então, automaticamente, os algoritmos que usualmente requerem um maior tempo de processamento, quando esse tempo não corresponde a soluções proporcionalmente melhores. Desta forma, garante-se que o sistema estará a utilizar o seu tempo de processamento em abordagens que oferecem melhores respostas no menor tempo possível.

Como apoio ao funcionamento adequado das estratégias implementadas foi criado um mecanismo de definição de perfis dos agentes competidores. Desta forma é possível obter previsões acerca das acções esperadas dos outros agentes participantes no mercado, tendo em conta as suas acções passadas e as reacções verificadas quando confrontados com situações específicas, como o sucesso ou o falhanço.

Palavras-chave:

Aprendizagem Adaptativa; Inteligência Artificial; Mercados de Energia Eléctrica; Simulação Multi-Agente; Sistemas de Apoio à Decisão

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

2E	**E**fficiency/**E**ffectiveness
AI	**A**rtificial **I**ntelligence
ALBidS	**A**daptive **L**earning strategic **Bid**ding **S**ystem
AMES	**A**gent-based **M**odelling of **E**lectricity **S**ystems
API	**A**pplication **P**rogramming **I**nterface
BNJ3	**B**ayesian **N**etwork tools in **J**ava **3**
DLL	**D**ynamic-**L**ink **L**ibrary
EMCAS	**E**lectricity **M**arket **C**omplex **A**daptive **S**ystem
FCT	Science and Technology Fundation (***F****undação para a* ***C****iência e a* ***T****ecnologia*)
FERC	**F**ederal **E**nergy **R**egulatory **C**ommission
GECAD	Knowledge Engineering and Decision Support Research Centre (***G****rupo de investigação em* ***E****ngenharia do* ***C****onhecimento e* ***A****poio à* ***D****ecisão*)
ICL	**I**nteragent **C**ommunication **L**anguage
IS	**I**ntelligent **S**erver
LPA	**L**ogic **P**rogramming **A**ssociates
MAPE	**M**ean **A**bsolute **P**ercentage **E**rror
MAS	**M**ulti-**A**gent **S**ystem
MASCEM	**M**ulti-**A**gent **S**imulator for **C**ompetitive **E**lectricity **M**arkets
MATLAB	**Mat**rix **Lab**oratory
MSE	**M**ean **S**quare **E**rror
MVC	**M**odel-**V**iew-**C**ontroller
NN	**N**eural **N**etwork
OAA	**O**pen **A**gent **A**rchitecture
OMEL	Iberian Electricity Market Operator *(****O****perador del* ***M****ercado* ***I****bérico de* ***E****nergia)*
PROLOG	***Pro****grammation en* ***Log****ique*
PSO	**P**article **S**warm **O**ptimization
QL	**Q**-**L**earning
RE	**R**oth-**E**rev
RLA	**R**einforcement **L**earning **A**lgorithm
SEPIA	**S**imulator for **E**lectric **P**ower **I**ndustry **A**gents
SREMS	**S**hort–medium **R**un **E**lectricity **M**arket **S**imulator
SVM	**S**upport **V**ector **M**achine
SWOT	**S**trengths-**W**eaknesses-**O**pportunities-**T**hreats
TS	**T**abu **S**earch
US	**U**nited **S**tates of **A**merica
VPP	**V**irtual **P**ower **P**layer
WPMP	**W**holesale **P**ower **M**arket **P**latform

NOMENCLATURE

A	Action
a	First cost coefficient
AD	Offers action domain
a_i^R	Reported first cost coefficient
amount	Price change amount (cent.€/kWh)
b	Second cost coefficient
b_i^R	Reported second cost coefficient
C	Confidence Value
C1	Social factor
C2	Cognitive factor
capacity_available	Available production capacity (MW)
Cap_i^L	producer i’s reported lower real-power production limit (MW)
Cap_i^L	True lower production limit for producer i (MW)
Cap_i^U	producer i’s reported upper real-power production limit (MW)
Cap_i^U	True upper production limit for producer i (MW)
C_i	Cooling parameter
$Cost_i$	Player i costs (€)
D	Day
Decrement	Risk decrement
degree	Weight of a learning case
Dif_i	Difference between a strategy’s prediction and the actual market price (cent.€/kWh)
E	Observed evidence
E_i	Energy amount (MW)
e_i	Experimentation parameter
energy_sold	Sold amount of power (MW)
exper	Experience
exper’	New experience
FC	Fixed costs (€)
G_b	Best global position of a particle
int	Interval
learningRate	Learning rate
M	Offers action domain cardinality
m’	Selected action
MC_i	Marginal cost function of producer i (€)
MCi	Unavailability or forced outage rate
min	Minimum considered bid value (cent.€/kWh)
n	Number of plays
New Decrement	Updated risk decrement

newValue	Updated value of the Q-Learning function
np	Number of possible bided prices
numb	Number of considered possible bids
number_of_bids	Number of bids
number_of_scenarios	Number of scenarios
O	Market offer
O_i	Possible Outcomes
oldValue	Q-Learning function’s old value
P	Bid price (cent.€/kWh)
p	Producer’s actual production (MW)
$P(O_i\|E,A)$	Conditional probability
$P_{0\ldots n}$	Possible bids
$p_{1\ldots n}$	Weighted average weights
P_b	Best individual position of a particle
$p_{im}(D)$	Probability of producer i choosing action m in day D
P_{market}	Market price (cent.€/kWh)
price	Price (cent.€/kWh)
probc	Probability
probc’	New probability
$Profit_{im'}(D)$	Profits attained by producer i in day D, in response to its choice of supply offer m’ (€)
Profits	Obtained Profits (€)
Q	Q-Learning function
q_{im}	Propensity of generator i for action m
r	Reward or cost
R1	Random number between 0 and 1
R2	Random number between 0 and 1
$RCap^L$	Reported lower production limit (MW)
$RCap^U$	Reported upper production limit (MW)
recency	Experimentation parameter
ref	Reference point
Reinf	Reinforcement amount
reinfValue	Value of the reinforcement
r_i	Recency parameter
$RIMax_i^L$	Lower range-index parameter for ADi construction
$RIMax_i^U$	Upper range-index parameter for ADi construction
$RIMin_i^C$	Capacity range-index parameter for ADi construction
Risk_factor	Risk assumed by the player
s	State
S	Reported supply offer
SS_i	Slope-start control parameter for ADi construction

t	Time
T_i	Temperature for generator i
Total Income	Total achieved income (MW)
U	Utility of each of the outcome states
V	Velocity
W	Weight of a particle’s inertia
X	Particle’s current position
x1...6	Fuzzy interval definition values
$x_{1...n}$	Strategy’s outputs (cent.€/kWh)
y1...6	Fuzzy interval definition values
α	Learning rate
α_i	Admissible offer for producer i
β	Scaling factor
λ	Temperature decreasing rate
ξ	Random number
σ	Range
γ	Discount factor

CHAPTER 1. INTRODUCTION

1.1. Motivation

The study of electricity markets operation has been gaining an increasing importance in the last years, as result of the new challenges that the electricity markets restructuring produced. This restructuring increased the competitiveness of the market, but with it its complexity. The growing complexity and unpredictability of the market's evolution consequently increases the decision making difficulty. Therefore, the intervenient entities are forced to rethink their behaviour and market strategies [Meeus *et al*., 2005].

In order to overcome the mentioned challenges, several market models have arisen. Some pioneer countries' experience provides some guidance in what regards the implemented market models' performance. However, it is still premature to take definitive conclusions [Shahidehpour *et al*., 2002]. Therefore, the use of tools allowing the study of different market mechanisms and the relationship between market entities becomes crucial.

Market players and regulators are very interested in foreseeing market behaviour: the regulators to test rules before they are implemented and to detect market inefficiencies; the market players to understand market's behaviour and operate in order to maximize their profits. The need for understanding those mechanisms and how the involved players' interaction affects the outcomes of the markets, contributed to the growth of usage of simulation tools, with the purpose of taking the best possible results out of each market context for each participating entity. Multi-agent based software is particularly well fitted to analyze dynamic and adaptive systems with complex interactions among its constituents, such as the electricity markets. Several modelling tools directed to the study of restructured wholesale power markets have emerged. Some relevant tools in this domain are EMCAS [Koritarov, 2004], AMES [Li and Tesfatsion, 2009], and MASCEM [Praça *et al.*, 2003], [Vale *et al.*, 2011a].

Although some works have arisen, which confirm the adequate applicability of simulation to the study of electricity markets, particularly by using multiagent systems, they present a common limitation: the lack of adaptive machine learning capabilities that allow these tools to effectively provide measurable support to market entities. Current tools are directed to the study of different electricity market mechanisms and to the analysis of the relationships between market entities, but they are not fitted to provide support to market negotiating players in what concerns their achievement of higher profits in power transactions.

These limitations point out the necessity for developing adaptive tools, able to provide effective support to market negotiating players. Such tools would provide the means for an actual improvement in players' results. By using intelligent tools, capable of adapting to different market circumstances and negotiating contexts, players can collect real and adequate suggestions on the optimal directions their actions should take, so that they can change their behaviour in a real market environment, and therefore pursuit the achievement of the best possible outcomes.

1.2. Objectives

Given that software applications directed to dynamic environments such as the electricity markets are especially relevant, it is essential to provide breakthroughs in what concerns the employment of learning abilities in multiagent systems, applied to such environments.

With this work a contribution in this area is projected, by conceiving a system that is capable of learning the best agent acting approaches, depending on each situation and context. This learning process must take into account the system's log, taking advantage of all the available information, including collected data during the use of the multiagent system itself. In order to achieve this purpose, several algorithms and learning methodologies must be proposed and tested, so that together they can contribute to the best decision making in each moment. It is expected that the algorithms' proposal provides an important contribution, including the algorithms' combination and knowledge collecting in what regards their criterious usage depending on the context, using the concept of context awareness.

The nature of the considered algorithms and approaches shall consider distinct areas, such as data mining techniques, data analysis and forecasting tools, pattern recognition, knowledge engineering approaches, artificial intelligence algorithms, and also the proposal of other algorithms directed to specific situations, such as the application of economic, mathematical, physics and psychology theories. The adequate combination of different types of algorithms is also considered relevant.

The considered approaches must be chosen in each moment according to the guarantees of success they offer in each context. The choosing process shall be performed through the application of algorithms driven by statistics and probabilities management. Through the consideration of several algorithms, based on completely different natures, a higher probability of at least one approach offering good results in each distinct situation and context is expected to be achieved.

This work is essentially directed to the definition of bidding strategies of market negotiating players. Given that the conclusions to be taken from this work are intended to be supported by experimentation based on real or almost real cases, the work to be developed must be integrated with the MASCEM simulator. This simulator allows the study of electricity markets, providing the construction of cases based on information from real electricity markets.

Taking into account the referred specifications, the following objectives are considered:

- Development of a multiagent decision support tool directed to the strategic behaviour of market negotiating players, including:
    - Learning of the most adequate acting approaches, depending on each situation and the context it occurs, considering the system's log and the historic of past actions of the other agents considered in the system;
    - Proposal and testing of several learning and data analysis algorithms and methodologies, so that they can contribute together to the best decision making of the supported market players, including the development of algorithms directed to specific situations;

  - Implementation of a learning mechanism with the capability of choosing the most appropriate strategies at each moment, based on the methodologies performance statistics and probabilities; and the respective capability of calculating the disparity between each algorithm’s proposals and the actual reality, so that it can properly update the confidence values of the algorithms’ performances for each situation and context.

- Simulation of the intelligent action of players in electricity markets, concerning the following:
  - Integration of the developed system with the MASCEM electricity market simulator;
  - Utilization of the developed system to analyze the results of scenarios based on real electricity markets’ data.

1.3. Outline and Main Contributions

With the purpose of accomplishing the proposed objectives, while providing the negotiating players with competitive advantage in electricity markets it is essential to endow them with strategies capable of dealing with the constant market changes, allowing adaptation to the competitors’ actions and reactions. For that, a new system has been created to integrate several distinct technologies and approaches. This system is named ALBidS – Adaptive Learning strategic Bidding System [Pinto *et al.*, 2011a], and it is implemented as a multiagent system (this system is presented in Chapter 3). There is one agent performing each distinct algorithm, detaining the exclusive knowledge of its execution. This way the system can be executing all the algorithms in parallel, preventing the system’s performance degradation, in the possible amount. As each strategy agent gets its answer, it sends it to the Main Agent (presented in section 3.6, along with the tools that support this agent’s performance), which is responsible for choosing the most appropriate answer among all that it receives, depending on each context [Vale *et al.*, 2011a]. Figure 1.1 presents the global structure of the ALBidS system, with the Main Agent as its central entity.

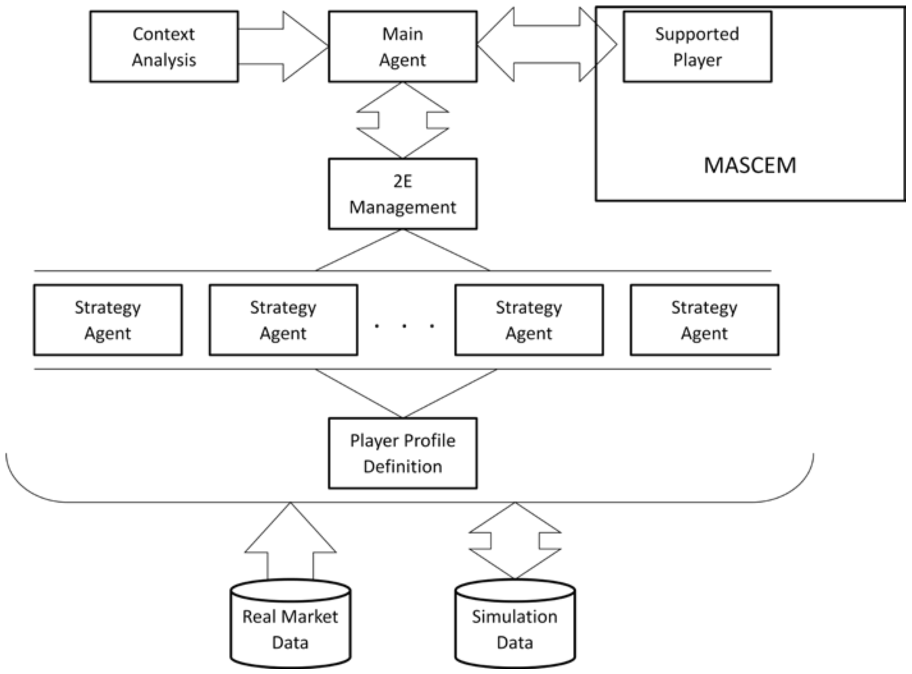


Figure 1.1 – ALBidS global structure.

Contexts are an important factor in what concerns the adaptation of the approaches to be chosen as the final action to be performed in the market by the supported player. A mechanism to analyse and define different market negotiating contexts is presented in section 3.5, hence providing the means for the chosen actions to be adapted and chosen depending of the different circumstances that are encountered at each moment.

The ALBidS system integrates several very distinct approaches in what concerns market negotiations, and prices forecast. These approaches, implemented and tested independently before their integration in ALBidS are presented in section 3.7 – Strategy Agents. Each of these strategies is introduced with an overview on the context of each approach. Afterwards, the details on these approaches implementations are described and the necessary experimental findings are presented, in order to test each approach's adequate performance, and prove the adequacy to its purpose [Pinto *et al.*, 2011b].

In order to support some of these approaches, a competitor player's profile definition mechanism has been implemented, with the point of creating adequate player profiles to be used by the ALBidS strategies which require such profiles for their execution [Pinto *et al.*, 2011c]. This mechanism is presented in section 3.8.

Additionally, a mechanism with the purpose of managing the balance between the efficiency and the effectiveness of the system has been developed (presented in section 3.9), so that the ALBidS system can be able to adapt to different simulation circumstances. This mechanism provides the means for the system to adapt its execution time depending on the purpose of the simulation, *i.e.*, if the expected results from ALBidS are as best as it is able to achieve, or, on the other hand, if the main requirement is for the system to be executed rapidly, since the purpose of the considered simulation is to analyse issues other than player's optimal performance in the electricity market [Pinto *et al.*, 2011d]. The Efficiency/Effectiveness Management mechanism manipulates the strategies both externally and internally. From the system's perspective this mechanism contributes by deciding which tools are used at each moment for each circumstance; depending on their observed performance in terms of efficiency and effectiveness. This way this mechanism can choose to exclude certain strategies when they are not fulfilling the ALBidS' requirements for the case in matter. The strategies chosen to be executed are also manipulated internally, so that they can adapt their individual results quality/execution time balance to the needs of the current simulation.

The details regarding ALBidS' internal design, its multiagent architecture, and its integration with MASCEM [Praça *et al.*, 2003], are presented in section 3.2. This section is supported and extended by sections 3.3 and 3.4, which respectively present the ALBidS facilitator built to support this system's agents communications independently of the rest of the simulation [Pinto *et al.*, 2011e]; and the ALBidS user interface which supports the interaction with the user, and the respective feedback from the system, regarding the presentation of the events that occur throughout a simulation, and consequent results. Figure 1.2 presents the ALBidS' integration with the MASCEM simulator.

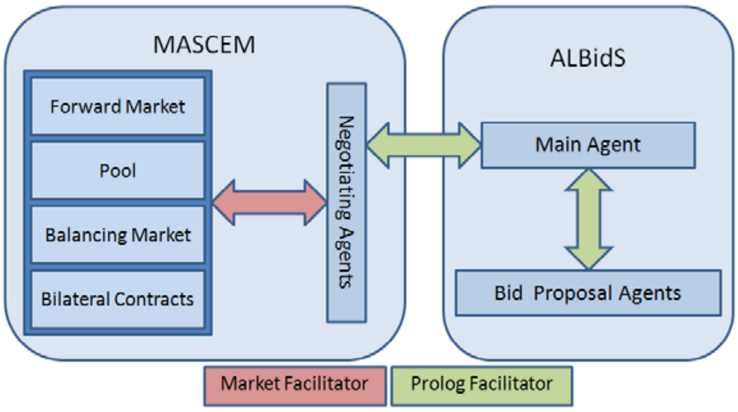


Figure 1.2 – ALBidS integration with MASCEM [Pinto *et al.*, 2011c].

Although all the proposed and developed work is tested individually, with the test results being presented in each respective section; the ALBidS system's performance in supporting a player in the market under different scenarios, using real electricity markets' data is tested and the results analysed in Chapter 4.

The work developed in the scope of this dissertation was supported by several projects funded by FCT *"Fundação para a Ciência e a Tecnologia"*, under the scope of the Knowledge Engineering and Decision Support research Centre – GECAD. The regarded projects are:

- ViP-DiGEM - VIrtual power Producers and DIstributed Generation trading in Energy Markets (PTDC/ 72889);
- ID-MAP - Intelligent Decision Support for Electricity Market Players (PTDC/EEA-EEL/099832/2008);
- FIGURE – Flexible and Intelligent Grids for Intensive Use of Renewable Energy Sources (PTDC/SEN-ENR/099844/2008).

Additionally, throughout the development of this work, several scientific papers were published, concerning the scientific advances achieved by the developed work:

Two scientific papers in SCI[1] **journals**:

- Zita Vale, Tiago Pinto, Isabel Praça, Hugo Morais, "MASCEM - Electricity markets simulation with strategic agents", in *IEEE Intelligent Systems*, vol. 26, n. 2, 2011 (presents an impact factor of 2.570, referring to the year 2010) [Vale *et al.*, 2011a]. This paper presents an early version of the ALBidS system, regarding only some of the final version strategies and mechanisms;
- Tiago Pinto, Hugo Morais, Pedro Oliveira, Zita Vale, Isabel Praça, Carlos Ramos, "A new approach for Multi-Agent coalition formation and management in the scope of Electricity Markets", in *Energy*, vol. 36, n. 8, pp. 5004-5015, August 2011 (presents an impact factor of 3.565, referring to the year 2010) [Pinto *et al.*, 2011e]. This paper presents the development of the ALBidS facilitator presented in

[1] *Science Citation Index® (SCI®); http://thomsonreuters.com/products_services/science/science_products/a-z/science_citation_index/*

section 3.3, the application of the neural network presented in sub-section 3.7.2, and the average and regression strategies presented in sub-sections 3.7.1.4 and 3.7.1.5.

Four papers in **book chapters**:

- Tiago Pinto, Zita Vale, Fátima Rodrigues, Hugo Morais, Isabel Praça, "Bid Definition Method for Electricity Markets Based on an Adaptive Multiagent System", in *Advances on Practical Applications of Agents and Multiagent Systems*, Springer Berlin / Heidelberg, vol. 88, pp. 309-316, 2011 [Pinto *et al.*, 2011d]. This paper is directed to the Efficiency/Effectiveness Management mechanism, presented in section 3.9 of this dissertation;
- Tiago Pinto, Zita Vale, Fátima Rodrigues, Hugo Morais, Isabel Praça, "Strategic Bidding Methodology for Electricity Markets using Adaptive Learning", in *Modern Approaches in Applied Intelligence*, vol. 6704, pp. 490-500, 2011 [Pinto *et al.*, 2011c]. This paper presents the Player Profile Definition mechanism, which is described in section 3.8;
- Tiago Pinto, Hugo Morais, Zita Vale, Isabel Praça, "Multi-Agent Negotiation for Coalition Formation and Management in Electricity Markets", in *Negotiation and Argumentation in MAS*, Bentham Science Publishers Ltd., 2011 [Pinto *et al.*, 2011f]. This paper presents the use of ALBidS in providing decision support to market negotiations of Virtual Power Players;
- Zita Vale, Hugo Morais, Tiago Pinto, Isabel Praça, Carlos Ramos, "Electricity Markets Simulation: MASCEM contributions to the challenging reality", in *Handbook of Networks in Power Systems*, Springer-Verlag, 2011 [Vale *et al.*, 2011b]. This paper presents, among other things, the development of the ALBidS facilitator presented in section 3.3, the application of the neural network presented in sub-section 3.7.2, and the average and regression strategies, presented in sub-sections 3.7.1.4 and 3.7.1.5.

Five scientific papers in top ISI[2] **conferences** of the research area:

- Tiago Pinto, Zita Vale, Fátima Rodrigues, Isabel Praça, Hugo Morais, "Multiagent System for Adaptive Strategy Formulation in Electricity Markets", in *IEEE Symposium Series on Computational Intelligence - IEEE Symposium on Intelligent Agents*, SSCI-IA 2011 [Pinto *et al.*, 2011a]. This paper presents a global view over the ALBidS system in an intermediary state of this system's development;
- Tiago Pinto, Zita Vale, Fátima Rodrigues, Isabel Praça, Hugo Morais, "Cost Dependent Strategy for Electricity Markets Bidding Based on Adaptive Reinforcement Learning", in *International Conference on Intelligent System Application on Power Systems*, ISAP 2011 [Pinto *et al.*, 2011b]. This paper presents the AMES strategy, described in sub-section 3.7.3 of this thesis;
- Gabriel Santos, Tiago Pinto, Hugo Morais, Isabel Praça, Zita Vale, "Complex Market integration in MASCEM electricity market simulator", in *International Conference on the European Energy Market 11*, EEM 2011 [Santos *et al.*, 2011]. This paper presents the application of ALBidS to support player negotiations in the Complex Market, a market type presented in sub-section 2.2.1.2;

[2] *Conference Proceedings Citation Index ®; http://thomsonreuters.com/products_services/science/science_products/a-z/conf_proceedings_citation_index/*

- Zita Vale, Tiago Pinto, Hugo Morais, Isabel Praça, "VPP's Multi-Level Negotiation in Smart Grids and Competitive Electricity Markets", in *IEEE PES General Meeting*, PES-GM 2011 [Vale *et al.*, 2011c]. This paper presents the use of ALBidS in providing decision support to Virtual Power Players in Smart Grids;
- Tiago Pinto, Tiago M. Sousa, Isabel Praça, Zita Vale, Fátima Rodrigues, Hugo Morais, "Adaptive Learning in Multi-Agent Systems: A Bidding Methodology Based on Forecasting Error Analysis" in *4th International Conference on Agents and Artificial Intelligence*, ICAART 2012. This paper was accepted to be published in February 2012, and presents the Error Theory strategy, described in sub-section 3.7.6.

Besides the mentioned papers, several others can be referred, which will soon be prepared for publishing:

- "Strategic Bidding in Electricity Markets: An Agent-based Simulator with Game Theory for Scenario Analysis", regarding the Game Theory strategy presented in sub-section 3.7.5;
- "An Adaptive Multi-Agent Decision Support Framework for Intelligent Bidding", presenting the Bayes Theorem based reinforcement learning algorithm, and its application to ALBidS;
- "ALBidS: Adaptive Learning for strategic Bidding System", presenting the ALBidS system as a whole, in its current version;
- "An Economic Approach to Support Electricity Market Players: Internal Data Analysis vs. Sectorial Analysis", concerning the Economic Analysis strategy presented in sub-section 3.7.7;
- "Application of the Determinism Theory to Optimize Electricity Market Players Strategic Bidding", presenting the Determinism Theory strategy, which is described in sub-section 3.7.8;
- "Negotiation Context Analysis in Electricity Markets", regarding the Context Analysis mechanism presented in section 3.5;
- "A new approach for Systems Efficiency/Effectiveness Balance Management", providing full details on the Efficiency/Effectiveness Management mechanism, presented in section 3.9;
- "Adaptive Learning in Agents Behaviour: A Mechanism for Agent Profile Definition", describing the Player Profile Definition mechanism, which is presented in section 3.8;
- "Optimizing the Performance of a Multiagent System", paper that provides some highlights on the performed developments that aimed at optimizing the performance of ALBidS, *e.g.* the implementation of the Determinism Theory's optimization algorithms in Prolog (see sub-section 3.7.8), or the decision of building an independent facilitator for managing this system's communications rather than using MASCEM's market facilitator (see section 3.3).

These scientific contributions support the relevance of the developed work to the scientific community, in several fields, such as: artificial intelligence, machine learning, multiagent systems, and intelligent power systems.

1.4. Document Structure

This document consists of five chapters. After this introductory chapter, Chapter 2 presents an overview of electricity markets simulation. This overview includes a discussion on the consequences of the electricity markets' restructuring, and its effects on these markets' participating entities. Regarding the electricity markets' structure, the typical regulatory models are described, with particular focus on the spot market - the main action scope of the developed work. Afterwards, an introduction to the multiagent systems theme is presented, which supports the ensuing approached issue – simulation and multiagent simulation of electricity markets. Here, the reasons for the increasing need of simulation tools in this scope are exposed, and some of the most important simulators for electricity markets, with special emphasis on those based on multiagent simulation, are presented. In this chapter the MASCEM simulator is also described with particular detail. The chapter is closed with an overview on the main regards of machine learning.

Chapter 3 presents the work developed in the scope of this dissertation. This chapter starts by describing ALBidS' global structure and its integration process with the MASCEM simulator. This includes the ALBidS multiagent model, and the facilitator that was developed to manage this system's agent communications. The ALBidS graphical interface and its main advantages in providing an adequate interaction with the user are also presented in this chapter. Afterwards, the core of ALBidS intelligence starts being described. Firstly, the context analysis mechanism is presented. This mechanism is used by the Main agent, the entity that is described next, together with the reinforcement learning mechanisms that it uses. Subsequently, the strategy agents are presented. These entities are the ones that perform the actual analysis of the market data, and execute the several approaches in market's negotiations. To finalize this chapter, two complementary mechanisms are described: the player profile definition and the efficiency/effectiveness management mechanisms.

Chapter 4 presents three case studies using the ALBidS system. These case studies are directed to illustrate the operation and testing of this system, by employing it to provide decision support to MASCEM's market negotiating players. These tests concern different market circumstances, which are conceived to demonstrate the ALBidS global performance, and that of some of its most important mechanisms.

This dissertation is completed with Chapter 5, which presents the most important conclusions that resulted from the developed work, and some suggestions of future work.

CHAPTER 2. ELECTRICITY MARKETS SIMULATION

2.1. Introduction

The electricity sector's restructuring process placed several challenges, demanding the transformation of the conceptual models that have previously dominated the sector [Shahidehpour *et al.*, 2002]. The restructuring made the market more competitive, but also more complex, placing new challenges to the participants. The growing complexity and unpredictability of the market's evolution consequently increases the decision making difficulty. Therefore, the intervenient entities are forced to rethink their behaviour and market strategies.

So that these entities can face the new challenges, it becomes crucial to use decision support tools, mainly those that are able to provide the means for the entities to analyse the market's evolution, and test alternative solutions, this way proving to be an important added value.

This chapter intends to provide answers to some matters, such as: which are the types of tools that can be useful for the electricity markets' entities to surpass the challenges that arose with the electricity markets' restructuring; in what way agents can be useful to the contribution of the electricity markets evolution's study; if are there solutions that support agents' applicability to this theme; what type of solutions for electricity markets' study exist.

For this reason, after the introduction to electricity markets, this chapter presents the theme of multiagent systems, so that the simulation based on this type of design can be properly exposed afterwards. In this exposure, the role that agents and multiagent simulation can perform in a decision support tool directed to the study of electricity markets, is analysed. Additionally, a description of the most important works that have arisen in this scope, and that support the potentialities that this type of technology presents in this area, is provided. This includes a detailed description of the MASCEM simulator, which is used to support the developed system in the scope of this work, providing the means for its testing in an electricity market's environment. Finally, an overview of the machine learning field is presented, complementing the lacking issue of most existing market simulators, and providing the basis for the adequate development of the ALBidS mechanisms.

2.2. Electricity Markets

All over the world the restructuring of the power sector placed several challenges to governments and to the companies that are involved in the area of generation, transmission, distribution and retail of electrical energy. One of the most significant changes is the introduction of electricity markets, aimed at providing competitive electricity service to consumers [Meeus *et al.*, 2005]. Potential benefits, however, depend on the efficient operation of the market. Definition of the market structure implies a set of complex rules and regulations that should not encourage strategic behaviours that might reduce market performance.

Electricity Markets are not only a new reality but an evolving one as the involved players and rules change at a relatively high rate [Ilic and Galiana, 1998]. The emergence of a diversity of new players (*e.g*. aggregators) and new ways of participating in the market (distributed energy resources and demand side are gaining a more active role) are signs of this [Hatziargyrious *et al.*, 2002].

The restructuring turned electricity markets into an attractive domain for developers of software tools. Simulation and Artificial Intelligence techniques may be very helpful under this context.

Real-world restructured electricity markets are sequential open-ended games with multiple participants trading electric power. Market players and regulators are very interested in foreseeing market behaviour: regulators to test rules before they are implemented and to detect market inefficiencies; market players to understand market's behaviour and operate in order to maximize their profits.

Electrical energy generation is a distributed problem by its own nature. Traditionally, electricity was based on a reduced number of plants (*e.g*. nuclear, other thermal and hydro power plants). However, guaranteeing sustainable development is an enormous challenge for Power Systems. This requires a significant increase in distributed generation, mainly based on renewable sources [Shahidehpour *et al*., 2002]. However, this leads to a system that is much more complex to control, since it includes many more power generation plants, and the generation is more unpredictable than before, due to the difficulty in forecasting the energy production originated by some renewable sources (*e.g*. wind and photovoltaic).

Electricity markets put a new emphasis on the economic dimension of the problem. However, the basic infrastructure, namely the power system network, has a real physical nature, with specific limitations [OMEL, 2011]. The introduction of electricity markets shows us the fragility of power systems infrastructures to operate in a competitive context. Several severe incidents, including blackouts, occurred (*e.g*. the 14th August 2003 Blackout in the US, and the 4th October 2006 quasi-blackout affecting nine European countries and some African nations as well).

2.2.1. *Regulatory Models*

The market environment typically consists of a pool, which may be symmetric or asymmetric, as well as a floor for bilateral contracts [EAC, 2009]. Additionally, balancing markets are also required. This implies that each market player must decide whether to, and how to, participate in each market type.

Besides the entities that negotiate in the market, namely electricity sellers and buyers, these types of market usually include also a market operator and a system operator. The market operator is the entity responsible for operating the market; it manages the pool using a market-clearing tool to establish the market price and a set of accepted selling and buying bids for every negotiation period. The system operator is usually responsible for the transmission grid management and for the technical constraints. Every established contract, either through bilateral contracts or through the pool, must be communicated to the system operator, who analyses its technical feasibility from the power system point of view.

2.2.1.1. Spot Market

The spot or day-ahead market [Sheblé, 1999], [Klemperer, 1999] is a daily basis functioning market, with the goal of negotiating electric power for each hour, or half an hour of the following day. This type of markets is structured to consider the daily production fluctuations, as well as the differentiated operation costs of production units.

One day is usually divided into 24 or 48 negotiation intervals [OMEL, 2011], referring to one hour or half hour periods. The participating entities must present their selling or buying proposals for each of those periods. These proposals, or bids, are typically characterized by (in symmetric pools, the most common type of market clearing mechanism): in the case of a seller, the amount of power to be sold, and the minimum selling price; in the case of a buyer, the amount of desired power, and the maximum accepted price.

When the negotiation is finished, the dispatch for each period is set by the market operator. This entity is responsible for the spot market functioning, initiating and controlling the full process. The market operator is also responsible for the definition of the market price, a unique price that is applied to all transactions.

Symmetric Pool

The symmetric pool is characterized by a competition between selling and buying entities. The price definition mechanism is usually based on a double auction.

In this type of pool, suppliers and consumers both submit bids. The market operator orders the selling and demand offers: selling bids start with the lowest price and move up, and demand bids start with the highest price and move down. Then, the proposed bids form the supply and demand step curves, and the point at which the two curves intersect determines the market price, paid to all accepted supplier and consumers. The bids of every supplier offering prices lower than the established market price and every consumer offering prices higher than the market price will be accepted.

Figure 2.1 shows the dispatch procedure of a symmetric market.

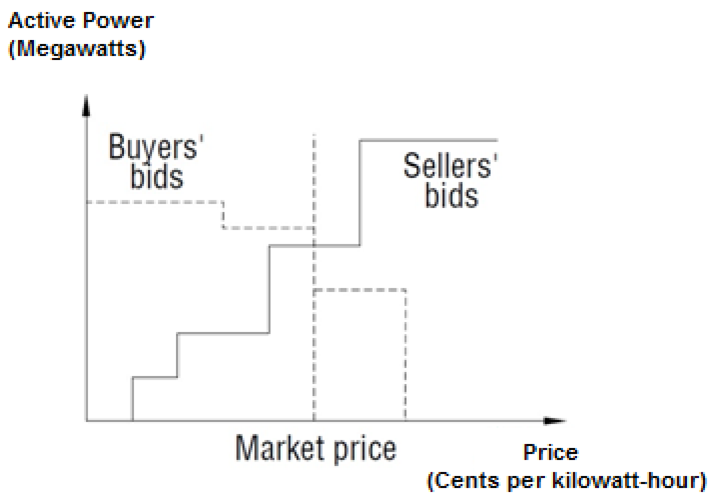


Figure 2.1 – Symmetric Pool, adapted from [Praça *et al.*, 2003].

Sellers and buyers submitting their bids in this type of markets is an indicator of the existence of behaviours with price sensitive consumptions. This observation is related to the dependence of this market's efficiency on the number of participating players, as well as on their proposals provision.

Asymmetric Pool

This type of pool allows the exclusive submission of selling proposals. In the asymmetric markets the buyer players do not present bids, rather only a demand estimative. This model assumes that buyers are able to pay any price which results from the market functioning, *i.e.*, the demand is considered inelastic.

In an asymmetric market, the sellers present their bids, and the market operator organizes them starting with the lowest price and moving up. The consumers reveal their needs to set up the demand. Once the market operator knows the demand, it accepts the suppliers' bids starting from the lowest and accepts as many as necessary to fill the demand. The market price - to be paid to all accepted suppliers - is that of the last accepted bid (the one with the highest price).

Figure 2.2 shows the dispatch procedure of a symmetric market.

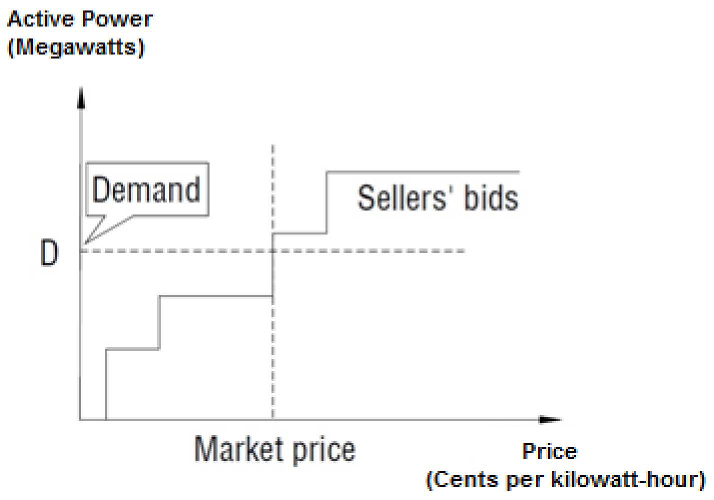


Figure 2.2 – Asymmetric Pool, adapted from [Praça *et al.*, 2003].

In this type of market the prices are highly influenced by the proposed selling prices and by the level of demand.

2.2.1.2. Complex Market

The complex market [OMEL, 2011], [Santos *et al.*, 2011] provides the opportunity for the presentation of restrictions that allow players to leave the market if they are not respected, meaning that players are not interested in participating unless those conditions are met. In addition, to meet the requirements of the simple day-ahead pool, complex market includes at least one of the following conditions: *Indivisibility, Charge Gradient, Minimum Income* and *Scheduled Stop*. Complex conditions are also used by market agents as strategies for achieving the highest possible profit.

The *Indivisibility* condition allows setting a minimum value of operation in the first offer of each period. Below this value, the participation of the production unit on the market is not possible. This condition applies to generating units that cannot work under a technical limit.

The *Charge Gradient* condition allows establishing the maximum difference between the initial and the final power, between periods, for a production unit. This allows avoiding abrupt changes between consecutive periods (resulting from technical impossibility of achieving such changes).

The *Minimum Income* condition is used to ensure that the production unit does not enter the market if it cannot obtain a minimum amount in Euros (€), in the total of all periods, plus a variable fee per transacted kWh. This restriction depends on the sales strategy of each agent.

The *Scheduled Stop* condition is used in situations when the production unit has been withdrawn for not meeting the condition of required Minimum Income. This condition ensures that the production stopping is not done abruptly, rather undertaking a scheduled stop in a maximum time of 3 hours, avoiding production to immediately decrease to zero, from the last period of one day to the first period of the next. This is done by accepting the first offer of the first three periods as a simple offer, with the sole condition that the offered power is decreasing in each period, to smooth the production decrease until it gets to zero.

The market operator must assure the economical dispatch taking into account the specified conditions, which may imply the renegotiation of the period or day in matter, depending on the possible removal of entities that have presented competitive bids but whose complex conditions were not satisfied. In day-ahead market, only seller agents may present complex conditions.

2.2.1.3. Bilateral Contracts

Bilateral contracts allow players to negotiate directly with each other, outside the spot market. This provides opportunities for reaching advantageous agreements from an economic perspective, but also from a spatial one, when negotiating with players that offer benefits resulting from their location. It is also a good opportunity to establish contracts with varying timelines, resulting on higher security for companies that require constant demands over time.

Bilateral contracts are established through requests for proposals sent by buyers or traders - the demand agents. If a demand agent chooses to participate in the bilateral market, it will first send a request for electricity with its price expectations to the interested sellers. In response, a seller analyzes its own capabilities, current availability, and past experience. The seller must be sure that it is feasible to deliver energy to the buyer’s location. So, it must get the network operator’s feedback before reaching an agreement with the demand agent. If the seller can make an offer according to the requested parameters, it formulates a proposal and sends a message to the demand agent. The demand agent evaluates the proposals and accepts or rejects the offers.

2.2.1.4. Balancing Market

The balancing market's [Olsson and Soder, 2008], [Veen and Vries, 2008] goal is to take care of the necessary adjustments on the viable daily program and the last final hourly program, correcting possible deviations from forecasted production or consumption. It is a market of voluntary participation, in which offers can be presented by all agents qualified for the daily market who have participated in the corresponding session of the same market, which performed a bilateral contract, or whose units were unable to attend due to unavailability in the daily market and return to be available. It is, therefore, a complementary platform to the day-ahead market.

Although usually only sellers may present complex conditions to the spot market, in the balancing market both sellers and buyers may present such conditions. Another important issue is that sellers may become buyers on the balancing market, while buyers may become sellers, depending on their forecasted production or consumption deviations, and also on their strategies. Players are therefore able to take advantage on this to define negotiation strategies which contemplate the buying of a higher amount of power when the prices are lower, knowing that they can later sell it at higher prices; or vice-versa, selling more than they are capable of producing, when the prices are high, and then buying the extra amount on a complementary market in which the prices are lower.

2.2.1.5. Complementary Markets

Electric energy is one of the most commonly used forms of energy. However, a substantial part of its production results from the burning of fossil fuels, which originates greenhouse gases emission to the atmosphere, namely carbon dioxide, methane, nitrous oxide and sulphur hexafluoride, originating climatic changes with harmful consequences to the environment [PSI, 2007].

With the shortage perspective of the non-renewable resources and the increasing usage of electric energy, it becomes imperative to develop new methods of energy production, investing in technologies that contribute to a more energetically rational way of living.

The verified growth of the investment on distributed generation, namely in wind and photovoltaic technologies, has been creating new opportunities for the promoters that own such technologies. Besides the selling of electrical energy to the system operators or in energy markets, the promoters can develop their activity in other markets, such as the Carbon Market, the Green Certificates emission, or the selling of water steam and hot water, among others [Matos, 2007]. An alternative potential business is the integration with industries as livestock, the treatment of municipal solid waste, cork, in order to significantly reduce investment and/or operation costs.

These market mechanisms are complementary to the electric market, originating a more dynamic and alternative global market. The complementarity between such different types of markets creates the opportunity for players to improve their negotiating approaches, considering the investments in different markets.

The increasing complexity brought by the conception of such a diversity of market types resulted in high changes concerning the relationship between the electricity sector entities. It also resulted on the emergence of new entities, mostly dedicated to the electricity sector and electricity energy trading management. In what regards the commercial transactions, the analysis of different market mechanisms and the relationship between market entities becomes crucial. Namely in the case of Portugal, where the Iberian market, in partnership with Spain, has materialized not long ago, there are many aspects to analyze, improve, and even redefine. All market participants develop interactions among them, needing information systems for that purpose. As the observed context is characterized as being of significant adaptation and change, the need for decision support tools directed to this markets' analysis is also accentuated. Multi-agent based software is particularly well fitted to analyze systems with such characteristics.

2.3. Multi-Agent Systems

With the purpose of solving problems for which a single entity equipped with Artificial Intelligence (AI) could not provide the best response [Davis, 1980], the field of Distributed AI emerges. This field is created in the 80s, resulting from the merging between AI and distributed computation.

Traditionally, the Distributed AI is divided into two main areas [Panait and Luke, 2005]. The distributed problem solving concerns the decomposing and distribution of a problem solving process with multiple knots, and regarding a collective solving for the problem. The second approach area – Multi-Agent Systems – highlights the common behaviours of the agents, with some degree of autonomy and complexity, which results from their interactions. This type of systems regards the coordination of behaviours of an agent community, in a way they can share knowledge, capabilities, and objectives, so that they are able to solve complex problems.

In order to correctly use a Multi-Agent System it is necessary to understand the concept of agent, realizing which their characteristics are, and in what way they are organized in the presence of other agents. Therefore, the concepts of agent and Multi-Agent System are described in this section.

2.3.1. *The Agent*

There is no common definition of an agent in the literature, in an AI context. However, from the different existing definitions, some of the agent characteristics are consensual. Some of these characteristics are: the sensorial capabilities over the surrounding environment, the capability of interacting and reacting on that same environment, and the social abilities that allow the interaction with other agents.

According to [Minsky, 1986], each agent is only able to perform simple tasks, which do not demand any reasoning. However, when these agents are joined in societies in special ways, this will lead to real intelligence. In [Brustolini, 1991], on the other hand, agents are defined as systems which are capable of autonomously perform important actions in the real world. In order to perform such actions in an effective way, agents are forced to coordinate their actions, by developing strategic decision making capabilities [Coelho, 1994]. The way agents perform their autonomous actions is based on information (sensors, and feedback) received from the environment [Panait and Luke, 2005].

These definitions of on agent are centred on these entities' capabilities of acting in an autonomous way, inside a certain environment, with which they interact through sensors and actuators, while trying to fulfil their objectives [Wooldridge, 2002]. In order to accomplish their goals, agents usually present some or even all of the following characteristics [Weiss, 2010]:

- Sensorial capability, when an agent has sensors to gather information about its environment;
- Reactivity, if an agent feels and acts, reacting to on-going environment changes;
- Autonomy, when it decides and controls its own actions;
- Pro-activity, meaning that the agent is goal driven, and goes beyond reacting to the environment;
- Persistency, when an agent exists during long periods of time;
- Social skills, allowing the agent to communicate and co-operate with other agents or even people, *i.e.* competing or negotiating;
- Learning, if an agent is able to change its behaviour based on prior experience;
- Mobility, when the agent is able to move from one computer to another;
- Flexibility, if its tasks don't need to be pre-determined;
- Agility, by being able to swiftly take advantage of new unforeseen opportunities;
- Character, when the agent presents a credible personality and emotional behaviour;
- Intelligence, if it is able to reason autonomously, to plan its actions, to correct its mistakes, to react to unexpected situations, to adapt and to learn.

2.3.2. *Agent Based Systems*

Agent based systems can be divided into two different types [Stone and Veloso, 1997]: the single-agent systems, and the multiagent systems. In the single-agent systems there is only one agent responsible for taking decisions, while all the other constituents work is based on these decisions. Although the agent is a part of the environment, it possesses additional characteristics apart from that environment. This is due to this agent being an independent entity with its own objectives, actions, and knowledge. In a single-agent environment, any other entity is recognized as an agent, rather elements that shape the environment.

Multiagent systems, on the other hand, feature several agents, which affect and shape other agents' objectives and actions. In a typical multi-agent scenario interactions between agents occur, *i.e.* they communicate among each other. From an individual agent's perspective, the main difference between these systems and single-agent systems resides on the possibility of the environment dynamics being influenced by other agents. This way, all multiagent systems are considered dynamic, *i.e.* the environment can be altered when an agent acts on it.

In [Panait and Luke, 2005] a multi-agent environment is defined as that in which *"there are several agents interacting with each other, in an environment where restrictions take place in a way an agent cannot, at a certain moment, be aware of all the knowledge other agents possess (including the internal status of the other agents)"*. These are important restrictions to the definition of a multiagent system in these authors' perspectives. In a different view, distributed agents can perform synchronized actions knowing exactly the status of the other agents, and which behaviours they will execute. This allows agents to behave as "slaves" to a master controller. Additionally, if the domain does not demand interactions, this will then be decomposed into separate and completely independent tasks, which are solved by a unique agent.

Multiagent systems have been standing out in the last decades as a particularly interesting paradigm in what concerns the modelling and implementation of applications in dynamic and unpredictable environments [Zambonelli *et al.*, 2004]. This type of systems is even referred to as the next great step in the computation's evolution, as was before the object oriented programming [Luck *et al.*, 2005].

Agent's usage is owed to a holistic vision, where the total is more than the simple sum of the parts. Therefore, multiagent systems can be used in problems that present an intractable complexity to humans or single-agent systems. Using problem decomposing techniques allows the computation's distribution, so that the resulting simpler problems can be mapped by a set of distinct agents which interact in a way they can obtain satisfactory global solutions.

Multiagent systems have been widely applied to several areas, namely in e-commerce, robotics, traffic control, production systems, and scheduling. Additionally, with the rising necessity of simulating complex environments, this type of systems has been increasing its coverage of topics. Sections 2.4 and 2.5 of this document approach multiagent systems' application to electricity markets.

2.4. Simulation and Multi-Agent Simulation of Electricity Markets

Electricity market simulators must be able to cope with this evolving complex dynamic reality and provide electricity market players with adequate tools to adapt themselves to the new reality, gaining experience to act in the frame of a changing economic, financial, and regulatory environment. With a multi-agent simulation tool the model may be easily enlarged and future evolution of markets may be accomplished.

Multiagent simulation combined with other artificial intelligence techniques may result in sophisticated tools, namely in what concerns players modelling and simulation, strategic bidding and decision-support [Wellman *et al.*, 2007], [Hortaçsu and Puller, 2008], [Amjady *et al.*, 2010]. For example, consumers' role has significantly changed in this competitive context, making load analysis, consumer profiling and consumer classification very important [Figueiredo *et al.*, 2005]. The data generated during simulations and by real electricity markets operation can be used for knowledge discovery and machine learning, using data mining techniques [Figueiredo *et al.*, 2005], [Jing *et al.*, 2009] in order to provide electricity markets players with simulation tools able to overcome the little experience they have in electricity markets operation. Some of the existent electricity markets simulators have machine learning abilities [Koritarov, 2004], [Li and Tesfatsion, 2009], [Vale *et al.*, 2011a], but huge advances are required so they can be a real added value for real electricity markets players.

Each player acting in an electricity market has its own goals and should use adequate strategies in order to pursuit those goals, its strategic behaviour being determinant for its success. Player behaviour exhibits changes in response to new information and knowledge that it may have; this may refer to his self knowledge, to knowledge coming from the exterior and from the dynamic complex interactions of the heterogeneous individual entities. Each agent has only partial knowledge of other agents and makes his own decisions based on his partial knowledge of the system. Methodologies for strategic bidding in electricity markets [Figueiredo *et al.*, 2005], [Wellman *et al.*, 2007], [Amjady *et al.*, 2010], [Vale *et al.*, 2011a] can help players making more successful decisions but they must be combined together with dynamic behaviour strategies able to take advantage from the knowledge concerning past experience and other players.

There are several experiences that sustain that a multiagent system with adequate simulation abilities is suitable to simulate Electricity Markets [Harp *et al.*, 2000], [Koritarov, 2004], [Zimmermann *et al.*, 2004], [Migliavacca, 2007], [Li and Tesfatsion, 2009], considering the complex interactions between the involved players. It is important to note that a Multi Agent System is not necessarily a simulation platform but simulation may be of crucial importance for electricity markets study, namely concerning scenarios comparison, future evolution study and sensitive analysis.

Simulator for Electric Power Industry Agents (SEPIA) [Harp *et al.*, 2000] is a Microsoft Windows platform oriented simulator. It is based on a Plug and Play architecture, allowing users to easily create simulations involving several machines in a network, or in a single machine, using various processing units. SEPIA allows specifying the number of participating agents, as well as their behaviours, interactions, and changes during the simulation. The simulation developments can be followed and oriented by mechanisms for that purpose.

The Electricity Market Complex Adaptive System (EMCAS) [Koritarov, 2004] uses an agent based approach with agents' strategies based on learning and adaptation. Different agents are used to capture the restructured markets heterogeneity, including generation, demand, transmission, and distribution companies, independent system operators, consumers and regulators. It allows undertaking Electricity Markets simulations in a time continuum ranging from hours to decades, including several Pool and Bilateral Contracts markets.

Power Web [Zimmermann *et al.*, 2004] is a Web-based market simulator, allowing the various participants to interact from very distinct zones of the globe. It is a rather flexible system that allows the definition of simulations with a large set of scenarios and rules. This simulator includes a centralized agent, acting as an independent system operator, to guarantee the reliability of the system, according to a defined group of entities, acting in various markets. It also allows users to enter an open market, competing against producers controlled by other users or computational algorithms.

The Short – Medium run Electricity Market Simulator (SREMS) [Migliavacca, 2007] is based on game theory and is able to support scenario analysis in the short-medium term and to evaluate market power, in some situations. Some of its main features are: short-medium run (a time horizon of one month or multiples of it) simulation of electricity markets based on game theory, calculating price-makers optimal hourly bids; inelastic load, defined hour by hour and zone by zone; tree-like network with interzonal transit limits; monthly scheduling of reservoir hydro pumping storage plants; highly realistic representation of thermal plants; possible quota appointed to physical bilateral contracts, depending on producers share and risk attitude. It is particularly suitable to study the Italian electricity market.

Agent-based Modelling of Electricity Systems (AMES) [Li and Tesfatsion, 2009] is an open-source computational laboratory for the experimental study of wholesale power markets restructured in accordance with

U.S. Federal Energy Regulatory Commission (FERC)'s market design. It uses an agent-base test bed with strategically learning electric power traders to experimentally test the extent to which commonly used seller market power and market efficiency measures are informative for restructured wholesale power markets. The wholesale power market includes independent system operator, load-serving entities and generation companies, distributed across the busses of the transmission grid. Each generation company agent uses stochastic reinforcement learning to update the action choice probabilities currently assigned to the supply offers in its action domain.

These are important contributions but, in general, lack flexibility as they adopt a limited number of market models and of players' methodologies. The evolution of some of these simulators is difficult to track but some of them, as is the case of AMES, are evolving in a very dynamic way. At the present state, it is important to go a step forward in Electricity Markets simulators as this is crucial for facing the changes in Power Systems. Increasing number and diversity of active players (due to high penetration of distributed energy resources and demand side participation) are a huge challenge [Figueiredo *et al.*, 2005].

The Multi-Agent Simulator for Competitive Electricity Markets (MASCEM) [Praça *et al.*, 2003], [Pinto *et al.*, 2011e], [Vale *et al.*, 2011a] was developed so that it could contribute to the overcoming of such challenges, and surpass the limitations that the referred simulators present. Section 2.5 describes the MASCEM simulator in detail, pointing out the improvements and model enlargement already planned and being implemented.

2.5. MASCEM: Multi-Agent Simulator for Competitive Electricity Markets

The Multi-Agent Simulator for Competitive Electricity Markets - MASCEM [Praça *et al.*, 2003], [Pinto *et al.*, 2011e], [Vale *et al.*, 2011a] is a modelling and simulation tool that has been developed with the purpose of studying complex restructured electricity markets operation. MASCEM models the complex dynamic market players, including their interactions and medium/long-term gathering of data and experience, to support players' decisions according to their very own characteristics and objectives. MASCEM most important features are presented in Figure 2.3.

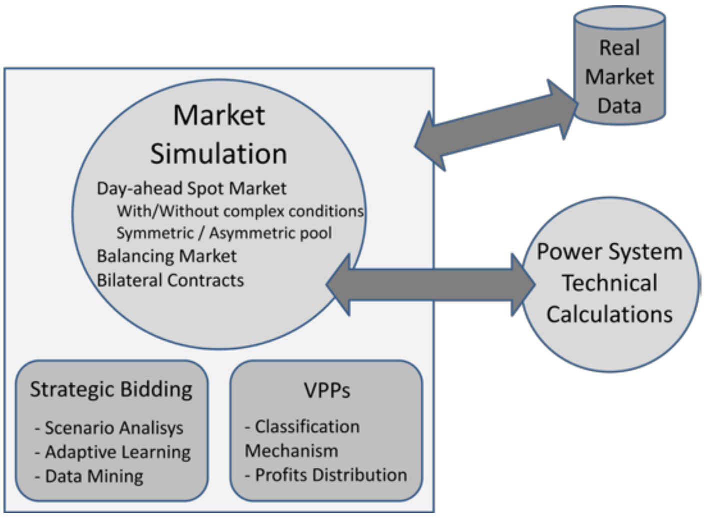


Figure 2.3 – MASCEM key features [Pinto *et al.*, 2011e].

MASCEM is implemented on the top of OAA [OAA, 2011], using OAA *AgentLib* library, and Java Virtual Machine 1.6.0 [JAVA, 2011]. The OAA's Interagent Communication Language (ICL) is the interface and communication language shared by all agents, no matter which machine they are running on or which programming language they are programmed in, which allows integrating a variety of software modules.

Communication and cooperation between agents are brokered by one or more facilitators, which are responsible for matching requests, from users and agents, with descriptions of the capabilities of other agents.

OAA is not a framework specifically devoted to develop simulations; some extensions were made to make it suitable to deal with the energy markets that MASCEM currently supports, namely to introduce the time evolution mechanism of the simulation.

MASCEM's goal is to be able to simulate as many market models and players types as possible so it can reproduce in a realistic way the operation of real electricity markets. This enables it to be used as a simulation and decision-support tool for short/medium term purposes but also as a tool to support long-term decisions, such as the ones taken by regulators.

2.5.1. *Multi-Agent Model*

There are several entities involved in the negotiations in the scope of electricity markets; MASCEM's multi-agent model represents all the involved entities and their relationships. MASCEM multi-agent model includes: a market facilitator agent, seller agents, buyer agents, virtual power player (VPP) [Praça *et al.*, 2007] agents, VPP facilitator agents, a market operator agent and a system operator agent. Figure 2.4 presents a general overview of MASCEM's main entities interaction.

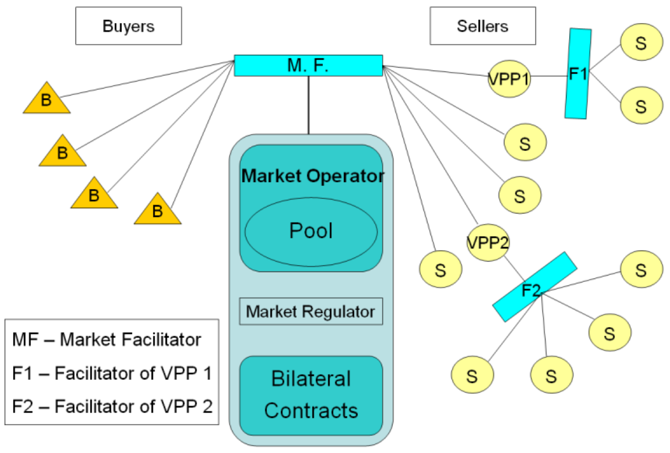


Figure 2.4 – MASCEM agent architecture [Vale *et al.*, 2011b].

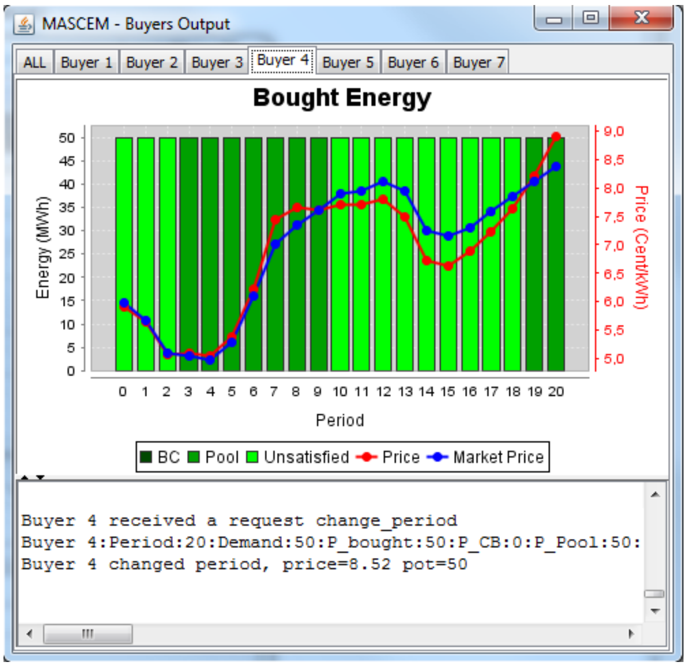
The market operator agent is only present in pool or hybrid markets simulations. It receives the bids from buyer and seller agents, validates and analyses them, and determines the market price, and the accepted and refused bids.

The system operator agent is always present. It ensures that all conditions are met within the system, and it is also responsible for the system security. After being informed of all negotiations to be held, examines the technical feasibility from the power system point of view and solves congestion problems that may arise. In fact, this agent makes a connection with a power system simulator, through which the power flow analysis is performed.

The market facilitator agent coordinates and ensures the proper operation of the market, regulating all the existing negotiations. It knows all the market players, as they are registered in advance to the facilitator, specifying their role and services.

Buyer and seller agents are the key elements of the market. Consumers and distribution companies are represented by buyer agents. The electricity producers are represented by seller agents. Seller agents compete with each other to maximize their profits. On the other hand, seller agents may also cooperate with buyers trying to establish agreements that meet the objectives of both parties. The number of buyers and sellers, and their intrinsic and strategic characteristics are defined by the user for each scenario.

Figure 2.5 displays a screenshot of a buyer agent running, showing its bids, sold and unsold power, and also the requests it is receiving.



```
Buyer 4 received a request change_period
Buyer 4:Period:20:Demand:50:P_bought:50:P_CB:0:P_Pool:50:
Buyer 4 changed period, price=8.52 pot=50
```

Figure 2.5 – Buyer agent's output [Vale *et al.*, 2011a].

The top part of Figure 2.5 shows the graphical representation of this agent's results in the Pool. It includes the amount of energy that it bought in each period, and the comparison between its bid price and the market price.

The bottom part presents the requests that this agent is receiving at each time, and some information about this agent's actions. In this case period 20 has ended and the agent is getting ready to start the negotiations of period 21. Three sets of information can be seen:

- First it received a notification from the market operator indicating that the time for the present period of negotiations ended. So, the agent performs the necessary arrangements to be ready for the next period;

- The second line presents a summary of the results that this agent obtained in the last period, including the information presented in the graph;
- Finally, the third line indicates the bid price and amount of power that this agent will negotiate in this market in the next period. These values are the output of the bidding strategy this agent is using for the next negotiation period.

Due to a significant increase of small independent producers negotiating in the market, comes the need to make alliances between them, in order to be able to compete on equal footing with the big producers. The VPP agents represent these alliances. They manage the information of their aggregates and are viewed from the market as seller agents. Each VPP is modelled as an independent multi-agent system, maintaining high performance and allowing agents to be installed on separate machines. To achieve this independence, individual VPP facilitators have been created to manage the communications between each VPP and its members independently from the rest of the simulation [Pinto *et al.*, 2009].

2.5.2. *Virtual Power Players*

Due to environmental and fossil fuels shortage concerns, renewable energy resources are being more used. The advantages of using renewable are clear from the environment point of view. From the technical and economical point of view there are problems that must be overcome to take advantage of an intensive use of renewables, which are mostly originated by distributed generation. An aggregating strategy can enable owners of renewable generation to gain technical and commercial advantages, making profit of the specific advantages of a mix of several generation technologies and overcoming serious disadvantages of some technologies. The aggregation of distributed generation plants gives place to the new concept of Virtual Power Player (VPP). VPPs integration into electricity markets is a very challenging domain that has been motivating MASCEM evolution.

VPPs are multi-technology and multi-site heterogeneous entities, being relationships among aggregated producers and among VPPs and the remaining electricity market agents a key factor for their success [Dang *et al.*, 2004], [Vale *et al.*, 2008], [Rahwan *et al.*, 2008]. Agent coalitions are especially important to address VPPs as these can be seen as a coalition of agents that represent the aggregated players [Dang *et al.*, 2004], [Vale *et al.*, 2008].

Coalition formation is the coming together of a number of distinct, autonomous agents that agree to coordinate and cooperate, acting as a coherent grouping, in the performance of a specific task. Such coalitions can improve the performance of the individual agents and/or the system as a whole. It is an important form of interaction in multi-agent systems. The coalition formation process comprises several phases: coalition structure generation, optimization of the value of the coalition and payoff distribution [Vale *et al.*, 2008], [Oliveira *et al.*, 2009], [Pinto *et al.*, 2009].

Regarding the coalition formation process, for VPP modelling, the three main activities of coalition structure generation, optimization of the value of the coalition and payoff distribution should be considered under a scenario where agents operate in a dynamic and time dependent environment. This entails significant changes on MASCEM core model and communications infrastructure.

To sell energy in the market VPP must forecast the generation of aggregated producers and “save” some power capacity to ensure a reserve to compensate a generation oscillation of producers with natural resources technologies dependent.

The VPP can use different market strategies, considering specific aspects such as producers established contracts and range of generation forecast. The prediction errors increase with the distance between the forecasting and the forecast times. The standard errors are given as a percent of the installed capacity, since this is what the utilities are most interested in (installed capacity is easy to measure); sometimes they are given as the mean production or in absolute numbers.

MASCEM’s modelling of VPPs enlarged the scope of negotiation procedures in this simulator, allowing the study of different types of negotiation outside the usual electricity markets’ regulatory models.

2.5.3. *Negotiation in MASCEM*

MASCEM includes several negotiation mechanisms usually found in electricity markets, being able to simulate several types of markets, namely: Pool Markets, Bilateral Contracts, Balancing Markets and Forward Markets. Figure 2.6 presents the negotiation sequence for one day simulation in MASCEM.

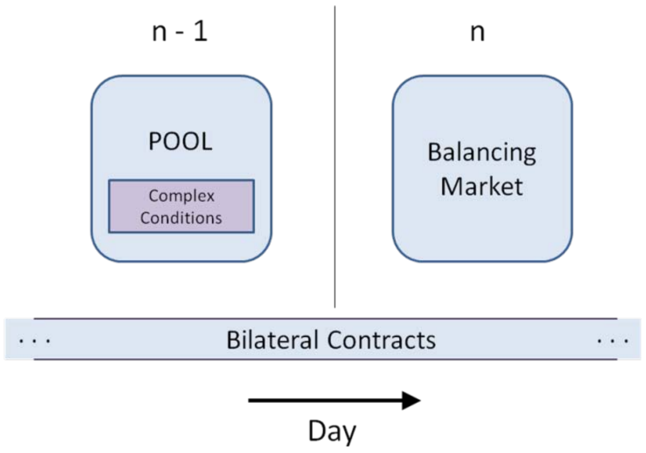


Figure 2.6 – Negotiations timing for day n [Santos *et al.*, 2011].

Based on the previously obtained results, buyer and seller agents review their strategies for the future. The strategic behaviour of each agent defines its desired price and amount of power to be negotiated in each market.

Time-dependent strategies and behaviour-dependent strategies are part of each agent, and define the price to negotiate in the next day according to the results obtained previously. There are four types of time-dependent strategies [Praça *et al.*, 2003]: Determined (prices remain constant throughout the period of negotiation); Anxious (minor changes to the price are made after little trading time); Moderated (small changes to the price are made in an intermediate stage of negotiation period); Gluttonous (the price is significantly changed, but only in late trading).

On the other hand, the behaviour-dependent strategies are [Praça *et al.*, 2003]: Composed Goal Directed (when an agent has two consecutive goals, in which the definition of the second objective depends on the fulfilment of the first); Adapted Derivative Following (the results of price changes made in previous trading periods are analyzed. If the agent finds that the change in the price of its proposals brought benefits, it maintains the same type of change for the next period. Otherwise, the change in price will go in the opposite direction); Market Price Following (this strategy bases the agent price fluctuations on the fluctuations of the market price).

Concerning the VPPs' operation, negotiations take place in some additional timings, namely in coalitions' formation and management [Vale *et al.*, 2011c], [Pinto *et al.*, 2011f]. This type of negotiations provides players with the capabilities of achieving the most advantageous coalition contracts, both for the aggregator (VPP) and for the members (sellers and buyers). These negotiations take into account the players' characteristics, objectives, and goals, and allows them to get alternative deals to those they could get by negotiating exclusively on the market.

The different types of negotiation approached in MASCEM, the different types of markets implemented, and the distinct interactions between the participating entities in different situations, create the fundamental need for the use of machine learning techniques in this simulator. An overview on the machine learning theme is provided in section 2.6.

2.6. Machine Learning

Machine learning is a field of artificial intelligence that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can teach themselves to adapt and expand when exposed to new data. Therefore, it can be said that *"The goal of machine learning is to build computer systems that can adapt and learn from their experience"* [Dietterich, 1991].

The process of machine learning is similar to that of data mining. Both systems search through data to look for patterns. However, instead of extracting data for human comprehension - as is the case in data mining applications - machine learning uses that data to improve the program's own understanding. Machine learning programs detect patterns in data and adjust program actions accordingly. Thus *"Learning denotes changes in a system that (…) enables a system to do the same task more efficiently the next time"* [Simon, 1973], this way making computer systems more capable of solving problems, by adapting and being more self-sufficient, with the purpose of supporting humans in their tasks.

Learning is essential for unknown environments. When the system's designer is not acquainted with the complete aspects of the "world", by exposing a program agent to reality rather than trying to write it down (programming it), it becomes a useful method in system construction [Minsky, 1961]. Also when the amount of information or knowledge is too large, it becomes impossible to be defined and encoded by humans, hence it becomes essential that the machine is capable of doing it itself. A similar situation is when the environments change over time, and when new knowledge about certain tasks is constantly being discovered by humans, it becomes impossible to re-design all systems by hand all the time.

Given that the environment is always evolving, adapting to that potentially dynamic environment is crucial. This adaptation can be performed by utilizing an implicit learning characteristic of evolution in the systems' algorithms. This way, by considering the algorithms' evolution, taking into account the own algorithm's perception of what it experiences, provided that *"Learning is constructing or modifying representations of what is being experienced"* [Michalski, 1980], computer programs that learn and evolve in uncertain environments can be created.

There are several learning techniques that can be used to develop this kind of evolutionary algorithms, acquiring the necessary knowledge to the system, and choosing the significant information, making them more appropriate to each distinct situation. In [Winston, 1992] and [Rich and Knight, 1991] it is said that "*Learning is the process that acquires knowledge to the system. It is hard to say that the machine is intelligent if it has no learning capability*". However, these authors offer different opinions when categorizing the different learning techniques. In [Rich and Knight, 1991] learning techniques are classified into five distinct groups:

- rote learning;
- learning by taking advice;
- learning from examples;
- learning in problem solving;
- discovery learning.

One year later, in [Winston, 1992] two types of learning techniques are presented:

- coupling new information to previously acquired knowledge;
- digging useful regularity out of data.

Later on, in [Ginsberg, 1993] learning techniques are categorized it into two major techniques: generalization and discovery learning. Being the first one divided into deductive and inductive learning.

Nowadays, the types of learning are usually grouped in:

- unsupervised learning:
    - clustering;
    - discovery;
    - genetic algorithms;
- supervised learning:
    - rote learning;
    - induction;
    - analogy;
- reinforcement learning.

These ideas are compared and confronted in Table 2.1, presenting the main points in which they converge and disagree.

Table 2.1 – Machine learning techniques classification.

P. H. Winston [Winston, 1992]	**Rich and Knight** [Rich and Knight, 1991]	**M. L. Ginsberg** [Ginsberg, 1993]	**Nowadays**
coupling new information to previously acquired knowledge	rote learning	generalisation learning	rote learning
digging useful regularity out of data	learning by taking advice	inductive learning	
	learning from examples	deductive learning	analogy
	learning in problem solving		clustering
	discovery learning		
			genetic algorithms
			reinforcement learning

As presented in Table 2.1, the discovery learning was the first technique to start being considered by the majority of authors. Discovery learning is a restricted form of learning that acquires knowledge without the help of a teacher. Unlike the other learning techniques, discovery learning can perform the tasks for which none, or little knowledge can be provided. In general, discovery learning can be broken down into three categories namely theory-driven discovery, data-driven discovery and clustering.

Afterwards, inductive learning started being consensual too. Inductive learning is the system that tries to induce a general rule based on observed instances. In other words, the system tries to infer an association between specific inputs and outputs. In general, the input of the program is a set of training instance where the output is a method of classifying subsequent instance. For example, the input of the program may be the colour of mushrooms where the output is classification that determines whether they are edible or not.

2.7. Final Remarks

With the restructuring of the electricity sector, its operation rules were altered so that this sector could provide higher efficiency and enhance the competitiveness of all the involved participants, from production to consumption. This process presents high complexity, placing several challenges to both the enterprise and scientific communities.

In order to overcome these challenges, several market models have arisen. Some restructuring pioneer countries' experience provides some guidance in what regards the implemented market models' performance.

However, it is still premature to take definitive conclusions. Therefore, tools allowing the study of different market models' performance prove to be a real added value.

The electricity market can be seen as a multiagent decision problem, where each market player is represented by an autonomous agent. Just as agents, the electricity markets players have their own objectives, and certain autonomy. Agents can behave as buyers, sellers, aggregators, intermediaries, traders, or market and system operators. Multiagent systems, particularly multiagent based simulation, proved to be an adequate way to develop tools that allow, not only to analyse the market's performance as a whole, but also the individual behaviour of the several entities.

In this sense, some works have arisen, which confirm the adequate applicability of agents and multiagent systems to the study of electricity markets. Some of the most important works in this scope have been presented, highlighting their main characteristics, and pointing out their main limitations. These limitations are mostly directed to the lack of machine learning capabilities that allow them to effectively provide support to market entities. To overcome these limitations is the main goal of this work; by developing an adaptive tool which is able to provide effective support to market negotiating players, by adapting to different circumstances and negotiation contexts, hence providing players with the most adequate acting support for each situation.

Chapter 3 presents ALBidS, an adaptive learning system equipped with artificial intelligence techniques, context analysis tools, and adaptation capabilities, able to endow market players with strategic acting capabilities, hence providing them competitive advantage in the market. This enables the contribution regarding the requirements that market players have, resulting from the electricity markets restructuring; and also concerning the limitations that the existing market simulators present.

CHAPTER 3. ALBIDS: ADAPTIVE LEARNING STRATEGIC BIDDING SYSTEM

3.1. Introduction

In order to provide the negotiating players with competitive advantage in the electricity market it is essential to provide them strategies capable of dealing with the constant market changes, allowing adaptation to the competitors' actions and reactions. For that, it is necessary to have adequate forecast techniques to analyze the market data properly, namely the historic market prices. Prices prediction can be approached in several ways, namely through the use of statistical methods [Zhao *et al.*, 2008], data mining techniques [Azevedo *et al.*, 2007], [Cao *et al.*, 2009], neural networks (NN) [Wilamowski and Yu, 2010], support vector machines (SVM) [Bayro and Daniel, 2010], or several other methods [Bompard *et al.*, 2005], [Jing *et al*., 2009]. There is no method that can be said to be the best for every situation, only the best for one or other particular case.

To take advantage of the best characteristics of each technique, a new system has been created to integrate several distinct technologies and approaches. This system is named ALBidS – Adaptive Learning strategic Bidding System, and it is implemented as a multiagent system. There is one agent performing each distinct algorithm, detaining the exclusive knowledge of its execution. This way the system can be executing all the algorithms in parallel, preventing as possible the degradation of the method's performance. As each strategy agent gets its answer, it sends it to the Main Agent (presented in section 3.6, along with the tools that support this agent's performance), which is responsible for choosing the most appropriate answer among all that it receives, depending on each context.

Contexts are an important factor in what concerns the adaptation of the approaches to be chosen as the final MASCEM's supported negotiating player action to perform in the market. A mechanism to analyse and define different market negotiating contexts is presented in section 3.5, therefore providing the means for the chosen actions to be adapted and chosen depending of the different circumstances that are encountered at each moment.

The ALBidS system integrates several very distinct approaches in what concerns market negotiations, and prices forecast. These approaches, implemented and tested independently before their integration in ALBidS are presented in section 3.7 - Strategy Agents. An overview on each approach's context is provided, in order to integrate the reader with the scope of each. Afterwards the details on these approaches implementations are described and the necessary experimental findings are presented, in order to test each approach's adequate performance, and prove the adequacy to its purpose.

Supporting some of these approaches, a competitor player's profile definition mechanism has been implemented, with the point of creating adequate player profiles to be used by the ALBidS strategies which require such profiles for their execution. This mechanism is presented in section 3.8.

Additionally, in order for the ALBidS system to able to effectively adapt to different simulation circumstances, a mechanism with the purpose of managing the balance between the efficiency and the effectiveness of the system has been developed, and it is presented in section 3.9. This mechanism provides the means for the system to adapt its execution time depending on the purpose of the simulation, *i.e.*, if the expected results from ALBidS are as best as it is able to achieve, or on the other hand if the main requirement is for the system to be executed rapidly, since the purpose of the considered simulation is no analyse issues other than a players’ optimal performance in the electricity market.

Finally, the details regarding ALBidS’ design, internally, and in terms of multiagent architecture; and its integration with MASCEM, are presented in section 3.2. This section is supported and extended by sections 3.3 and 3.4, which respectively present the ALBidS facilitator built to support this system’s agents communications independently from the rest of the simulation; and the ALBidS user interface which supports the interaction with the user, and the respective feedback from the system, regarding the presentation of the happenings along a simulation, and consequent results.

3.2. Global Structure and Integration with MASCEM

ALBidS integration with MASCEM is an essential process for the success of this work. This integration had to take into account the good practices in terms of software engineering, and high care had to be taken when defining ALBidS architecture.

The definition of the ALBidS global structure has been based on a careful analysis on how the system should behave, both in an internal, independent perspective, and also in what concerns its connection and efficient communication with MASCEM. For that it was necessary to consider in a high account the design of its structure in terms of programming issues, while at the same time guaranteeing the higher possible performance in what concerns the agent’s (both ALBidS’ internally, and with MASCEM’s) interconnectivity and communications.

To provide players adequate decision support, the system agents access a database containing the log of the system’s events, and the historic electricity market data. Figure 3.1 presents the data model used by ALBidS.

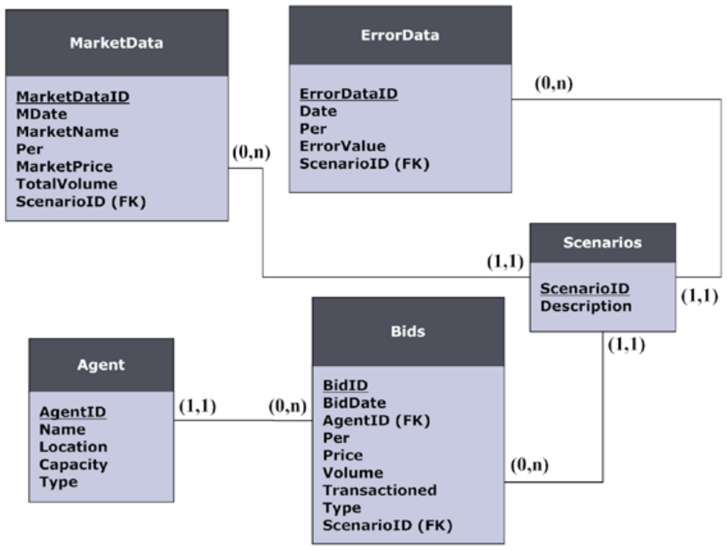


Figure 3.1 – Data Model [Sousa, 2011].

Each MASCEM negotiating agent bids in the market for each period of each day. The information regarding all the agents' bids is stored in the **Bids** table. This data includes the indication of the success that each bid got on the market. These bids are subject to an error, which is especially relevant in the case of bids resulting from strategies based on forecasting techniques. The error is saved in the **ErrorData** table. This information is used by several strategies which require the competitors' log of actions to support the self strategy's action proposal. These tables are dependent on a ScenarioID, indicating the scenario that the simulator is running. When a new simulation is started, the user is able to choose to continue an existing scenario, with a specific number of agents, with already set characteristics, or, on the other hand, to create a new scenario, defining all the MASCEM simulation options. When ALBidS is being used for a previously existing scenario, it is able to use the simulation data that was saved before to support the strategic behaviour. The scenario dependent data includes the statistics used by the Main Agent to choose among the proposed bids, and also the historic market prices, saved in the **MarketData** table. This table is filled with the historic market prices, which may be from different markets relevant to the required studies. This historic data is used as reference for the market price forecasts, and it is updated, depending on the scenario, with the prices resulting from the simulation, this way it is kept updated with the actual market prices.

When designing the ALBidS system, high importance had to be given to defining its architecture, as it is one of the first decisions and steps to take when one pretends to develop a computer system. The architecture depends on the characteristics of the system, and will influence all its conception. Choosing the MVC (Model-View-Controller) architecture (Figure 3.2) ensured the independence between the data (model), the user interface (view), and the business layer (controller).

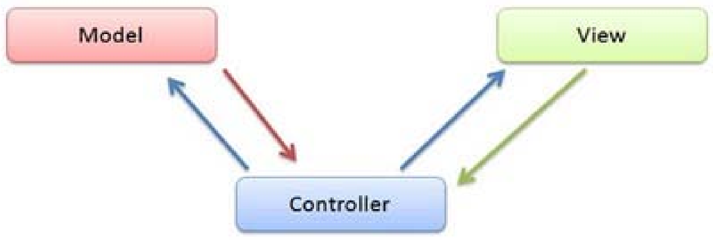


Figure 3.2 – Interaction between the MVC architecture components.

ALBidS is integrated with the MASCEM simulator through the *MainGUI* class, the class that is responsible for the ALBidS user interface (presented in section 3.4) and for creating the ALBidS model classes. The *MainGUI* class contains an instance of the *AlbidsController* class, which is responsible for making the connection with the rest of the ALBidS system, manipulating the domain objects and the system's agents (Figure 3.3).

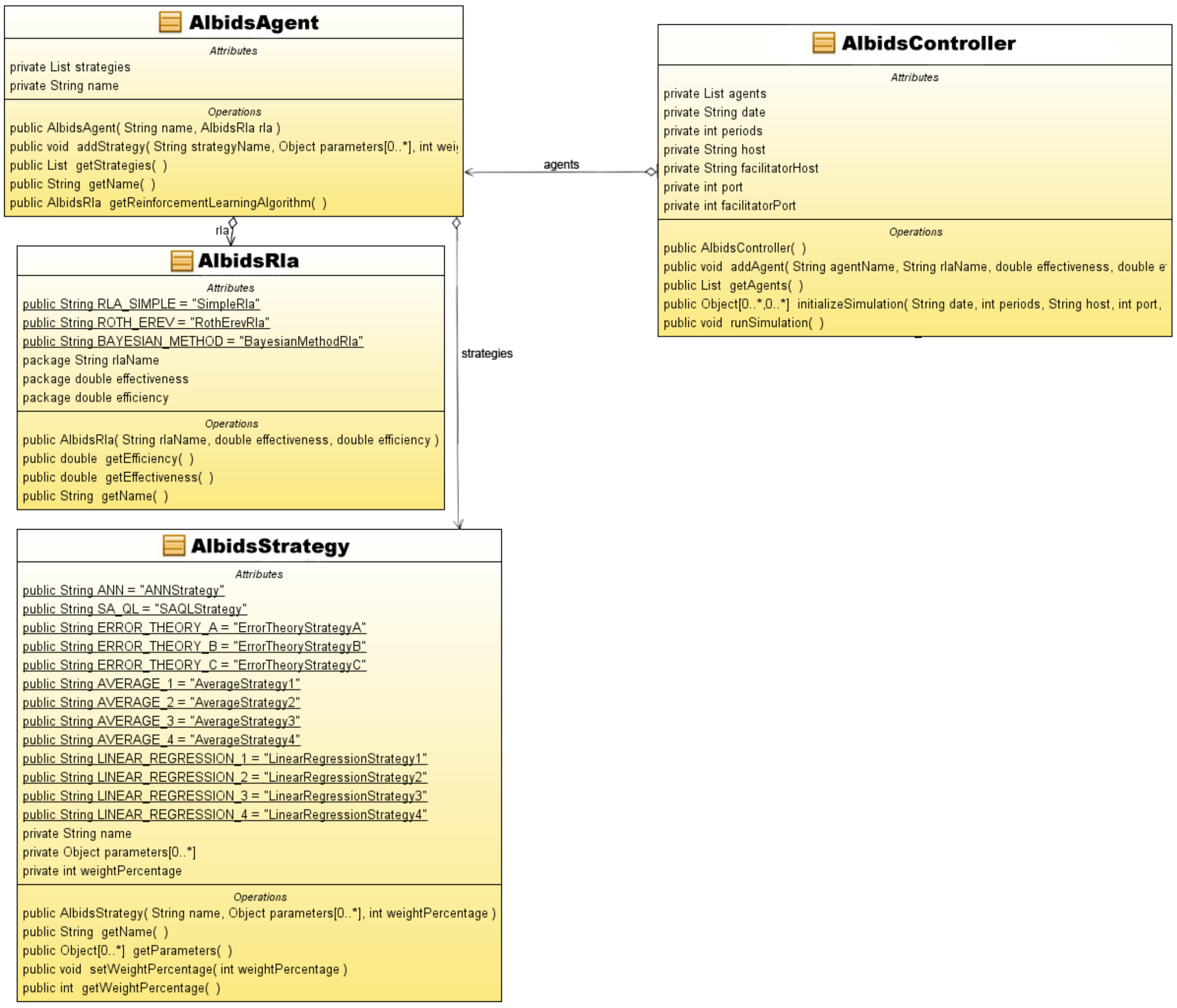

AlbidsAgent
Attributes
private List strategies
private String name
Operations
public AlbidsAgent(String name, AlbidsRla rla)
public void addStrategy(String strategyName, Object parameters[0..*], int wei
public List getStrategies( )
public String getName( )
public AlbidsRla getReinforcementLearningAlgorithm( )
agents
AlbidsController
Attributes
private List agents
private String date
private int periods
private String host
private String facilitatorHost
private int port
private int facilitatorPort
Operations
public AlbidsController( )
public void addAgent(String agentName, String rlaName, double effectiveness, double e
public List getAgents( )
public Object[0..*,0..*] initializeSimulation(String date, int periods, String host, int port,
public void runSimulation( )
rla
strategies
AlbidsRla
Attributes
public String RLA_SIMPLE = "SimpleRla"
public String ROTH_EREV = "RothErevRla"
public String BAYESIAN_METHOD = "BayesianMethodRla"
package String rlaName
package double effectiveness
package double efficiency
Operations
public AlbidsRla(String rlaName, double effectiveness, double efficiency)
public double getEfficiency( )
public double getEffectiveness( )
public String getName( )
AlbidsStrategy
Attributes
public String ANN = "ANNStrategy"
public String SA_QL = "SAQLStrategy"
public String ERROR_THEORY_A = "ErrorTheoryStrategyA"
public String ERROR_THEORY_B = "ErrorTheoryStrategyB"
public String ERROR_THEORY_C = "ErrorTheoryStrategyC"
public String AVERAGE_1 = "AverageStrategy1"
public String AVERAGE_2 = "AverageStrategy2"
public String AVERAGE_3 = "AverageStrategy3"
public String AVERAGE_4 = "AverageStrategy4"
public String LINEAR_REGRESSION_1 = "LinearRegressionStrategy1"
public String LINEAR_REGRESSION_2 = "LinearRegressionStrategy2"
public String LINEAR_REGRESSION_3 = "LinearRegressionStrategy3"
public String LINEAR_REGRESSION_4 = "LinearRegressionStrategy4"
private String name
private Object parameters[0..*]
private int weightPercentage
Operations
public AlbidsStrategy(String name, Object parameters[0..*], int weightPercentage)
public String getName( )
public Object[0..*] getParameters( )
public void setWeightPercentage(int weightPercentage)
public int getWeightPercentage( )

Figure 3.3 – Controller and Domain classes.

As it can be seen in Figure 3.3 the *AlbidsController* class contains the list of the MASCEM agents (*AlbidsAgent* instance) to which the ALBidS system will be providing decision support. The selection of these MASCEM agents is done by the user when creating a new scenario or starting a simulation. Each *AlbidsAgent* instance contains an *AlbidsRla* instance, with the information regarding the reinforcement learning algorithm that will be used (see section 3.6 – Main Agent), and a list of the strategy agents that will contribute to the decision support of this particular MACEM agent (*AlbidsStrategy* instances).

These domain objects are created when the user chooses to initiate a simulation in MASCEM. The *controller* provides the required information to the ALBidS agents (see the sequence diagram of Figure 3.5), completing the connection between the MASCEM simulator and ALBidS.

Figure 3.4 shows the connection between the *controller* class and the ALBidS Main Agent.

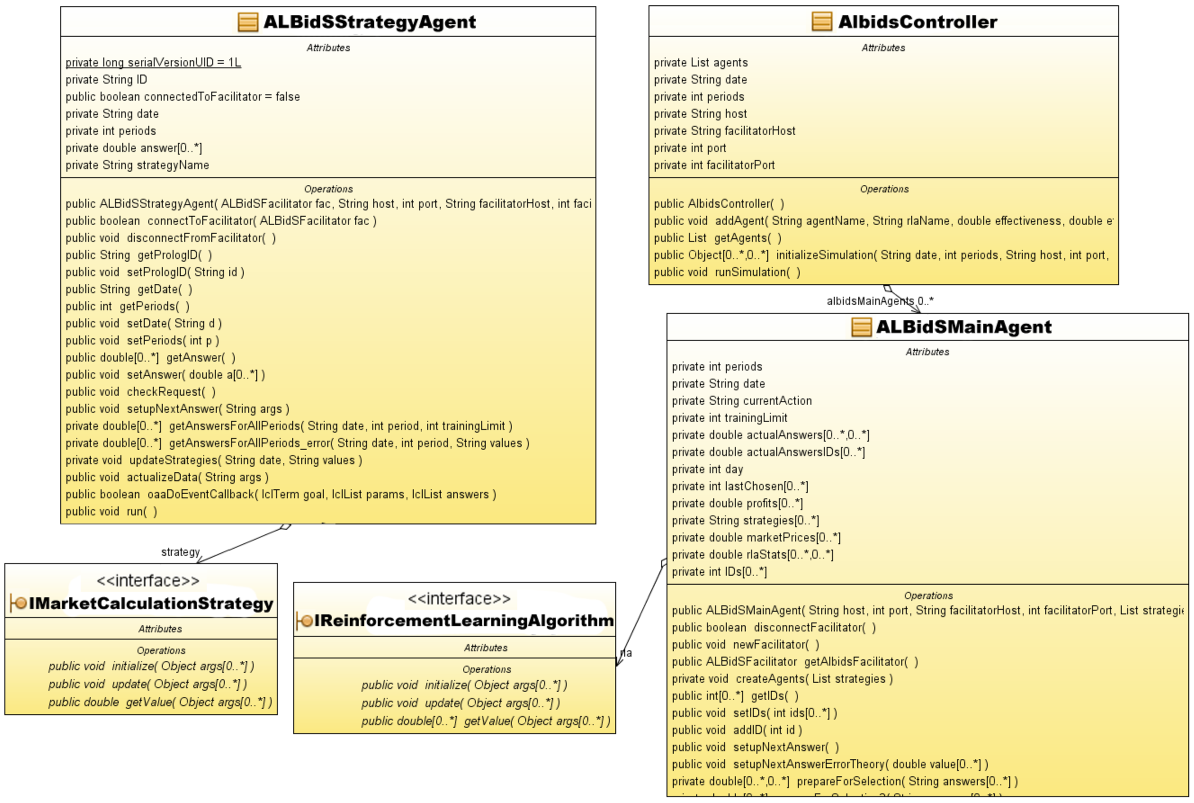

ALBidSStrategyAgent
Attributes
private long serialVersionUID = 1L
private String ID
public boolean connectedToFacilitator = false
private String date
private int periods
private double answer[0..*]
private String strategyName
Operations
public ALBidSStrategyAgent(ALBidSFacilitator fac, String host, int port, String facilitatorHost, int faci
public boolean connectToFacilitator(ALBidSFacilitator fac)
public void disconnectFromFacilitator( )
public String getPrologID( )
public void setPrologID(String id)
public String getDate( )
public int getPeriods( )
public void setDate(String d)
public void setPeriods(int p)
public double[0..*] getAnswer( )
public void setAnswer(double a[0..*])
public void checkRequest( )
public void setupNextAnswer(String args)
private double[0..*] getAnswersForAllPeriods(String date, int period, int trainingLimit)
private double[0..*] getAnswersForAllPeriods_error(String date, int period, String values)
private void updateStrategies(String date, String values)
public void actualizeData(String args)
public boolean oaaDoEventCallback(IclTerm goal, IclList params, IclList answers)
public void run( )
AlbidsController
Attributes
private List agents
private String date
private int periods
private String host
private String facilitatorHost
private int port
private int facilitatorPort
Operations
public AlbidsController( )
public void addAgent(String agentName, String rlaName, double effectiveness, double e
public List getAgents( )
public Object[0..*,0..*] initializeSimulation(String date, int periods, String host, int port,
public void runSimulation( )
albidsMainAgents 0..*
ALBidSMainAgent
Attributes
private int periods
private String date
private String currentAction
private int trainingLimit
private double actualAnswers[0..*,0..*]
private double actualAnswersIDs[0..*]
private int day
private int lastChosen[0..*]
private double profits[0..*]
private String strategies[0..*]
private double marketPrices[0..*]
private double rlaStats[0..*,0..*]
private int IDs[0..*]
Operations
public ALBidSMainAgent(String host, int port, String facilitatorHost, int facilitatorPort, List strategie
public boolean disconnectFacilitator( )
public void newFacilitator( )
public ALBidSFacilitator getAlbidsFacilitator( )
private void createAgents(List strategies)
public int[0..*] getIDs( )
public void setIDs(int ids[0..*])
public void addID(int id)
public void setupNextAnswer( )
public void setupNextAnswerErrorTheory(double value[0..*])
private double[0..*,0..*] prepareForSelection(String answers[0..*])
strategy
<<interface>>
IMarketCalculationStrategy
Attributes
Operations
public void initialize(Object args[0..*])
public void update(Object args[0..*])
public double getValue(Object args[0..*])
<<interface>>
IReinforcementLearningAlgorithm
Attributes
Operations
public void initialize(Object args[0..*])
public void update(Object args[0..*])
public double[0..*] getValue(Object args[0..*])
rla

Figure 3.4 – Controller and ALBidS agents' classes.

The *ALBidSMainAgent* class creates an agent for each strategy (instances of the *ALBidSStrategyAgent* class), which will be responsible for executing the corresponding strategy.

Figure 3.5 contains a sequence diagram, demonstrating the full system interaction, from MASCEM to ALBidS, starting from the request for the beginning of a simulation. Figure 3.6 presents the detailed process of the *initializeSimulation* method, of the *AlbidsController* class, where the creation of the ALBidS system classes takes place.

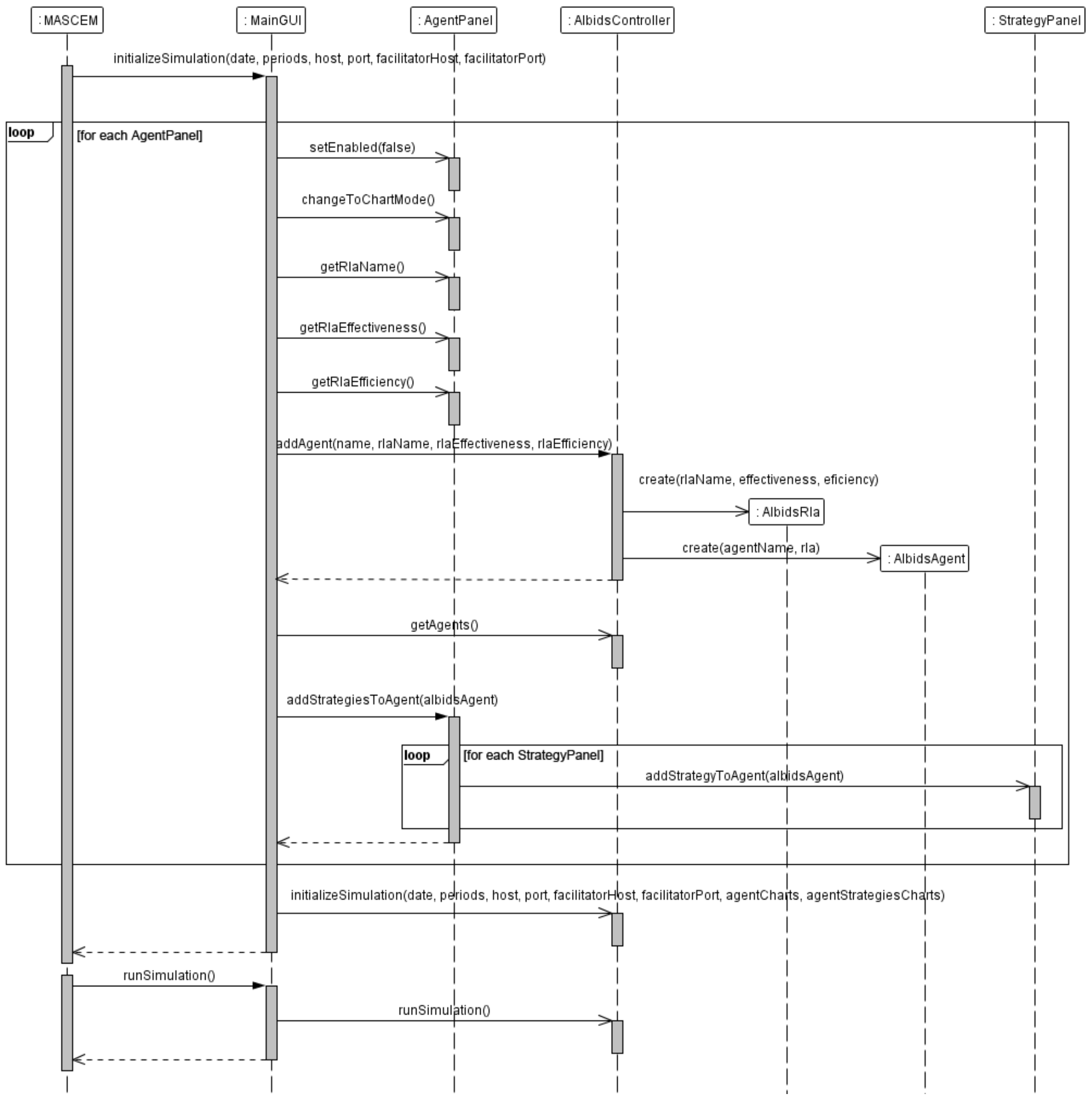

: MASCEM
: MainGUI
: AgentPanel
: AlbidsController
: StrategyPanel
initializeSimulation(date, periods, host, port, facilitatorHost, facilitatorPort)
loop
[for each AgentPanel]
setEnabled(false)
changeToChartMode()
getRlaName()
getRlaEffectiveness()
getRlaEfficiency()
addAgent(name, rlaName, rlaEffectiveness, rlaEfficiency)
create(rlaName, effectiveness, eficiency)
: AlbidsRla
create(agentName, rla)
: AlbidsAgent
getAgents()
addStrategiesToAgent(albidsAgent)
loop
[for each StrategyPanel]
addStrategyToAgent(albidsAgent)
initializeSimulation(date, periods, host, port, facilitatorHost, facilitatorPort, agentCharts, agentStrategiesCharts)
runSimulation()
runSimulation()

Figure 3.5 – Sequence diagram of when a simulation is started.

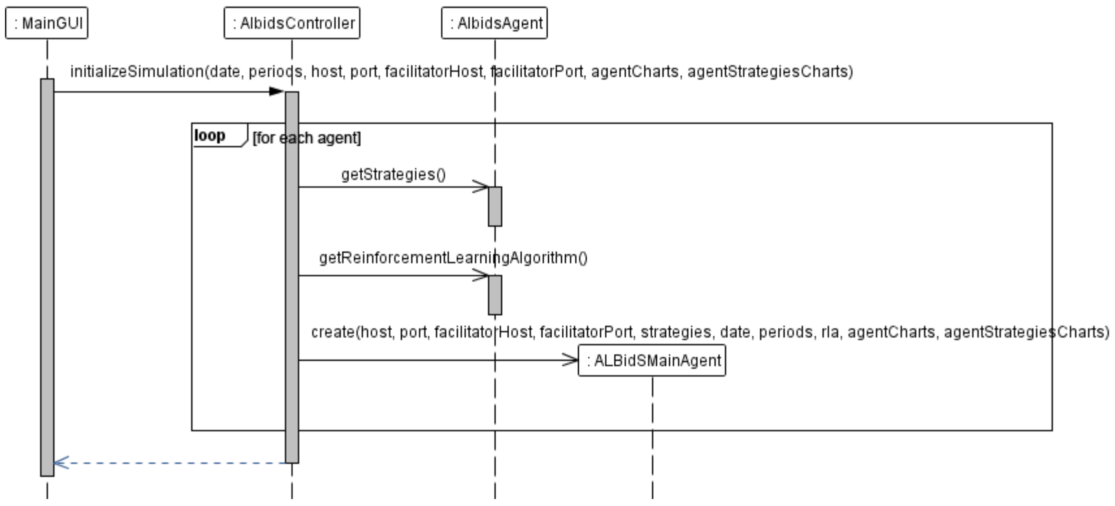

: MainGUI
: AlbidsController
: AlbidsAgent
initializeSimulation(date, periods, host, port, facilitatorHost, facilitatorPort, agentCharts, agentStrategiesCharts)
loop
[for each agent]
getStrategies()
getReinforcementLearningAlgorithm()
create(host, port, facilitatorHost, facilitatorPort, strategies, date, periods, rla, agentCharts, agentStrategiesCharts)
: ALBidSMainAgent

Figure 3.6 – ALBidS classes instantiation.

From an agents' perspective this system is composed by the following entities, which will be presented throughout this chapter: Main Agent, Prolog Facilitator Agent, Game Theory Agent, Regression Agents, Composed Goal Directed Agent, Adapted Derivative Following Agent, Market Price Following Agent, NN Agent, AMES Agent, Average Agents, SA-QL Agent, Error Theory Agents, Economic Analysis Agent, Determinism Theory Agent, and Metalearner Agents.

ALBidS is connected with the MASCEM simulator, providing a response to the negotiating players when they require intelligent support to act in the market. The connection between the two systems is managed by the Main Agent, using the Prolog Facilitator. This agent, presented in section 3.6, acts as an intermediary between the two systems. It receives requests from the negotiating players when they require decision support, and provides them the corresponding answers. These answers are provided after managing the ALBidS internal mechanism, including the interactions with the strategy agents (or bid proposal agents) – the agents responsible for executing the different strategies. The ALBidS integration with the MASCEM simulator is presented in Figure 3.7.

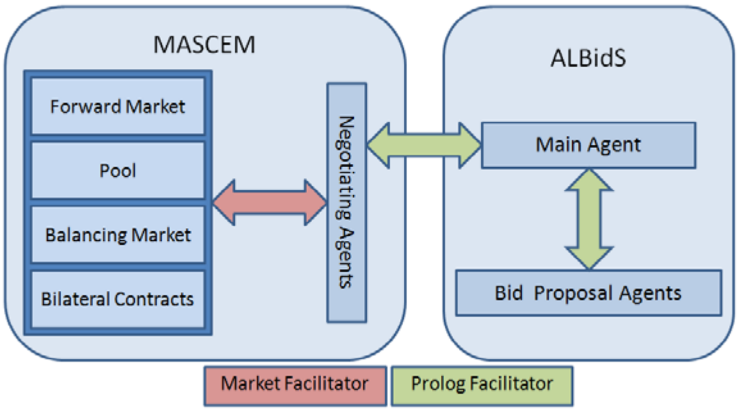


Figure 3.7 – ALBidS integration with MASCEM [Pinto et al., 2011c].

The communications between the agents of this system are managed by the Prolog Facilitator presented in the section 3.3, this way making it possible for this system to be integrated in MASCEM, with the communications treated separately from the rest of the simulation, guaranteeing the independence and parallelism between the distinct groups of agents.

3.3. ALBidS Facilitator

Being ALBidS a multi-agent system with an independent purpose from the MASCEM simulator, with its agents' interactions irrelevant to the functioning of the agents of MASCEM, and vice-versa, it became evident that this system should be managed independently from the rest of the simulation, from a communication perspective [Pinto *et al.*, 2011a]. This means having the communications between agents independent from those of MASCEM, to ensure the parallel processing from the two groups of agents. To achieve this, it became necessary to assign the ALBidS communications management to an independent facilitator.

Due to the OAA [OAA, 2011] restrictions with the use of more than one facilitator simultaneously [Vale *et al.*, 2011b], it was decided that a new version of the OAA facilitator, to manage the ALBidS agents' communications should be developed. ALBidS facilitators are implemented in LPA Prolog [Prolog, 2011], as it guarantees a level of efficiency and speed of processing that JAVA [JAVA, 2011] cannot give. ALBidS facilitators have been implemented so they seem as an adaptation of the OAA facilitator's basic functionalities [Pinto *et al.*, 2009]. For this purpose, it has been decided to maintain as long as possible the name of the OAA facilitator methods, so that it would grant an easier adaptation from people who are used to the OAA. The ALBidS facilitator allows the agents to:

- connect to the facilitator, sending the list of their capabilities;
- disconnect from the facilitator, and so, from that point on, not being asked to solve requests from this facilitator;
- send "solve" requests;
- send "solve" requests for a specific agent;
- get the agents' answers to a solvable;
- check for requests to solve;
- respond to a "solve" request;
- get information about:
  - the list of capabilities of each agent that is connected to this facilitator;
  - the agents that already replied to a current request;
  - the agents that did not yet give their response;
  - all the actions that are being solved;
  - the historic of all the actions that were requested, the agents that replied to each of them, and respective answers.

ALBidS facilitators are implemented in a Prolog file named facilitator.pl (Figure 3.8), which is integrated in the JAVA program by using the LPA Win-Prolog Intelligence Server [IS, 2011]. The LPA Intelligence Server provides a Dynamic-Link Library (DLL) interface between Win-Prolog and other applications, and so it allows the LPA-based components to be embedded in applications written in many Windows programming languages.

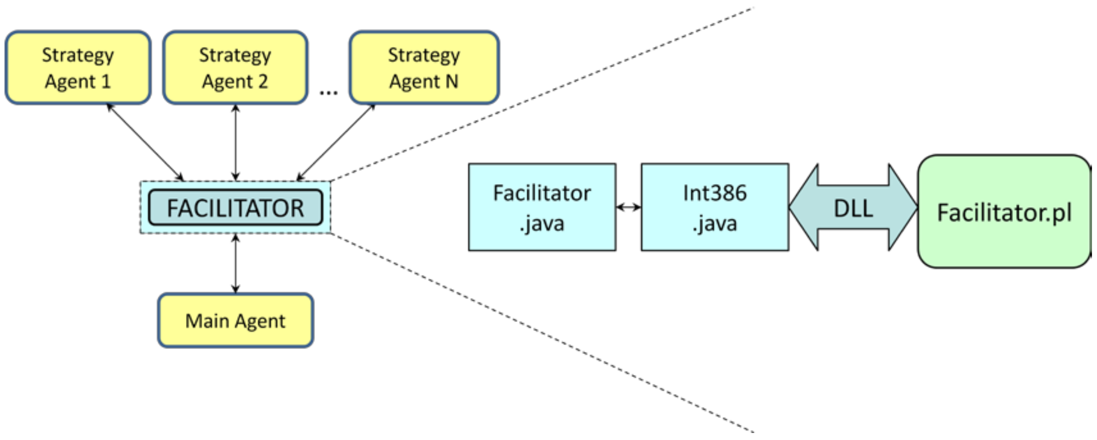


Figure 3.8 – ALBidS Facilitator.

Adding the Intelligence Server functionality is obtained by importing the provided lib file which defines the Intelligence Server function prototypes for the DLL. Then, just calling Prolog goals as a string, retrieves the results. These are also provided in the form of a string. The first parameter is the return value of the goal (true or false), after this, the console output results. So, the way to send back results other than true/false information is to write the results to the console, and they will be returned together with the result of the goal.

The native code interface to the Intelligent Server is represented in the class *Int386w*. The functions are accessed through an instance of this class.

The *facilitator* class, to which the agents have access, provides an easier call to the Prolog facilitator goals. This class contains the methods that allow the agents to:

- connect to an existing instance of the Prolog facilitator;
- create a new instance of the Prolog facilitator;
- simplify the calls of the goals. As mentioned before, the usage of the *Int386w* class to establish the connection to the Intelligence Server requires using the methods of this class to access the Prolog (facilitator) goals. To make it easier to use and similar to the OAA facilitator, the facilitator class provides the methods to make the calls more intuitive. So, to solve a goal, instead of having to make the calls:
    - Int386w.initGoal("solve(goal)") ;
    - Int386w.callGoal() ;
    - Int386w.exitGoal() ;

With this class it is just needed to call:

- facilitator.solve("goal").

Still with the intention of simplifying, this class also contains the necessary methods for the agent to get the results of an asked goal in the most appropriate way, by transforming the returned string into variables that can be easily understood by the agent. Figures 3.9 and 3.10 provide a better understanding of how these communications are processed.

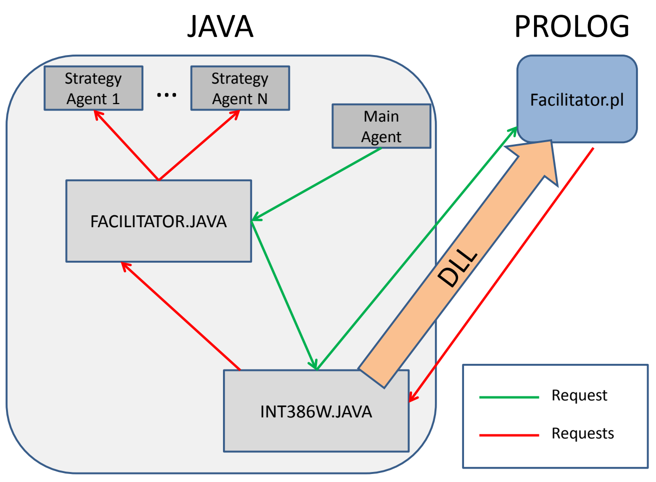


Figure 3.9 – Main Agent "Solve" Request.

Figure 3.9 shows an example of how the requests to solve a certain action are processed. The Main Agent sends its request to the JAVA *facilitator* class, which through the calls of the necessary methods of the *Int386w* class (initializing the goal, calling it, and then exiting the call) makes the request get to the Prolog facilitator. When the facilitator receives the request, it checks the agents that are connected and that have the ability to solve it, and then sends the request to be solved to each of those agents, via the same way – *Int386w* class, and JAVA *facilitator* class.

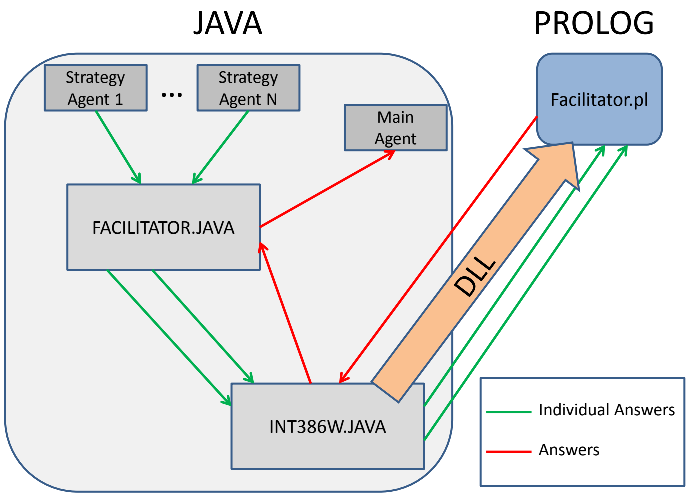


Figure 3.10 – Answer to a Request.

Once the agents get the request to solve the action, the reverse process occurs (Figure 3.10), they solve it and return each individual result to the java facilitator class, which makes sure it gets to the Prolog facilitator, through the necessary calls of the *Int386w* class once more.

When the Prolog facilitator receives the answers, it combines them and sends the final answer back to the Main Agent, which initially made the request.

3.4. User Interface

According to the defined architecture – MVC (section 3.2), the user interface (view) must communicate with a *controller*, which will manipulate the domain objects. This way it is possible to deal with separate computer code from the interface, business and model. Additionally, the possibility to change the graphic interface in the future, without the need to change the structure of the learning system, is facilitated.

The user interface must be simple and intuitive to the user, so that he/she does not feel "lost" in it. It must provide the means to choose for each MASCEM agent using ALBidS: the desired efficiency/effectiveness balance (see section 3.9), the desired reinforcement learning algorithm (see section 3.6), and the strategies it will use, with the required parameters. Once a simulation starts, to facilitate the user's accompanying of the happenings regarding this system, several graphs are presented, namely:

- Global graphs, presenting:
  - A comparison between all the agents' proposed bids, and the actual market price, to provide the user with the chance to follow all the agents proposed actions;
  - A comparison between the confidence values of all strategies, regarding the reinforcement learning algorithm (see section 3.6), allowing the user to see which agents are gaining an higher importance in the system;
  - A comparison between the execution times of each agent when running the corresponding strategies, indicating which are having more influence over the simulation's eventual time degradation (important for the efficiency/effectiveness management. See section 3.9).
- Individual graphs for each strategy agent, displaying its proposed bids, and, in case of its proposal being chosen as the system's final response, the profit it originated to the MASCEM agent to which ALBidS is providing decision support to.

All the values are presented, not throughout the day of the simulation, but over the sequence of periods under a defined context (see section 3.5). For example, if a certain period is defined as being one context, indicating that throughout the days, this period presents similar characteristics in what concerns the actual market results; the reinforcement learning algorithm will consider all the negotiations occurring in this period as independent events from the rest, and so, it will decide on the best proposal to choose, taking into account only the strategy agents' results concerning this period. For this reason, it becomes much more interesting to analyze the results along events of each context, and effectively understanding the system's development, than following the results throughout each day, when many of the periods' results of that day certainly will have no relation with each other, from the system's progression point of view.

The user interface's creation is triggered by MASCEM, in the way that Figure 3.11 presents.

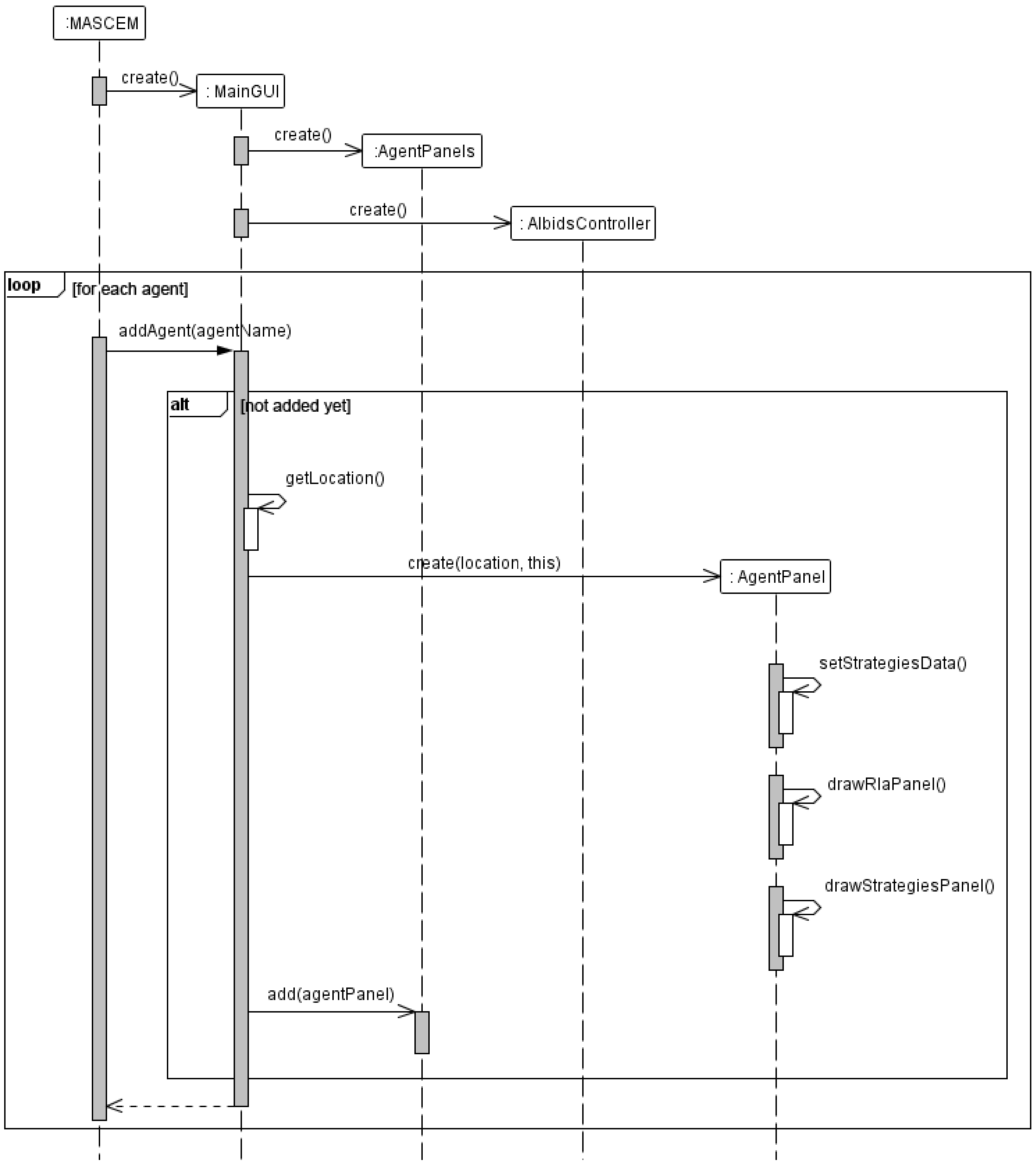
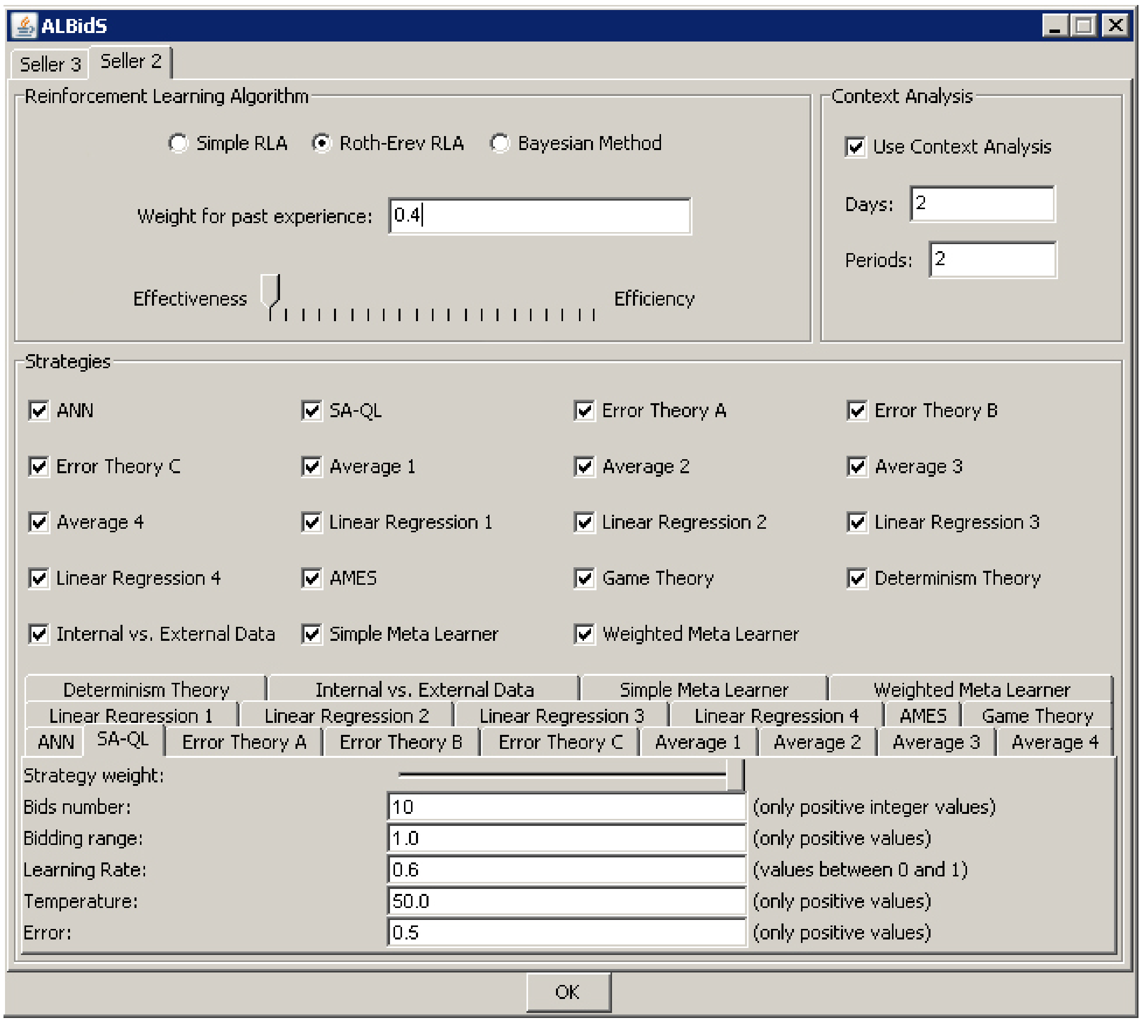


Figure 3.11 – ALBidS' user interface creation.

After constant improvements throughout its development [Sousa, 2011], the implemented interface, regarding the necessary inputs, resulted on the look presented in Figure 3.12.



Simple RLA [✓ Roth-Erev RLA] Bayesian Method

Weight for past experience: 0.4

[✓ Use Context Analysis]

Days: 2

Periods: 2

[✓ ANN]	[✓ SA-QL]	[✓ Error Theory A]	[✓ Error Theory B]
[✓ Error Theory C]	[✓ Average 1]	[✓ Average 2]	[✓ Average 3]
[✓ Average 4]	[✓ Linear Regression 1]	[✓ Linear Regression 2]	[✓ Linear Regression 3]
[✓ Linear Regression 4]	[✓ AMES]	[✓ Game Theory]	[✓ Determinism Theory]
[✓ Internal vs. External Data]	[✓ Simple Meta Learner]	[✓ Weighted Meta Learner]	

Strategy weight:		
Bids number:	10	(only positive integer values)
Bidding range:	1.0	(only positive values)
Learning Rate:	0.6	(values between 0 and 1)
Temperature:	50.0	(only positive values)
Error:	0.5	(only positive values)

Figure 3.12 – ALBidS user display.

In the example presented in Figure 3.12, ALBidS will be supporting two MASCEM agents (Seller 2 and Seller 3), as can be seen by the two tabs located on the top. In the reinforcement learning algorithm panel, it is possible to choose the desired algorithm, define its parameters, and define the efficiency/effectiveness balance. In the context analysis panel it is possible to choose whether the Context Analysis mechanism (presented in section 3.5) will be used in the simulation, and in case of it being chosen, it provides the means to define the number of days and periods that will be used by this mechanism. Finally, in the strategy panel, it is possible to select the strategies to use, and define their parameters. By omission all the strategies are selected, and when one is selected or deselected, the corresponding tab where the parameters are defined is created or eliminated.

The content of the strategies tabs changes for each strategy, depending on the required parameters. To make it more dynamic, and easing future changes, an abstract class named *StrategyPanel* has been created. Each strategy has its own panel, represented by a derivate class from *StrategyPanel*, containing the specific attributes and methods of each strategy, which are not shared by others. The common ones are provided by the abstract class.

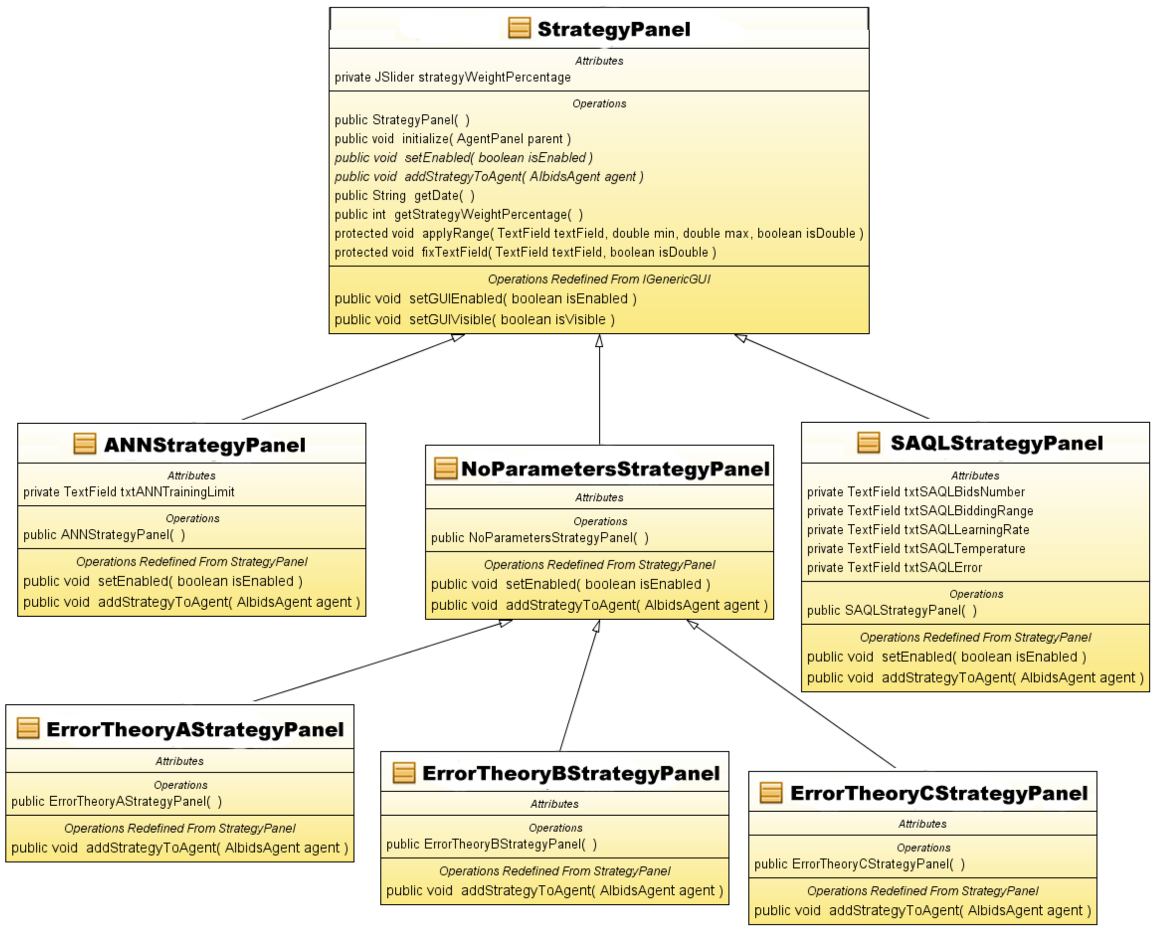

StrategyPanel
Attributes
private JSlider strategyWeightPercentage
Operations
public StrategyPanel( )
public void initialize(AgentPanel parent)
public void setEnabled(boolean isEnabled)
public void addStrategyToAgent(AlbidsAgent agent)
public String getDate( )
public int getStrategyWeightPercentage( )
protected void applyRange(TextField textField, double min, double max, boolean isDouble)
protected void fixTextField(TextField textField, boolean isDouble)
Operations Redefined From IGenericGUI
public void setGUIEnabled(boolean isEnabled)
public void setGUIVisible(boolean isVisible)
ANNStrategyPanel
Attributes
private TextField txtANNTrainingLimit
Operations
public ANNStrategyPanel( )
Operations Redefined From StrategyPanel
public void setEnabled(boolean isEnabled)
public void addStrategyToAgent(AlbidsAgent agent)
NoParametersStrategyPanel
Attributes
Operations
public NoParametersStrategyPanel( )
Operations Redefined From StrategyPanel
public void setEnabled(boolean isEnabled)
public void addStrategyToAgent(AlbidsAgent agent)
SAQLStrategyPanel
Attributes
private TextField txtSAQLBidsNumber
private TextField txtSAQLBiddingRange
private TextField txtSAQLLearningRate
private TextField txtSAQLTemperature
private TextField txtSAQLError
Operations
public SAQLStrategyPanel( )
Operations Redefined From StrategyPanel
public void setEnabled(boolean isEnabled)
public void addStrategyToAgent(AlbidsAgent agent)
ErrorTheoryAStrategyPanel
Attributes
Operations
public ErrorTheoryAStrategyPanel( )
Operations Redefined From StrategyPanel
public void addStrategyToAgent(AlbidsAgent agent)
ErrorTheoryBStrategyPanel
Attributes
Operations
public ErrorTheoryBStrategyPanel( )
Operations Redefined From StrategyPanel
public void addStrategyToAgent(AlbidsAgent agent)
ErrorTheoryCStrategyPanel
Attributes
Operations
public ErrorTheoryCStrategyPanel( )
Operations Redefined From StrategyPanel
public void addStrategyToAgent(AlbidsAgent agent)

Figure 3.13 – StrategyPanel class and its derivate classes.

In Figure 3.13 it is also visible that the *NoParametersStrategyPanel* class, subclass of *StrategyPanel*, is also abstract. It is used to represent the strategies which require no definition of input parameters.

The user interface is represented by the *MainGUI* class, containing several instances of the *AgentPanel* class, which represents the panel for the definition of the preference values referring to each MASCEM agent. The *AgentPanel* class contains a list of the *StrategyPanel* instances, the strategies that will be used to support this agent's decisions, as presented in Figure 3.14.

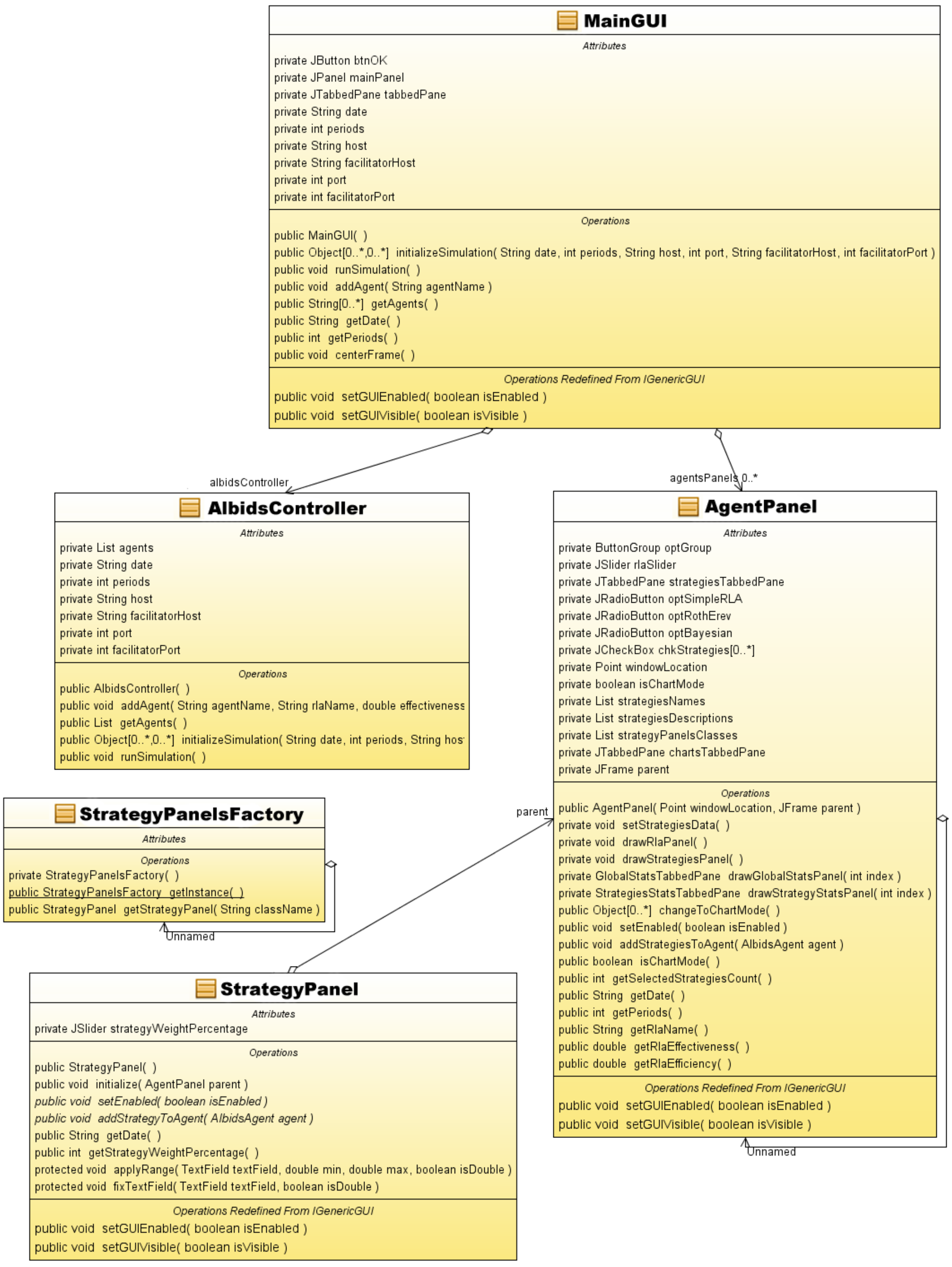

MainGUI
Attributes
private JButton btnOK
private JPanel mainPanel
private JTabbedPane tabbedPane
private String date
private int periods
private String host
private String facilitatorHost
private int port
private int facilitatorPort
Operations
public MainGUI( )
public Object[0..*,0..*] initializeSimulation(String date, int periods, String host, int port, String facilitatorHost, int facilitatorPort)
public void runSimulation( )
public void addAgent(String agentName)
public String[0..*] getAgents( )
public String getDate( )
public int getPeriods( )
public void centerFrame( )
Operations Redefined From IGenericGUI
public void setGUIEnabled(boolean isEnabled)
public void setGUIVisible(boolean isVisible)
albidsController
agentsPanels 0..*
AlbidsController
Attributes
private List agents
private String date
private int periods
private String host
private String facilitatorHost
private int port
private int facilitatorPort
Operations
public AlbidsController( )
public void addAgent(String agentName, String rlaName, double effectiveness
public List getAgents( )
public Object[0..*,0..*] initializeSimulation(String date, int periods, String hos
public void runSimulation( )
AgentPanel
Attributes
private ButtonGroup optGroup
private JSlider rlaSlider
private JTabbedPane strategiesTabbedPane
private JRadioButton optSimpleRLA
private JRadioButton optRothErev
private JRadioButton optBayesian
private JCheckBox chkStrategies[0..*]
private Point windowLocation
private boolean isChartMode
private List strategiesNames
private List strategiesDescriptions
private List strategyPanelsClasses
private JTabbedPane chartsTabbedPane
private JFrame parent
Operations
public AgentPanel(Point windowLocation, JFrame parent)
private void setStrategiesData( )
private void drawRlaPanel( )
private void drawStrategiesPanel( )
private GlobalStatsTabbedPane drawGlobalStatsPanel(int index)
private StrategiesStatsTabbedPane drawStrategyStatsPanel(int index)
public Object[0..*] changeToChartMode( )
public void setEnabled(boolean isEnabled)
public void addStrategiesToAgent(AlbidsAgent agent)
public boolean isChartMode( )
public int getSelectedStrategiesCount( )
public String getDate( )
public int getPeriods( )
public String getRlaName( )
public double getRlaEffectiveness( )
public double getRlaEfficiency( )
Operations Redefined From IGenericGUI
public void setGUIEnabled(boolean isEnabled)
public void setGUIVisible(boolean isVisible)
Unnamed
parent
StrategyPanelsFactory
Attributes
Operations
private StrategyPanelsFactory( )
public StrategyPanelsFactory getInstance( )
public StrategyPanel getStrategyPanel(String className)
Unnamed
StrategyPanel
Attributes
private JSlider strategyWeightPercentage
Operations
public StrategyPanel( )
public void initialize(AgentPanel parent)
public void setEnabled(boolean isEnabled)
public void addStrategyToAgent(AlbidsAgent agent)
public String getDate( )
public int getStrategyWeightPercentage( )
protected void applyRange(TextField textField, double min, double max, boolean isDouble)
protected void fixTextField(TextField textField, boolean isDouble)
Operations Redefined From IGenericGUI
public void setGUIEnabled(boolean isEnabled)
public void setGUIVisible(boolean isVisible)

Figure 3.14 – Part of the class diagram.

The *MainGUI* class also contains an instance of the *AlbidsController* class, which, as referred in section 3.2, is responsible for making the connection with the rest of the ALBidS system, by manipulating the domain objects, and the ALBidS' agents.

After all user decisions are made, and the simulation is started, the ALBidS interface displays the graphs presenting the simulation data. As the ALBidS system is responsible for "feeding" the graphs, it was found necessary to send the graphs instances by parameter from the interface to the core ALBidS system, at the time of the simulation beginning request. This way the graphs can be directly manipulated by ALBidS.

When a simulation is started, the user interface changes its aspect, hiding the input options presented at first and giving place to the simulation data graphs. When the window changes its appearance, it keeps presenting one tab per MASCEM agent, and for each of those, one tab concerning each context or period.

There are two types of graphs: the global stats graphs, providing a global view of the simulation; and the strategy individual graphs, presenting the information regarding each of the strategies. The graphs are grouped in panels, once again separated by tabs. This second mode of the interface is showed in Figure 3.15.

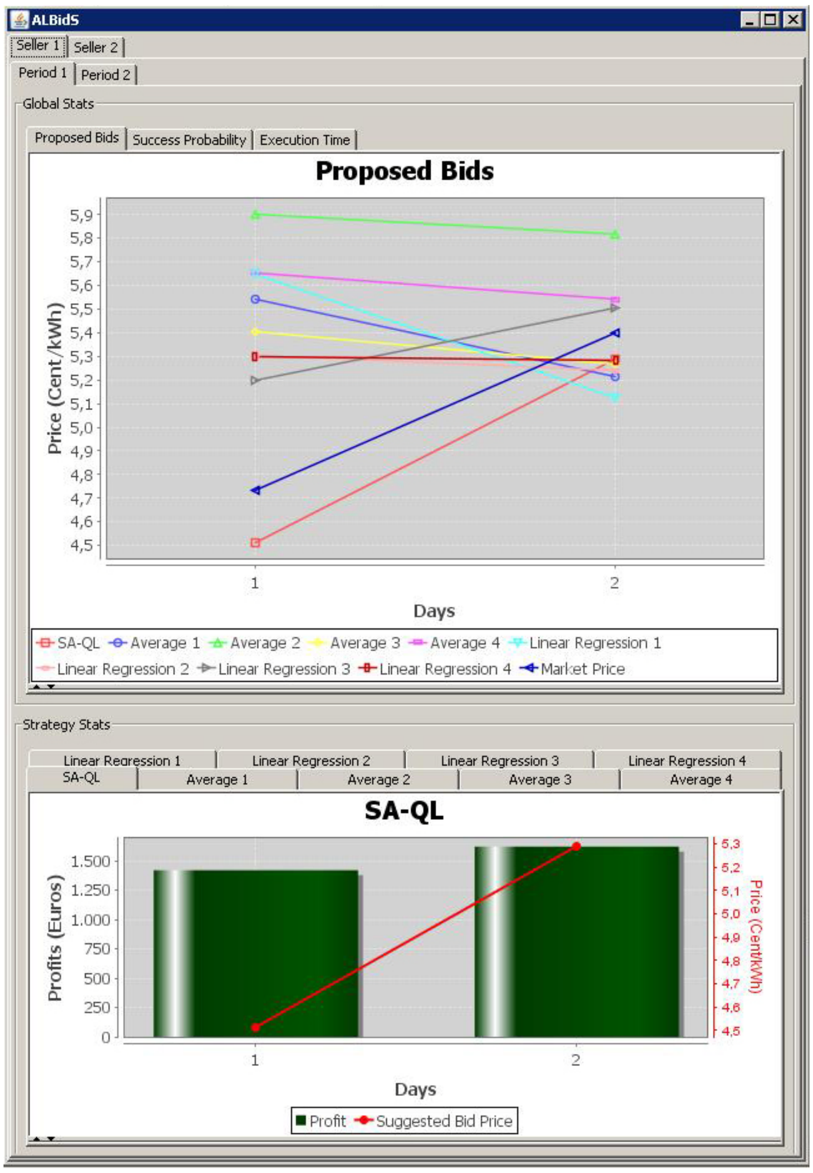


Figure 3.15 – ALBidS interface module for simulation results display.

In the first panel, presenting the global stats, the three required graphs are displayed, presenting: all strategies' bids, and the actual market price, comparing the different strategies' proposals; the performance of each strategy, concerning the reinforcement learning algorithm; and the comparison of the execution times of all strategies.

The second panel presents the individual strategy graphs, displaying for each one the bid prices and the obtained profits. The class diagram concerning the graphs is presented in Figure 3.16.

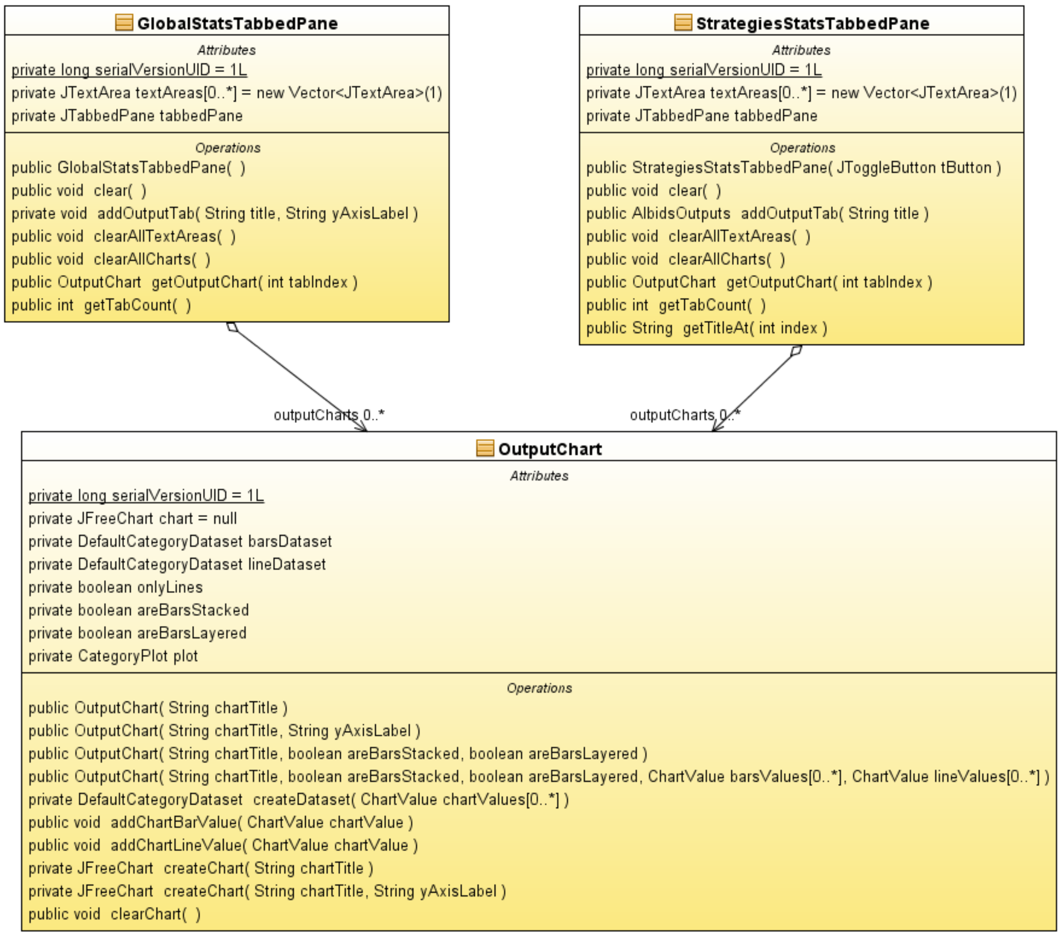

GlobalStatsTabbedPane
Attributes
private long serialVersionUID = 1L
private JTextArea textAreas[0..*] = new Vector<JTextArea>(1)
private JTabbedPane tabbedPane
Operations
public GlobalStatsTabbedPane( )
public void clear( )
private void addOutputTab(String title, String yAxisLabel)
public void clearAllTextAreas( )
public void clearAllCharts( )
public OutputChart getOutputChart(int tabIndex)
public int getTabCount( )
StrategiesStatsTabbedPane
Attributes
private long serialVersionUID = 1L
private JTextArea textAreas[0..*] = new Vector<JTextArea>(1)
private JTabbedPane tabbedPane
Operations
public StrategiesStatsTabbedPane(JToggleButton tButton)
public void clear( )
public AlbidsOutputs addOutputTab(String title)
public void clearAllTextAreas( )
public void clearAllCharts( )
public OutputChart getOutputChart(int tabIndex)
public int getTabCount( )
public String getTitleAt(int index)
outputCharts 0..*
outputCharts 0..*
OutputChart
Attributes
private long serialVersionUID = 1L
private JFreeChart chart = null
private DefaultCategoryDataset barsDataset
private DefaultCategoryDataset lineDataset
private boolean onlyLines
private boolean areBarsStacked
private boolean areBarsLayered
private CategoryPlot plot
Operations
public OutputChart(String chartTitle)
public OutputChart(String chartTitle, String yAxisLabel)
public OutputChart(String chartTitle, boolean areBarsStacked, boolean areBarsLayered)
public OutputChart(String chartTitle, boolean areBarsStacked, boolean areBarsLayered, ChartValue barsValues[0..*], ChartValue lineValues[0..*])
private DefaultCategoryDataset createDataset(ChartValue chartValues[0..*])
public void addChartBarValue(ChartValue chartValue)
public void addChartLineValue(ChartValue chartValue)
private JFreeChart createChart(String chartTitle)
private JFreeChart createChart(String chartTitle, String yAxisLabel)
public void clearChart( )

Figure 3.16 – Part of the class diagram, presenting the classes of each graph panel.

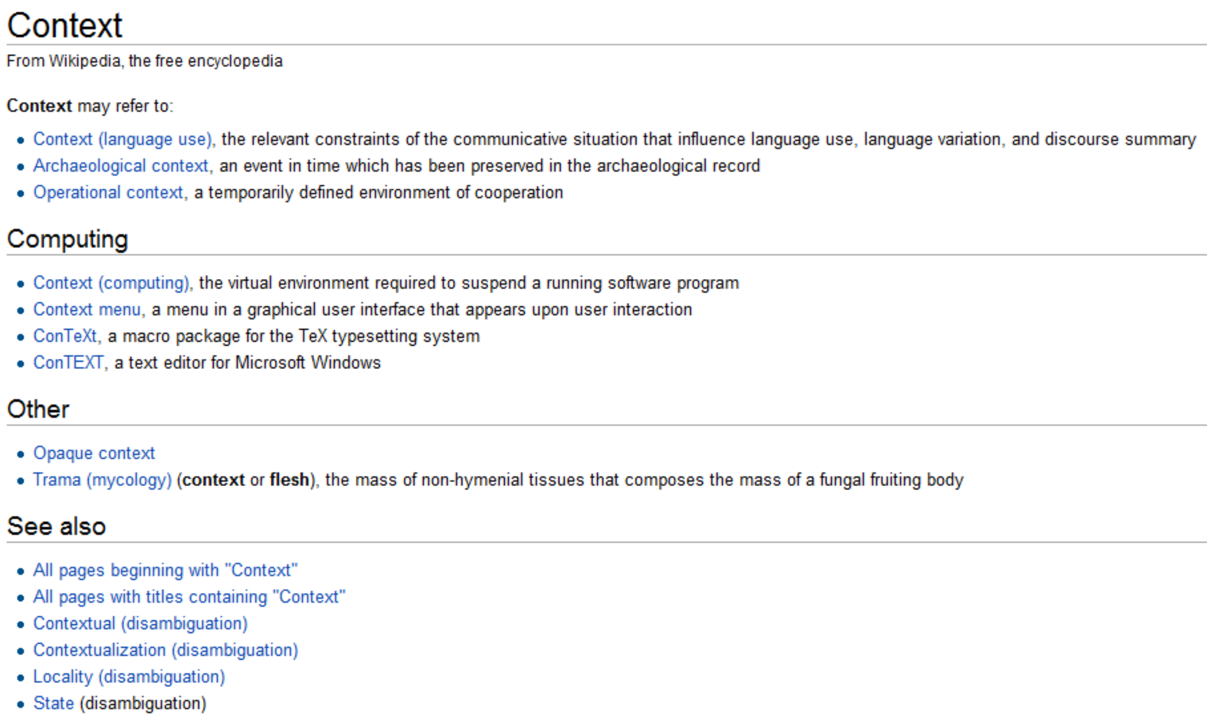
3.5. Context Analysis

Regardless the scope of where it is acting, when a certain subject desires or needs to take the maximum advantage on its surrounding environment, a full understanding of that environment's characteristics and particularities is a critical issue. These characteristics' particularities and conditions define the circumstances in which a certain event exists or occurs inside the considered environment. This is called context.

The human mind instinctively uses context to manage a massive amount of information from different every-day situations, such as work, community, family and friends. Using context, people quickly decipher what information is relevant in a given situation. As people, we can recognize the contexts we are in, know what information is applicable to each context, and derive information from each context.

In fact, not only humans, but also animals use context instinctively. For example, when a lion is hunting, a specific set of conditions and characteristics in its environment must be gathered in order for it to strike its prey, *e.g.* if the wind is on the opposite direction, to avoid the prey smelling it; and if there are no other predators that may put the hunting at risk.

Context is present everywhere, and unquestionably influences the way information is processed in every situation [Dekel *et al.*, 2009], [Pal, 2010]. In fact, the definition of the own word "context" is dependent on the context itself! When searching for the definition of context in Wikipedia[3] [Wiki, 2011a], a set of different results, depending on the context the word is used, was found (Figure 3.17).



Context

From Wikipedia, the free encyclopedia

Context may refer to:

- Context (language use), the relevant constraints of the communicative situation that influence language use, language variation, and discourse summary
- Archaeological context, an event in time which has been preserved in the archaeological record
- Operational context, a temporarily defined environment of cooperation

Computing

- Context (computing), the virtual environment required to suspend a running software program
- Context menu, a menu in a graphical user interface that appears upon user interaction
- ConTeXt, a macro package for the TeX typesetting system
- ConTEXT, a text editor for Microsoft Windows

Other

- Opaque context
- Trama (mycology) (**context** or **flesh**), the mass of non-hymenial tissues that composes the mass of a fungal fruiting body

See also

- All pages beginning with "Context"
- All pages with titles containing "Context"
- Contextual (disambiguation)
- Contextualization (disambiguation)
- Locality (disambiguation)
- State (disambiguation)

Figure 3.17 – Definition of context depending on the context [Wiki, 2011b].

While context is critical to information processing in all kinds of situations, it is almost fully absent from the modern information technology infrastructure. There is some work done in providing computer systems with context awareness [Sama *et al.*, 2010], namely in multiagent simulation [Padovitz *et al.*, 2010]. However, the concept of context awareness is very far from being widely used in the computer system's area. The fact that this is an important issue to consider and its lack of consideration in decision support systems made it essential to include context analysis in this work. The analysis and definition of different contexts of negotiation in the electricity market are performed to support an adequate acting of negotiating players, adapting their actions to best suit the context they are encountering at each moment.

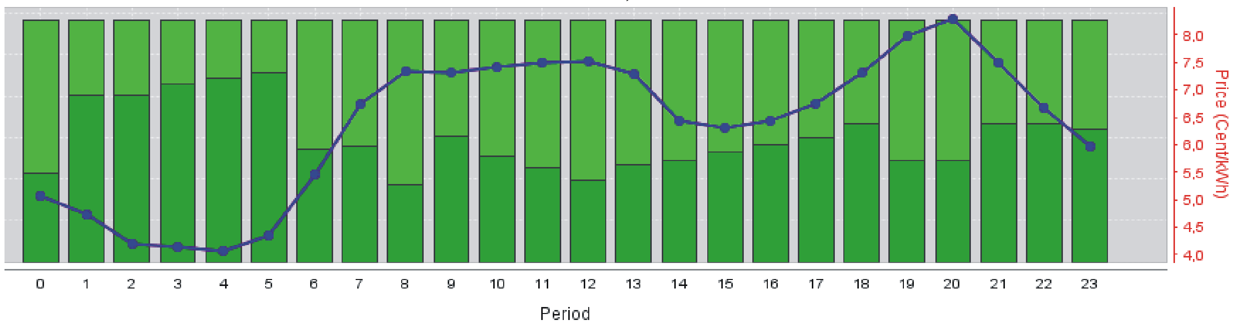
The first step when analyzing context in the electricity market environment is to consider its most basic conditionings, *i.e.* on what these negotiations depend: days and periods. These are the two main factors to consider

[3] Although being a fact that Wikipedia cannot be considered the most reliable source, the way this web page presented the results for the "Context" definition was found interesting, as it shows the usage differences of the word "Context", depending itself on the context of utilization. This is the reason why this figure and consequent definitions were considered relevant to be presented in this scope. The adequacy of the definitions is supported by references (dictionaries) of higher relevance, such as [Oxford, 2011] and [Houaiss, 2001].

when bidding in the market, since each bid much be submitted for each period of each day (see sub-section 2.2.1.1 - Spot Market).

Figure 3.18 a) presents the comparison between the typical market price curve along the 24 one hour periods of one day. Figure 3.18 b) presents the price curve and along the days, considering the same period. Both these curves were taken from MASCEM's market simulations, using real data from the Iberian electricity market – OMEL [OMEL, 2011].

a)



b)

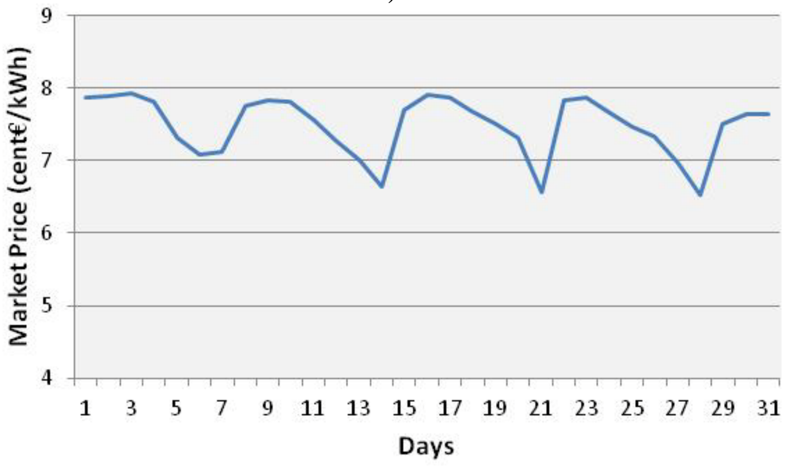


Figure 3.18 – Evolution of the market price throughout: a) the periods of one day, b) the days concerning one period.

When analyzing the typical market price throughout the periods of one day, it is visible that the market prices' variation is much higher than throughout the days for a certain period. In the first case, for the presented example, the market prices vary from 4.0 to 8.0 cent€/kWh, while in the second case they only vary from 6.5 to 8 cent€/kWh. The difference in these tendencies is behind the first approach of this work when dealing with data analysis. This first approach in considering contexts concerned a data analysis throughout the days, for each period independently, taking advantage on the prices much lower variation, and so facilitating the prices forecast process.

The developed context definition process takes into consideration the analysis of the situations concerning both perspectives, evolution throughout the days and throughout day periods. To perform this analysis, some influential conditionings that affect the prices in both cases were considered. The considered conditionings, or characteristics of a day and period are:

- the market price for the period and day in matter;
- the amount of transacted power in the market;

- the wind intensity verified in that period of the day (this is important because it affects the production of wind plants, and therefore the total negotiated amount of power);
- the type of the day (whether it is a working day or weekend; if it is a holiday, or a special situation day, *e.g.* a day of an important event, such as an important game in a certain sport, which affects the energy consumption in that day, both because of the consumption in the stadium, and for increasing the number of people with the TV on to watch it).

The grouping of a day's periods depending on their context is performed through the application of a clustering mechanism. The clustering mechanism analyses the characteristics of each period throughout the days, and attributes each period to the cluster that presents the most similar characteristics. The clustering is performed using the K-Means clustering algorithm provided by MatLab [MatLab, 2011]. Table 3.1 presents the results of the tests performed to three distinct data sets taken from OMEL. For each data set the clustering algorithm is applied, using a matrix where the lines represent the periods, and the columns represent days. For each day, the information regarding the four conditionings presented before is used. The considered number of days for analyzing the periods, and the number of clusters are variable, depending on the preference for efficiency/effectiveness balance (see section 3.9). For this case, the tests are performed for two and three clusters, and for seven (one week), fourteen (two weeks) and thirty one (one month) days. The table presents a number and associated colour, in each cell, which represent a different cluster. The rows represent the different periods of one day, and the columns represent different days (depending on the data set, and the type of clustering applied – whether using two or three clusters, and using 7, 14 or 31 days for the clustering process).

Table 3.1 – Clustering results for grouping periods with similar characteristics.

	Days	7						14						30					
	Clusters	2			3			2			3			2			3		
Period	Data Set	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1		1	1	1	3	3	1	1	1	1	3	3	1	1	1	1	3	3	3
2		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7		1	1	1	1	1	3	1	1	1	1	1	3	1	1	1	1	1	1
8		1	1	2	3	3	3	1	1	2	3	3	3	1	1	1	3	3	3
9		1	1	2	3	3	3	1	1	2	3	3	3	1	1	1	3	3	3
10		2	2	2	2	2	3	2	1	2	2	2	2	2	2	2	2	2	2
11		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
12		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
13		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
14		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
15		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
16		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
17		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
18		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
19		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
20		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
21		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
22		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
23		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
24		1	1	2	3	1	2	1	1	1	3	3	3	1	1	1	3	3	3

From Table 3.1 it is visible that, considering two clusters, the tendency is the grouping of from ten to twenty three periods in one cluster, and the rest in another. This separation reflects the peak and off-peak periods of the day in what concerns electrical energy consumption. This tendency is more evident and subject to less misclassified cases, when the considered number of days for analysis is increased. Regarding the tests using three clusters, the same tendency is verified, with the additional cluster being used for grouping the transition periods between peak and off-peak.

The same type of tests is performed to analyze the similarity of the days (see Table 3.2). In this case the matrix rows represent the days of the month, and the columns represent the periods. For each period the same four characteristics presented before are considered. The considered number of periods to analyze the days, for this test is twelve (two hour periods), and twenty four (one hour periods), as displayed in the first row of Table 3.2. Note that for the three used data sets the month in question starts on a Monday, to ease the interpretation of the results.

Table 3.2 – Clustering results for grouping days with similar characteristics.

	Periods	12									24								
	Clusters	2			3			5			2			3			5		
Day	Data Set	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1		1	1	1	1	1	1	4	4	4	1	1	1	1	1	1	4	4	4
2		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3		1	1	1	1	1	1	5	5	5	1	1	1	1	1	1	5	5	5
4		1	1	2	1	1	2	5	1	2	1	1	1	1	1	1	1	1	1
5		1	1	1	1	1	1	4	4	4	1	1	1	1	1	1	4	4	4
6		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
7		2	2	2	3	3	3	3	3	3	2	2	2	3	3	3	3	3	3
8		1	1	1	1	1	1	4	4	4	1	1	1	1	1	1	4	4	4
9		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10		1	1	1	1	1	1	5	5	5	1	1	1	1	1	1	5	5	5
11		1	1	1	1	1	1	1	4	1	1	1	1	1	1	1	1	4	1
12		1	1	1	1	1	1	4	4	4	1	1	1	1	1	1	4	4	4
13		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
14		2	2	2	3	3	3	3	3	3	2	2	2	3	3	3	3	3	3
15		1	1	1	1	1	3	4	4	4	1	1	1	1	1	1	4	4	4
16		1	1	1	1	1	1	5	4	4	1	1	1	1	1	1	4	4	4
17		1	1	1	1	1	1	5	5	5	1	1	1	1	1	1	5	5	5
18		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19		1	2	1	1	2	1	4	4	4	1	1	1	1	1	1	4	4	4
20		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
21		2	2	2	3	3	3	3	3	3	2	2	2	3	3	3	3	3	3
22		1	1	1	1	1	1	4	4	4	1	1	1	1	1	1	4	4	4
23		1	1	1	1	1	1	5	5	5	1	1	1	1	1	1	5	5	5
24		1	1	1	1	1	1	5	5	5	1	1	1	1	1	1	5	5	5
25		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
26		2	1	1	2	1	1	4	4	4	2	1	1	2	1	1	4	4	4
27		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
28		2	2	2	3	3	3	3	3	3	2	2	2	3	3	3	3	3	3
29		1	1	1	1	1	1	4	4	4	1	1	1	1	1	1	4	4	4
30		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

For the cases concerning two clusters, Table 3.2 shows a clear tendency in separating the work days from the weekend days. That tendency is clearer when the considered number of periods for analyzing the days is increased. Concerning the cases with three clusters, the grouping additionally considers the separation between Sundays and Saturdays. A special regard for day twenty six, which, for the first data set is always placed on the same cluster as the weekends, for the cases of two and three clusters. That fact is due to this day being a special situation day (a holiday). This day's characteristics were found to be more similar to weekends than to working days.

Regarding the cases with five clusters, the tendencies are not very clear. It is visible that Mondays and Fridays are placed in cluster 4 in the majority of the cases, but that is not a constantly observed event. Therefore, until further studies are performed, allowing the understanding of the reasons behind the clustering results for these cases, the consideration of a high number of clusters is discouraged.

The actual context definition is performed through an overlapping of the two result matrixes, concerning the analysis of the days, and the analysis of the periods. The result matrixes concern only one option in what regards the number of clusters and the number of days or periods for analysis. Table 3.3 presents an excerpt of the results of the overlapping of the results matrixes, concerning only two clusters, for an easier understanding of the results, and considering 24 periods in the analysis of the days, and 30 days in the analysis of the periods.

This excerpt considers the necessary days and periods to demonstrate the adequacy of the Context Analysis mechanism. Regarding the days, using two clusters and twenty four periods for the analysis, an evident separation between working days and weekends was found. Regarding the periods, when using two clusters and thirty days for their analysis it was clear a separation between periods from ten to twenty three from the rest. The defined contexts result from the combination of both result matrixes. D1 represents the days that were grouped in the first cluster, D2 the days that were grouped in the second cluster, P1 the periods grouped in the first cluster of the period analysis, and finally, the periods grouped in the second cluster are represented by P2. The different colours displayed in Table 3.3 represent the different combinations of days and periods clusters, hence representing the different contexts.

Table 3.3 – Context definition results.

		Period					
		7	8	9	10	11	12
Day	1	D1 P1	D1 P1	D1 P1	D1 P2	D1 P2	D1 P2
	2	D1 P1	D1 P1	D1 P1	D1 P2	D1 P2	D1 P2
	3	D1 P1	D1 P1	D1 P1	D1 P2	D1 P2	D1 P2
	4	D1 P1	D1 P1	D1 P1	D1 P2	D1 P2	D1 P2
	5	D1 P1	D1 P1	D1 P1	D1 P2	D1 P2	D1 P2
	6	D2 P1	D2 P1	D2 P1	D2 P2	D2 P2	D2 P2
	7	D2 P1	D2 P1	D2 P1	D2 P2	D2 P2	D2 P2
	8	D1 P1	D1 P2	D1 P3	D1 P2	D1 P2	D1 P2

Analyzing Table 3.3, D1P1, presented in yellow, represents the first context - off-peak periods of working days. D1P2, presented in green, represents the second context, peak periods of working days. D2P1, presented in orange, represents the third context, off-peak periods during weekend days. Finally, D2P2, presented in blue, represents the fourth context, peak consumption periods of the weekend days.

These contexts are defined in order to support the adaptation of the action approaches on behalf of a market negotiation agent, depending on the different contexts this agent will act on during a market simulation.

The context definition mechanism proved to efficiently recognize and group distinct market contexts. This mechanism's applicability and influence in supporting the ALBidS system for providing adequate support to a market negotiation player is demonstrated in the case study presented in section 4.3 - Contexts and efficiency/effectiveness balance.

3.6. Main Agent

The Main Agent [Pinto *et al.*, 2011a] is the central entity in ALBidS. It is responsible for communicating with the MASCEM agent that is using the decision support. The communications occur whenever the MASCEM player is going to negotiate, by sending a request to the Main Agent for it to return the final answer; and also when the negotiations are finished. When the MASCEM agent sends back the feedback with the information regarding the market results and competitor players' actions, the Main Agent updates its stats, "feeds" the ALBidS interface graphs, and distributes the information to the strategy agents, for them to individually update as well.

Besides dealing with the communications and containing the reinforcement learning algorithms, the Main Agent also integrates the Context Analysis module, and the Efficiency/Effectiveness Management module (see section 3.9). The Main Agent's role in ALBidS is presented in Figure 3.19.

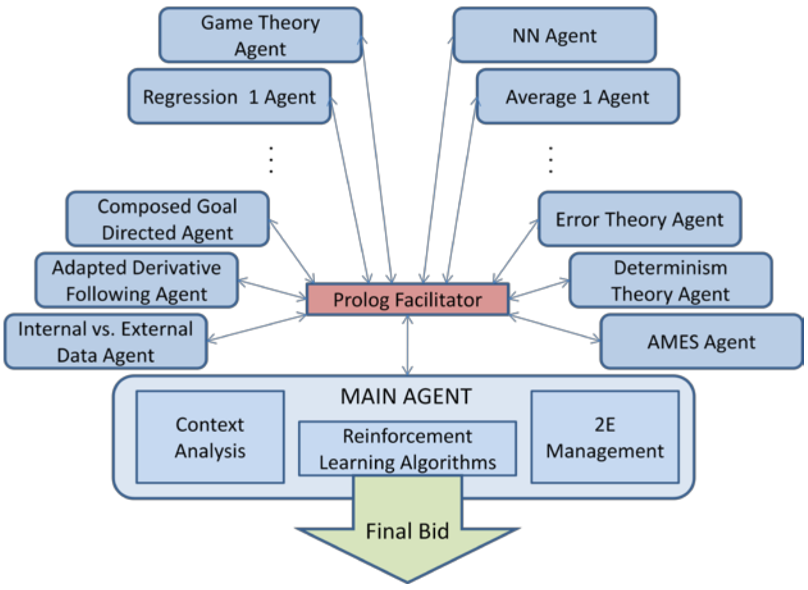


Figure 3.19 – Main Agent's role in ALBidS.

This agent's main responsibility is to execute the reinforcement learning algorithms [Vale *et al.*, 2011a]. In each moment and in each circumstance the technique that presents the best results for the actual scenario is chosen as the simulator's response. So, given as many answers to each problem as there are algorithms, the reinforcement learning algorithm will choose the one that is most likely to present the best answer according to the past experience of their responses and to the present characteristics of each situation, such as the considered day, the period, and the particular market context that the algorithms are being asked to forecast.

The main reinforcement algorithm presents a distinct set of statistics for each market context, as presented in section 3.5, which means that an algorithm that may be presenting good results for a certain context, with its output chosen more often when bidding for this period, may possibly never be chosen to provide the final answer for another period that is defined as being in another context.

The way the statistics are updated, and consequently the best answer chosen, can be defined by the user. MASCEM provides three reinforcement learning algorithms that can be chosen, all having in common the starting point. All the algorithms start with the same value of confidence, and then according to their particular performance that value will be updated. All algorithms also have the option of being attributed a weight value that defines their importance to the system. This means that a strategy that has a higher weight value will detach faster from the rest in case of either success or failure. There have been implemented three reinforcement learning algorithms, which differ in the way the stats for each strategy are updated. This way the system has the capability of choosing among the proposals using different perspectives, which improves the chances of one being more advantageous for each situation.

3.6.1. *Simple Reinforcement Learning Algorithm*

This sub-section presents the implementation of a simple version of a reinforcement learning algorithm. This version considers the basic characteristics of this type of algorithms, the most simplified as possible, in order to provide a fast response.

3.6.1.1. Context

Reinforcement learning is an area of machine learning in computer science, and therefore also a branch of Artificial Intelligence. It deals with the optimization of an agent’s performance in a given environment, in order to maximize a reward it is awarded [Tokic, 2010].

In this type of problems, an agent is supposed to decide the best action to take, based on his current state. When this step is repeated, the problem is known as a Markov Decision Process [Kaelbling *et al*., 1996]. This automated learning scheme implies that there is little need for a human expert who knows about the domain of application. Much less time will be spent designing a solution, since there is no need for the definition of rules as with expert systems.

A reinforcement learning agent interacts with its environment in discrete time steps [Sutton and Barto, 1998]. At each time t, the agent receives an observation o_t, which typically includes the reward r_t. It then chooses an action a_t from the set of available actions. The environment moves to a new state s_{t+1} and the reward r_{t+1} associated with the transition (s_t, a_t, s_{t+1}) is determined. The goal of a reinforcement learning agent is to collect as much reward as possible. The agent can choose any action as a function of the history and it can even randomize its action selection. A reinforcement learning global view is presented in Figure 3.20.

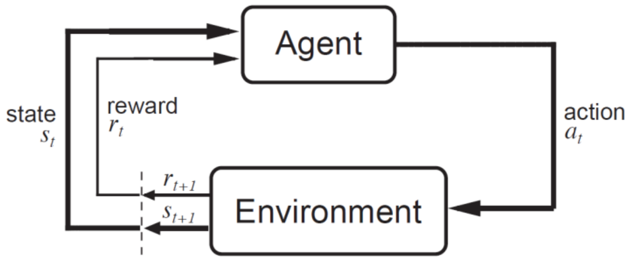


Figure 3.20 – Reinforcement learning overview (adapted from [Sutton and Barto, 1998]).

When an agent's performance is compared to that of an agent which acts optimally from the beginning, the difference in their performances gives rise to the notion of regret [Auer *et al.*, 2010]. In order to act near optimally, the agent must reason about the long term consequences of its actions. As the agent is not told which action to take, as in most forms of machine learning, but instead must discover which actions provide 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.

One of the challenges that arises in reinforcement learning and not in other kinds of learning is the trade-off between exploration and exploitation. To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. But to discover which actions are the most profitable, it has to select actions that it has not tried before. The agent has to exploit what it already knows in order to obtain reward, but it also has to explore in order to make better action selections in the future. The dilemma is that neither exploitation nor exploration can be pursued exclusively without failing at the task. Thus, reinforcement learning is particularly well suited to problems which include a long-term versus short-term reward trade-off [Sutton, 1988].

3.6.1.2. Implementation

A reinforcement learning algorithm provides the means to choose, in a given moment and circumstance, the strategy that presents the best results for the current scenario. This way, given the responses for each problem, the reinforcement learning algorithm chooses the response that is more likely to provide the best results. For that it takes into account the past experience concerning each alternative strategy's answers, and its results, for each context.

For each context, the Simple Reinforcement Learning Algorithm starts with the same value of confidence for each strategy. As the simulation progresses the values of confidence are updated for each context, through the application of a reinforcement that is defined according to the strategies' performance.

The confidence value *C* for each possible action (strategy), at the time *t*, is done through the application of (3.1).

$$C_{t+1} = C_t + Reinf \tag{3.1}$$

where *Reinf* refers to the reinforcement of the considered action for time *t+1*. The reinforcement is calculated as in 3.2.

$$Reinf_i = p \times Dif_i \tag{3.2}$$

where Dif_i is the difference between a strategy's prediction and the actual price that was verified in the market. The difference value is subject to a penalization *p* in case of the prediction value being above the actual market price, in case of seller agents, or below the market price in case of buyer agents. This is because if a seller's final bid is located above the market price, the agent will not be able to sell (see sub-section 2.2.1.1 – Spot Market). The same for buyers when biding below the market price. In these cases, the penalization will assume the value 3, otherwise *p* will be equal to 1, therefore having no influence on the difference value. Note that, according to way this algorithm is implemented, the reinforcement will be higher; the higher is the difference between a prediction and the actual value. Therefore, the *C* value does not represent a confidence value in the strategy's success, rather the confidence in its failure. So, the best action to be chosen in this scope is the one that presents the lower *C* value.

The Simple Reinforcement Learning Algorithm, similarly to the other reinforcement learning algorithms presented in the scope of this work, allows the attribution of a weight value for each strategy. This weight traduces the importance of a strategy to the system, *i.e.*, a strategy presenting a bigger weight will detach faster from the others, both in case of success or failure.

3.6.1.3. Experimental Findings

In order to verify the correct performance of the Simple Reinforcement Learning Algorithm, three tests are presented. Each of the tests considers a different simulation for fifteen consecutive days, including three seller agents, for whom the results are analyzed, and seven buyer agents. The simulations were performed considering solely the first period of the day. These simulations were built with the purpose of being simple, only to allow a demonstration of the algorithm's adequate performance. More elaborated case studies are presented in Chapter 4 - Case-Studying.

The first test presents a simulation using only three strategies. Figure 3.21 presents the bids each of the three strategies proposed, with equal weights for all strategies, and Figure 3.22 presents the confidence values that each strategy presents to the Simple Reinforcement Learning Algorithm throughout the days.

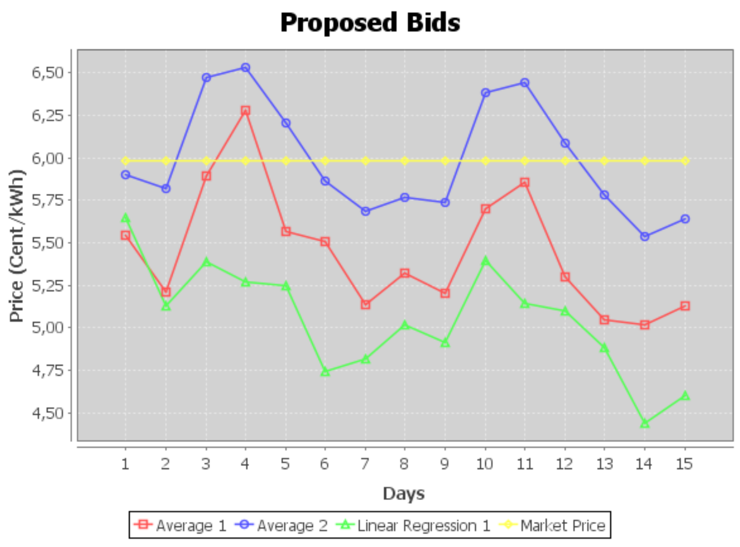


Figure 3.21 – Strategies' bid proposals.

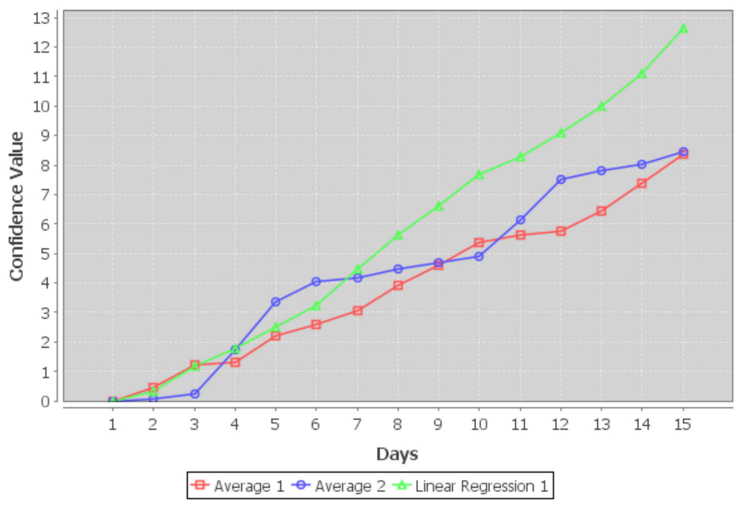


Figure 3.22 – Strategies' confidence values.

Comparing Figures 3.21 and 3.22 it is visible that the Average 2 strategy was the one that presented closer values to the market price, and located below the market price, for the first two days. This is reflected on this strategy's lower failure confidence value for the following days (days 2 and 3), as the performance of a strategy in one day is used to update the values for the next. From day 3 forward, the Average 2 strategy started presenting bids above the market price. For that reason its confidence value in failing started to rise, and Average 1, which only surpassed the market price barrier in one day passed it in what concerns being the best proposal strategy. The only day in which that does not verify is day 10, in which Average 1 is able to very temporarily surpass Average 2. Figure 3.23 presents the incomes achieved by the Simple Reinforcement Learning Algorithm's choice of Average 1's bid proposal.

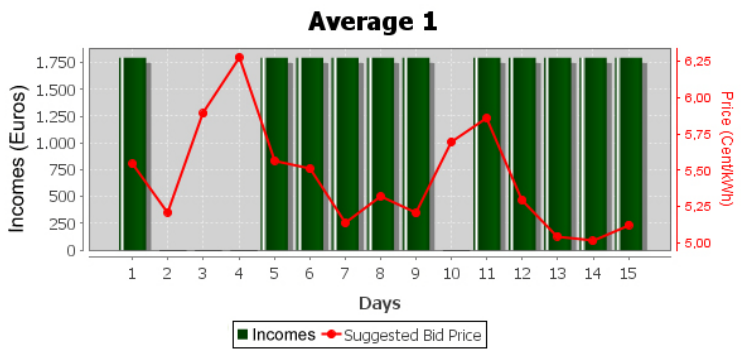


Figure 3.23 – Average 1's achieved incomes.

From Figure 3.23 it is visible that the Average 1's proposed bid was chosen and achieved positive incomes in all days that it presented the best (smaller) failure confidence value; except for period 4, the only one in which it was chosen as the algorithm's final bid, for being the best option (as seen in Figure 3.22), however as its proposed bid was located above the market price (as seen in Figure 3.21), it was unable to achieve incomes. The other days in which Average 1 did not achieve incomes (days 2, 3 and 10) it was because it wasn't the one presenting the best confidence value; therefore it did not had its proposal chosen as the algorithm's response.

Regarding day 1, in which all strategies present equal confidence value, Average 1 was chosen as the algorithm's response, and consequently was able to achieve incomes, only by chance, because in this case the choice was made randomly, and it just "got lucky" in being chosen.

Figure 3.24 presents the confidence values of the same three strategies, in the same exact scenario, but this time with Linear Regression 1 being attributed a weight of 100%, while the other two strategies receiving only half of that value. This test intends to show the influence of the weight value.

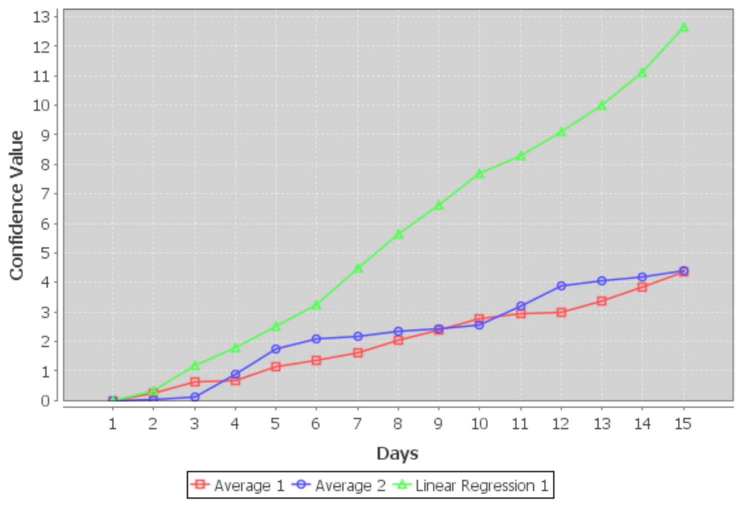


Figure 3.24 – Strategies' confidence values with Linear Regression 1's higher weight.

Figure 3.24 shows that the difference between the confidence value of the Linear Regression1 and the confidence values of the other strategies is much higher than in the first test (Figure 3.22). It detached faster from the rest because it was attributed a higher weight.

The third test considers a much higher number of proposing bid's strategies. Figure 3.25 presents the confidence values of these strategies to the Simple Reinforcement Learning Algorithm.

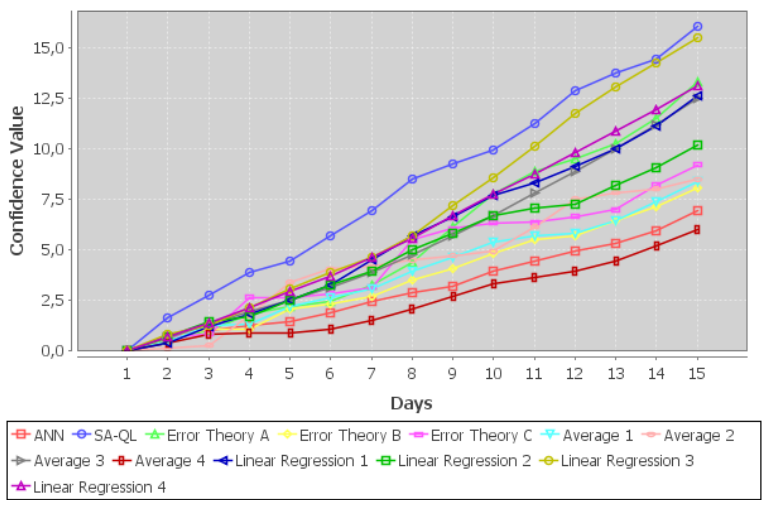


Figure 3.25 – Strategies' confidence values in the third test.

From Figure 3.24 the evolution of the confidence values of the several strategies can be seen. This shows the capability of the Simple Reinforcement Learning Algorithm in considering a high number of alternative strategies.

3.6.2. *Roth-Erev Reinforcement Learning Algorithm*

The application of the Roth-Erev reinforcement learning algorithm for choosing the best bid option is presented in this sub-section.

3.6.2.1. Context

The main reasoning behind a reinforcement learning algorithm is that the tendency to perform an action should be strengthened, or reinforced, if it produces favourable results and weakened if it produces unfavourable results.

The main principle behind a reinforcement learning algorithm is that, not only are choices that were successful in the past more likely to be employed in the future, but similar choices will be employed more often as well. This is referred to as law of effect [Erev and Roth, 1998].

Roth and Erev [Roth and Erev, 1995] take this law of effect principle, widely accepted in the psychological learning literature, as the basic starting point in their search for a robust model of individual learning, *i.e.* the understanding of how people learn individually to behave in games with multiple strategically interacting players. Roth and Erev argue that, in such contexts, the law of effect is not sufficiently able to support the learning process; therefore they have also included in their studies an additional learning principle, also widely accepted in the psychological learning literature, which they refer to as the power law of practice [Jing *et al.*, 2009]. This principle is based on the latter principle, which suggests that learning curves initially tend to be abrupt, and after some time they flatten out.

In order to experiment on each of these learning principles' responsibility over a person's learning process, Roth and Erev have developed a reinforcement learning algorithm, named as the Roth-Erev algorithm [Roth and Erev, 1995].

This algorithm's way of testing such principles uses an experimentation or recency parameter, which defines the weight that the past experience will have on a subject's learning process [Nicolaisen *et al.*, 2001]. Roth and Erev show that this algorithm is able to successfully track the observed intermediate-term behaviour of human subjects over a wide variety of multiagent repeated games with unique equilibrium achievable, using stage-game strategies.

3.6.2.2. Implementation

Following the principles proposed by Roth and Erev [Roth and Erev, 1995], the Roth-Erev Reinforcement Learning Algorithm has been implemented to support the reinforcement learning algorithm's purpose in the scope of ALBidS.

The differences from this methodology to the Simple Reinforcement Learning Algorithm concern the inclusion of the *recency* parameter, which defines the importance of the past experience to the evolution of the learning process. This way, taking advantage on this parameter, it is possible to define if, for each case, it will be attributed a higher importance for the recent events, adapting faster to the changing environment; or on the other hand, if the accumulated past experience, and the achieved results in the past, will be the most influent factor in defining the confidence in an action's performance [Vale *et al.*, 2011a].

The main procedure behind this algorithm is very similar to the Simple Reinforcement Learning Algorithm's. The reinforcement calculation is done in the same way (as presented in equation (3.2)); and similarly, the same value of confidence in assigned for each strategy (action) at the moment of initialization. As the simulation progresses the values of confidence are updated for each context, through the application of the strategies' performance based reinforcement. The application of this reinforcement, in the Roth-Erev Reinforcement Learning Algorithm's case, concerning the *recency* parameter *r*, is done according to (3.3).

$$C_{t+1} = C_t \times r + Reinf \times (1 - r) \tag{3.3}$$

As can be seen by the *r* parameter's inclusion, which assumes values between 0 and 1, the higher this value is, the higher is the importance that will be given to the consideration of the previous days' confidence value. On the other hand, if this value is small, the accumulated experience from the past days will vanish faster, and the new reinforcement for the current case will present a much higher influence over the new confidence value that is being updated.

The Roth-Erev Reinforcement Learning Algorithm also allows the attribution of a weight value for each strategy, reflecting the user's preference for a strategy's importance to the system.

3.6.2.3. Experimental Findings

The testing of the Roth-Erev Reinforcement Learning Algorithm's performance includes the presentation of two simulations, concerning different values of the *recency* parameter, so that conclusion can be taken about the influence of this parameter, and the consequent variation in terms of results. Each of the tests considers a different simulation for fifteen consecutive days, including three seller agents, for whom the results are analyzed, and seven buyer agents. The simulations were performed considering solely the first period of the day, in order to ease their results comprehension. The proposed bids from the considered seller agents in both simulations are presented in Figure 3.26.

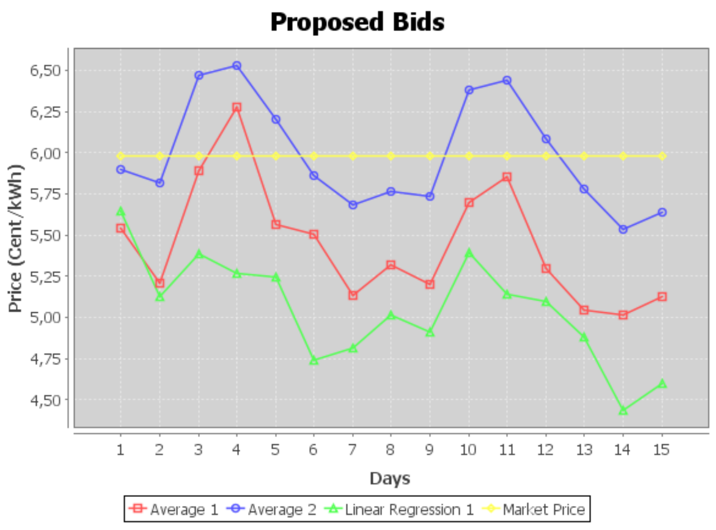


Figure 3.26 – Seller agents' proposed bids.

In the first test the *recency* parameter is assigned a value of 0.2, a small value to enhance the influence of recent events. Regarding the second simulation, the *recency* parameter assumes a value of 0.8, a large value, so that the past experience is taken into higher consideration for this case. Figure 3.27 presents the confidence values for each strategy throughout both simulations. Note that, similarly to the Simple Reinforcement Learning Algorithm's case, the confidence value reflects the confidence in a strategy's failure, so the smaller this value is, the better the proposal is expected to be.

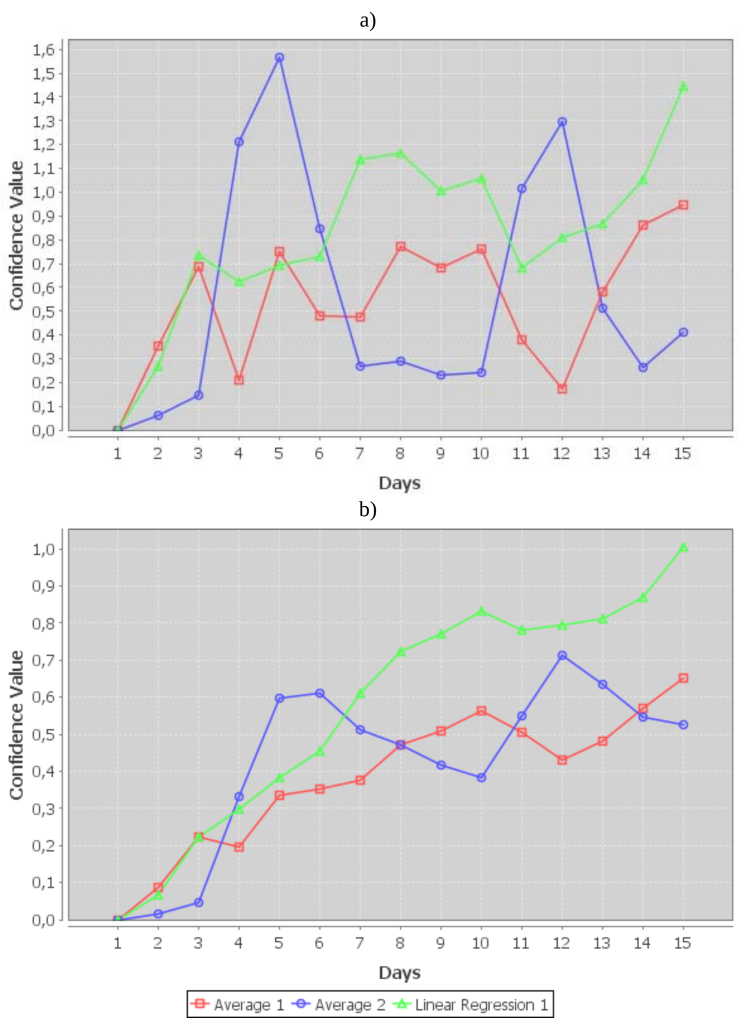


Figure 3.27 – Seller agents' failure confidence values for a *recency* parameter's value of: a) 0.2, b) 0.8.

Analysing Figure 3.27 it is visible that for a higher importance for recent events (Figure 3.27 a)), the variations of the confidence values are highly abrupt. A strategy presenting the best confidence value for one day, may present the worst in the following, in case of its results being ineffective for a single case. Regarding Figure 3.27 b), concerning the case where the higher importance is attributed to the past experience, the confidence values' evolution is much smoother, since a good or bad result in one day does not present such an higher influence over the confidence value's update.

Figure 3.28 presents the results, in incomes, achieved by ALBidS use of the strategy that presented better results (Average 1), in both simulations.

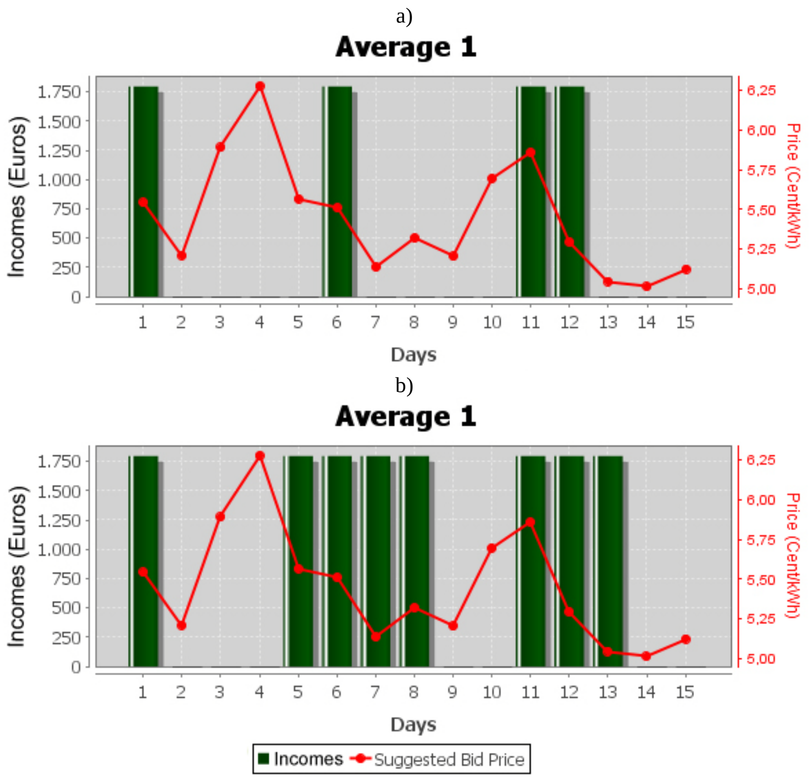


Figure 3.28 – Average 1's achieved incomes for a *recency* parameter's value of: a) 0.2, b) 0.8.

These results are obvious in what regards the difference in achieved results by the Average 1 strategy. The higher volatility in the confidence values for the first simulation reflects a higher uncertainty in what regards the achieved incomes. This uncertainty is smothered by the considering of a higher importance for past experience, which resulted in a higher consistency of results for this strategy, for this case.

The balance between past experience and recent results is an important issue to be dealt with when defining the preferences for the execution of the Roth-Erev Reinforcement Learning Algorithm. These tests showed that for this case the past experience's higher weight proved to be more advantageous; however, this is a fact that may depend on the circumstances and on the context of the market negotiations.

3.6.3. *Bayes Theorem*

This strategy implements an application of the Bayes Theorem, directing this theorem's principles to the development of a reinforcement learning algorithm based on the estimation of success probabilities [Vale *et al.*, 2011a].

3.6.3.1. Context

The Bayes Theorem has been applied in several scopes throughout the times, taking advantage on this probability theory's capabilities of supporting applications directed to the most alternative contexts. One of this theorem's advantages that has been considered interesting was its applicability to be used as a reinforcement learning algorithm [Pinto *et al.*, 2011a]. This applicability is based on the use of a probability estimation to determine which of the different alternatives (strategies' suggestions) presents a higher probability of success in each context, therefore being considered as the most appropriate approach.

Bayesian networks have been developed to facilitate the task of prediction and abduction in artificial intelligence systems. Simplistically, these networks, also known as *causal networks*, or *probabilistic networks*, are graphic models for uncertainty based reasoning [Dore and Regazzoni, 2010]. A Bayesian network is represented by a directional acyclic graph in which the knots represent domain variables, and the arches represent the conditional or informative dependency between the variables. In order to represent the dependency strength, probabilities are used, associated to each network's group of parent-son knots.

The values represented by the domain variables must be mutually exclusive and exhaustive, *i.e.* a variable must always assume one and only one of those values. Usually these values types are boolean, enumerated, or numeric values. All values must efficiently represent the domain, however with enough detail to be distinguished.

Two knots must be connected directly, in case of one affecting the other, with the arch's direction pointing the effect direction. Once the network's topology is defined, the following step is to quantify the relationships between the connected knots, specifying the conditional probability distribution for each knot [Korb and Nicholson, 2003]. The conditional probability is represented as (3.4):

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{3.4}$$

This conditional probability provides the basis for the propagation of each knot's probability given the facts observed by variables each influence them. The expected utility of the action each knot represents, is given by (3.5), being *E* the available evidences, *A* an action with possible outcomes O_i, $U(O_i|A)$ the utility of each of the outcome states given that action *A* is taken, $P(O_i|E,A)$ the conditional probability distribution over the possible outcome states, given that evidence *E* is observed and action *A* taken.

$$EU\left(A|E\right) = \sum_i P(O_i \mid E, A) \times U(O_i \mid A) \tag{3.5}$$

3.6.3.2. Implementation

The Bayes Theorem reinforcement learning algorithm applies the Bayes Theorem through the implementation of a Bayesian network. In this adaptation, each strategy (suggested action) is represented by a different knot. All strategy knots are connected to an output knot, which is responsible for the final decision on which is the most advantage action. This decision is based on the calculation of the success probability of each strategy, based on the observed events: if a strategy has accomplished to be the most successful amongst all in a certain negotiation period (event *Yes*), or not (event *No*). Figure 3.29 presents the topology of the *Bayesian network*, with an example considering three strategies.

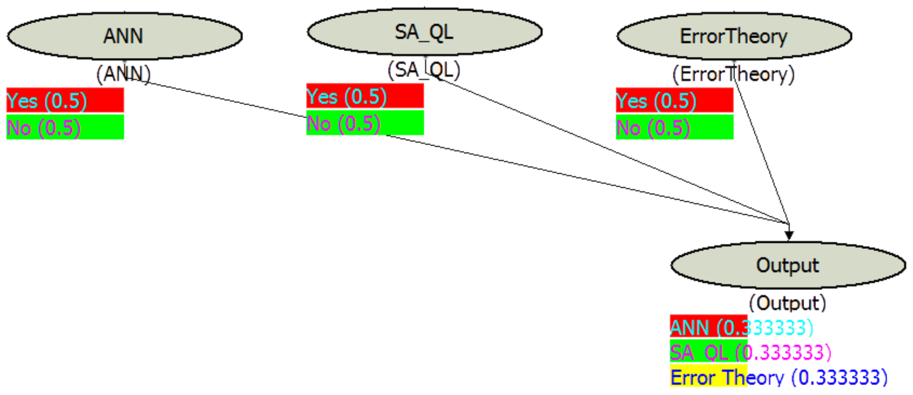


Figure 3.29 – Bayesian network's topology considering three strategies-

At the initial point of a simulation all strategies must be initialized with the same probability of success, value that throughout the simulation shall be updated according to the performance of each. However, a Bayesian network does not represent a temporal probabilistic model. Therefore, it has been found necessary to use a dynamic *Bayesian network*. This dynamism has been achieved by contemplating the inclusion of the *Counting-Learning* algorithm.

Counting-Learning Algorithm

The *Counting-Learning* algorithm considers, for each case about to be learned, an experience value, which defines in what degree the considered case will affect the conditional probability of each knot. This algorithm guarantees that all knots' probabilities are updated.

For each new case, the updated experience value *exper'*, is calculated depending on the previous experience value *exper*, as presented in (3.6).

$$exper' = exper + degree \tag{3.6}$$

where *degree* represents the weight of the current case. The common approach (applied in this implementation) is defining the *degree* as assuming the value 1, meaning that the case will be learnt as having the same importance as all the others. However, the *Counting-Learning* algorithm is prepared for other situations, in which the attribution of a higher value, *e.g.* 2, makes the algorithm learn this case as if it represented two similar cases at once (giving it the double of the importance). It can also be attributed a negative value, *e.g.* -1, so that it can "unlearn" a case.

The experience value and the *degree* are used in the updating process of a knot's probability. The probability update of a knot that has observed an event is done accordingly to (3.7). A new probability *probc'*, is calculated, taking into account the previous probability *probc*.

$$probc' = \frac{(probc \times exper + degree)}{exper'} \tag{3.7}$$

The remaining knots *i* (for which the event was not observed) are updated according to (3.8), so that the probabilities vector (which contains all knots probabilities for each context) is kept normalized.

$$probi' = \frac{(probi \times exper)}{exper'} \tag{3.8}$$

Once the network’s dynamics are defined, the following step is to integrate them with the reinforcement learning algorithm, while guaranteeing the independence between probabilities concerning different contexts.

The adopted approach has been to use an existing and tested *API* which already implemented the basics of a Bayesian network, so that this strategy could abstract from the probabilistic calculations and from the probabilities propagation. Several Java *API*s have been tested, however many of them did not work properly, others were incomplete, and others did not present the necessary and updated documentation, so that one could properly work with it and manipulate its functionalities, in order to achieve the required procedure. Some of the *API*s that were testes but did not qualify as adequate were: *JavaBayes* [JAVABAYES, 2011], *jBNC* [JBNC, 2011] and *Banjo* [BANJO, 2011].

Finally, an adequate *API*, which fulfils all the requirements, was found: *Bayesian Network tools in Java 3* (BNJ 3) [BNJ3, 2011]. In order to define a Bayesian network with this *API*, it is necessary to create the knots, and the connections between them, so that its structure is built. Afterwards, the initial probabilities for each knot must be defined, in order to indicate the way each is going to affect the network. Once these initial definitions are completed, the network is able to be executed and interacted with. With the objective of “hiding” these calculations from the Main Agent class, an auxiliary class *BayesianNetworkStatsCalculator* has been created, containing the network, and automatically defining its structure, initializations, probabilities, and possible actions to perform with the network. This way, by instantiating an object of this class and initializing it with the required parameters, it is possible to create an adequate Bayesian network without the need to constantly repeat the low-level steps, what makes its use and manipulation much simpler. This class also applies the *Counting-Learning* algorithm’s application over the network, while hiding its complexity.

The most important methods of the *BayesianNetworkStatsCalculator* class are:

- public void initialize(Object args[]), which receives variables. The first two containing vectors with: the names of the knots (strategies), and the preferred weights for each strategy; and the third indicating the number of contexts that were defined for the current simulation, so that the probability vectors are defined accordingly;
- public double getStats(), which returns the current probabilities of each knot, for the current context, so that the most adequate one can be selected as the chosen action;
- public void updateStats(boolean[] values), which receives a vector of boolean variables, indicating for each strategy if it was or not the one that effectively presented the best results in the market.

3.6.3.3. Experimental Findings

This sub-section presents three tests, in order to verify the correct performance of the Bayes Theorem reinforcement learning algorithm. Each of the tests considers a different simulation for fifteen consecutive days, including three seller agents, for whom the results are analyzed, and seven buyer agents. The simulations were performed considering solely the first period of the day.

The first two tests present simulations using only three strategies. Figure 3.30 presents the bids each of the three strategies proposed in both simulations

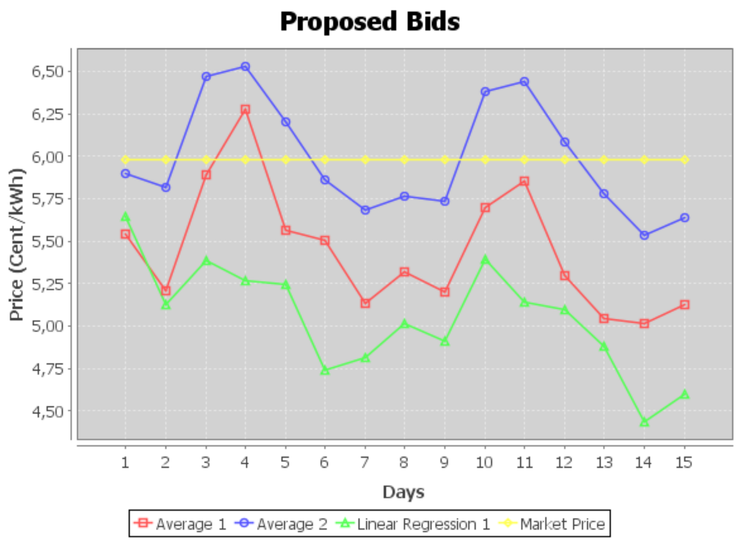
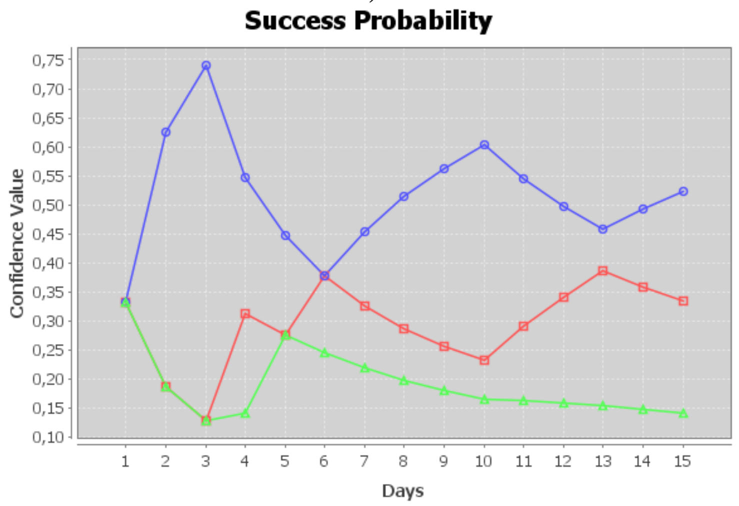


Figure 3.30 – Seller agents' proposed bids.

In the first test all strategies are assigned an equal importance weight; while in the second, the Average 2 strategy presents a double weight value comparing to the other considered strategies. Figure 3.31 presents the success probability for each strategy throughout both simulations. Note that, on the contrary of the Simple Reinforcement Learning Algorithm's and the Roth-Erev Reinforcement Learning Algorithm's case, the success probability of the Bayes Theorem reinforcement learning algorithm reflects the actual confidence in a strategy's success, and not in its failure. So, the higher this value is, the better the strategy is expected to perform.

a)



b)

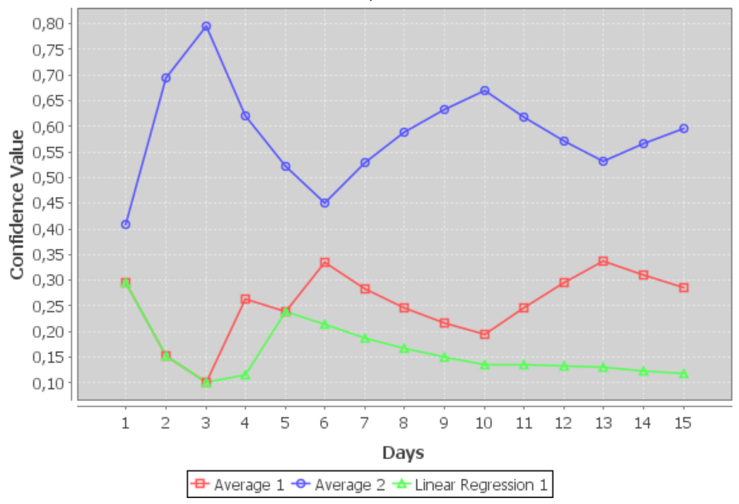


Figure 3.31 – Seller agents' success probability for an Average 2 weight: a) equal to the other strategies, b) double of the other strategies'.

Analysing Figure 3.31 it is visible that when the Average 2 strategy is assigned a higher weight value, this strategy's confidence values detach faster from the others. This detachment reflected an isolation of Average 2 as presenting the best confidence values for all days.

Figure 3.32 presents the results, in incomes, achieved by ALBidS use of the Average 2, in both simulations.

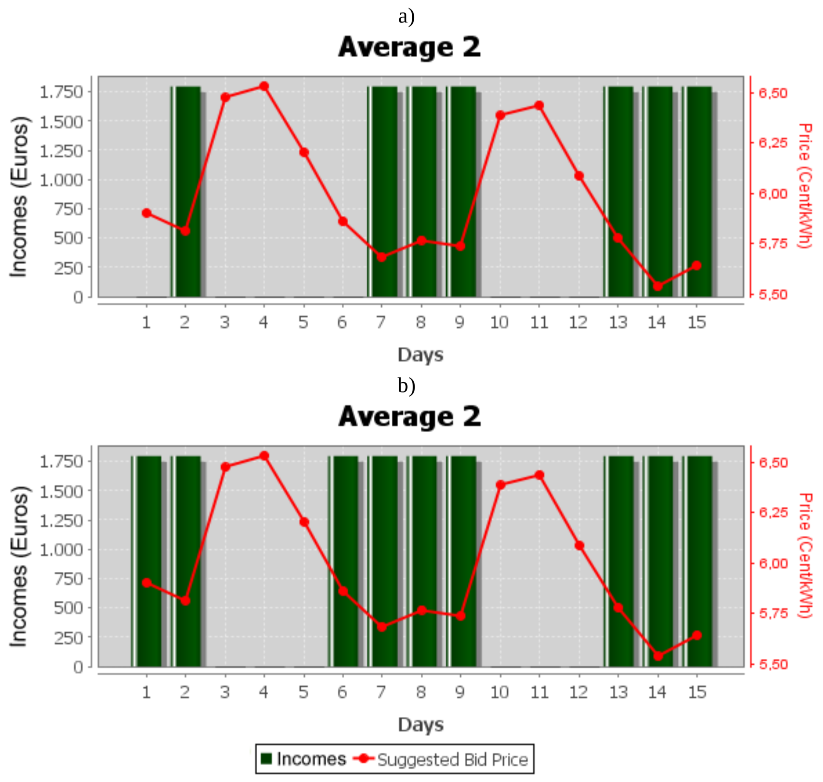


Figure 3.32 – Average 2's achieved incomes with a preference weight: a) equal to the other strategies, b) double of the other strategies'.

The Average 2 detachment as the strategy with the best confidence value reflected an improvement in the incomes obtained by the use of this strategy. This is due to the constant use of this strategy's proposal as the ALBidS system's final action. By using the strategy more often, its contribution to the system increased its achieved results.

The third and final test intends to show the Bayes Theorem reinforcement learning algorithm's ability to support various alternative strategies, *i.e.* reflected by an increase in the network's knots. Figure 3.33 presents the evolution of the strategies' confidence values, concerning an higher number of strategies.

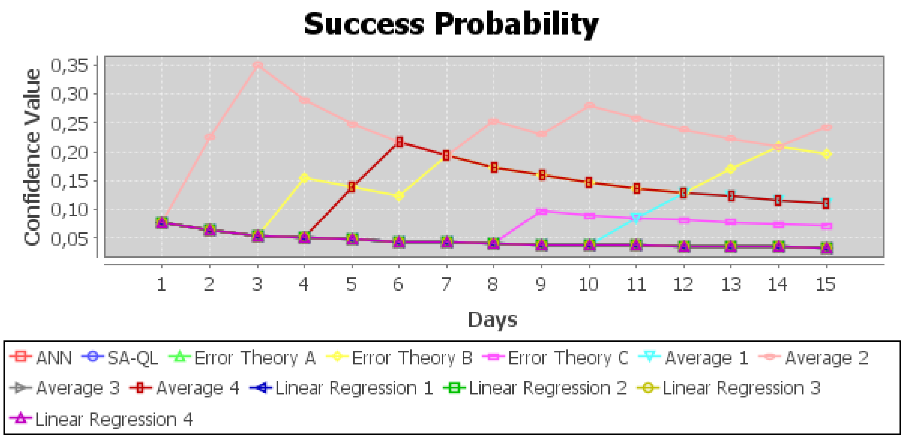


Figure 3.33 – Strategies confidence values in the third test.

From Figure 3.33 the evolution of the confidence values of the several strategies can be seen. This shows the capability of the Bayes Theorem reinforcement learning algorithm in considering a high number of alternative strategies.

3.7. Strategy Agents

This section presents the agents that have been developed with the purpose of executing different strategies to provide proposals of actions for the Main Agent to choose from.

Many different methodologies are able to be applied to the purpose of these agents, either in the electricity market's perspective or in what concerns forecasting methods, applied to time sequences, in this case, the electricity market prices [Bompard *et al.*, 2005], [Azevedo *et al.*, 2007], [Cao *et al.*, 2009], [Jing *et al.*, 2009]. None of them presents an evident superiority relatively to the others in terms of results, especially when applied to different contexts and circumstances of simulation with distinct characteristics. For this reason, this system integrates a large number of distinct approaches: data mining techniques [Cao *et al.*, 2009], forecasting methods [Wilamowski and Yu, 2010], artificial intelligence methodologies [Bayro and Daniel, 2010], application of electricity markets directed strategies [Li and Tesfatsion, 2009], mathematic approaches [Gintis, 2000], economic theory based models [Porter, 2008], and the adaptation of physics theories [Einstein, 1905]. This way the system is able to take advantage of the best characteristics of each approach whenever they show to be advantageous.

The system is then prepared to deal with different contexts and scenario situations, guaranteeing a large scope of approaches, which offer a greater chance of having appropriate responses even in very distinct situations. The strategies can be looked at as tools to be used by the Main Agent, which decides when are the appropriate moments to use each of them, and the best way to use them, depending on its analysis of the negotiation contexts, the simulation characteristics, and the results each strategy is presenting in each context.

Each strategy has been developed independently from the others, and the maximum attention was given to the optimization of each one. Note that the majority of the strategies resulted in a scientific contribution by its own, reflecting its individual relevance to the scientific community, and cooperation in what concerns the improvement in markets' strategic operation.

The strategies differ from each other only from an internal point of view, *i.e.* their behaviour, while for the system they are all viewed in the same way, with the communications treated equally for all. This is due to the use of the polymorphism mechanism. This way the integration of new strategies is highly facilitated, as it is one of the main future advantages of this system – being constantly complemented with new approaches.

The strategies require a process of initialization, of updating, and of returning values – the proposed bids. Taking this into account, the following Software Engineering standards were applied: *Strategy*, *Factory*, and *Singleton*, as presented in the class diagram of Figure 3.34.

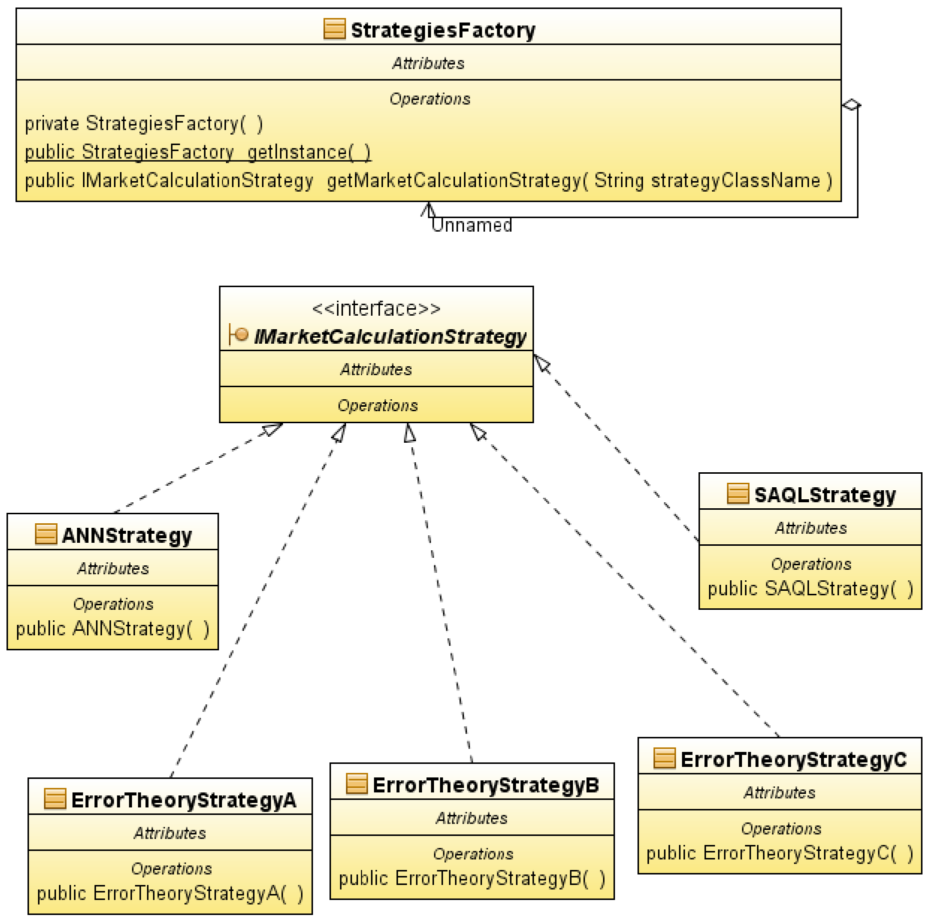


Figure 3.34 – Application of the *Strategy*, *Factory* and *Singleton* standards.

With the application of these standards, an *interface* named *IMarketCalculationStrategy* represents the strategies, which are classes that implement this *interface*. The ALBidS agent that is responsible for using a certain strategy must indicate the required strategy to the *StrategiesFactory*, which stays responsible for the creation of the instance of the desired strategy. This way all new strategies only need to implement the *interface* to be integrated, so that the system can be able to recognize and use them.

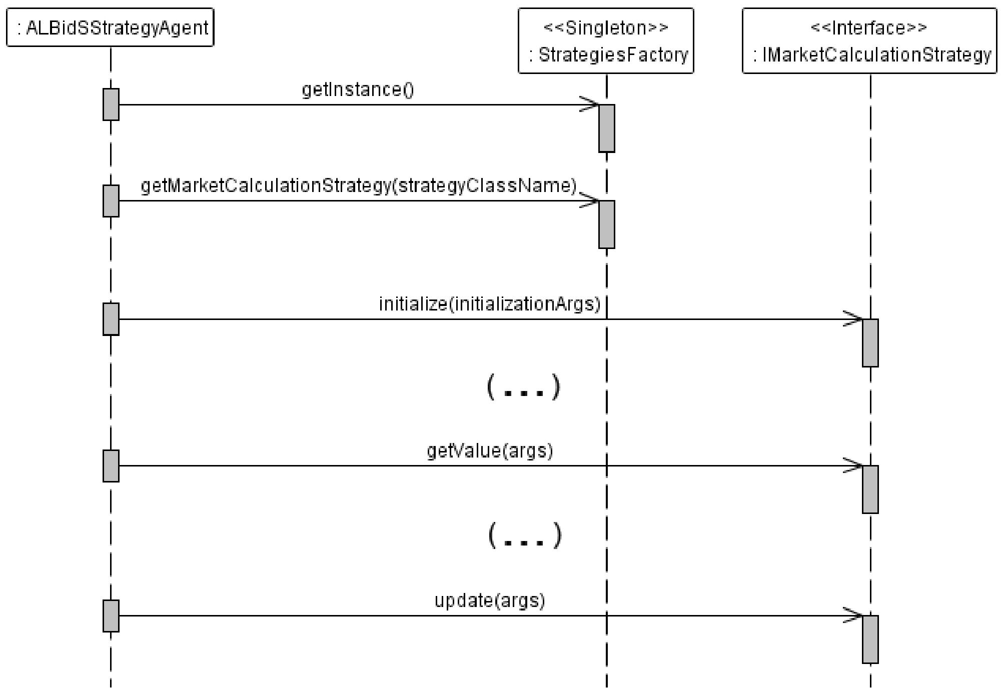


Figure 3.35 – Sequence diagram of the use of a strategy.

The implementation of the interface (*IMarketCalculationStrategy*) at the time of a new strategy integration implies the rewriting of the following methods:

- public void initialize(Object args[]), which is the method that is called after the creation of a strategy, to allow it to receive the strategy initialization parameters chosen by the user;
- public void update(Object args[]), which receives the values required for updating the strategies, after the end of each negotiation period. Since the parameter of this method is an Object list, this method can be global for all strategies, even though each strategy requiring different information from each other when updating the strategies;
- public double getValue(Object args[]), the method that asks the strategy to execute, and return the proposed bid. This method’s parameter is also an Object list, to allow the strategies to receive new execution parameters if necessary, *e.g.* for execution time reduction (see section 3.9).

Each part that composes each strategy was tested separately at the time of its implementation. This process comes to ensure the correct performance of each composing part, before the integration of all, giving origin to the final version of each strategy. The strategies’ final versions are tested individually, to demonstrate their suitability to their purpose (*e.g.* forecasting market prices, constructing possible scenarios, or developing intelligent action proposals). Additionally, most of the strategies require a demonstration in a market environment, to show their performance when acting in a simulation. To facilitate the comparison between strategies’ performances, and provide a better understanding of each strategy’s acting in that environment, a test market simulation scenario has been defined. The agents that are tested in the market use this test scenario, which has been based on real data extracted from the Iberian market – OMEL [OMEL, 2011].

Test Scenario

The test scenario involves 7 buyers and 5 sellers (3 regular sellers and 2 VPPs). This group of agents has been created with the intention of representing the Spanish reality, reduced to a smaller group, containing the essential aspects of different parts of the market, allowing a better individual analysis and study of the interactions and potentiality of each of those actors [Vale *et al.*, 2011a]. Figure 3.36 presents the test scenario structure.

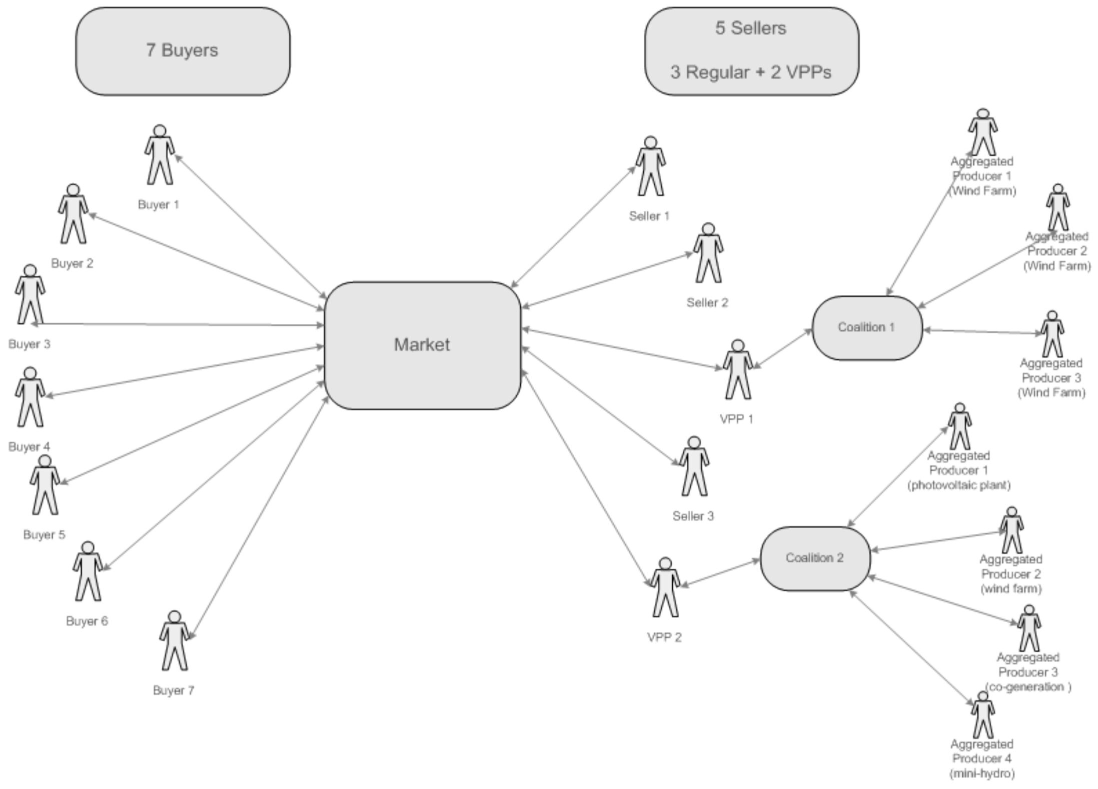

7 Buyers
5 Sellers
3 Regular + 2 VPPs
Buyer 1
Buyer 2
Buyer 3
Buyer 4
Buyer 5
Buyer 6
Buyer 7
Market
Seller 1
Seller 2
VPP 1
Seller 3
VPP 2
Coalition 1
Coalition 2
Aggregated Producer 1 (Wind Farm)
Aggregated Producer 2 (Wind Farm)
Aggregated Producer 3 (Wind Farm)
Aggregated Producer 1 (photovoltaic plant)
Aggregated Producer 2 (wind farm)
Aggregated Producer 3 (co-generation)
Aggregated Producer 4 (mini-hydro)

Figure 3.36 – Test scenario structure.

The simulations consider different biddings for each agent. Seller 2, which is used as the test reference, will use each presented strategy with different parameters, depending on what it is important to be shown. Also, the number of simulations with different parameters for each strategy, and the simulated days, varies depending on the tested strategy.

The other players' bids are defined as follows:

- Buyer 1 – This buyer buys power independently of the market price. The offer price is 18.30 c€/kWh (this value is much higher than average market price);
- Buyer 2 – This buyer bid price varies between two fixed prices, depending on the periods when it really needs to buy, and the ones in which the need is lower. The two variations are 10.00 and 8.00 c€/kWh;
- Buyer 3 – This buyer bid price is fixed at 4.90 c€/kWh;
- Buyer 4 – This buyer bid considers the average prices of the last four Wednesdays;
- Buyer 5 – This buyer bid considers the average prices of the last four months;

- Buyer 6 – This buyer bid considers the average prices of the last week (considering only business days);
- Buyer 7 – This buyer only buys power if market prices are lower than the usually verified market price (around 4.0 to 8.0 c€/kWh), by bidding a much lower value: 2.0 or 3.0 c€/kWh, depending on whether the current negotiation period is at a peak time of the day;
- Seller 1 – This seller needs to sell all the power that he produces. The offer price is 0.00 c€/kWh;
- Seller 3 – This seller bid considers the average prices of the last four months with an increment of 0.5 c€/kWh;
- VPP 1 – Includes four wind farms and offers a fixed value along the day. The offer price is 3.50 c€/kWh;
- VPP 2 – Includes one photovoltaic, one co-generation and one mini-hydro plants; the offer price is based on the costs of co-generation and the amount to sell is based on the total forecasted production.

In order to facilitate the understanding of the results, all the considered market negotiating entities will be attributed an absent total cost. This way, this value will not affect the results display in the analysis of each test. Therefore the examinations are done by considering incomes rather than profits (costs subtracted to incomes).

Another important remark is that for each strategy, the presented tests consider the adequacy of the strategy in what regards its purposes. The analysis regarding the strategies' execution time is performed in section 3.9 – Efficiency/Effectiveness Management mechanism. This way all strategies' execution times can be compared in the scope of the mechanism that was developed to manage them. All tests and case studies were performed on a computer with two Intel® Xeon® X5450 3.0GHz processors, each one with 2 cores, 4GB of random-access-memory and Windows Server 2008 32 bits operating system.

3.7.1. *Integration of the Previously Existing MASCEM Strategies*

The first created strategy agents were attributed the capability of executing the strategies that MASCEM included already, which were presented in sub-section 2.5.

Since these strategies were already implemented, tested, and validated in previous work [Praça *et al.*, 2003], this sub-section will not present further tests on these strategies. Rather it will present a brief description of the strategic behaviour these strategies execute.

These strategies integration is characterized by a simple change on how the execution is made. The strategies were executed directly by a particular MASCEM negotiating agent. With the integration of ALBidS they started being executed by ALBidS agents.

3.7.1.1. Composed Goal Directed

This strategy is based on two consecutive objectives, the first one may be increasing the income (reducing the payoff), and the second one reducing the greenhouse gas emissions. This strategy will try to obtain the highest profit, decreasing the price if in the same period of the previous day the first objective was not completely satisfied, and then try to fulfil the second goal, while maintaining the first satisfied.

3.7.1.2. Adaptive Derivative Following

This strategy is based on a Derivative-Following strategy proposed by Amy Greenwald [Greenwald *et al.*, 1999]. The Adapted Derivative-Following strategy adjusts its price by looking at the amount of revenue earned in the same period of the previous day, as a result of that period's price change. If that period's price change produced more revenue per good than the same period of two days before, then the strategy makes a similar change in price. If the previous change produced less revenue per good, then the strategy makes a different price change.

The price changes, or adjustments, are calculated by the following formulas [Praça *et al.*, 2003]:

$$price_{i+1} = price_i \pm amount_{i+1} \tag{3.9}$$

$$amount_{i+1} = price_i * \left(\beta + \frac{\Delta_i}{capacity\_available_i * \alpha} \right) \tag{3.10}$$

$$\Delta = capacity\_available - \text{energy}\_sold_i \tag{3.11}$$

This allows scaling the change by a ratio instead of adjusting the price each day by a fixed percentage. In case of a seller, the change amount increases with the difference between the amount of energy the seller wanted to sell and the amount it actually sold. α and β are scaling factors.

3.7.1.3. Market Price Following

As the name suggests, this strategy follows the market price of the same period of the previous day. It is a very simple strategy, but it presents good results when prices show a tendency to stabilize in a certain period, for some consecutive days.

3.7.1.4. Average Agents

The average agents perform averages on the log of historic market prices. There are three agents in this category, whose bids differ concerning the way they analyze the market database. The three agents are:

- Average Agent 1 - This agent performs an average of the prices from the market historic database. It uses the data from the 30 days prior to the current simulation day, considering only the same period as the current case, of the same week day. This allows having a strategy based on the tendencies per week day and per period;

- Average Agent 2 - This agent performs an average of the market prices considering the data from one week prior to the current simulation day, considering only working days, and only the same period as the current case. This strategy is only performed when the simulation is at a business day. This approach, considering only the most recent days and ignoring the distant past, provides a proposal that can very quickly adapt itself to the most recent changes in the market;
- Average Agent 3 - This agent uses an average of the data from the four months prior to the current simulation day, considering only the same period as the current case. This offers an approach based on a longer term analysis.

Even though this type of strategies, based on average, may seem too simple, they present good results when forecasting the market prices, taking only a small amount of time for their execution.

3.7.1.5. Regression Agents

The regression agents, analogously to the average agents, perform a statistical analysis on the log of historic market prices. However, their output is based on linear regressions on the historic data, rather than averages. There are two agents in this category:

- Regression Agent 1 - This agent performs a regression on the data from the four months prior to the current simulation day, considering only the same period of the day, similarly to the method used by Average 3 Agent.
- Regression Agent 2 - This agent performs a regression on the data of the last week, considering only business days. This strategy is only performed when the simulation is at a business day.

3.7.2. *Dynamic Artificial Neural Network Agent*

This strategy implements an artificial neural network, with the special feature of being dynamic. The dynamism is characterized as a re-training of the network in every iteration, so that it is able to consider the most recent data at all times, and constantly adapt itself to the most recent happenings.

3.7.2.1. Context

Artificial Neural Networks (NN) are inspired on the human brain, consisting in a huge number of neurons with high interconnectivity. A NN is constituted by a series of nodes, or neurons, organized in different levels, and interconnected by numeric weights. They resemble to the human brain in two fundamental points: the NN knowledge being acquired from the surrounding environment, through a learning process; and the network's nodes being interconnected by weights (synaptic weights), used to store the knowledge [Amjady *et al.*, 2010].

Figure 3.37 presents the global structure of a NN, where the input vector *X*, of dimension *n*, is mapped into variable *Y* through the scalar product of the vector *W* weights, and the non-linear mapping function *f*.

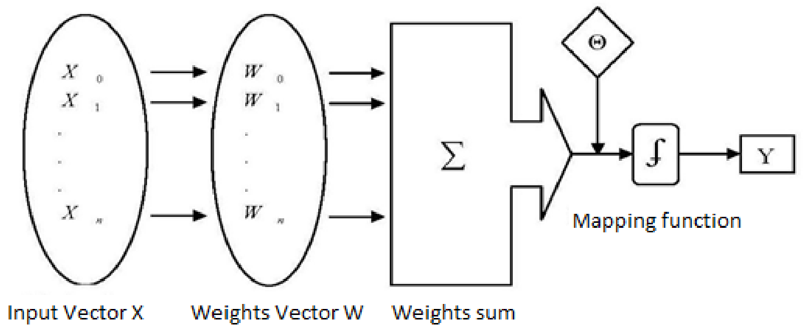


Figure 3.37 – Neural Network structure.

Each neuron executes a simple operation, the weighted sum of its input connections, which originates the exit signal that is sent to the other neurons. The network learns by adjusting the connection weights, in order to produce the desired output - the output layer values. The basic concept consists in providing the network with a large number of correct examples, so that through the comparison of the NN's output with the correct response, it can slowly change the connection weights until it is able to generate outputs that are coincident with the correct values. This way, the NN is able to extract basic rules from real data, differing from the programmed computation, where a set of rigid and pre-defined set of rules and algorithms is necessary.

An NN usually presents a high accuracy in classification and forecasting, even for complex problems. Also, NNs are advantageous when dealing with redundant attributes, since the weights assigned to these attributes are usually very small. Concerning the generated outputs, NNs present the advantage of being able to originate results based on discrete values, real values, or vectors of values [Wilamowski and Yu, 2010].

3.7.2.2. Implementation

The main contribution of this strategy is the development of a dynamic artificial neural network. The dynamism of the NN is achieved by a retraining of the network in each iteration, in order to always consider the most recent information, instead of the usual approach when managing NNs, which is training it once, and then use that trained NN to perform the forecasts from that point forward. With the dynamism of the NN, the constant adaptation and adequacy of the forecasts taking into account the most recent events, is the main goal.

The considered NN is characterized as a feedforward neural network, receiving as inputs the market prices and total amount of negotiated energy in the market, referring to: the day before to the desired forecasted day, one week before, two weeks before, and three weeks before. The NN considers four nodes in the intermediate layer, and one output – the forecasted market price. This topology was defined after some analysis and experiences, previous to the development of this work [Pinto *et al.*, 2011e], [Vale *et al.*, 2011b]. The structure of this NN is presented in Figure 3.38.

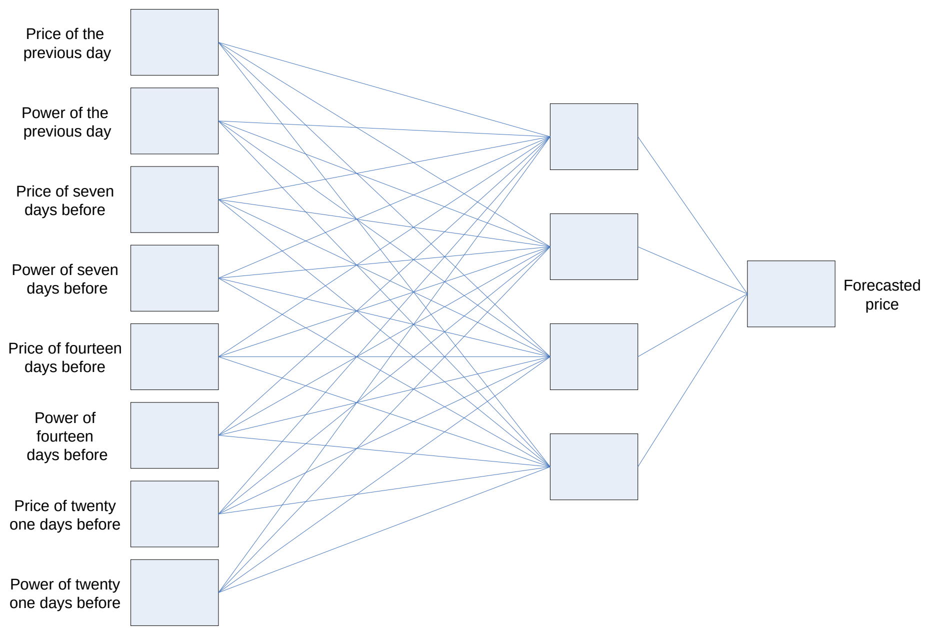

Price of the previous day
Power of the previous day
Price of seven days before
Power of seven days before
Price of fourteen days before
Power of fourteen days before
Price of twenty one days before
Power of twenty one days before
Forecasted price

Figure 3.38 – Neural Network topology.

The NN is implemented in MATLAB [MatLab, 2011], taking advantage of the mathematical tools this language offers. In order to effectively train and execute the NN it was necessary to provide a connection between the JAVA main program and MATLAB. This connection was achieved through the use of the *JMatLink* library [JMatLink, 2011]. This library provides the native methods that enable the use of the MATLAB computational engine inside JAVA applications. With this connection, training and running the NN are done simply by invoking the intended MATLAB function and receiving its return value. The MATLAB function responsible for running the NN is based on the reading of a MS Excel file, which contains the input data for the NN. Supported by this data, the MATLAB file trains the NN, runs it, and saves the output value in a variable that is returned to the JAVA program. The MATLAB function responsible for this process receives three input parameters:

- *rangeInTraining*, representing the Excel file cells interval containing the data used to train the NN;
- *rangeTargetTraining*, containing the Excel file cells interval containing the target results of the training;
- *excelFileName*, indicating the name of the Excel file.

Therefore, the use of this MATLAB function requires the creation of a MS Excel file containing the input data for the training of the NN, based on the log of market history. This file consists in two calculus sheets, one for the training data, and one for the testing data. The test sheet contains eight lines, concerning each of the NN inputs. Regarding the training sheet, it uses the same structure, but with the data referring to the previous days. It also includes a ninth line, with the value of the desired output for each training column.

The number of previous days considered for the training, *i.e.* the number of columns of the training sheet, is defined by the *TrainingLimit* variable. This variable indicates if the NN will use more or less values for training, meaning a faster but worse forecast, or a slower but more effective forecast. Note that considering an exaggerated amount of data may lead to over-training, making the NN memorize the examples instead of learning the relationship between the data. Also, it may lead to the consideration of inadequate data, from a long time before, which possibly has no added value to the learning of the most recent tendencies of the data. Since this strategy can be used by several ALBidS agents at the same time, providing decision support to different MASCEM's market negotiating agents, it was found necessary to make the Excel files' names dynamic, making this name dependent on the agent using it.

In order to fill the Excel files with the necessary values, a query process to the MASCEM database is necessary. The consult of the values from the database originates the creation of matrixes of data, which afterwards are saved into the Excel file. During the implementation process it was verified that, as the necessary amount of data increases, so does the required time to retrieve the data from the database, due to the necessary number of consults. In order to smooth this time's increase, parallel programming was used, dividing each data matrix into various, and creating a different *thread* to fill each of these smaller matrixes. Table 3.4 presents the difference in this strategy's execution time when using parallel computing.

Table 3.4 – NN's average execution time, before and after the implementation of parallel computing.

	Execution Time (milliseconds)	
Training Limit (days)	**Without parallelism**	**With Parallelism**
60	15.000	11.000
120	18.000	13.000
200	22.000	14.000
365	30.000	17.000
730	49.000	20.000

It is visible in Table 3.4 that the reduction in execution time when using parallel programming is more evident when the amount of data is larger. The considered number of *threads* is also adaptive. It depends on the *trainingLimit*, for as larger the matrix is, the larger the number of necessary *threads*. All *threads* are contained by a *ThreadPool,* also adaptive depending on the number of *threads*, which presents several advantages, such as delegating the *threads* scheduling to the operating system, since they cannot be executed all at once due to the high number of *threads* created. The *ThreadPool* also ensures that all *threads* finish their processing before the execution of the global code continues.

3.7.2.3. Experimental Findings

This sub-section presents the tests performed to the Dynamic Artificial Neural Network strategy (NN), in order to validate its adequate execution.

When performing a forecast, this strategy receives as inputs the day and period for which the prediction is required, additionally to the desired *training limit*. All presented tests have been performed for the same starting day, October 4th, and concerning the same period of the day, the twelfth, in order to guarantee the consistency in the comparison of the various results. The considered values for these tests concerning the number of previous days considered for training (*training limit*) are: 60, 200 and 730. Figure 3.39 presents the comparison between the forecasts and the real market price values, referring to twenty four consecutive days, concerning the period and starting day in matter.

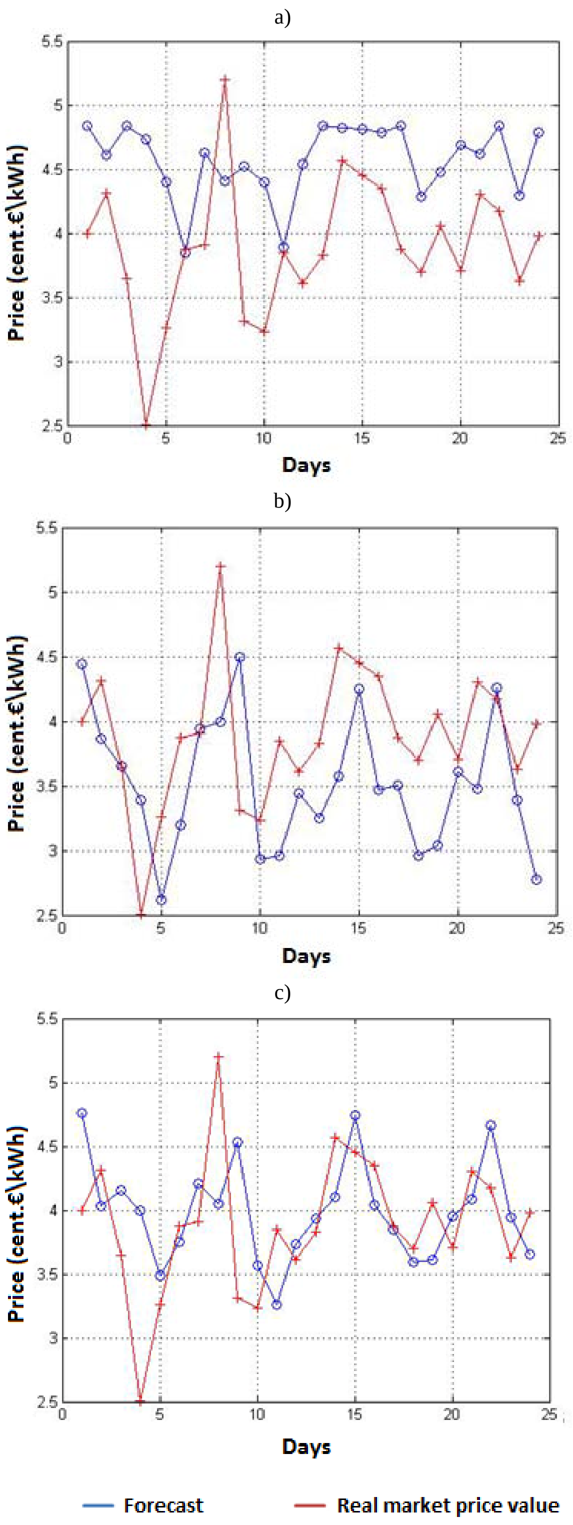


Figure 3.39 – Real and forecasted values, considering *training limits* of: a) 60 days, b) 200 days, c) 730 days.

Figure 3.39 shows that, in the case of a *training limit* of 60 days, although the predictions are not that approximate to the reality, it is visible that the real value tendencies are followed. Regarding the case of a *training limit* equal to 200 days, the predictions are much closer to the real values.

Finally, concerning a *training limit* of 730 days (two years), the predictions are even closer to the real values. Besides the presented tests, other *training limits* were experimented, namely around 1000 days. However, using such an enlarged set of days, the predictions tended to get worst, probably due to the inclusion of data with no longer up-to-date characteristics. Regarding this conclusions, it is suggested that the considered training limit for the predictions should not be above 730 days. Therefore, this is the limit of considered days used by the NN strategy when supporting ALBidS decisions.

Once the strategy is integrated in the ALBidS system, it is important to verify if the quality of the forecasts is translated into incomes when negotiating in the market. Figure 3.40 presents the NN strategy's achieved incomes when supporting Seller 2's negotiations in the market, using the test scenario. These simulations concern the same inputs as the previous tests, for 61 consecutive days.

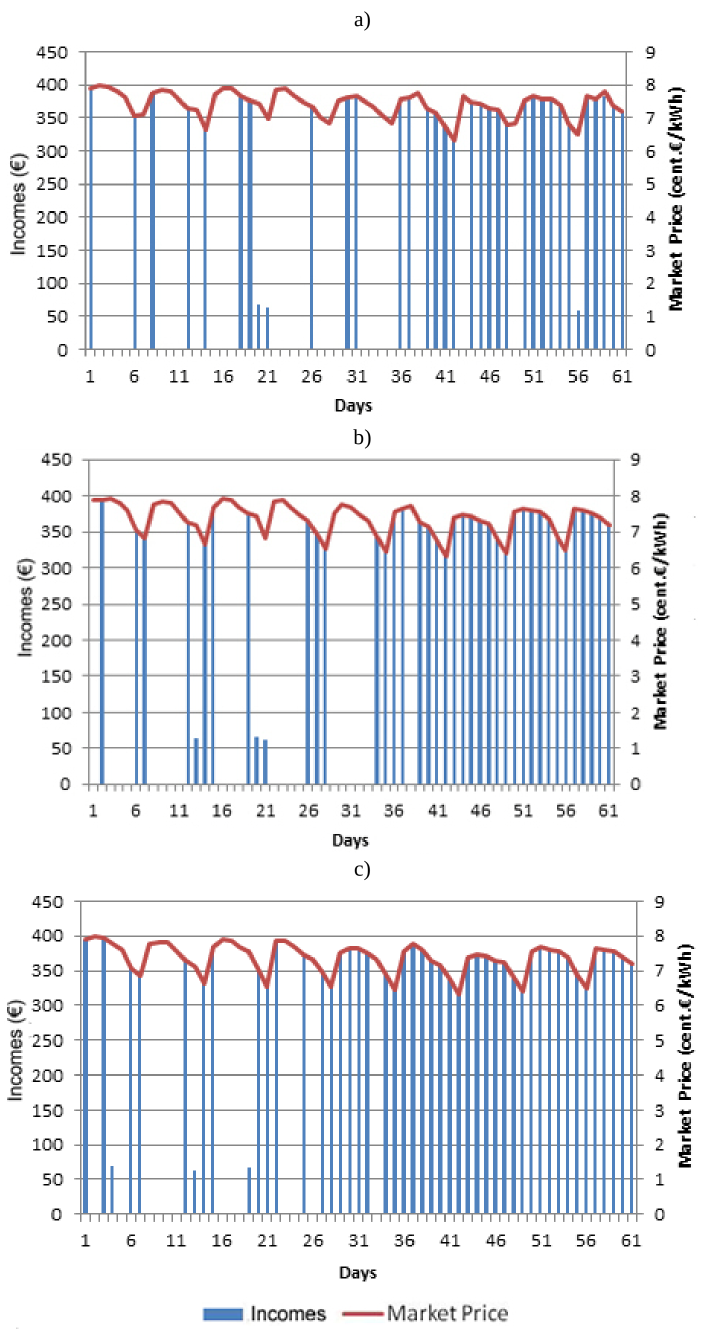


Figure 3.40 – Incomes obtained by Seller 2 using the NN with *training limits* of: a) 60 days, b) 200 days, c) 730 days.

As presented in Figure 3.40, for this case, as the *training limit* increases, so do the incomes and the number of days in which the agent is able to sell energy. This is supported by Figure 3.41, which presents the comparison between the total incomes obtained by Seller 2 throughout all the considered days, for the 24 periods.

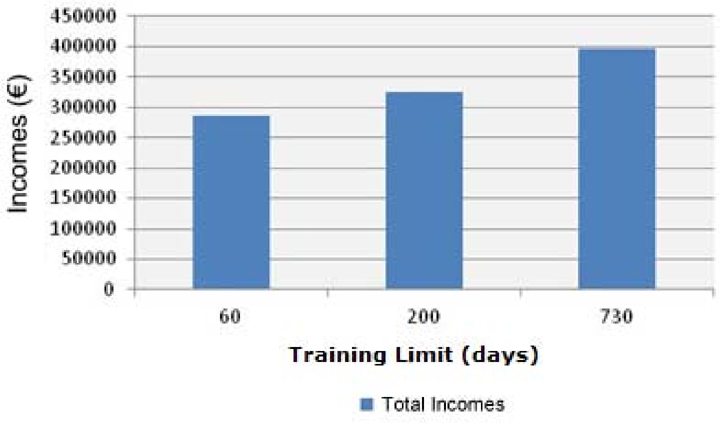


Figure 3.41 – Total incomes obtained by Seller 2 using the NN with *training limits* of: a) 60 days, b) 200 days, c) 730 days.

Analyzing the presented results, it can be concluded that the higher the NN's *training limit* is, the bigger the incomes it is able to achieve in the market, taking advantage on the higher forecasting capabilities. Another important conclusion is the fact that the NN presents higher incomes as the days pass by. This is verified due to the NN being dynamic, and including along the time the most recent events and data in its training, this way adapting to the circumstances. This is verified in all presented cases, independently of the *training limits*, demonstrating the advantage of the NN being dynamic, for a prediction and learning system such as ALBidS.

3.7.3. *AMES Strategy Agent*

This methodology is based on the AMES simulator [Li and Tesfatsion, 2009] (presented in section 2.4) generators' strategic behaviour, which is fully detailed in [Sun and Tesfatsion, 2007].

This strategy uses a reinforcement learning algorithm to choose the best from a set of possible bids, which are defined accordingly to the cost function of each producer. Additionally, a simulated annealing heuristic [Tellidou and Bakirtzis, 2007], [Bandyopadhyay, 2011] is tested, to compare the results when forcing the method to converge faster to a result [Pinto *et al.*, 2011b].

3.7.3.1. Context

This strategy is based on a study of the efficiency and reliability of the Wholesale Power Market Platform (WPMP), a market design proposed by the U.S. Federal Energy Regulatory Commission for common adoption by all U.S. wholesale power markets [FERC, 2003], [FERC, 2011].

The architecture of this type of electricity markets defines the generators' offers as a reported supply offer *S*, which includes the reported cost coefficients *a* and *b*, the reported lower production limit $RCap^L$, and the reported upper production limit $RCap^U$. So, for any generator *i*, a supply offer can be represented in the form (3.12):

$$S_i = \left(a_i, b_i, RCap_i^L, RCap_i^U\right) \tag{3.12}$$

This offers' representation format demands an exhaustive study of the optimal cost coefficients to be reported, in order to get the values that guarantee the higher possible profits. Such studies are performed by simulators of this type of markets, namely the AMES simulator [Li and Tesfatsion, 2009].

MASCEM is a simulator directed to markets with different characteristics, including asymmetrical and symmetrical pool markets, such as the Iberian Market – OMEL [OMEL, 2011], which require a different offers *O* format, including a bid price *P*, and an energy amount E_i. So, for any player *i*, either a generator or consumer, an offer can be represented in the form (3.13):

$$O_i = \left(P_i, E_i\right) \tag{3.13}$$

In spite of the difference between these markets' offers formats, the study of the optimal cost coefficients for WPMP can be used as basis for the calculation of the production cost function for the other type of markets' generators, and consequently define the minimum price that must be bided in order to cover such costs, and provide profits [Pinto *et al.*, 2011b].

3.7.3.2. Implementation

This sub-section is divided in three parts. The first presents the reinforcement learning algorithm used in this strategy, to choose among the possible bids that constitute the *action domain*; the second presents the simulated annealing based heuristic, implemented to facilitate the converging process; finally, the third describes how the *action domain* is created, taking into account the analysis of the producers' cost coefficients [Pinto *et al.*, 2011b].

Reinforcement Learning Algorithm

The producers using this strategy select their supply offers based on their past profits, using the Roth-Erev reinforcement learning algorithm [Jing *et al.*, 2009].

Producer *i* must choose a supply offer from its *action domain ADi*, with finite cardinality *Mi*.

The initial propensity of Generator *i* to choose supply offer *m* ∈ *ADi*, which is equal for all considered supply offers, is given by (3.14):

$$q_{im}(0) = q_i(0), \ \forall \ m \in AD_i \tag{3.14}$$

Now let us consider the beginning of any day *D* > 1, and suppose that the current propensity of producer *i* to choose supply offer *m* ∈ *ADi* is given by $q_{im}(D)$. The choice probabilities that producer *i* uses to select a supply offer for day *D* are then constructed from these propensities (3.15):

$$p_{im}(D) = \frac{\exp(q_{im}(D)/C_i)}{\sum_{j=1}^{M_i} \exp(q_{ij}(D)/C_i)}, \ m \in AD_i \tag{3.15}$$

In (3.15), *Ci* is a smoothing parameter that affects the degree to which producer *i* makes use of propensity values in determining its choice probabilities. As $C_i \to \infty$, then $p_{im}(D) \to 1/M_i$, so in the limit, paying no attention to propensity values in forming its choice probabilities. On the other hand, as $C_i \to 0$, the choice probabilities become increasingly peaked over the particular supply offers *m* having the highest propensity values $q_{im}(D)$, thereby increasing the probability that these supply offers will be chosen.

At the end of day D, the current propensity $q_{im}(D)$ that producer *i* associates with each supply offer $m \in ADi$, is updated in accordance with the following rule (3.16). Let *m'* denote the supply offer that was actually selected and reported into the Day-Ahead Market by producer *i* in day *D*, and let $Profit_{im'}(D)$ denote the profits (positive or negative) attained by this producer in the settlement of the Day-Ahead Market at the end of day *D* in response to its choice of supply offer *m'*. Then, for each supply offer $m \in ADi$:

$$q_{im}(D+1) = [1 - r_i]\, q_{im}(D) + Reinforcement_{im}(D) \tag{3.16}$$

where:

$$Reinforcement_{im}(D) = \begin{cases} [1 - ei]\ Profit_{im'}(D) & \text{if } m = m' \\ e_i\, q_{im}(D)/[M_i - 1] & \text{if } m \neq m', \end{cases} \tag{3.17}$$

where $m \neq m'$ implies $M_i > 2$. The introduction of the *recency* parameter r_i in (3.16) acts as a damper on the growth of the propensities over time. The experimentation parameter e_i in (3.17) permits reinforcement to spill over to some extent from a chosen supply offer to other supply offers to encourage continued experimentation with various supply offers in the early stages of the learning process.

Note that producer *i* in day *D* does not necessarily choose the supply offer with the highest accumulated profits to date. The supply offers generating the highest accumulated profits tend to have a relatively higher probability of being chosen, but there is always a chance that other supply offers will be chosen instead.

Being the objective of this strategy the achievement of the best possible profits, it becomes necessary to adequate this algorithm with a mechanism able to make the choices converge to the best options, and stop allowing bad choices. Because even though this algorithm tends to provide higher probabilities for the best choices, there are always chances for bad choices to be performed. And being this a necessary evil at start, to allow experimentation, when a certain level of experience is achieved, the criterion over the quality of the choices must be adequate, to guarantee a constant good performance.

Simulated Annealing Heuristic

The simulated annealing heuristic has been applied to several learning algorithms [Tellidou and Bakirtzis, 2007], [Bandyopadhyay, 2011] in order to determine their action-selection strategy and to balance exploration and exploitation, which can accelerate the convergence to the optimum avoiding the excessive exploration during the learning process. This heuristic is performed as follows:

Step 1

Select a supply offer $m_{ir} \in ADi$ randomly*;*

Step 2

Select a supply offer $m_{ib} \in ADi$ following the policy (3.18):

$$m_{ib} = \max q_{im}(D),\ \forall\ m \in AD_i \tag{3.18}$$

Step 3

Generate a random number $\xi \in (0, 1)$;

Step 4

$$m'_i(D+1) = \begin{cases} m_{ib} & \text{if } \xi \geq \exp\left[\dfrac{q_{mir}(D) - q_{mib}(D)}{T_i(D)}\right] \\ m_{ir} & \text{otherwise;} \end{cases} \tag{3.19}$$

Step 5

Calculate $T_i(D+1)$ by the temperature-dropping criterion:

$$T_i(D+1) = \lambda T_i(D),\ \lambda \in (0.5, 1) \tag{3.20}$$

Notice that the constantly decreasing value of the temperature is translated into a decreasing acceptance of choices with lower values of propensity. And consequently an increasing demand in choices quality.

Action Domain Construction

In order to build the *action domain ADi* for producer *i*, there will be performed an analysis on the cost coefficients *a* and *b*, which define the variation of a producer costs depending on its actual production *p*. The marginal cost function MC_i is defined as in (3.21):

$$MC_i\ (p) = a_i + 2b_i\ p \tag{3.21}$$

defined over a feasible production interval

$$\mathrm{Cap}_i^L \leq p \leq \mathrm{Cap}_i^U \tag{3.22}$$

where a_i and b_i are producer *i*'s reported cost coefficients, *p* denotes producer i's hourly real-power production level, and Cap_i^L and Cap_i^U are producer *i*'s reported lower and upper real-power production limits.

The *action domain ADi* for producer *i*, is characterized as a matrix of cardinality *Mi*, in which each line, or admissible offer, takes the form (3.23):

$$\alpha_i = \left(RI_i^L, RI_i^U, RCAP_i^U\right) \tag{3.23}$$

This matrix is parameterized by three density-control parameters $M1_i$, $M2_i$ and $M3_i$, satisfying $M1_i \times M2_i \times M3_i = M_i$; three range-index parameters $RIMax_i^L$, $RIMax_i^U$, and $RIMin_i^C$; and a slope-start parameter SS_i.

Defining $M1_i \geq 2$ and $M2_i \geq 2$, step increments are defined as follows:

$$Inc1 = \frac{RIMax^L{}_i}{M1_i - 1} \tag{3.24}$$

$$Inc2 = \frac{RIMax^U{}_i}{M2_i - 1} \tag{3.25}$$

Then specify $M1_i$ equally spaced lower range-index values $\{v_1, \ldots, v_{M1i}\}$ starting at $v_1 = 0.00$ and ending at $v_{M1i} = RIMax^L{}_i$ and spaced at distance Inc1 from each other. Similarly, M2 is specified equally spaced upper range-index values $\{w_1, \ldots, w_{M2i}\}$ starting at $w_1 = 0.00$ and ending at $w_{M2i} = RIMax^U{}_i$ and spaced at distance Inc2 from each other.

The *ADi* matrix is then created with each line being a distinct combination of the previous mentioned values, and so achieving the desired M_i x 3 matrix.

When a certain admissible offer α_i is selected, it must take the following steps, in order to get the required results: a_i^R and b_i^R values:

Step 1

Let l_i and u_i denote producer $_i$'s true marginal costs for producing at its true lower and reported upper production limits, respectively (3.26) and (3.27):

$$l_i = MC_i\,(Cap^L{}_i) = a_i + 2b_i\ Cap^L{}_i \tag{3.26}$$

$$u_i = MC_i\,(Cap^U{}_i) = a_i + 2b_i\ Cap^U{}_i \tag{3.27}$$

Step 2

Now define

$$l_i^R = \frac{l_i}{1 - RI_i^L} \geq l_i \tag{3.28}$$

Step 3

To get $u_i^{start} > l_i^R$

$$u_i^{start} = \begin{cases} u_i & if \quad u_i > l_i^R \\ l_i^R + SS_i & if \quad u_i \leq l_i^R \end{cases} \tag{3.29}$$

Step 4

u_i^R can now be defined as

$$u_i^R = \frac{u^{start}}{1 - RI_i^U} \geq l_i^R \tag{3.30}$$

Step 5

To get b_i^R

$$b_{\mathrm{i}}^{\mathrm{R}} = \frac{1}{2}\,\frac{u_i^R - l_i^R}{Cap_i^U - Cap_i^L} > 0 \tag{3.31}$$

Step 6

To get a_i^R

$$a_{\mathrm{i}}^{\mathrm{R}} = l_i^R - 2\mathrm{b}_{\mathrm{i}}^{\mathrm{R}} Cap_i^L \tag{3.32}$$

Once the required values of a_i^R and b_i^R are achieved, the marginal cost function MC_i (10) can use be used with these values of coefficients for the real production *p*, and determine the minimum value that must be bided to the market, in order to cover such costs. Note that the description of this methodology, which is fully implemented in ALBidS as presented, is completely presented in [Sun and Tesfatsion, 2007], including a numerical example, for an easier comprehension of this procedure.

3.7.3.3. Experimental Findings

This sub-section presents three simulations undertaken using MASCEM with the test scenario, described in the beginning of section 3.7, referring to the same 16 referring to the same 61 days, two consecutive months, starting from September 1st.

This test allows comparing the performance of the AMES strategy when using distinct parameterizations and taking conclusions on its suitability and the influence of the different parameters.

In the first simulation the test subject - Seller 2 will use the proposed method, with the Roth-Erev reinforcement learning algorithm without the simulated annealing heuristic. In the second simulation, Seller 2 will use the simulated annealing to test the acceleration of the convergence to the optimum avoiding the excessive exploration.

After the simulations, the incomes obtained by Seller 2 using both approaches can be compared, as well as the time each takes to converge.

The input parameters for this experience are the following:

For the Roth-Erev reinforcement learning algorithm:

- qi(0) = 0;
- Ci = 0.9;
- e = 0.1;
- r = 0.2.

For the simulated annealing heuristic:

- Ti(0) = 50;

- λ = 0.9.

For the action domain construction:

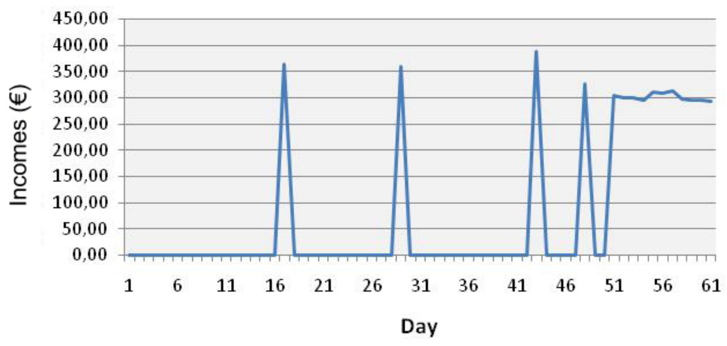
- M1i = 5;
- M2i = 3;
- M3i = 1;
- RIMaxiL = 0.4;
- RIMaxiU = 0.4;
- RIMiniC = 1;
- SSi = 1;
- Mi = M1i x M2i x M3i = 15.

Internal producer's characteristics:

- CapiL = 0 Megawatt (MW);
- CapiU = 50 MW;
- ai = 10.00;
- bi = 0.025.

Figure 3.42 presents the evolution of Seller 2's incomes in the first period of the day, along the 61 days, using each of the two considered approaches (with or without SA).

a)



b)

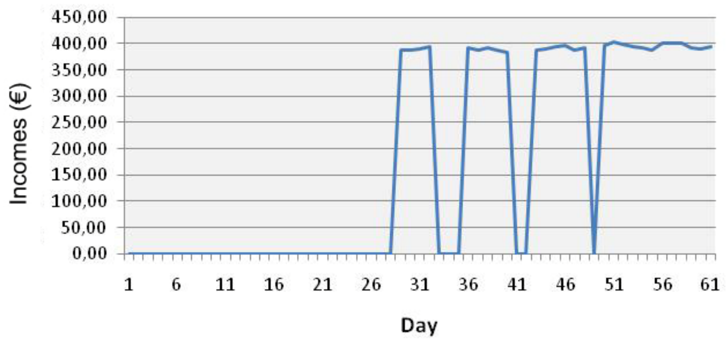


Figure 3.42 – Incomes obtained by Seller 2 in the first period of the considered 61 days, using the proposed method: a) without SA, b) with SA.

Comparing the graphs presented in Figure 3.42, it is visible that the simulation using the proposed method with SA was clearly the most profitable for Seller 2 for this period.

In both simulations the initial periods were unsuccessful, as expected, due to the initial inexperience of the method. As experimentation progresses, the results start to be positive. However, it is obvious, analyzing these results, that in the second simulation, as time progresses and the temperature of SA decreases, forcing the method to choose the best options at an earlier time, the results start to improve much earlier, around day 28. In the approach without SA these results only start to appear around day 42.

These results are a consequence of the convergence time of each method, as presented in Figure 3.43:

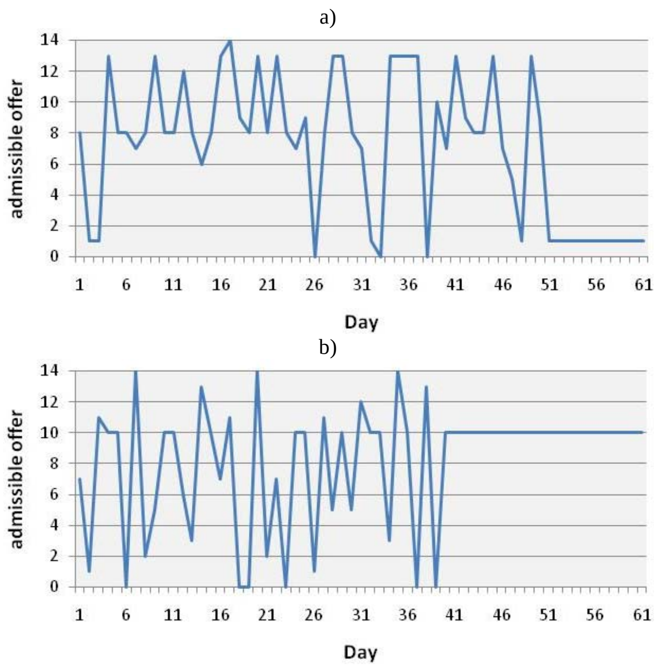


Figure 3.43 – Offer chosen by Seller 2 in the first period of the considered 61 days, using the proposed method: a) without SA, b) with SA.

Both approaches went through an experimentation period, after which they eventually converged to an offer, as supported by Figure 3.43. It is visible that the method without SA took longer to converge.

Comparing Figure 3.43 a) with Figure 3.42 a) it can be concluded that day 51 is the day when the method without SA stabilized for admissible offer 1, coinciding with the day when it started obtaining incomes in a more regular way.

Comparing Figure 3.43 b) with Figure 3.42 b) it is visible that the method with SA stabilized around day 39; however in what concerns the obtained incomes, it still meant a failure in some days after that, as well as success in many days before that point. This means that the offer in which it converged is a more risky offer than in the other approach, translating into more incomes when it is able to sell, comparing to the other approach, but also meaning that this offer is not capable of obtaining incomes in every day.

Note that a certain offer that performs well in a certain day does not guarantee success in all days. There are safer offers, which guarantee incomes in almost all cases, but being incapable of achieving the highest possible gain in each day; and riskier offers, which are able to obtain the higher possible gains, but that risk translates into failure in some other days. This is what can be observed in this case study, with the first approach converging to a safer offer, and the second to an offer with higher gains but higher risk.

3.7.4. *Simulated Annealing Q-Learning Agent*

This strategy uses the Simulated Annealing heuristic to accelerate the process of convergence of the Q-Learning algorithm in choosing the most appropriate from a set of different possible bids to be used by the market negotiating agent whose behaviour is being supported by ALBidS.

3.7.4.1. Context

The Q-Learning algorithm [Rahimi-Kian *et al.*, 2005] is a very popular reinforcement learning method. It is an algorithm that allows the autonomous establishment of an interactive action policy. It is demonstrated that the Q-Learning algorithm converges to the optimal proceeding when the learning state-action pairs Q is represented in a table containing the full information of each pair value [Juang and Lu, 2009].

The basic concept behind the Q-Learning is that the learning algorithm is able to learn a function of optimal evaluation over the whole space of state-action pairs *s x a*. The *Q* function performs the mapping in the way (3.33):

$$Q : s \times a \mapsto U \tag{3.33}$$

where *U* is the expected utility value when executing an action a in the state s. As long as the state and action states do not omit relevant information, nor introduce new information, once the optimal function *Q* is learned, the agent will know precisely which action results on the higher future reward, in a particular situation *s* [Juang and Lu, 2009].

The Q(s, a) function, regarding the future expected reward when action a is chosen in the state s, is learned through try and error, following the equation (3.34):

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha[r_t + \gamma V_t(s_{t+1}) - Q_t(s_t, a_t)] \tag{3.34}$$

where α is the learning rate; *r* is the reward or cost resulting from performing the action a in the state *s*; γ is the discount factor; and *U* (3.35) is the state *s* utility resulting from action *a*, obtained using the *Q* function learned so far.

$$U_t(S_{t+1}) = \max_a Q(s_{t+1}, a) \tag{3.35}$$

The Q-Learning algorithm is executed as follows:

- For each *s* and *a*, initialize *Q(s,a)* = 0;
- Observe s;
- Repeat until the stopping criterion is satisfied:
    - Select action a, using the current action policy;

- Execute action *a*;
- Receive immediate reward *r(s,a)*;
- Observe new state s';
- Update *Q(s,a)*;
- $s \leftarrow s'$.

As the visiting of all state-action pairs tends to infinite, the method guarantees a generation of an estimative of Q_t which converges to the value of Q. In fact, the actions policy converges to the optimal policy in a finite time, however slowly.

The acceleration of the learning process, and consequently of the convergence process can be performed through the application of an heuristic, such as the Simulated Annealing, as shown by several studies [Tellidou and Bakirtzis, 2007], [Zhifeng *et al.*, 2009].

3.7.4.2. Implementation

The Simulated Annealing Q-Learning (SA-QL) strategy's purpose is to adapt the AMES strategy, presented in sub-section 3.7.3, using its fundamentals: the use of a reinforcement learning algorithm to choose the best from a set of possible actions. This adaptation concerns two main aspects:

- the experimentation of an alternative reinforcement learning algorithm, well known for its good performance reputation – Q-Learning (QL);
- and the consideration of a new type of possible actions (or *Action Domain*) definition.

The SA-QL strategy also implements the Simulated Annealing (SA) heuristic to accelerate the convergence process, as it proved to be effective in its purpose in the AMES strategy's scope.

Regarding the possible actions for this domain, the desire is for them to cover a large set of possibilities, surrounding the point where the actions are optimal. In order to do this, firstly, the SA-QL strategy receives as input (from the user, when starting its execution) the number of desired possible actions *numB*, and a range, or interval *int*, which defines the length of the considered bids' space. These two input variables define the dispersion of the possible bids along the area surrounding the expected optimal bid.

The reference point *ref*, around which the possible bids are defined, must be a value that guarantees a proximity to the optimal value. In order to find this value, the average market price of the previous day is used as reference. It is known that the variance in the market prices can be smaller or bigger depending on each situation, but using the average price of the day before guarantees that a recent reference is considered, and that this value points in the direction of the area where the market price for the next days will be located.

The definition of the exact values that the possible actions will assume, stored in the *possibleBids* vector requires the initial calculation of the minimum considered bid value, *min*, as in (3.36).

$$\text{min} = \text{ref} - \frac{\text{int}}{2} \tag{3.36}$$

This minimum value is used as starting point for the application of an increment *inc* (3.37), which defines the set of possible bids.

$$\text{inc} = \frac{\text{int}}{\text{numB} - 1} \tag{3.37}$$

Using these two calculated values, the *possibleBids P* are defined as in (3.38).

$$\begin{cases} P_0 = \text{min} \\ P_i = \text{min} + \text{inc} \times i \quad , \quad i = [1, \text{numB} - 1] \end{cases} \tag{3.38}$$

Each of the *possibleBids* has associated a confidence value, which represents the propensity or *Q* value of each possibility. These values are initialized equally for all possibilities, and are then updated throughout the days, depending on the performance of each action.

To assist in the update of each action's *Q* value, two additional inputs are required - the *learning rate* which defines the speed of the learning process; and the *error*, representing the maximum tolerated error, or difference, between an action's output, and the actual market price. This means that, in order to properly update the Q-Learning algorithm, and make it learn properly, it is required that the SA-QL method also receives the actual market prices in each moment, to evaluate the quality of an action's performance.

The evaluation of an action's performance is done through the calculation of the difference between the actual market price, and the suggested bid. If this difference value is located below the tolerated *error* value, the attributed reinforcement value *reinfValue* will be positive; otherwise, the reinforcement is negative, as the selected action has originated a result with an error superior to the allowed value.

The *reinfValue* is what in fact will "correct" the *Q* values, and consequently "teach" the Q-Learning algorithm. The update of the *Q* confidence value is performed by applying (3.39).

$$newValue = (1-learningRate) * oldValue + learningRate * reinfValue \tag{3.39}$$

This process will update the *Q* value associated to the considered element of the *possibleBids*, which suggested the current action.

The experimentation of the actions, leading to the convergence on the optimal one, is slow. For this reason the *Simulated Annealing* heuristic is applied, in order to increase the speed of convergence. The way this heuristic is applied is done in the same way as in its application to the AMES strategy. Consequently, the process is not described in this sub-section.

3.7.4.3. Experimental Findings

This sub-section presents some of the tests performed to the SA-QL strategy, in order to verify its correct performance. These tests are intended to provide conclusions about the influence of the input variables. Therefore, the results of a total of six simulations are presented, concerning the variation of: the *learning rate*, the *admissible error*, and the SA initial temperature.

All simulations are performed using the test scenario, for 61 consecutive days, starting on the October 4th. For all tests, the number of desired possible bids and the possible bids interval, are maintained constant, both assuming the value 10.

Regarding the first two simulations, with the purpose of analyzing the influence of the *learning rate*, this variable assumes the value of 0.5 in the first test and 0.2 in the second. The *admissible error* and the SA initial *temperature* present the same values in the two simulations, respectively 0.8 and 50. This way it is possible to analyze the influence of reducing the *learning rate*, under the same circumstances in what concerns all the other variables. Figure 3.44 presents the results of the first two simulations.

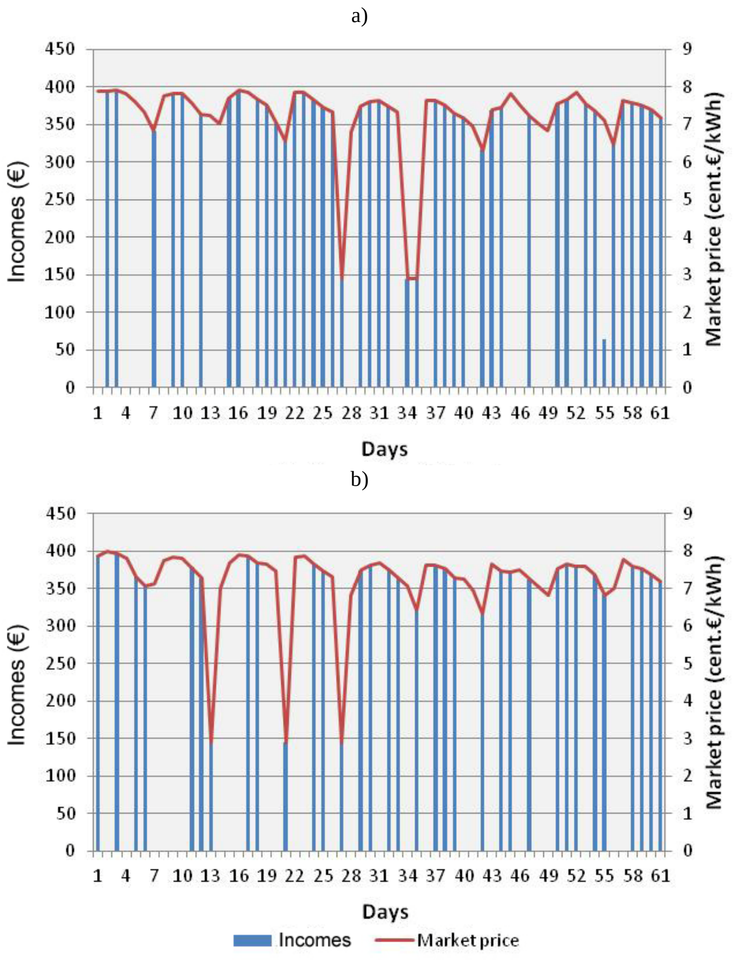
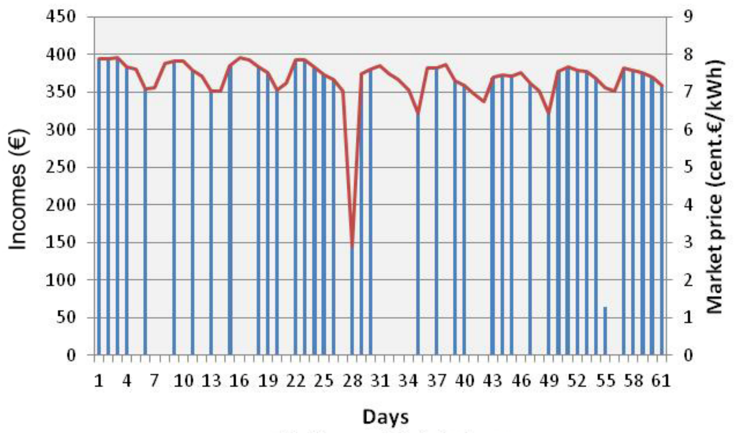


Figure 3.44 – Incomes obtained by Seller 2 using the SA-QL strategy with a *learning rate* of: a) 0.5, b) 0.2.

Comparing the graphs presented in Figure 3.44, it is visible that the results of the first simulation are better than the second, in what concerns the achieved incomes. The conclusions taken from this case are that, lowering the *learning rate* to very small value, the SA-QL strategy takes longer to learn, and that is translated into a worst adaptation to the market circumstances, and therefore, reflected in its results.

The following test takes the SA temperature as the main object of study. The goal is to understand how the temperature variation affects the results. In order to achieve that, both the *learning rate* and the *error* value are assigned a value of 0.8, which remains static for the next two simulations. Regarding the temperature, it presents a value of 50 in the first simulation, and a value of 20 in the second.

a)



b)

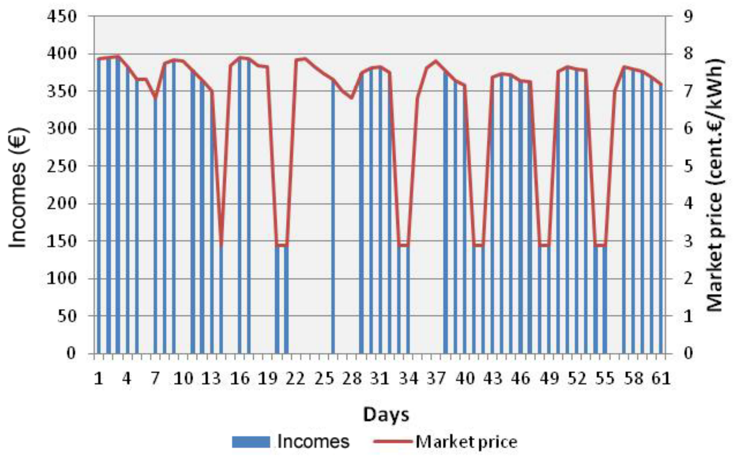


Figure 3.45 – Incomes obtained by Seller 2 using the SA-QL strategy with a *temperature* of: a) 50, b) 20.

Comparing the results from the two simulations presented in Figure 3.45, a reduction in incomes is visible when lowering the temperature too much. This incomes reduction, even though not very accentuated, is due to the premature lowering of the learning process, because of the lower temperature, which made the strategy converge very quickly to an offer that proves not to be the best one. The total incomes obtained by Seller 2 in the first simulation of this test were equal to 14,736.50€, and in the second simulation they were equal to 14,390.96€.

In the final test, the temperature assumes the value 50 for both simulations, while the learning rate is fixed at 0.2. The error will take the value of 0.8 for the first simulation, and 0.4 for the second. Figure 3.46 presents the results for this final test.

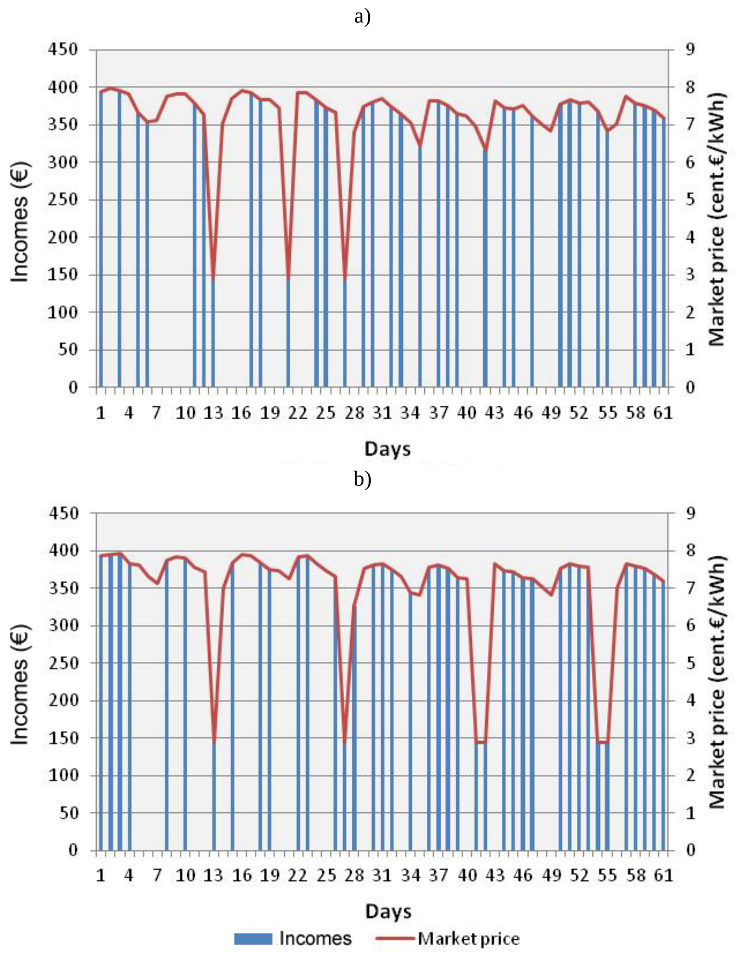


Figure 3.46 – Incomes obtained by Seller 2 using the SA-QL strategy with a *learning rate* of: a) 0.5, b) 0.2.

Comparing the results presented in the graphs of Figure 3.46, it is visible that the incomes increase when lowering the error value. The achieved incomes in the first simulation totalized 11,919.50€, while in the second simulation they were equal to 13,677.50€. As the second simulation presents a smaller admissible error, the strategy converges to a solution that is closer the real values. That proved to be a successful action, providing the opportunity to achieve higher incomes.

The tests presented in this sub-section allowed addressing the importance that the refining of the several parameters has to the performance of the SA-QL strategy. The presented tests provided some indications about the adequate actions to take when choosing the parameters for this strategy's successful functioning. However, it is certain that the testing of a much larger combination of alternatives would surely be interesting for taking more effective and final conclusions.

3.7.5. *Game Theory Agent*

This agent’s strategy is characterized as a scenario analysis algorithm able to support strategic behaviour, based on the application of the game theory.

3.7.5.1. Context

Game theory deals with circumstances where a person’s success is based upon the choices of others. This theory has been applied in many areas, namely in mathematics, economics, political science, psychology, and computer science. Its classic uses include a sense of balance in numerous games, where each person has found or developed a tactic that cannot successfully improve his results, given the other approach [Osborne and Martin, 2004].

Game theory has started to play an increasingly important role in logic and computer science. Several logical theories have a basis in game semantics. In addition, computer scientists have used games to model interactive computations. Also, game theory provides a theoretical basis to the field of multi-agent systems [Shoham *et al.*, 2009].

The field of algorithmic game theory combines computer science concepts of complexity and algorithm design with game theory and economic theory [Gintis, 2000]. The emergence of the internet has motivated the development of algorithms for finding equilibrium in games, markets, computational auctions, peer-to-peer systems, and security and information markets.

An important issue in game theory is the concept of perfect information. A game is regarded to as being of perfect information if all players know the moves previously made by all other players. Thus, only sequential games can be games of perfect information, since in simultaneous games not every player knows the actions of the others. Most games studied in game theory are imperfect information games, such as poker and contract bridge. Although there are some interesting examples of perfect information games, including chess, go, and mancala. Perfect information is often confused with complete information, which is a similar concept. Complete information requires that every player know the strategies and payoffs available to the other players but not necessarily the actions taken.

3.7.5.2. Implementation

The scenario analysis algorithm supports strategic behaviour with the aim of providing complex support to develop and implement dynamic pricing strategies.

This agent develops a strategic bid, taking into account not only its previous results but also other players’ bids and results and expected future reactions. This is particularly suitable for markets based on a pool or for hybrid markets, to support sellers’ and buyers’ decisions for proposing bids to the pool and accepting or not a bilateral agreement. The algorithm is based on the analysis of several bids under different scenarios. The analysis’ results are used to build a matrix which supports the application of a decision method to select the bid to propose. Each agent uses the historical information about market behaviour and about other agents' characteristics and past actions. This algorithm's organization is presented in Figure 3.47.

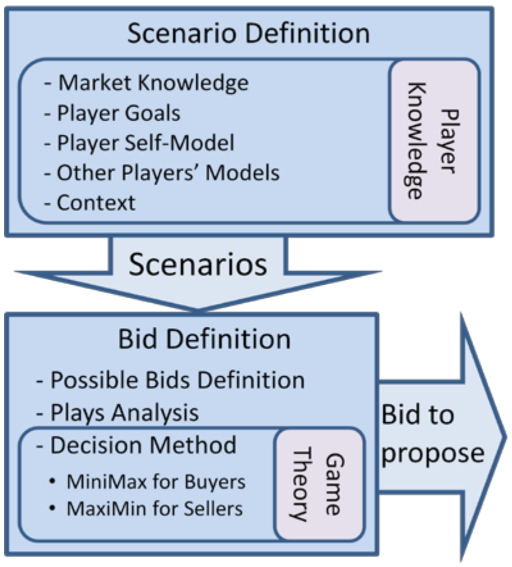


Figure 3.47 – Scenario Analysis Algorithm [Pinto *et al.*, 2011a].

To get warrantable data, the agents using this method perform an analysis of the historical data. With the gathered information, agents can build a profile of other agents including information about their expected proposed prices, limit prices, and capacities (see section 3.8 – Player Profiles Definition). With these profiles, and based on the agent's own objectives, several scenarios, and the possible advantageous bids for each one, are defined.

Seller and buyer agents interact with each other, in MASCEM environment, taking into account that their results are influenced by competitor's decisions. Game theory is well suited for analyzing these kinds of situations [Owen, 1995], [Neumann *et al.*, 2003].

Scenario definition

MASCEM is implemented as a decision support tool, so the user should have the flexibility to decide how many and which scenarios should be analyzed. To do so, the user must define the scenarios to be simulated by specifying the price that competitor agents will propose (3.40):

$$price_{i+1} = price_i \pm amount_{i+1} \tag{3.40}$$

where λ and φ are scaling factors that can be different for each agent and for each scenario.

Let us suppose that the user selects λ=0 and φ=1 for every seller and λ=1 and φ=0 for every buyer; this means an analysis of a pessimistic scenario. If the user selects λ=1 and φ=0 for every agent, then the most probable scenario will be analyzed. Using this formula the user can define for each agent the proposed prices for every scenario that it desires to consider.

In order to build adequate profiles for competitor agents, taking into account their past actions and reactions to different events, a player profile definition mechanism was developed [Pinto *et al.*, 2011c], and will be presented in section 3.8.

Bid definition

An agent should analyze the income that results from bidding its limit, desired, and competitive prices - those that are just slightly lower (or higher, in the buyers' case) than its competitors' prices.

A play is defined as a pair bid – scenario, so, the total number of plays to analyze is (3.41):

$$n = number\_of\_bids \times number\_of\_scenarios \tag{3.41}$$

and the maximum value it can achieve is (3.42):

$$\left(2 \times n + 2\right) \times 2^{n} \tag{3.42}$$

considering that agents only bid their limit or expected prices. However, an agent may bid prices between its limit and expected prices, or even above that limit price. If we consider each agent may bid *np* prices, the number of scenarios becomes equal to np^n, and the number of plays to analyse is (3.43).

$$\left(np \times n + 2\right) \times np^{n} \tag{3.43}$$

The user is also allowed to choose the number of bids that will be considered as possibilities for the final bid. In this case, the value of the bids is calculated depending on an interval of values that can also be defined by the user. That interval is always centred on a trusted value, the value of the market price of the same period of the previous day. This way the considered possible bids are always around that reference value, and their range of variance depends on the bigger or smaller value of the user defined interval.

So, being *nb* the number of bids defined by the user, *int* the value defining the interval to be considered, and *mp* the market price from the same period of the previous day, the possible bids *b1..nb* are defined as (3.44) and (3.45):

$$\mathrm{b}_1 = mp - \frac{int}{2} \tag{3.44}$$

$$b_m = b_{m\text{-}1} + \left(\frac{int}{nb-1}\right), \quad m \in \left[2, nb\right] \tag{3.45}$$

After defining all the scenarios and bids, market simulation is applied to build a matrix with the expected results for each play.

The matrix analysis with the simulated plays' results is inspired on the game theory concepts for a pure-strategy two-player game, assuming each player seeks to minimize the maximum possible loss or maximize the minimum possible gain.

After each negotiation period, an agent may increase, decrease or maintain its bid, increasing the number of scenarios to analyse. So, after k periods, considering only three possible bid updates, the number of plays to analyse becomes (3.46):

$$\left(np \times n + 2\right) \times np^{n} \times 3^{(k-1)\times n} \tag{3.46}$$

Game Theory for Scenario Analysis

A seller, like an offensive player, will try to maximize the minimum possible gain by using the *MaxiMin* decision method. A buyer, like a defensive player, will select the strategy with the smallest maximum payoff by using the *MiniMax* decision method.

Buyers' matrix analysis leads to the selection of only those situations in which all the consumption needs are fulfilled. This avoids situations in which agents have reduced expenses but cannot satisfy their consumption needs completely.

The state space to be searched is related to the possible plays of other agents, regarding possible bids from one agent. Figure 3.48 illustrates this procedure.

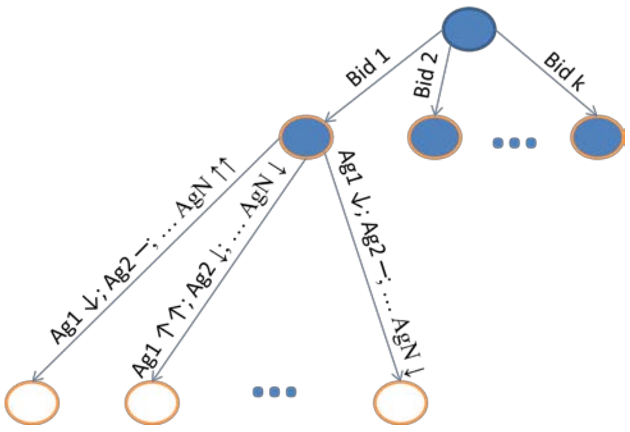


Figure 3.48 – Game theory for scenario-analysis space search.

Each bid of a specific agent (*e.g. Ag_i*) is analyzed by considering several possible scenarios, in order to support the decision of this agent. The scenarios are evaluated by considering the prices other agents may propose, regarding the previous proposed prices. It is also considered that each agent may change its price: increasing a lot (↑↑), increasing a little (↑), maintaining (—), decreasing a little (↓), or decreasing a lot (↓↓) its bid price. Here the concepts of little and lot will consider the historic data of agents' bids and will be converted to variations in cents. It is important to observe that it is impossible to consider all kind of variations, due to the complexity of the problem, as it has been seen before. The required time for solving the problem with a large set of combinations would be impractical since a complete market simulation is required for each scenario.

Each leaf node of the tree in Figure 3.48 corresponds to a possible scenario. The idea is to evaluate each one of these scenarios and apply a *MiniMax* or *MaxiMin* based algorithm to select the safest bid to be offered by agent Agi.

Notice that the use of game theory is intended for supporting one specific agent and not for achieving the equilibrium in the market. The idea of this methodology is to provide a specific agent with decision support.

For each simulated scenario (leaf of Figure 3.48) the price P_{market} for each MW.h (Megawatt hour) is calculated, defined as the result of the simulated market. For the support of seller agents the evaluation of the scenario is made by the product of the energy sold by the supported agent Ag_i, $Energy\_Sold_i$, by the profit, obtained from the difference between P_{market} and the cost associated to each MW.h sold by Ag_i, $Cost_i$, according to (3.47):

$$F = Energy\_sold_i \cdot (P_{market} - Cost_i) \tag{3.47}$$

Notice that the part of this formula that demands the higher processing cost is the calculation of the value P_{market}, since it implies to run the simulation of the scenario in order to determine the market clearing price.

Additionally, there are two methods for solving problems of equality in the evaluation of scenarios. In case of a seller, the *MaxiMin* algorithm chooses the bid that offers the maximum gain, from the worst possible scenario. In case of more than one scenario being evaluated with equal value as worst scenario, the options for choosing among them are:

- A greedy approach, choosing the scenario, among the equally worst ones, that presents the bid that allows the higher payoff from all the possible bids;
- An average of the results of all possible bids for these scenarios, choosing the one that gets the worst average as the worst possible scenario.

The user is able to choose among these two methods for solving the problems of equality. He/she can also choose a third option that is a mechanism that chooses automatically among these two options, accordingly to the success that each of them is presenting. This mechanism uses a reinforcement learning algorithm, with initial equal values of confidence for the two options. As the time evolves, the values of success of each option are updated, and the one that presents the best confidence in each run, is the chosen option.

The updating of these confidence values is performed by running the two options and saving the answer proposed by each one. Later, after the bid is chosen as the agent's action for the actual market, this method analyzes the market values and checks which of the outputs proposed by each method would have led to the best results.

3.7.5.3. Experimental Findings

This sub-section presents three simulations undertaken using MASCEM with the test scenario, presented in the beginning of section 3.7, referring to the same 16 consecutive days, starting from Friday, 15th October.

This test allows comparing the performance of this method when using distinct parameterizations and taking conclusions on its suitability and the influence of the different parameters. The parameters influence is an important issue to be tested, because of the necessity for the method to run under different requirements in terms of execution times, due to the efficiency/effectiveness mechanism (see section 3.9 - Efficiency/Effectiveness Management).

The common parameters in all the simulations are: the selection of the automatic mechanism for solving the problems of equality among scenarios; for all seller agents the limit price is fixed as 0 c€/kWh, for it does not make sense to bid negative values; for all buyer agents the limit price is 20 c€/kWh, a high value for allowing the players to consider a good margin of prices. Also, the selected reinforcement learning algorithm for the players' profiles definition has been the revised Roth-Erev, with equal value of the algorithms weight. The past experience weight *W* value is set to 0.4, a small value to grant higher influence to the most recent results, so that it can quickly learn and catch new tendencies in players' actions. For each scenario the scaling factors for competitors' probable price λ and limit price φ, will be equal for every competitor agent, in order to give the same importance to the price forecast of each agent. These scaling factors will only vary from scenario to scenario, but always maintaining the equality among agents.

The variations introduced in each simulation are concerning the test subject – Seller 2:

- In the first simulation Seller 2 will use the scenario analysis method with a small number of considered scenarios and possible bids. This test will allow us to realize if a restrict group of scenarios, and consequent advantage in processing speed, will be reflected on a big difference in the results quality. For this simulation the number of considered scenarios is 3, the number of considered bids is 5, and the interval for the possible bids definition is 8. Considering the 3 scenarios, the first will assign to all agents λ=1 and φ=0; the second λ=0,95 and φ=0,05; and the third λ=0,9 and φ=0,1. These values give higher importance to the most probable prices, in order to consider the most realistic scenarios.
- In the second simulation Seller 2 will use the scenario analysis method with an intermediate number of considered scenarios and possible bids. The number of considered scenarios is 5, the number of considered bids is 7, and the interval for the possible bids definition is 8. Considering the 5 scenarios, the first will assign to all agents λ=1 and φ=0; the second λ=0,95 and φ=0,05; the third λ=0,9 and φ=0,1; the fourth λ=0,8 and φ=0,2; and the fifth λ=0,7 and φ=0,3.
- Finally, in the third simulation Seller 2 will use the method with a higher number of considered scenarios and possible bids, in order to obtain a more detailed analysis. The number of considered scenarios is 7, the number of considered bids is 10, and the interval for the possible bids definition is 10, granting also a bigger interval for considered bids. Considering the 7 scenarios, the first will assign to all agents λ=1 and φ=0; the second λ=0,95 and φ=0,05; the third λ=0,9 and φ=0,1; the fourth λ=0,8 and φ=0,2; the fifth λ=0,7 and φ=0,3; the sixth λ=0,5 and φ=0,5; and the seventh λ=0,2 and φ=0,8.

After the simulations, the incomes obtained by Seller 2 using the proposed method with each of the three combinations of parameters can be compared. This agent's power production to be negotiated in the market will remain constant at 50MW for each period throughout the simulations.

Since the reinforcement learning algorithm for the players' profiles definition treats each period of the day as a distinct case, the analysis of the development of the performance must be done for each period individually. Figure 3.49 presents the evolution of Seller 2 incomes in the first period of each considered day, along the 16 days, using each of the three considered combinations of parameters.

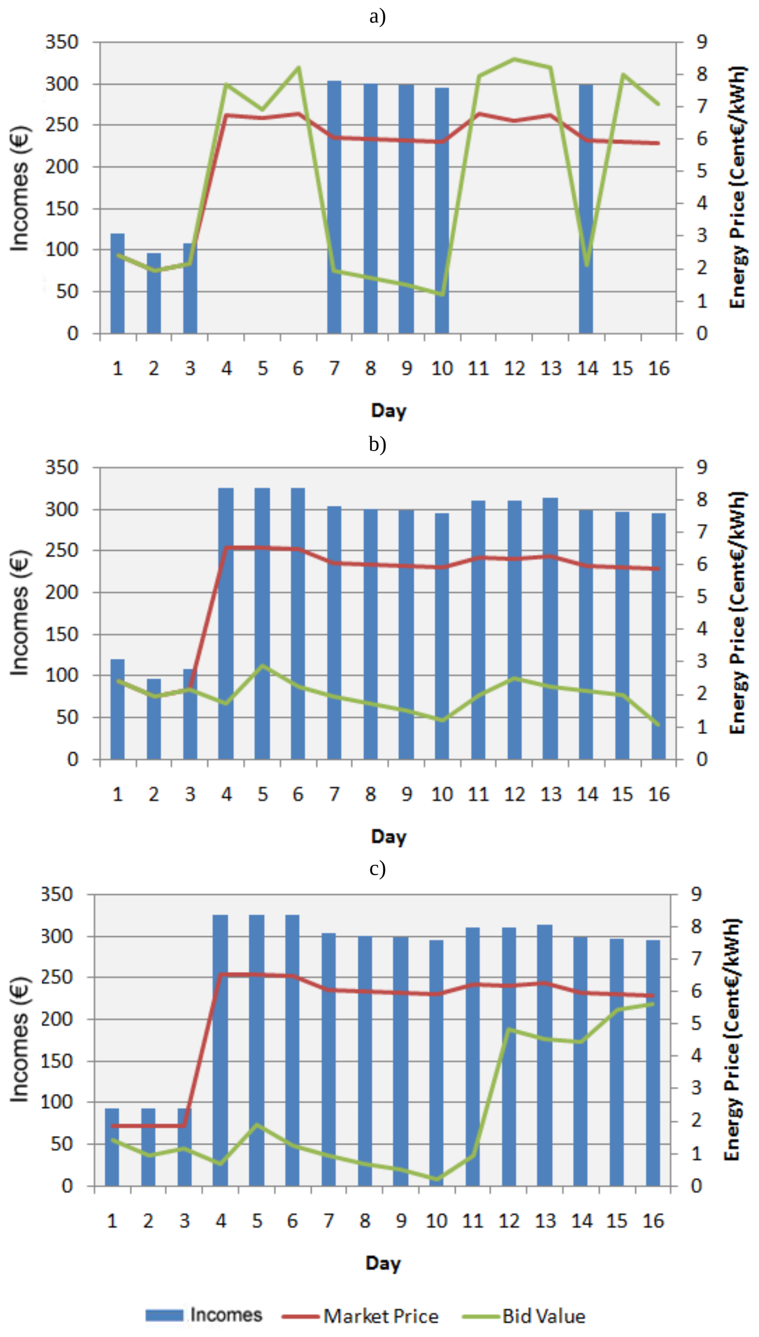


Figure 3.49 – Incomes obtained by Seller 2 in the first period of the considered 16 days, using: a) the first parameterization, b) the second parameterization, c) the third parameterization.

Figure 3.50 presents the results of Seller 2 in the twelfth period of each considered day.

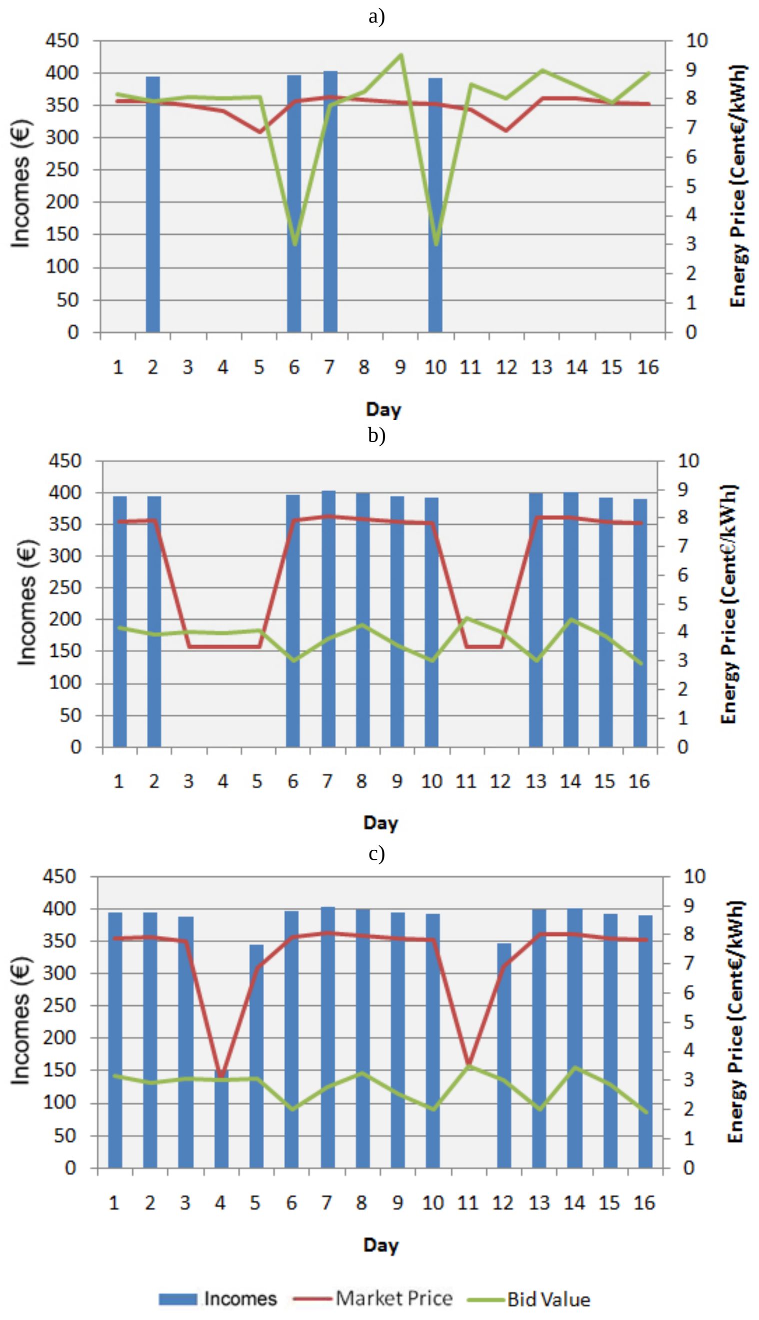


Figure 3.50 – Incomes obtained by Seller 2 in the twelfth period of the considered 16 days, using: a) the first parameterization, b) the second parameterization, c) the third parameterization.

Figure 3.51 presents the global results of the three cases along the 24 periods of the considered 16 days.

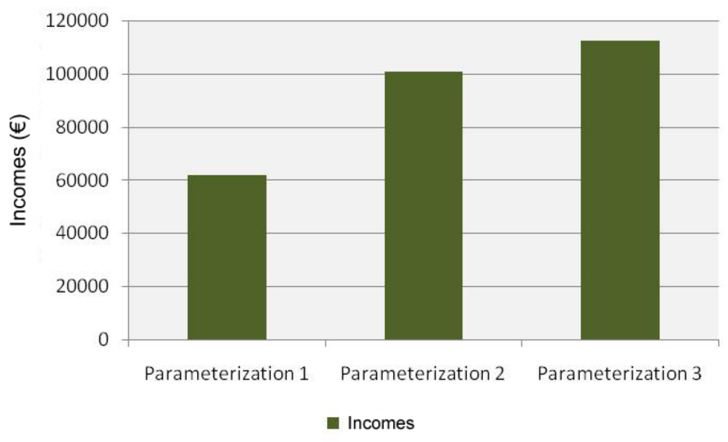


Figure 3.51 – Total incomes obtained by Seller 2 for the considered 16 days.

Comparing the graphs presented in Figure 3.49, it can be concluded that the first simulation was clearly the most disadvantageous for Seller 2 for this period. The second and third simulations present very similar results in what concerns the incomes obtained by this agent in the first period.

The results of the twelfth period show the first parameterization worst results when compared with the other two. However, in this case, the third parameterization clearly obtained better results than the second one. The global results for all periods of the considered 16 days, presented in Figure 3.51, support this tendency; a large difference between the first parameterization and the others, and a smaller difference between the results achieved by the second and third parameterizations can be clearly seen.

The comparison of the different parameterizations' performances also allows taking an important conclusion: when it is required for the simulations to improve the processing times, a criterious reduction of the search space may not represent a significant decrease of the method's effectiveness. As proven by simulation 2, which even though considering fewer scenarios and possible bids than the parameterization of simulation 3, its results were still acceptable for situations for which the method's processing time is crucial (see section 3.9 – efficiency/effectiveness management).

3.7.6. *Error Theory Agents*

Given that forecasts and predictions are always subject to some error, it is important to analyze that error properly, in order to try to overcome that. When analyzing the forecasting errors' distribution over time, it becomes visible that many times those errors show patterns in their occurrence. In some cases, forecasts fail by predicting higher or lower values than the reality, in recurrent events that may have something in common (*e.g.* their context).

This strategy's goal is to analyze the forecasting errors' evolution of a certain forecasting method, to try finding patterns in that error sequence and provide a prediction on the next error, which will be used to adequate the initial forecast.

3.7.6.1. Context

When a prediction is made, there will always be some error or uncertainty. For any measurement, there are a set of factors that may cause deviation from the actual (theoretical) value. Most of these factors have a negligible effect on the outcome and usually can be ignored. However, some effects can cause a significant change, or error, in the final result. In order to achieve a useful prediction, it is necessary to have an idea of the amount of the errors [Principe, 2010].

In adaptive systems, the data input is important in order to adjust the system so that it learns the best way. The data introduced must take into account the outputs generated by the system in previous iterations, appropriate to reflect the difference between the reality and the forecast results. This is called the prediction error. To represent this error a Gaussian distribution is used most of the time [Rao and Principe, 2008].

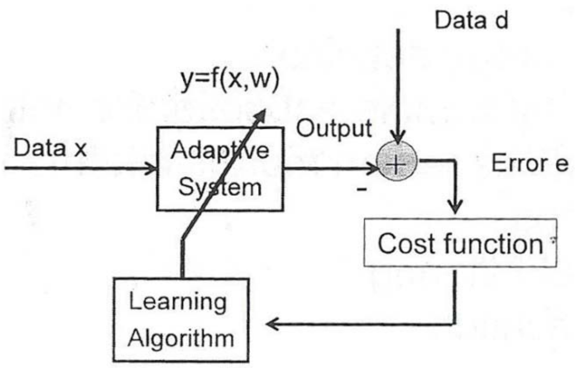


Figure 3.52 – Optimal adaptive models [Principe, 2010].

The cost function, responsible for calculating the prediction error, is the most widely used Mean Square Error (MSE), because it offers the best solution, although it is not guaranteed when using a Gaussian distribution. Another measure of accuracy to qualify the error is the Mean Absolute Percentage Error (MAPE), which expresses the precision in percentage and is defined by (3.48):

$$M = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \tag{3.48}$$

However, for nonlinear systems this type of error analysis is not adequate. The criterion for determining that error may be based on entropy or divergence [Rao and Principe, 2008].

There are different approaches regarding the definition of entropy:

- Hartley Information (1928) [Hartley, 1928]
  - Large probability – Small information

$$p_x(x) = 1 \rightarrow S_H = 0 \tag{3.49}$$

  - Small probability – Large information

$$p_x(x) = 0 \rightarrow S_H = \infty \tag{3.50}$$

  - Two identical channels should have twice the capacity as one

$$g(p_x(x)^2) = 2g\big(p_x(x)\big) \tag{3.51}$$

- Log2 is a natural measure for additivity

$$S_H = -\log_2 p_x(x) \tag{3.52}$$

- Shannon Entropy [Shannon, 1948]:

$$H_S(X) = E[S_H] = -\sum p_x(x)\log \quad p_x(x) \tag{3.53}$$

- Rényi Entropy [Rényi, 1970]:

$$H_\alpha(X) = \frac{1}{1-\alpha}\log \sum p_X^\alpha(x) \tag{3.54}$$

Rényi entropy becomes entropy when Shannon's entropy when:

$$\alpha \to 1 \tag{3.55}$$

- Fisher Entropy (local) [Fisher, 1922]:

$$H_f(X) = \int \frac{[p'_X(x)]^2}{p_X(x)}\,dx \tag{3.56}$$

For divergence, there are the following approaches:

- Kullback-Leibler divergence [Kullback and Leibler, 1951]

$$K(f,g) = \int f(x)\log(f(x)/g(x))\,dx \tag{3.57}$$

$$I(x,y) = \int\int f_{xy}(x,y)\log(f_{xy}(x,y)/f_x(x)f_y(y))\,dx\,dy \tag{3.58}$$

- Mutual information [Principe, 2010]

$$I(x,y) = H(x) + H(y) - H(x,y) = H(x) - H(x|y) = H(y) - H(y|x) \tag{3.59}$$

To address the existing error problem determined by the entropy and divergence, José C. Príncipe proposes the use of the Information Theory [Príncipe, 2010].

The Information Theory [Liu *et al.*, 2008] is a probabilistic description of random variables that quantify the very essence of the communication process. It has been fundamental in the design and quantification of communication systems. The Information Theory provides a consistent and quantitative framework to describe processes with partial knowledge (uncertainty). Figure 3.53 shows that not all random events are also random.

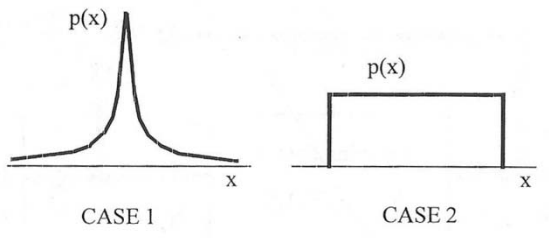


Figure 3.53 – Different random events [Principe, 2010].

Thus, it is suggested to overlap the samples in standard distributions to determine which fits the best.

Laplace distribution:

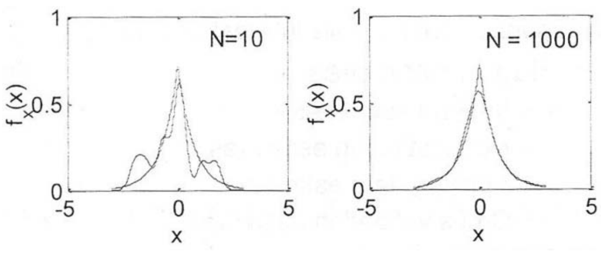


Figure 3.54 – Overlapping samples in Laplace distributions [Principe, 2010].

Uniform distribution:

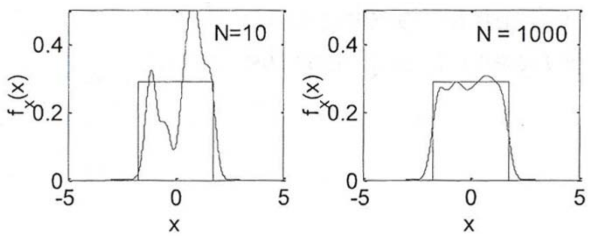


Figure 3.55 – Overlapping samples in uniform distributions [Principe, 2010].

According to Information Theory, to choose the sample size, which is just a scale value, it is suggested to use the dynamic range (3.60):

$$3\sigma \sim 0.1 \tag{3.60}$$

Or use the rule of Silverman [Silverman, 1986] (3.61):

$$\sigma = 0.9AN^{-1/5} \tag{3.61}$$

3.7.6.2. Implementation

This proposed approach makes a prediction of the error value, taking into account the trends that this error has over time, depending on circumstances and context, in order to be able to correct the value of another prediction made earlier, and try to obtain a final value closer to the real one.

For this problem, a dynamic neural network (NN) is used to forecast the errors. The NN is re-trained in each iteration, so that the forecast values are always updated according to the most recent observations. The NN receives as input the prediction error history data of the market’s prices, and is trained to generate an output value, which will be the expected error. Then, this error is used to adjust the value of a prediction made by other forecasting strategy. Errors are stored in a market’s history, registered in the database presented in section 3.2.

When defining the architecture of the NN, a matter of high importance had to be analyzed – the way the errors’ sequence is “looked at” when trying to forecast it. This is a very important issue, for it does not matter how much data one has access to if that data is not properly used. For this reason, three different approaches were considered.

These three approaches use a NN with one value in the output layer - the value of the expected error, two intermediate nodes, and an input layer of four units. The input layer considers different values in the different approaches. These values depend on how the history of the error is considered:

- **Error Theory A** - This strategy makes a prediction along the 24 periods per day, using for the training of each period the error of the same period for:
    - the previous day;
    - the previous 7 days;
    - the previous 14 days;
    - the previous 21 days.

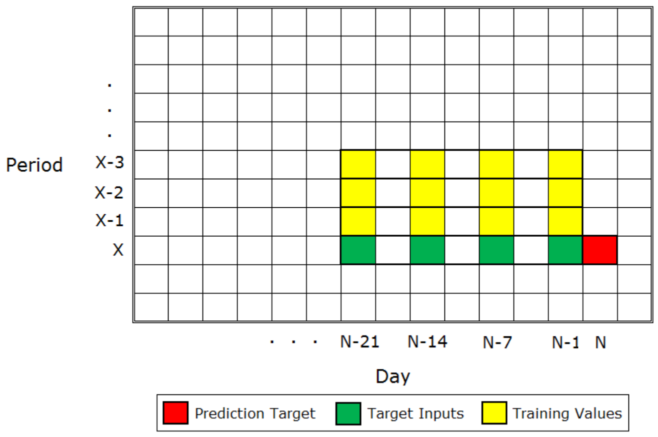


Figure 3.56 – Error Theory A training data structure.

Having Figure 3.56 as support, if an error prediction is required for day N and period X, the input values of the NN are N-1, N-7, N-14 and N-21, all for period X. The data of the previous periods is used to train the NN, considering the same days.

- **Error Theory B** - This strategy makes a prediction along the days, using the error of the following periods:
    - prior period;
    - 2 previous periods;
    - 3 previous periods;
    - 4 previous periods.

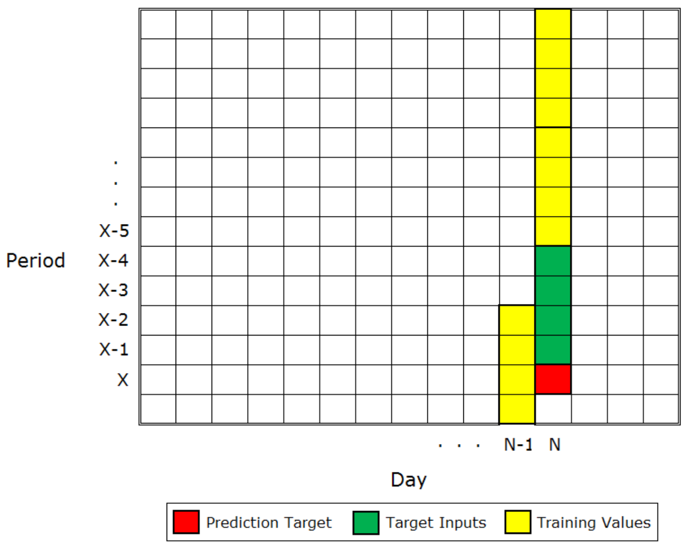


Figure 3.57 – Error Theory B training data structure.

If an error prediction is required for day N and period X, the input values of the NN are the previous four periods, considering the same day. To train the NN the previous periods are considered. When all the periods of the current day have contributed to the training, the periods of the prior day start being considered.

- Error Theory C - this strategy makes a prediction considering always the errors of the same period (the time period in question), using the error for:
  - the previous day;
  - the previous 7 days;
  - the previous 14 days;
  - the previous 21 days.

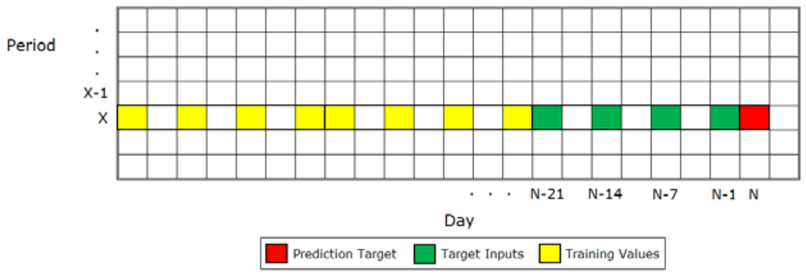


Figure 3.58 – Error Theory B training data structure.

This approach always considers the same period. It ignores the data concerning all periods other than the required one, and uses the previous days to train the NN.

3.7.6.3. Experimental Findings

This sub-section presents three tests, one for each strategy: Error Theory A, Error Theory B, and Error Theory C. All tests concern the same start day and time period, so one can also take a conclusion about the most appropriate strategy for the period under review.

The dynamic feed-forward neural network trained with the historic market prices, presented in sub-section 3.7.4, is used as a reference value to forecast market prices, which afterwards will be adjusted by using the proposed approaches,

The first test concerns Error Theory A strategy, comparing its prediction with the value received by the auxiliary strategy, to which the expected error is applied. In this comparison there is presented the actual market price, so one can check the quality of the prediction strategy. Figure 3.59 presents a chart with the results of this test.

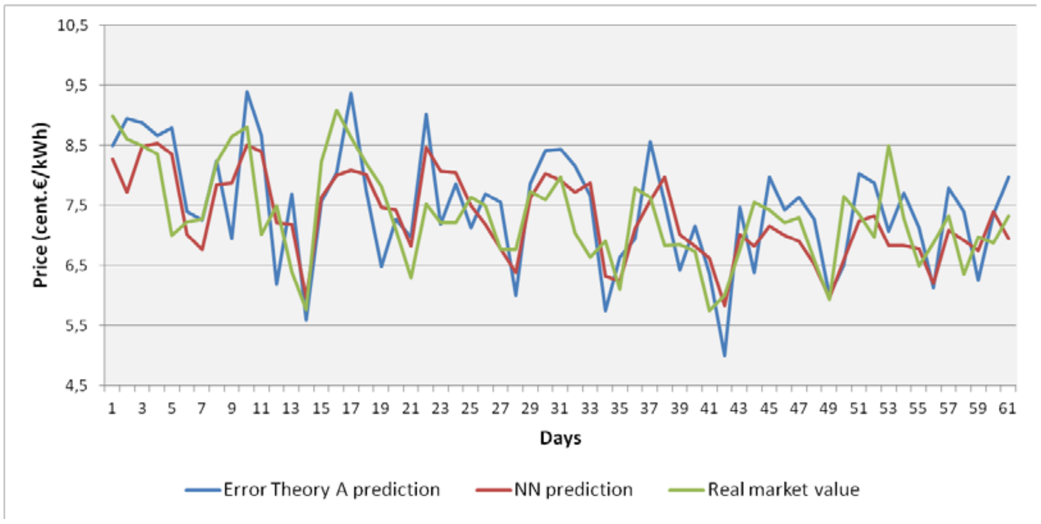


Figure 3.59 – Results of the test on Error Theory A strategy.

As can be seen from the chart presented in Figure 3.59, the calculation and application of the expected error, using the Error Theory A strategy, is very close to the actual market value, such as the auxiliary strategy. The comparison of the MAPE for both predictions is presented in the final of this sub-section.

In the second test, the same process was taken for Error Theory B. Figure 3.60 presents a chart with the test results.

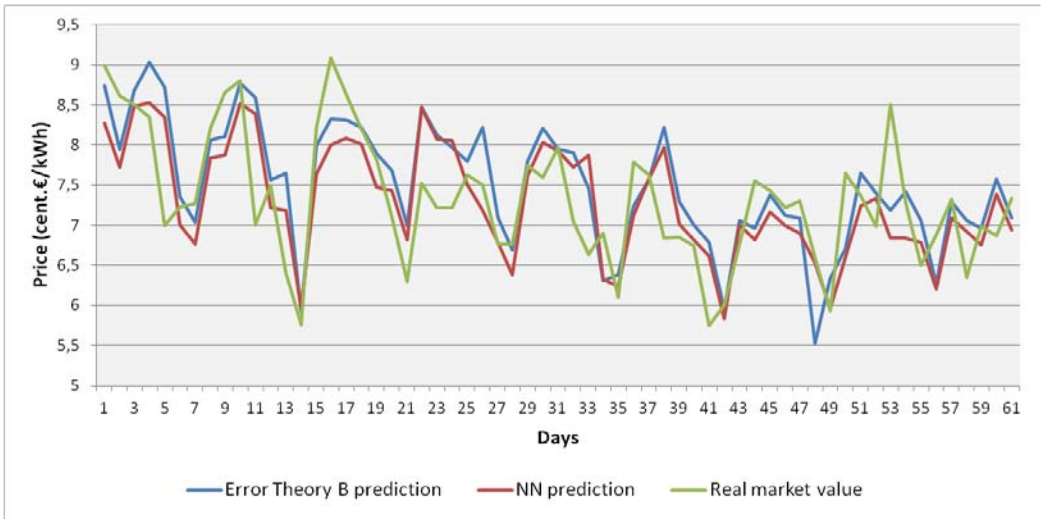


Figure 3.60 – Results of the test on Error Theory B strategy.

In this second test, one can see that the Error Theory B strategy had worst results when compared with Error Theory A strategy, for the tested starting day and period.

Finally, the third test is similar to the previous ones, but it is applied to Error Theory C strategy. These test results are presented in Figure 3.61.

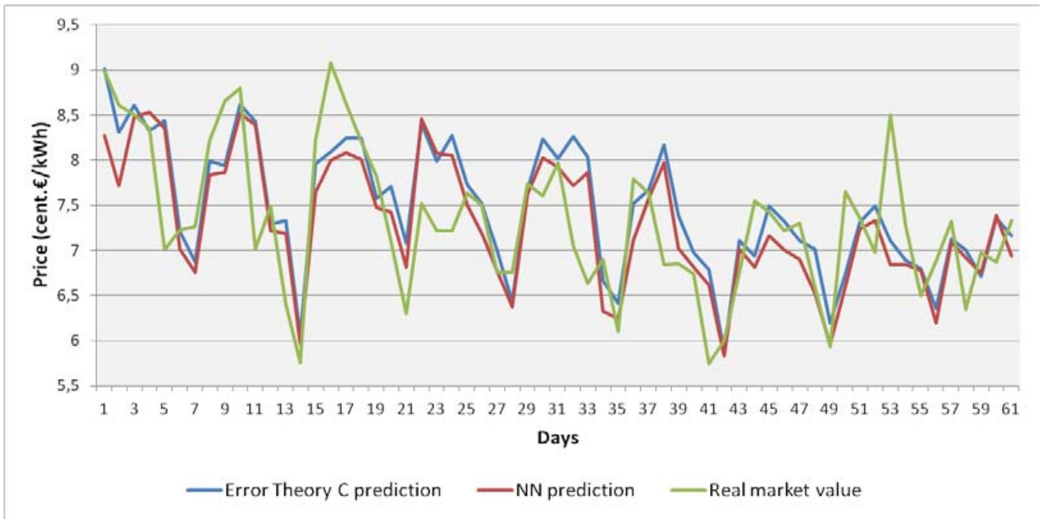


Figure 3.61 – Results of the test on Error Theory C strategy.

Analyzing the results of the third test, it seems that the Error Theory C strategy, concerning the tested start day and time, has a better performance than the Error Theory strategy B, regarding the level of proximity of the forecast with the actual values of the market, but worst than Error Theory strategy A. These results are supported by Table 3.5, which presents the comparison of the MAPE values of the presented forecasts.

Table 3.5 – MAPE comparison of each proposed strategy with the auxiliary strategy.

	Error Theory A	*Error Theory B*	*Error Theory C*
Forecasting strategy	0,064907955	0,06651243	0,065098301
Auxiliary forecast strategy (NN)	0,066845633		

In spite of the three strategies having got better results than the auxiliary strategy, Error Theory A proves to be the best of the three strategies, for the day and period used in the tests, as can be seen by the MAPE values, which relate the actual market value to the predictions of the auxiliary strategy, and the presented approaches.

The observed results are motivating, since all the approaches were able to adjust the NN forecasted values, and generate predictions closer to the actual goal. This was achieved by considering the prediction of the forecasting error that was expected.

3.7.7. *Economic Analysis Agent*

The Economic Analysis Agent implements a strategy based on the two most commonly used approaches of forecasting in a company’s scope. These approaches are the internal data analysis of the company, and the external, or sectorial, data analysis.

3.7.7.1. Context

The most important and widely used strategic planning methods for a companies’ business, base their analysis in the identification and grouping of the key pieces of information into two main categories: Internal and External data.

Perhaps the most famous strategic planning method, the *SWOT analysis* [Hill and Westbrook, 1997], is based on the identification of the key internal and external factors that are important to a companies’ achievement of its objective. The internal factors considered by this method are the strengths and weaknesses internal to the organization. The external factors are the opportunities and threats presented by the external environment to the organization.

The internal factors may be viewed as strengths or weaknesses depending upon their impact on the organization's objectives. What may represent strengths with respect to one objective may be weaknesses for another objective. SWOT analysis is just one method of categorization and has its own weaknesses. For example, it may tend to persuade companies to compile lists rather than think about what is actually important in achieving objectives. It also presents the resulting lists uncritically and without clear prioritization so that, for example, weak opportunities may appear to balance strong threats [Menon *et al.*, 1999].

In order to overcome these weaknesses, other methods for analyzing and developing companies businesses’ strategies arose. Another widely used method is the *Porter's Five Forces* [Porter, 1979]. This strategy was developed by Michael E. Porter of Harvard Business School in 1979. It presents five forces that determine the competitive intensity and therefore attractiveness of a market. Attractiveness in this context refers to the overall industry profitability. An unattractive industry is one in which the combination of these five forces acts to drive down overall profitability.

Three of Porter's five forces refer to competition from external sources: threat of substitute products, the threat of established rivals, and the threat of new entrants [Porter, 2008]. The other two are internal threats: the bargaining power of suppliers and the bargaining power of customers.

One important issue concerning the external analysis of a company is the sector analysis. Sector analysis is based on a review and assessment of the current condition and future prospects of a given sector of the economy [Brealey and Myers, 2003]. Sector analysis serves to provide an investor with an idea of how well a given group of companies are expected to perform as a whole [Church and Ware, 2000]. Using the sector analysis, companies can adequately select companies to negotiate with, which may ensure significant profit or negotiation advantage [Wooldridge, 2002].

3.7.7.2. Implementation

The Economic Analysis strategy's main principle is to decide when are the most appropriate moments to opt by a riskier or a safer approach in negotiating in the market. This decision is based on the connection between the internal and external data analysis of a company.

The internal analysis of the company can be viewed as the company's economic development, *i.e.*, the increasing or decreasing of the achieved profits. The profits take into consideration the company's fixed costs *FC*, such as the personnel expenses, the infrastructures' costs, the overheads, continuous maintenance, etc. Additionally, it also considers the variable costs, which are dependent on the production *P*, and are usually represented as two factors: *a* and *b*. the profits can be defined as in (3.62).

$$Profits = Total\ Income - (FC + P.a + P.b^2) \tag{3.62}$$

The analysis on the profits evolution is performed through a comparison between the most recent achieved profits, with the immediately previous ones. If the evolution is crescent, *i.e.*, the recent profits are increasing, it is considered that the company is consolidating its position on the market, and therefore it is in a position where it can afford to take risks, in order to try obtaining higher profits.

On the other hand, if the recent profits tendency is decreasing, the company must play safe, acting towards equilibrium, in a way to assure the continuous achievement of incomes, even if they are not as high as they could be.

When the decision goes for risking, and trying to achieve the higher possible profits, the *Economic Analysis* strategy uses as reference for the bid price, the forecasted market price, as it is the threshold of where a bid should be located, in order to obtain profits. The market price forecast used by this strategy is provided by the Dynamic Artificial Neural Network strategy, presented in sub-section 3.7.2.

When the decision is acting towards equilibrium, and consequently playing safe, safeguarding the achievement of profits, even if they are not optimal, the reference value used for bidding is based on the external or sectorial analysis. The sectorial analysis intends to: firstly, understand how the sector (in this case the electricity sector) is moving inside the global market; secondly, realize in which sub-sector (most influent companies, less powerful ones, etc.) the company in matter is located; and thirdly, analyze how each of these sub-sectors is developing, *e.g.*, going towards the unification of several sub-sectors, or otherwise, distance themselves. Figure 3.62 presents a global view over the external or sectorial analysis.

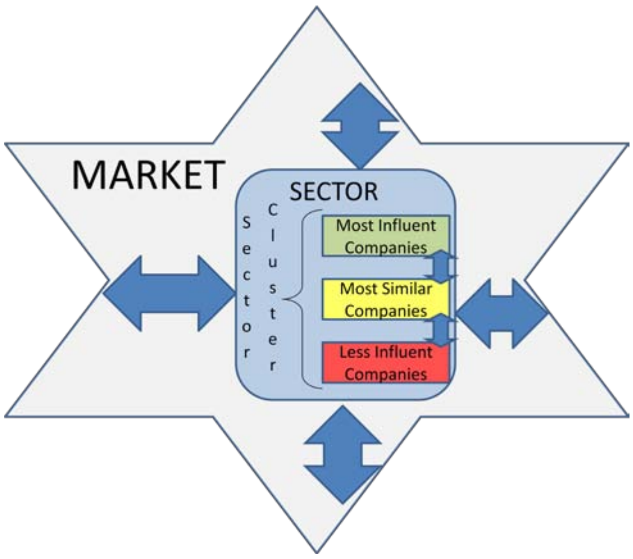


Figure 3.62 – External/Sectorial Analysis.

In order to define the sub-sectors inside the electricity market sector, a clustering mechanism is used to group the companies that act in the electricity market, in different groups, according to their characteristics. The clustering is performed using the K-Means algorithm provided by MATLAB. The companies are grouped according to their similarity in what concerns their dimension (amount of produced energy), the prices they are practicing recently (the most recent bid prices), and the average price they presented for the last month and year.

The clustering mechanism groups the companies into three different clusters; one representing the most influential companies, one representing the most similar companies to the one ALBidS is supporting, and one representing the less influent companies over the market.

Once the clustering is finished, the average bid price of the companies grouped in the same cluster as the supported one, is determined as the sector reference price, as it is used as reference for the situations in which the decision was to act towards equilibrium. The only situation in which this may not apply is when the supported company is placed in a highly competitive cluster, with high risking prices. In this case it is used the lower from the cluster reference or market price forecast as reference value.

Concerning both reference values (sector reference price in case of equilibrium, or market price in case of risking), they are subject to a decrement before being defined as the bid price for the supported player. This decrement is based on a *risk factor*, chosen by the user, with the purpose of guaranteeing that the final bid price is located below the reference, regardless of which.

The *risk factor* is a value between 0 and 1, and the higher the value is, the smaller is the decrement, meaning a higher proximity to the used reference values, and therefore increasing the risk to which it is subject. The initial decrement is calculated as in (3.63).

$$Decrement = 1 - Risk\ factor \tag{3.63}$$

If the decision in a certain moment is acting towards equilibrium, the decrement stays fixed at this value. Otherwise, if decision is risking for a continuous number of periods, the decrement lowers according to (3.64), slowly increasing the risk, until the bid price is equal to the reference value. If the sequence of risking periods is interrupted, the decrement returns to its initial value.

$$\textit{New Decrement} = \textit{Decrement x } (1 - \textit{Risk factor}) \tag{3.64}$$

An adequate balance between the decision of taking higher risks, and acting safe, towards equilibrium, is the main goal of the Economic Analysis strategy. The decision making in what concerns the adequate times to risk is the essential concern of this strategy. For that, the internal vs. external data analysis gives its contribution.

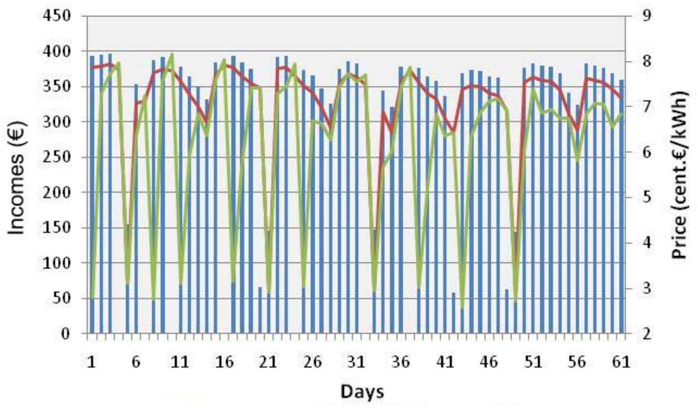
3.7.7.3. Experimental Findings

This sub-section presents four tests using the test scenario, for 61 consecutive days, starting from the 15th October. In the first three tests Seller 2 will use the proposed approach with different *risk factor* values. The first test concerns a *risk* value of 0.2, a small value, meaning a low risk. In the second, an intermediate value of 0.5, and in the third a higher value, 0.8. The fourth and final test will present Seller 2's results using the proposed approach exclusively directed to equilibrium.

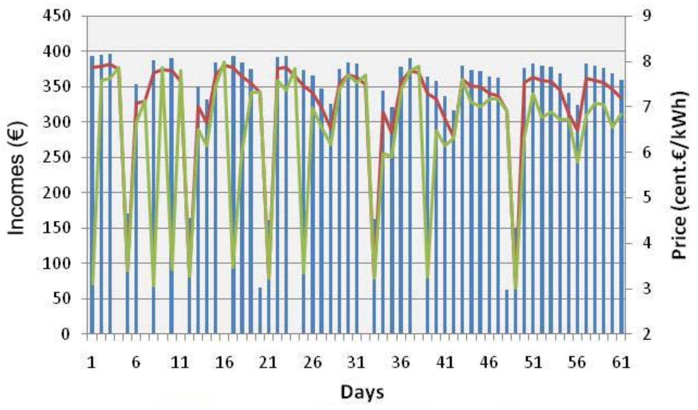
These tests intend to show a comparison of this strategy's performance using different *risk factors*, and also to demonstrate the advantage of both decisions: risking or playing safe, towards equilibrium.

Figure 3.63 presents the incomes obtained for this agent in the simulations, and a comparison between the market price and Seller 2's bid price, for the twelfth period of each day, for the test scenario.

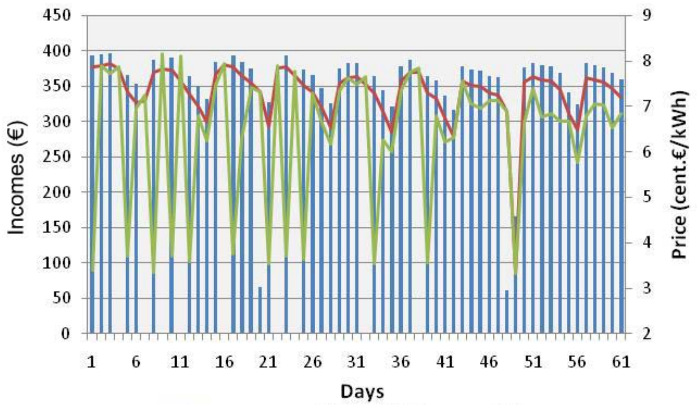
a)



b)



c)



d)

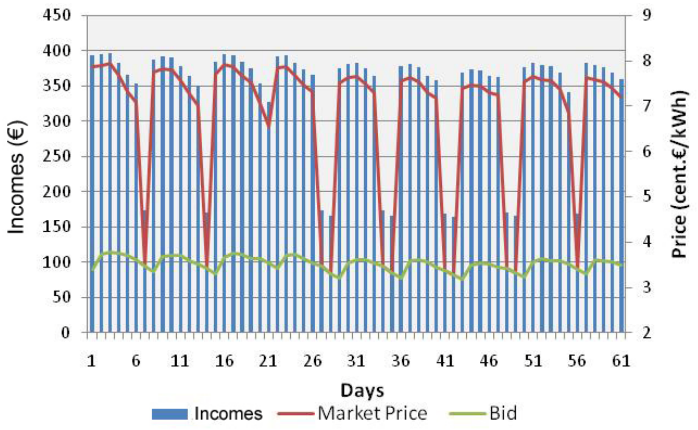


Figure 3.63 – Incomes obtained by Seller 2 in the twelfth period of the considered 61 days, with: a) a risk of 0.2, b) a risk of 0.5, c) a risk of 0.8, d) equilibrium.

The graphs in Figure 3.63 show that Seller 2 was able to obtain incomes in almost all days regarding the first three tests, while in the fourth it could effectively sell at all times. Additionally, in the first three cases, it is visible that following a day of bad results – not selling, or selling a low amount of power – the agent's bid price decreases, lowering the risk, and acting towards equilibrium. When the incomes are higher, the bid price for the following day is much closer to the market price, meaning a higher risk and the possibility to achieve higher profits.

The selling of power in all days in the fourth test is an evident proof of the advantage of using the equilibrium approach, as supported by the incomes in the total of the 24 periods, for the 61 considered days, obtained in each of the four tests, presented in Table 3.6.

Table 3.6 – Total incomes obtained in the twelfth period.

	Risk factor			Equilibrium
	0.2	0.5	0.8	
Total Income (€)	435482.16	429537.36	438655.92	494574

Table 3.6 shows that the equilibrium approach allowed the player to achieve higher global incomes, even though the tendency when increasing the risk not proving to be increasing. Regarding the three risking approaches, the third, which presents the higher risk, is the one that proved to be the most advantageous. However, although the equilibrium approach presented higher global incomes for this case study, it does not mean that this is the best decision to take in all situations. In fact, when risking in the right moments, a strategy can actually get even higher incomes. In Figure 3.63 it was visible that the first three approaches presented some periods in which they could not sell, and consequently not obtain incomes, but when considering the selling periods, the incomes obtained by increasing the risk are higher than the equilibrium approach. This is supported by Table 3.7, which presents the comparison of the total incomes obtained in the third and fourth tests, starting from day 37 - the following day to the last one in which Seller 2 could not sell using a risk factor of 0,8.

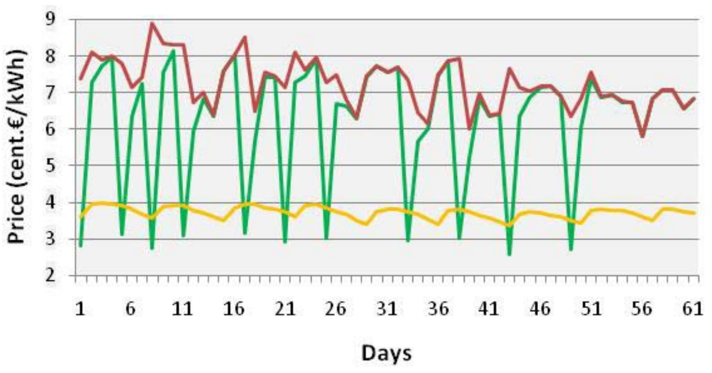
Table 3.7 – Total incomes obtained in the twelfth period starting from day 37.

	Risk factor 0,8	Equilibrium
Total Income (€)	7871.75	7497.12

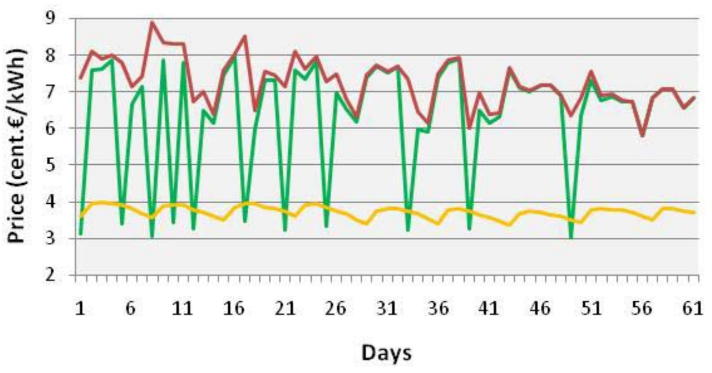
It is visible in Table 3.7 that the approach using a risk of 0.8 obtained higher incomes than the equilibrium approach when considering the periods in which both approaches were able to sell. This means that risking is advantageous for achieving higher incomes, when this is done in the adequate moments. The simple settling for equilibrium does not provide the best solution when the goal is achieving the highest possible incomes, rather a conjugation of both, done in the right times, and in the adequate amounts. This is an important conclusion to support the advantage of risking.

The way bids are defined in the first three tests, whether risking or going for equilibrium are dependent on the players' recent results. Figure 3.64 presents the bid definition in the four tests, in comparison with the two reference values: market price forecast and cluster reference price, which are the basis for the equilibrium and the risk actions.

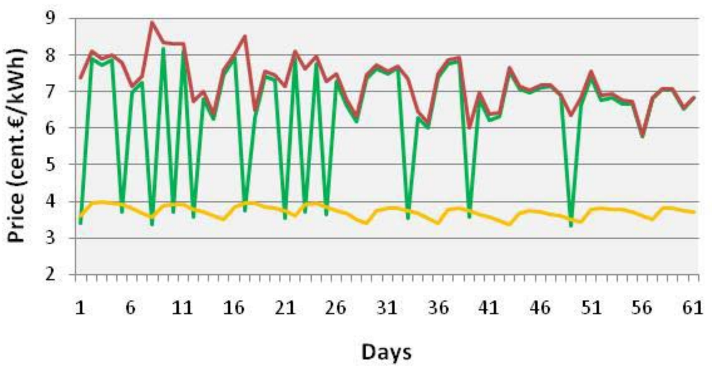
a)



b)



c)



d)

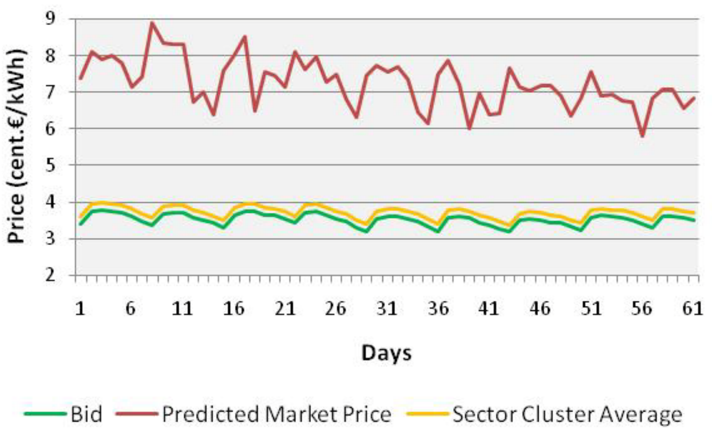


Figure 3.64 – Seller 2's bid comparison with the reference values in the twelfth period of the considered 61 days, with: a) a risk of 0.2, b) a risk of 0.5, c) a risk of 0.8, d) equilibrium.

The graphs in Figure 3.64 show that Seller 2's bid is always located below a reference value, either the forecasted market price, when assuming a higher risk, or the cluster reference value when going for equilibrium. In the fourth test, as the player uses the equilibrium approach at all times, the bid is always closely below the cluster reference value. Regarding the first three tests, it depends on the recent results. When comparing Figure 3.64 with Figure 3.63 it is visible that the cluster reference value is used after the periods in which the player got worst results. Comparing the bids among these first three tests, it is visible that the higher the risk of the approach, the closer the bid value gets to the reference values. Figure 3.65 presents the differences of the bids in each case, from the used reference value.

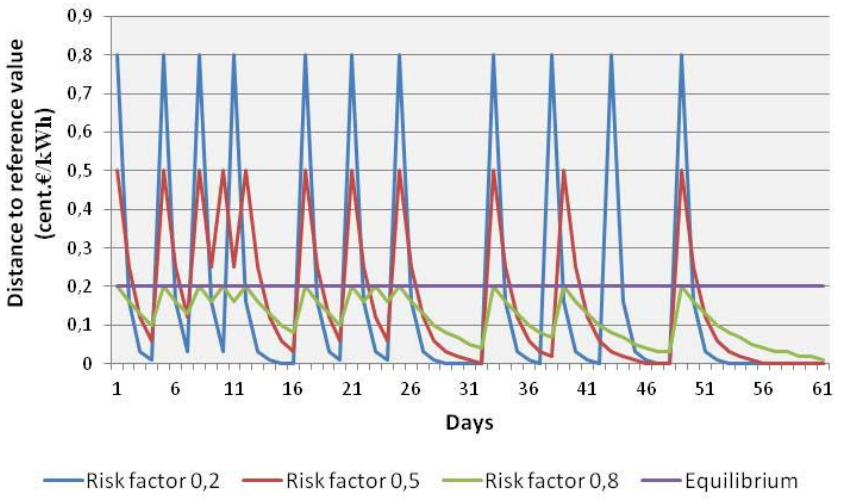


Figure 3.65 – Difference between the reference value and Seller 2's bid in the three considered cases.

In the first case, using a risk factor of 0.2, the bid value is located much lower than the references, as shown in Figure 3.65, presenting an average difference of 0,18 cent.€/kWh. In the second case, the differences are intermediate, with an average of 0.16 cent.€/kWh. Regarding the third case, where the risk factor is defined as 0.8, the bid comes very close the reference values, being the difference average of 0.11 cent.€/kWh. Finally, in what concerns the equilibrium approach, the difference is constant, being maintained at 0.2 cent.€/kWh in all situations.

The tests presented in this sub-section allowed demonstrating the adequacy and advantage of using the proposed approach, as it proved being able to provide high incomes to the supported player. Additionally, the combination of risk with equilibrium depending on the internal and external analysis of a company has been demonstrated.

3.7.8. *Determinism Theory Agent*

This agent executes a strategy based on the principles of the Determinism Theory. A theory that states that due to the laws of cause and effect, which apply to the material universe, all future events are predetermined.

3.7.8.1. Context

Determinism is the theory that all events, all happenings, are no more than consequences of a certain set of causes [Winnie, 1996], and therefore all events are determined, because the event that was caused by other, is itself one of the causes for the determination of various other happenings [Einstein, 1905]. This theory is based on the concept of *cause and effect* [Bunge, 1959], which states that things do not happen by chance. All things result from the causes that impelled that happening to occur.

According to this theory, although all events are determined, and there is no chance of things happening any other way, it is impossible to predict a complex event based on this paradigm, because that event is caused by infinite other events, and, there does not exist the capability of considering, analyzing, and combining infinite variables, to perform that prediction [Green¸ 2003].

Determinism has, since many years ago, intrigued many thinkers and scientists, and therefore triggered various different views and approaches based on this issue. With origin in the Atomism of Democritus and Lucretius, *physical determinism* is the theory that human interaction can be reduced to relationships between biological, chemical, or physical entities; this formulation is fundamental to modern Socio-biology and neuropsychology [Berofsky, 1971]. The *historical determinism* of Karl Marx, on the other hand, is transpersonal and primarily economic.

B. F. Skinner's [Skinner, 1971] *behavioural determinism* is a modification of this view, in that Skinner reduces all internal psychological states to publicly observable behaviour. His stimulus - response account - also uses modern statistical and probabilistic analyses of causation.

Jean Paul Sartre and other contemporary philosophers have argued that determinism is controvert by introspection, which reveals actions to be the result of our own choices and not necessitated by previous events or external factors. Determinists respond that such experiences of freedom are illusions and that introspection is an unreliable and unscientific method for understanding human behaviour. This view has been modified within the scientific community, however, with the enunciation of the *Uncertainty Principle* by the physicist Werner Heisenberg [Heisenberg, 1962]. Ramifications of his work in quantum mechanics led Heisenberg to assert that the scientist, as much a participant as an observer, interferes with the neutrality and very nature of the object in question. His work also questions whether it is possible to determine an objective framework through which one can distinguish cause from effect, and whether one can know an objective effect if one is always a part of its cause.

According to Donald Davidson [Davidson, 1980] "*a mental event and a simultaneous neural event constitute what can be called a psychoneural pair. Such a pair is a single effect of antecedents and a cause of subsequent* *events*". This account of the psychoneural relation gives place to the conviction that both mental and neural events have causal roles with respect to our actions and also subsequent mental events [Mellor, 1986].

The main conclusion to take from the analysis of the *Determinism Theory* is that it is considered impossible to predict a future event, or effect, through the analysis of its causes, although such event is predetermined, because of the impossibility of considering all the causes, or variables, which affect the event, for they are infinite.

However, in a controlled environment, such as in simulation, which represents the global reality in a condensed and simplified environment, the consideration of all variables becomes much closer to being possible. Although a simulation environment and its results cannot fully reflect the reality; as they represent the essential aspects of a certain part of the reality, they can be (and are widely) used to provide important pointers and possibilities about how this reality shall react to the events tested in the simulation environment. Therefore, the application of such theory can prove both, to be possible in this environment, and also to produce results which can reflect the reality.

3.7.8.2. Implementation

Physics theories' application to computer systems, in the intelligent systems' area, has been performed before [Liu *et al.*, 2005], [Wand *et al.*, 2008] and has proven to be effective. For this reason, and due to the characteristics of the main principles of the Determinism Theory, it was found interesting to transpose such principles to a decision support tool, in this case regarding a strategy to support the decisions of a software agent.

In order to implement the Determinism Theory's principles in this environment (simulation, and electricity markets), it was found essential to, in a first instance, find the world variables that are required to be predicted. In this case they are: the market price for a certain period of a certain day, and the total amount of power traded by each market agent. These are the main variables, because they are the ones that will determine the amount of incomes each agent will achieve, and the main goal of the agent ALBidS is supporting is the achievement of the higher possible profits.

Secondly, it is necessary to realize which variables influence those goal variables, *i.e.* which are the causes to the effect that is going to be predicted. In this case, the variables that influence the state of the world are the behaviours of the market negotiating agents, namely their proposed bid to the market. For these are the factors which will determine the market price and amounts of traded power, through the application of the spot market symmetric algorithm.

Realizing that the player which ALBidS is supporting has equivalent influence to the future state of the world as the other competing players do, the idea is to, by predicting the others' actions; optimize the supported player's action, in order to accomplish the best possible results. Once the competitor players' actions are predicted, they are considered as static for the period in matter, therefore, the only variation that can, from this point, affect the future state of the world, is the action of the ALBidS supported player, for this is the only action that is not predicted, and that will vary, according to the results of the optimization. Figure 3.66 presents a global overview over the Determinism Theory strategy approach.

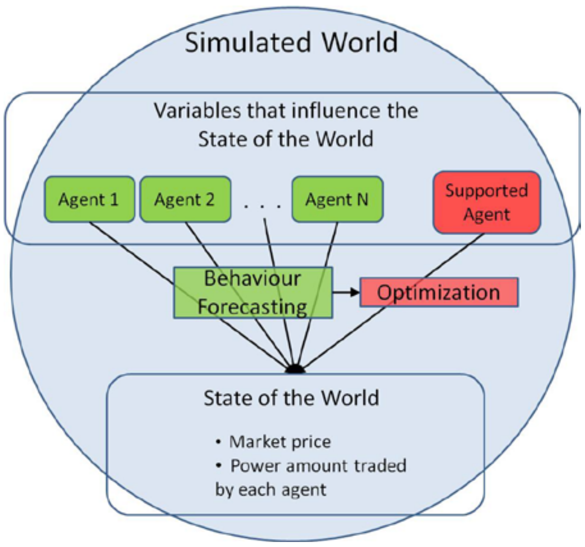


Figure 3.66 –Determinism theory approach.

The first step taken when implementing this approach was the forecasting of the market negotiating players' next action. This is done taking advantage on the Player Profile Definition mechanism, presented in section 3.8, which, according to each player's past actions and reactions, depending on the context, provides a prediction of the next action each player will perform.

Once these predictions are performed, they are used to optimize the supported player's bid price. The optimization considers as input variables the predicted bid prices and powers of the market players, and the bid power of the supported player (as this amount of power is fix for each period). The decision variable is the supported agent's bid price, which is optimized according to the objective function: the market clearing algorithm, of the symmetric pool (presented in sub-section 2.3.1.1). This algorithm is what defines the market clearing price and the power amount traded by each agent.

The optimization's goal is to find the bid price that provides the maximum profit for the supported agent. The way the optimization is performed depends on the efficiency/effectiveness preference (see section 3.9 - Efficiency/Effectiveness Management). So, for a high preference on the effectiveness of the method, the optimization is performed using an explicit enumeration of all the admissible solutions, *i.e.*, the experimentation of all the possible bid prices for this agent.

Explicit Enumeration

As prices are real values, the explicit enumeration considers a minimum and maximum admissible price (the minimum price being 0 cent€/kWh, for it does not make sense to bid negative values; and the maximum price being dependent on whether the supported player is a buyer or a seller).

Between these minimum and maximum values, all possible bid prices are considered, depending on a desired increment. This increment determines the interval between each considered bid. The smaller the interval is, the higher the number of considered bids is, as their values are closer to each other.

This is a simple procedure, but it guarantees the experimentation of all options, therefore guaranteeing to find the optimal solution to the problem. However, the required execution time for processing it is high. This is no problem when there is a high preference on the effectiveness of the method, but it is an essential issue when dealing with high exigency in what considers the reduction of the execution time.

For these cases, heuristics are used. Following the same principle as in the designing of the global ALBidS system, there is no heuristic that proves to be the best for all optimization situations, so it was decided to implement several ones, and let their performances decide which shall be used for each case, both in terms of their results performance, and in what regards the execution time they require. The implemented heuristics are: the *Simulated Annealing*, the *Tabu Search*, and the *Particle Swarm Optimization*.

Simulated Annealing

The *Simulated Annealing* (SA) heuristic's application to this case follows the same principles as in the previously presented strategies that use this heuristic, namely the AMES (sub-section 3.7.3) and the Simulated Annealing Q-Learning (sub-section 3.7.4) strategies. For this reason, this sub-section will not present further details on this heuristic's implementation.

Tabu Search

The *Tabu Search* (TS), proposed by Glover [Glover, 1989] consists in an adaptive procedure that guides a local search algorithm in the continuous search of the search space. The TS starts from an initial position, and moves to the best neighbourhood solution in each iteration, while not accepting movements to previously visited positions, stored in a *tabu list* with variable size. This list remains in memory for a certain number of iterations.

The TS uses flexible memory structures to store knowledge. Thanks to this adaptive memory use, the TS is capable of achieving good results. Additionally, the final solution presents a low dependency from the initial solution, since the method's implemented mechanisms escape from local optimal positions.

Particle Swarm Optimization

The *Particle Swarm Optimization* (PSO) is an evolutionary technique developed by James Kennedy and Russel Eberhart [Kennedy and Eberhat, 1948]. The initial idea was to demonstrate the behaviour that a fleet of birds or a shoal of fishes assume in their apparent individually random, but globally determined movements.

In a computational perspective, the PSO algorithms show an abstraction from that biological behaviour, and search for a better position in what concerns the search for the optimal solution. The search space is the space of possible solutions.

Each particle's behaviour is based on its previous experience as well as in the experience of those it relates to. The PSO considers a number of different particles, which optimize their moves based on two parameters: the social factor (C1), which determines the attraction, or convergence of the particles to the best global solution found so far; and the cognitive factor (C2), which determines the attraction of the particle to the best position found by itself.

Besides these factors, the minimum and maximum velocity at which the particles are allowed to move must also be specified. The velocity *V* of particle *i* change is essential to the position change of a particle and it is determined by (3.65).

$$Vid = W * Vid + C1 * R1 * (Pid - Xid) + C2 * R2 * (Gd - Xid) \tag{3.65}$$

where *W* is the weight of a particle’s inertia, R1 and R2 are random values between 0 and 1, *P* is the best individual position of a particle, *G* is the best global position, and *X* is the particle’s current position.

The heuristic algorithms were initially implemented in JAVA, as this is the main programming language of ALBidS and MASCEM. However, the execution time they were presenting in the tests was much higher than desired. Therefore, an alternative version of each one has been developed in LPA-Prolog in order to reduce the execution times.

Case-based Reasoning

In order to choose the heuristic approach that is used in each case, a case-based reasoning [Madureira and Pereira, 2010] approach has been developed. The case base contains the cases previously observed. Each case contains the information presented in Table 3.8.

Table 3.8 – Case base structure.

ID	Method	Parameters	Execution Time	Number of Players	Own Power	Profits

The *ID* refers to the identification of the case. The *Method* is the name of the heuristic, and *Parameters* is the definition of the method’s parameterization. *Execution Time* is the time that the method took to execute the optimization for the case, an important information in what regards the choice of the method under a situation with low preference for effectiveness. *Number of Players* indicates number of players negotiating in the market in this case. *Own Power* regards the amount of power negotiated by the agent ALBidS is providing support to. Finally, *Profits* concerns the amount of profits that this method and its parameterization were able to achieve in the actual market negotiation.

The *Own Power* and *Number of Players* are used to calculate the similitude between cases. When a new iteration starts, this information will determine which cases are considered similar to the current one, therefore being considered for the results analysis, in order to choose the most appropriate one. Cases are considered similar when their similitude is above 75%, considering both the amount of power, and the number of negotiating players.

The choice of the methods is dependent not only on the similitude but also on the current circumstances for which the method is required, *i.e.*, the preference of efficiency/effectiveness. This choice is done the following way:

- For an effectiveness preference over 90%, the execution time of the method is ignored, therefore the chosen method is always the *Explicit Enumeration*, as it guarantees the finding of the optimal solution;

- For an effectiveness preference between 65% and 90%, the method and respective parameterization is chosen randomly between two options considering only the similar cases: the method and parameterization that presented the best profits; or a random choice between all the similar cases;
- For an effectiveness preference between 35% and 65%, the method and respective parameterization is chosen randomly, in order to increase the variety of cases in the case base;
- For an effectiveness preference between 10% and 35%, the method and respective parameterization is chosen randomly between two options considering only the similar cases: the method and parameterization that presented the smaller execution time; or a random choice between all the similar cases;
- For an effectiveness preference between 0% and 10% the chosen method is the one that presents the smaller execution time from all the similar ones.

After the optimization is performed, the bid price chosen, and the market negotiations finished, the new case, concerning the current characteristics and results, is saved in the case base, in order to enrich it, allowing future cases to consider this new case in its choosing of the most appropriate method of optimization.

Once the optimization is finished, the process of prediction using the principles of the Determinism Theory is completed. This process considered the prediction of the causes (players' actions) that influence the effect (market results), while adapting (optimizing) the action of the supported player, so that it could take the most advantage out of its environment.

3.7.8.3. Experimental Findings

This sub-section presents the experimental findings that support the adequacy of the use of this strategy to support a market player's decisions.

Firstly, three simulations are presented. These simulations differ in the efficiency/effectiveness preference (see section 3.9 - Efficiency/Effectiveness Management) for which the method is performed. These tests have two main goals: showing the influence of the players' profiles definition (see section 3.8 – Players Profiles Definition) over this method's performance; and demonstrating a market player's performance when negotiating in the market, supported by the proposed strategy.

After this demonstration, the optimization part of the method is analyzed, through a comparison of the optimization algorithms' performance, concerning both their effectiveness and effectiveness.

Figure 3.67 presents the incomes obtained by Seller 2 in the three simulations, using the test scenario, for 61 consecutive days, starting from the 15th October. In the first simulation Seller 2 is supported by the proposed method with a 100% preference for the effectiveness of the method, meaning a faster but less adequate response. In the second simulation, a 50%-50% preference for efficiency/effectiveness is used. Finally, in the third simulation, the preference goes 100% for the efficiency. Also, for all tests a comparison between the market price and Seller 2's bid price, in the twelfth period of each day is presented.

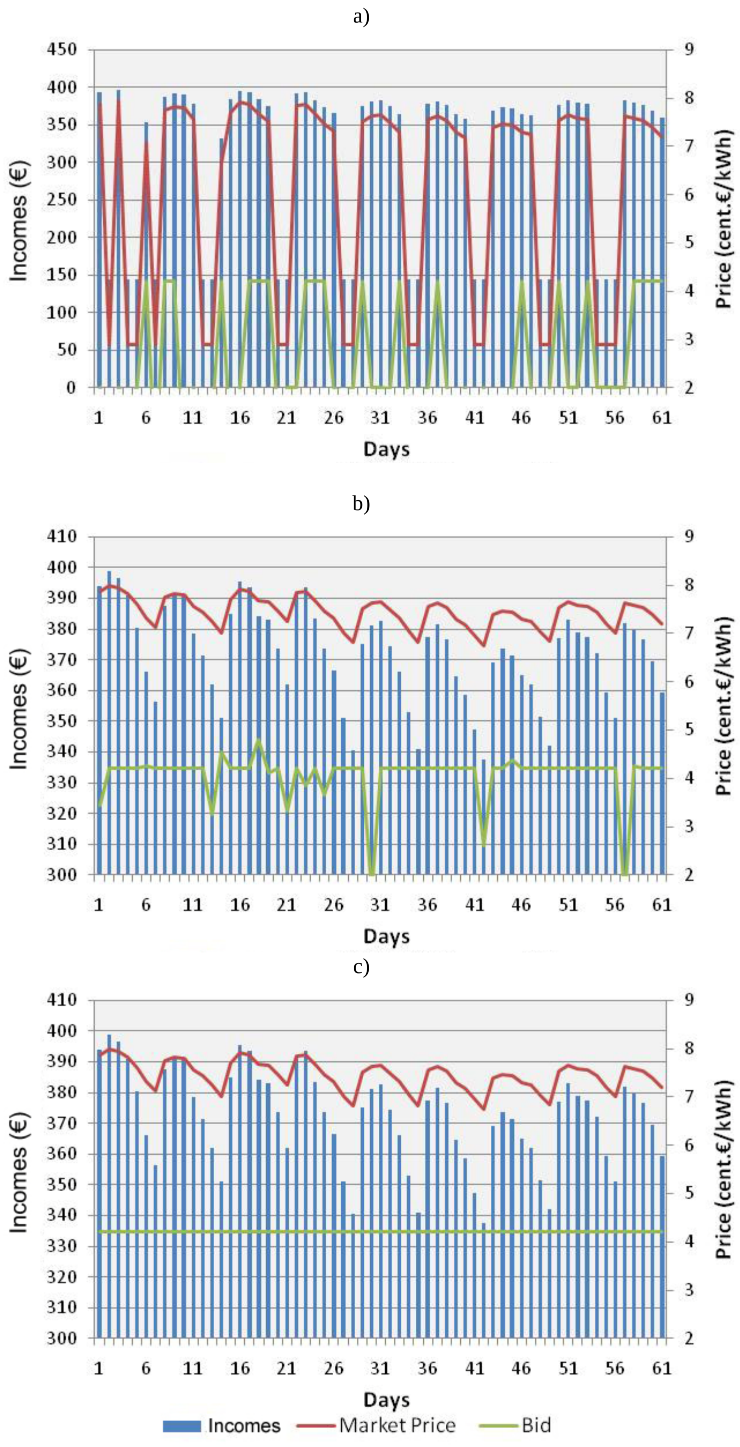


Figure 3.67 – Incomes obtained by Seller 2 in the twelfth period of the considered 61 days, with an effectiveness preference percentage of: a) 0%, b) 50%, c) 100%.

The graphs in Figure 3.67 show that Seller 2 was able to obtain incomes in all days in the three simulations. This proves the methods' high adequacy for supporting the player's actions when negotiating in the market, independently of the processing times that the method is obliged to respect.

Additionally, when comparing the income amounts along the days in the three cases, it is visible that the second and third simulations are very similar, meaning that the reduction of the players' profiles definition quality is not critical for the method's performance for a 100% to 50% preference on effectiveness.

In what concerns the proposed bid in these two cases, the differences are due to the choice of the optimization algorithm. In the third case, with the maximum importance being given to the effectiveness of the method, the chosen algorithm is always the *explicit enumeration* algorithm. The constant bided price reflects the optimization algorithm's process of choice when finding intervals in which the profit is maximum, instead of only one value. In these cases the algorithm chooses the smaller value from that interval.

In what concerns the bids for the second case, their variance is higher because of the randomness in the choice of the optimization algorithms for a mid point in efficiency/effectiveness preference.

Regarding the first case, it led to lower incomes, going below 140€ in some days, while in the other two cases the incomes stay above 335€ at all times. This remark points out the fact that the method is affected by the competitor players' actions' worst forecasts, resulting from the effectiveness preference of 0%. Also, the selected optimization algorithm for this case may not be the best as well, since for this scenario the most relevant aspect taken into account is the execution time of the algorithms, and not their quality in achieving good results.

Concerning the optimization to generate the best bid to be chosen, several parameterizations for all the algorithms were defined for testing. This way it is possible to compare the algorithms performance when using different parameters.

Tables 3.9, 3.10, 3.11 and 3.12 present the parameterizations used for the optimization algorithms.

Table 3.9 – Parameterizations for the *explicit enumeration* algorithm.

E1	
Increment interval	0,2
E2	
Increment interval	0,1
E3	
Increment interval	0,05

Table 3.10 – Parameterizations for the *PSO* algorithm.

PSO1	
Number of Particules	25
Number of Iterations	1000
C1	0.1
C2	0.1
W	0.1
Maximum Speed	1
Minimum Speed	-1
PSO2	
Number of Particules	40
Number of Iterations	1250
C1	0.3
C2	0.3
W	0.3
Maximum Speed	1
Minimum Speed	-1
PSO3	
Number of Particules	50
Number of Iterations	1500
C1	0.5
C2	0.5
W	0.5
Maximum Speed	1
Minimum Speed	-1

Table 3.11 – Parameterizations for the *Tabu Search* algorithm.

TS1	
Size of Tabu List	2
Number of Iterations	150
Type of neighborhood generation	Range 2
TS1	
Size of Tabu List	3
Number of Iterations	200
Type of neighborhood generation	Range 2
TS1	
Size of Tabu List	4
Number of Iterations	500
Type of neighborhood generation	Range 2

Table 3.12 – Parameterizations for the *SA* algorithm.

SA1	
Temperature	50
Number of Iterations	1000
SA2	
Temperature	50
Number of Iterations	500
SA3	
Temperature	80
Number of Iterations	1500

These parameterizations were defined to allow a comparison between some alternative combinations. However, it is certain that the consideration of innumerous other combinations could be interesting for testing.

In order to test these parameterization performances, two sets of five market simulations using the proposed method were performed. One set considers 11 competitor players, and the other considers only 5, since the number of players is the most influential factor over the optimization's performance, as it defines the number of variables to be considered in the optimization. Each set of simulations regards the method's use with: 0%, 20%, 50%, 80% and 100% preference for efficiency. This means that for each case, each parameterization has a different chance of being chosen more often to be executed by the case-base learning, depending on the results it originated: obtained incomes, and execution time. The achieved results are presented in Table 3.13.

Table 3.13 – Results of the several parameterizations (incomes in €).

		5 Players			11 Players		
		Best	Average	Worst	Best	Average	Worst
Optimization Algorithms Parameterizations	EX1	399,0	372,2	297,8	386,5	302,2	223,5
	EX2	360,3	340,5	272,4	389,0	378,3	315,8
	EX3	399,0	372,3	297,8	396,5	385,2	362,3
	PSO1	382,0	372,5	298,0	364,5	218,2	62,5
	PSO2	394,0	376,5	301,2	390,5	242,1	68,2
	PSO3	399,0	384,4	307,5	369,0	255,3	140,5
	TS1	391,5	371,4	297,1	393,5	382,6	322,5
	TS2	396,5	372,4	297,9	387,5	370,5	351,0
	TS3	394,6	375,7	300,6	382,6	372,8	245,0
	SA1	384,0	374,5	299,6	383,0	238,3	148,4
	SA2	379,5	364,5	291,6	374,5	364,8	337,6
	SA3	337,5	367,5	294,0	369,0	145,0	64,0

Analyzing these results it is visible that for the cases concerning 5 players, the performances are very similar. This fact is certainly due to the small number of variables, and consequently, smaller complexity in solving the optimization problem, which allowed all the parameterizations to get good results. The emphasis in this case goes to the *explicit enumeration* and *PSO* algorithms, for these were the ones that got the best results among all, even if with a small difference from the other algorithms.

Regarding the 11 players cases, the differences are much more evident, being the *explicit enumeration*, namely with the third parameterization, the algorithm that got the best results. This is the expected result, since this algorithm ensures the testing of all possible solutions. These results support the decision of choosing the *explicit enumeration* as the standard algorithm for the *Determinism Theory Strategy* when the higher importance is to achieve the most profitable solution, without taking into account the execution times. The second best algorithm, in terms of results is the *Tabu Search*. The PSO, which presented good results in the 5 players' case, has shown that it does not respond well in cases with a higher number of variables.

Besides the objective function results, expressed in incomes, obtained by each parameterization, another important result is the execution time that each one presents. Table 3.14 presents a comparison between the average execution times of each algorithm using each parameterization. Additionally, to support the choice of LPA-Prolog as programming language for implementing the optimization algorithms, the execution times of the same algorithms implemented in JAVA (the main programming language of ALBidS and MASCEM). The presented execution times are the average of 5 runs with each parameterization.

Table 3.14 – Average execution times of the tested parameterizations, presented in milliseconds.

		Optimization Algorithms Parameterizations											
		EX1	EX2	EX3	PSO1	PSO2	PSO3	TS1	TS2	TS3	SA1	SA2	SA3
5 Players	LPA-Prolog	4632,6	8421,2	15263,6	3212,3	6560,2	9739,4	197,3	258,7	321,4	151,0	72,6	233,0
	JAVA	662461,8	1204231,6	2182694,8	459363,7	938108,6	1392738,3	28213,9	36999,4	45960,2	21593,0	10381,8	33319,0
11 Players	LPA-Prolog	7412,2	13473,9	24421,8	4240,7	8085,4	12237,0	171,0	210,2	283,1	205,4	102,0	310,0
	JAVA	1245242,9	2263618,6	4102855,7	712432,0	1358347,2	2055816,0	28728,0	35318,8	47560,8	34507,2	17136,0	52080,0

The results presented in Table 3.14 are clear in what concerns the comparison of the execution times between the versions of the algorithms implemented in LPA-Prolog and the ones implemented in JAVA. All parameterizations take over 100 times longer to execute when implemented in JAVA. Even though it was expected that the differences would be evident due to the characteristics of the two programming languages, the amount of the difference is huge. These are very important results, which support the decision of implementing these algorithms in Prolog.

Regarding the comparison between the different algorithms and their parameterizations, the faster one is the SA2, the *SA* parameterization with the smaller number of iterations. In terms of execution times the *SA* and *Tabu Search* are the algorithms that present the best results. The EX3 parameterization, which presented the best income results, is the one presenting the worst execution times, fact that was expected from this algorithm's characteristics.

The presented results are relevant to support the decisions taken when developing the Determinism Theory Strategy, and to show the importance of the various factors that influence the method's performance.

3.7.9. *Metalearner Agents*

This sub-section presents two ALBidS' strategy agents. These agents are characterized as meta-learners, as they use the results of the learning process from all the other strategies presented before, as inputs to apply their own learning, and therefore create new outputs.

3.7.9.1. Context

Metalearning is one-step higher than ordinary learning and means Learning-About-Learning. As described in [Biggs, 1985], metalearning is the state of "being aware of and taking control of one's own learning". It can be defined as an awareness and understanding of the phenomenon of learning itself as opposed to subject knowledge.

Regarding the context of computer science, and Artificial Intelligence (AI), metalearning is a subfield of machine learning, where automatic learning algorithms are applied on meta-data about machine learning experiments [Bruha, 2004]. The main goal is to use such meta-data to understand how automatic learning can become flexible in solving different kinds of learning problems, hence to improve the performance of existing learning algorithms.

Flexibility is very important because each learning algorithm is based on a set of assumptions about the data. A learning algorithm may perform very well on one learning problem, but very badly on the next. From a non-expert point of view, this poses strong restrictions on the use of machine learning or data mining techniques, since the relationship between the learning problem and the effectiveness of different learning algorithms is not yet understood.

By using different kinds of meta-data, like properties of the learning problem, algorithm, or patterns previously derived from the data, it is possible to select, alter or combine different learning algorithms to effectively solve a given learning problem [Goldkuhl *et al*., 1998].

Metalearning is strongly related to *stacked generalization*, which works by combining a number of (different) learning algorithms. The meta-data is formed by the predictions of those different algorithms. Then another learning algorithm learns from this meta-data to predict which combinations of algorithms give generally good results. Given a new learning problem, the predictions of the selected set of algorithms are combined, *e.g.* by the use of weights, to provide the final prediction. Since each algorithm is deemed to work on a subset of problems, a combination is hoped to be more flexible and still able to make good predictions. *Stacked generalization* leads to the concept of *Boosting* [Boyarshinov *et al.,* 2007], which uses the same algorithm multiple times, but the examples in the training data get different weights over each run. This provides different predictions, each focused on rightly predicting a different subset of the data, and combining those predictions usually leads to better results, although more expensive in terms of processing time.

3.7.9.2. Implementation

Taking advantage on the concept of metalearning, and its adequacy to the scope of this work, it has been decided to develop two versions of metalearners, in order to test the advantage of using such type of learning algorithms, which perform a combination of the outputs of the several different approaches previously developed.

Simple Metalearner

The idea for the development of the Simple Metalearner has arisen from the analysis of the work of Hans Georg Zimmermann, a Siemens AG, Corporate Technology [Siemens, 2011] researcher whose work deals with, among many other issues, forecasting using neural networks (NN). In his widely recognized work [Zimmermann *et al.*, 2001], [Schaefer *et al.*, 2001], he recurrently uses simple averages of the outputs of the used NN, to originate a final output, in order to overcome the uncertainty that affects the forecasts. Zimmermann states that *"Uncertainty shows up because we do not know the true scenario. Stochasticity is not seen as a feature of the real world, but a consequence of partial observability."* [Zimmermann *et al.*, 2005].

In order to overcome such uncertainty, Zimmermann uses different initializations for the used NN, covering a larger space of solutions, and then uses an ensemble averaging to create the final solution. *"Large recurrent neural networks with tanh nonlinearities can be brought to work if we start initially with a sparse interconnectivity (structural sparsity). Obviously the solution of the system identification will depend on the initialization – which can be handled by an ensemble averaging."* [Zimmermann *et al.*, 2005].

The scope of the ALBidS system is similar, although many other approaches other than NNs are used. For this reason it has been found interesting to experiment if the averaging of the outputs of the various ALBidS strategies could lead to interesting results.

The procedure of the Simple Metalearner's processing is, therefore, a simple averaging between all the outputs of the strategies that are used in a certain situation. Note that the used strategies for this average depend both on the user initial preferences, and also on the requirements demanded by the Efficiency/Effectiveness Management mechanism (see section 3.9).

Weighted Metalearner

The Weighted Metalearner intends to extend the Simple Metalearner, creating an adaptive tool that can in fact be called a metalearner, using the concept of *stacked generalization*. This metalearner uses as inputs, the outputs of the various approaches, but presents the additional feature of attributing importance weights to each of those inputs. The weights provide the chance for the metalearner to adapt its output, giving higher focus to the results of the strategies that are proving to be more adequate, while partially or completely ignoring the contribution of the strategies which are presenting worst results.

This procedure allows the Weighted Metalearner to adapt its output according to the observed results of each of its inputs. The weights used for defining the importance that each input has for the metalearner, are based on the confidence values of the main reinforcement learning algorithm used by ALBidS (see section 3.6 – Main Agent). The reinforcement learning algorithm’s confidence values are adapted and updated according to the results each strategy is presented, hence being exactly what this metalearned requires understanding which are the strategies’ outputs that it should consider as most influent to the final output.

The generation of this output $\bar{x}_p$ is performed through a weighted average, using the reinforcement learning algorithm’s confidence values as weights p_1, p_2, p_3, ..., p_n for each strategy’s output’s x_1, x_2, x_3, ..., x_n contribution to the final metalearner’s solution. The procedure is expressed in (3.66).

$$\bar{x}_p = \frac{p_1 \times x_1 + p_2 \times x_2 + \ldots + p_n \times x_n}{p_1 + p_2 + \ldots + p_n} = \frac{\sum_{i=1}^{n} p_i \times x_i}{\sum_{i=1}^{n} p_i} \qquad (3.66)$$

The adapted output of the Weighted Metalearner is expected to be able to generate better results than the Simple Metalearner, since it takes higher regard for the inputs that are expected to point the final result towards a better solution.

3.7.9.3. Experimental Findings

This sub-section presents two simulations using the test scenario, for 61 consecutive days, starting from the 15th October. In the first simulation Seller 2 uses the Simple Metalearner, and in the second it uses the Weighted Metalearner.

These results purpose is to demonstrate the use of these strategies to support a market player’s decisions and to show the influence of the supporting strategies over these metalearners performance.

For these simulations, the main reinforcement learning algorithm, whose confidence values are used by the Weighted Metalearner, is the Bayes Theorem algorithm. Additionally, as supporting strategies, nine random strategies from all the ones presented in this document, were selected. These strategies are the same for both simulations.

Figure 3.68 presents the incomes obtained by Seller 2 in the first simulation, and a comparison between the proposed bid and the market price.

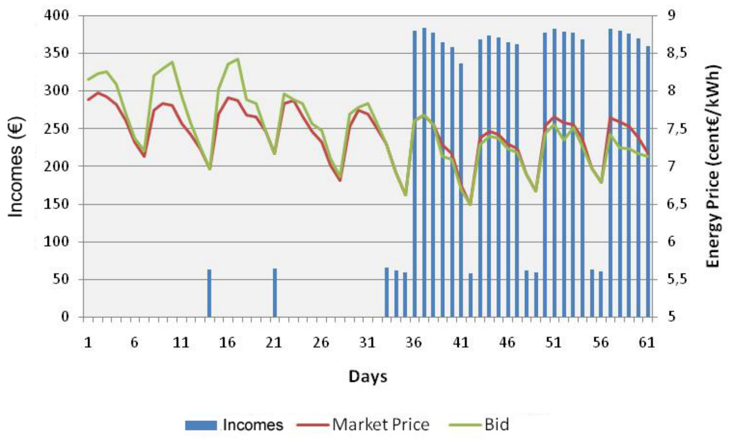


Figure 3.68 – Results of the Simple Metalearner.

Analyzing Figure 3.68 it is visible that this strategy starts by achieving bad results, which start to improve with the passing of the days. This is due to some supporting strategies' worst suggestions at the start, while they do not have the experience to learn adequately yet. As this metalearner considers all suggestions in a similar way, the good outputs that some strategies may be presenting are muffled by the bad ones. As the time progresses and the strategies start to learn and provide best individual results, the metalearner's results improve as well. Figure 3.69 presents a comparison between the metalearner's bid price and the supporting strategies' bid proposals.

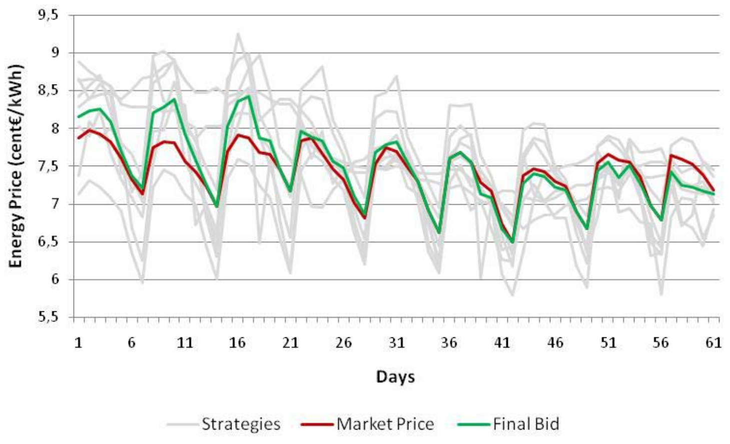


Figure 3.69 – Simple Metalearner's and supporting strategies' bids.

Figure 3.69 shows that, in the first days, the strategies' proposals present a high variance among each other, and so the metalearner's bid varies much as well. In the later days, the proposals start to converge and improve their quality, and consequently, the metalearner's bid gets closer to the market price.

The consideration of all suggestions in an equal way results in a bad performance of the Simple Metalearner especially in the first days. This is why the use of the Weighted Metalearner is important. This method considers adequate weights to instigate the attribution of higher importance to the strategies that are presenting best results at each time. Figure 3.70 presents the results of Seller 2 using the Weighted Metalearner.

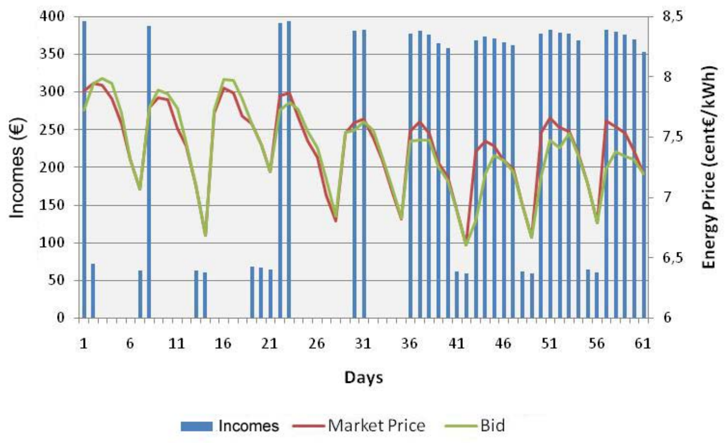


Figure 3.70 – Results of the Weighted Metalearner.

Figure 3.70 shows that, in spite of these results following the same tendency as the Simple Metalearner, with bad results at first, and a visible improvement over time, it could achieve good results in some early days, proving the advantage of attributing higher weights to the most appropriate supporting strategies' proposals. Figure 3.71 presents the development of the supporting strategies' confidence values, which determine the weights each one will be attributed.

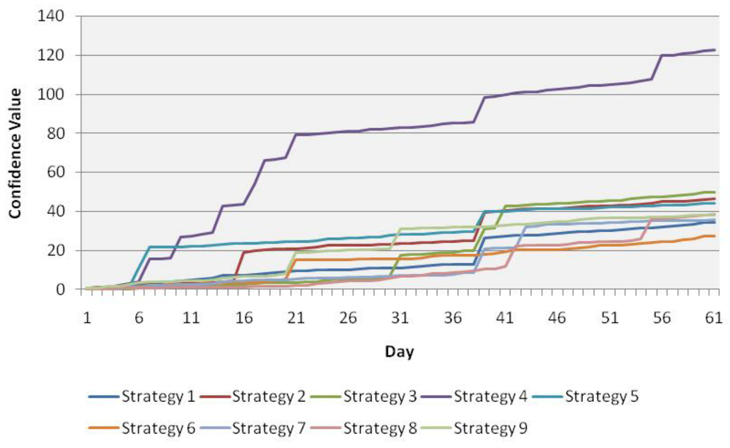


Figure 3.71 – Supporting strategies' confidence values.

From Figure 3.71 it is visible that *Strategy 4* detaches from the others. This means that this strategy is the one that influences the most the final bid from the Weighted Metalearner. Its influence increases over time. Figure 3.72 shows how this metalearner's bid follows the *Strategy 4*'s responses, in comparison with the rest of the supporting strategies.

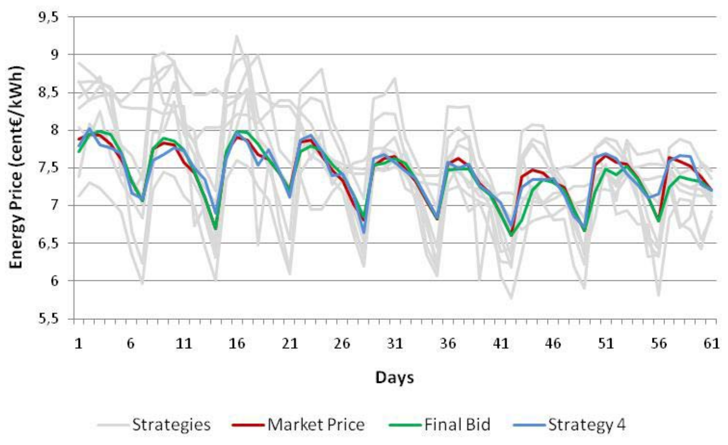


Figure 3.72 – Weighted Metalearner's and supporting strategies' bids.

Figure 3.72 shows that, in spite of in the first days the strategies' proposals presenting an high variance among each other, the metalearner's bid stays close to the market price, influenced by *Strategy 4*. This tendency remains visible throughout the days.

Comparing the Weighted Metalearner's performance with the Simple Metalearner, a much more constant behaviour is found. The adjustment of the final bid, taking into account the strategies' results has proven to be an added value in adequating the metalearner's results.

3.8. Player Profile Definition

In order to build suitable profiles of competitor agents, it is essential to provide players with strategies capable of dealing with the possible changes in competitors' behaviour, allowing adaptation to their actions and reactions [Pinto *et al.*, 2011c]. For that, it is necessary to have adequate techniques to analyze the data properly, namely the historic of other agents past actions. Analogously to the definition of market operation strategies, the way each agent's bid is predicted can be approached in several ways. So, the way to deal with this issue was to follow the same idea as for the main system's methodology. There are several algorithms for defining the players' profiles, all providing their predictions, and on top of that a reinforcement learning algorithm that chooses the one that is more likely to present the best answer according to the past experience of their responses for each particular market context.

The used reinforcement algorithm is the *Roth-Erev* algorithm [Roth and Erev, 1995], presented in sub-section 3.6.2. It presents a distinct set of statistics for each acting agent, for their actions to be predicted independently from each other, and also for each period or market context. This means that an algorithm that may be presenting good results for a certain agent in a certain context, with its output chosen more often when bidding in this context, may possibly never be chosen as the output for another context.

The update of the stats is done accordingly to the difference between the predictions and the action each player actually performed. The best rewards are attributed to the profile definition algorithms that present the smaller difference. This way, all algorithms' confidence values are updated in every prediction, whether a prediction is chosen as the final answer or not.

3.8.1. *Strategies*

There are several algorithms based on different approaches, used for predicting competitor players' actions. Some of these algorithms are reused and adapted from the bids' definition scope. Namely:

- A **dynamic feed-forward artificial neural network** trained with the log of historic market actions of a competitor player. The structure is the same as in the NN presented in sub-section 3.7.2, with an input layer of eight units, regarding the prices and powers that compose the player's bid of the same period of the previous day, and the same week days of the previous three weeks. The intermediate hidden layer has four units and the output has one unit – the predicted bid price of the analyzed agent for the period in question. This NN is executed and trained, analogously to the previously presented one.
- **Statistical approaches based algorithms**, for which the following are used:
    - Average of prices and powers from the agents' past actions database, using the data from the 30 days prior to the current simulation day, considering only the same period as the current case, of the same week day. This allows us to have a strategy based on the tendencies per week day and per period;
    - Average of the agent's bid prices considering the data from one week prior to the current simulation day, considering only business days, and only the same period as the current case. This strategy is only performed when the simulation is for a business day. This approach, considering only the most recent days and ignoring the distant past, gives us a proposal that can very quickly adapt to the most recent changes in this agent's behaviour. It is also a good strategy for agents that tend to perform similar actions along the week;
    - Average of the data from the four months prior to the current simulation day, considering only the same period as the current case. This offers an approach based on a longer term analysis. Even though this type of strategies, based on averages, may seem too simple, they present good results when forecasting players' behaviours, taking only a small amount of time for their execution;
    - Regression on the data from the four months prior to the current simulation day, considering only the same period of the day;
    - Regression on the data of the last week, considering only business days. This strategy is only performed when the simulation is for a business day.

Besides the adapted algorithms, whose utility is obvious when applying to this context as well, there are some other algorithms, built specifically for competitors' actions prediction. This new group of algorithms has been developed in LPA Prolog [Prolog, 2011], to increase the speed of processing.

To support the predictions, these algorithms have access to a *KnowledgeBase* file, where the log of each player's actions is saved. The knowledge base includes the following information regarding players' actions: the players bid prices, their desired amount of power, and the success that action presented (*i.e.* if that action allowed to player to buy or sell all the desired power, just a part, or none of that power). This information is of the greater importance for some of these algorithms, especially those based on *history matching*.

When a negotiation period is finished, and the update of the algorithms' stats is done, the update of the knowledge based is also undertaken. The update is performed, not only by the inclusion of the new data regarding each player's verified action, but also by an update in the saved patterns found so far. The patterns in each player's actions are searched for in every update, by matching the new information that is provided in the update, with the data already known. This allows the patterns to be found much faster than having to go through all the data every time a prediction is required.

Besides the *KnowledgeBase* file there is also the *Search* file, which implements the algorithms, and the *Main* file, which contains the main predicates to be used by the JAVA program, and accesses the two other files when required. The *Search* file contains the following algorithms:

- Algorithms based on **pattern analysis**:
  - Sequences in the past matching the last few actions. This approach considers the sequences of at least three actions found along the historic of actions of this player. The sequences are treated depending on their size. The longer matches to the recent history are attributed an higher importance;
  - Most repeated sequence along the historic of actions of this player;
  - Most recent sequence among all the found ones.
- Algorithm based on **history matching**, regarding not only the player actions, but also the result they obtained. This algorithm finds the previous time that the last result happened, *i.e*., what the player did, or how he reacted, the last time he performed the same action and got the same result.
- Algorithm returning the **most repeated action** of this player. This is an efficient method for players that tend to perform recurrent actions.
- **Second-Guessing** the predictions. Assuming that the players whose actions are being predicted are equipped with intelligent behaviour, it is essential to shield this system, avoiding being predictable as well. So this strategy aims to be prepared to situations when the competitors are expecting the actions that the system is performing.
  - Second-Guess: if the prediction on a player action is *P*, and the player is expecting the system to perform an action *P1* that will overcome his expected action, so in fact the player will perform an action *P2* that overcomes the system's expected *P1*. This strategy's prediction of the player's expected action is *P2*.
  - Third-Guess: this is one step above the previous strategy. If a player already understood the system's second guess and is expecting the system to perform an action that overcomes the *P2* action, than it will perform an action *P3* that overcomes the system prediction, and so, this strategy returns *P3* as the predicted player action.

- **Self Model prediction**. Once again, if a player is equipped with intelligent behaviour, he can perform the same historical analysis on the system's behaviour as the system performs on the others. This strategy performs an analysis on the supported player's own historic of actions, to predict what itself is expected to do next. From that the system can change its predicted action, to overcome the players that may be expecting it to perform that same predicted action.
- **Second-Guessing the Self Model** prediction. The same logic is applied as before, this time considering the expected play resulting from the Self Model prediction.

The strategies that are used to predict an agent's behaviour depend on the efficiency/effectiveness balance (see section 3.9) which is defined. If the preference is fully attributed to the efficiency of the method, only the Prolog based strategies are used, allowing a huge reduction of the execution time. Otherwise, all strategies are used. However, the NN is adapted for each circumstance. The higher the preference for effectiveness, the bigger the amount of data considered for training the NN. This allows reducing the execution time when required, even if just for a small amount, or increasing the quality of the predictions, when that is the main goal.

3.8.2. *Experimental Findings*

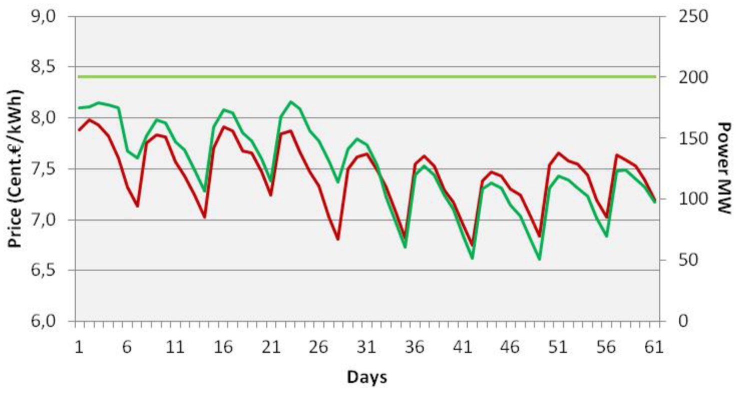
This sub-section presents the results of the tests on the Player Profile Definition mechanism‘s suitability in predicting agents' actions.

The presented results refer to this mechanism's tests when predicting two agents' behaviour, one buyer agent, and one seller agent. The tests were performed using the test scenario, for 61 consecutive days, starting on October 4th. The two test subjects are Buyer 5 and VPP 2. Buyer 5 presents a bid based on the average prices of the last four months, while requesting for a fix amount of power to buy regardless of the situation. This test allows focusing on the players' bid price prediction, while ignoring the prediction of its demanded power. VPP 2, however, by representing a coalition of renewable generation sources, presents a variable amount of power to sell, which is based on the total forecasted production. Its offer price is based on the costs of co-generation. This allows testing the Player Profile Definition mechanism‘s applicability to predicting a player's bid price, and negotiated amount of power.

The tests regarding the prediction of both agents are performed under two circumstances, a preference for efficiency of 100% (see section 3.9), and a preference for efficiency of 0%. This allows comparing the Player Profile Definition mechanism‘s performance when working at its best, whit the situation when it is restricted to the Prolog algorithms due to execution time limitations.

Figure 3.73 presents the results of the tests regarding the prediction of Buyer 5's bids, for both referred preference percentages for efficiency.

a)



b)

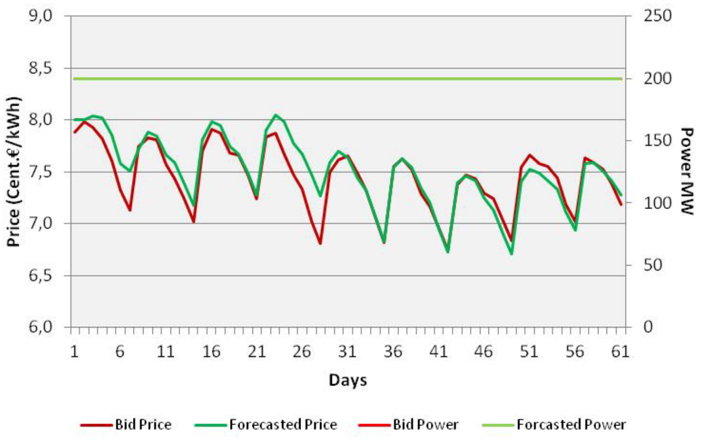


Figure 3.73 – Predicted and actual bid prices and powers, regarding Buyer 5, for an efficiency preference of: a) 100%, b) 0%.

Figure 3.73 shows that Buyer 5's bid price prediction was more accurate when the preference was attributed totally to the effectiveness of the method. This was an expected result, as in this case the Player Profile Definition mechanism has at its disposal a higher number of profile definition strategies, providing a larger number of results, which grants a higher chance of being more accurate. However, for this case, the differences, in spite of being noticed, are not that big. In what regards the power prediction, as it is fixed in a constant amount, it created no difficulty of prediction in none of the cases, with both achieving an expected 100% accuracy.

The differences in performance when considering a higher or lower preference for efficiency or effectiveness are dependent on the evolution of the considered supporting strategies' individual performance. The way the strategies perform affects the reinforcement they will be attributed by the reinforcement learning algorithm. This reinforcement defines the chances of each strategy's response being chosen as the final prediction. The evolution of the stats each strategy presented for the reinforcement learning algorithm, in both cases of the considered efficiency/effectiveness preference percentage is presented in Figure 3.74.

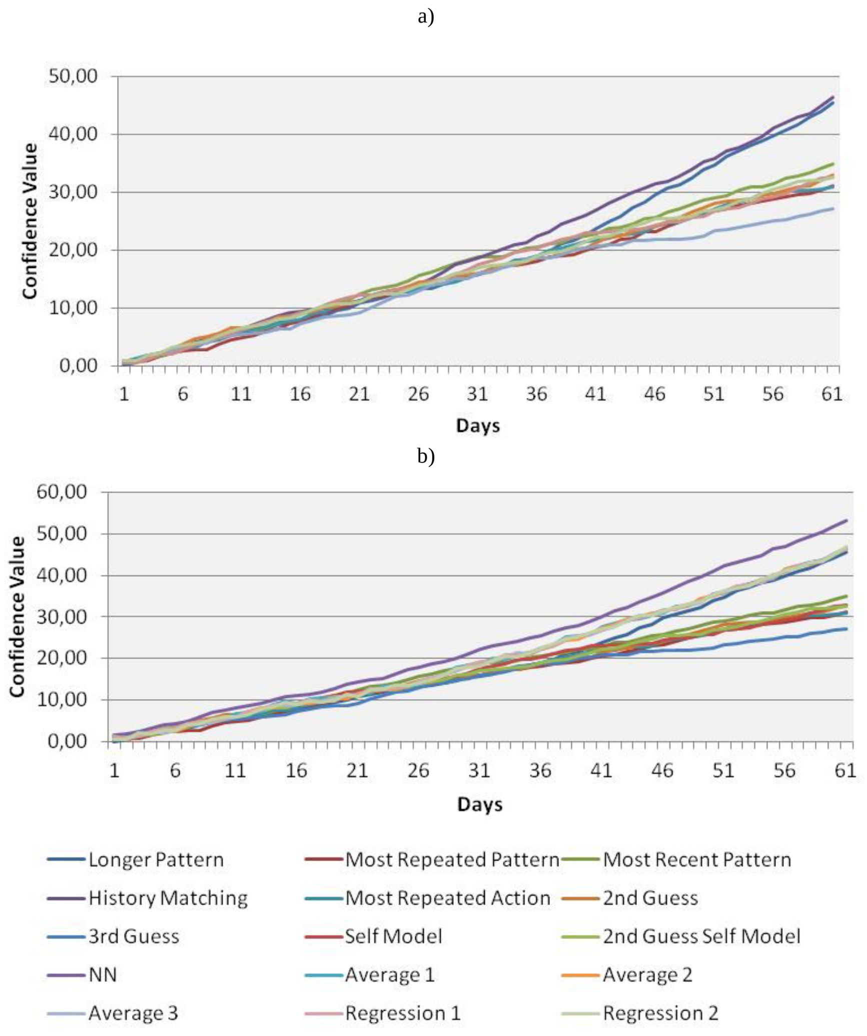


Figure 3.74 – Strategies' stats, for an efficiency preference of: a) 100%, b) 0%.

With the exclusion of the NN, the average, and the regression strategies, for the case with 100% preference for the efficiency, the Player Profile Definition mechanism strategies that presented higher stats were the *longer pattern*, and the *history matching*.

Regarding the case with full preference for the effectiveness, the strategy that most detached from the rest was the NN. The fact that different strategies got the bests results for the two cases explains why the results for the two cases are distinct.

The tests for the prediction of VPP 2 actions include both the prediction of the bid price, and of the bid power. The results regarding the tests on VPP 2's bid price prediction, for both preference percentages for efficiency, are presented in Figure 3.75.

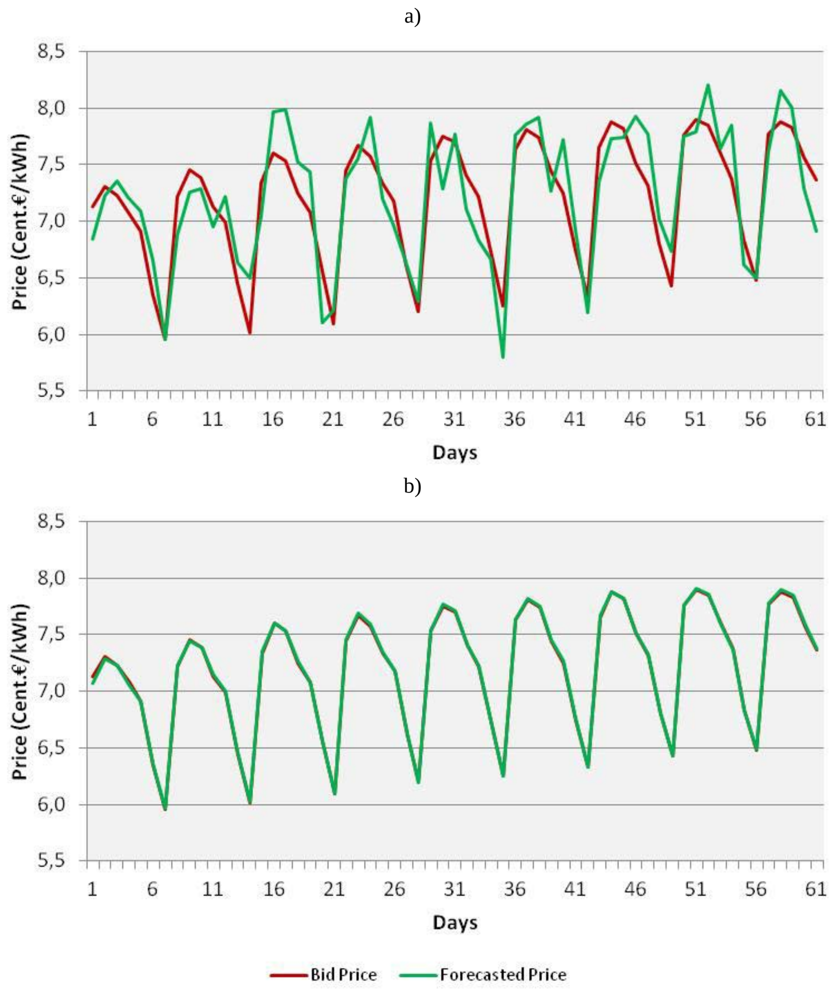


Figure 3.75 – Predicted and actual bid price of VPP 2, for an efficiency preference of: a) 100%, b) 0%.

Figure 3.75 shows that the bid price prediction for this agent was once again more accurate when the preference was attributed totally to the effectiveness of the method. The results concerning VPP 2 are more evident than the ones regarding Buyer 5, with the difference between the forecasted and actual bid prices being highly reduced from the case with 100% preference for efficiency to the case with 0% preference.

These differences are shown in Table 3.15, which presents the MAPE values of Buyer 5 and VPP 2’s bid price and power predictions, for several values of preference for the efficiency/effectiveness balance.

Table 3.15 – MAPEs of the predictions for several values of preference for efficiency.

Efficiency preference (%)	Buyer 5	VPP 2	
	Price (€)	Price (€)	Power (MWh)
100%	0,026477	0,034703	0,329787
80%	0,016846	0,009432	0,215466
50%	0,015582	0,006959	0,189349
20%	0,014566	0,002849	0,172547
0%	0,014101	0,001363	0,164893

Analyzing Table 3.15 it is visible that the predictions present best results, in all cases, when applied in situations with 0% preference for efficiency. The difference of the results is more evident in the case of predicting Buyer 5's bid prices.

Regarding the power prediction, the best results continue to be achieved when the preference falls completely on the effectiveness of the mechanism. However, these results are much worse than the ones obtained when predicting the bids prices. This can be seen in Figure 3.76, which presents a comparison between the predicted and actual bid power of VPP 2, for a preference value for efficiency of 0% and 100%.

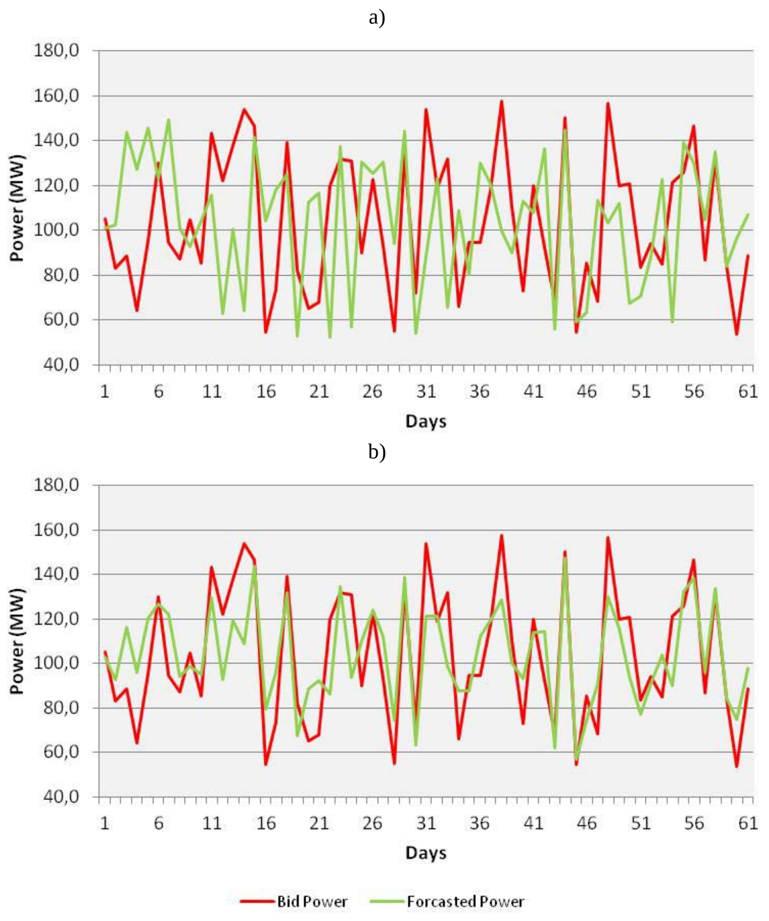


Figure 3.76 – Actual and predicted bid power of VPP 2, for an efficiency preference of: a) 100%, b) 0%.

Analyzing Figure 3.76, it is visible that the predictions of VPP 2 bid power are much worse than the predictions of its bid price. Regarding the prediction performance, when comparing the cases with full preference for efficiency and for effectiveness, an evident improvement is verified in the case with 0% preference for efficiency.

The stats of the different strategies regarding the prediction of the VPP bid price are presented in Figure 3.77.

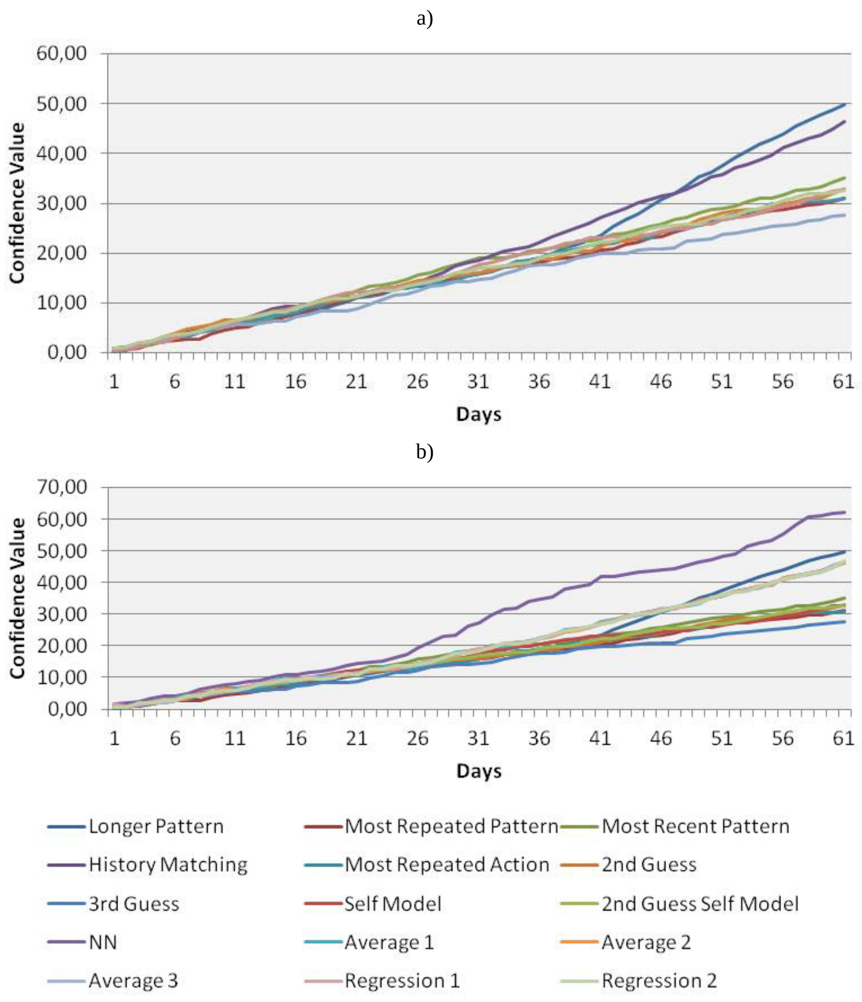


Figure 3.77 – Strategies' stats, for an efficiency preference of: a) 100%, b) 0%.

From Figure 3.77, it is visible that in the case of 100% preference for efficiency, the strategies presenting higher confidence values are the *Longer Pattern*, and the *History Matching*, such as occurred in the Buyer 5's bid prediction case. In the case of 100% preference for the effectiveness of the method, the NN maintains its leadership, followed by the *Longer Pattern* and *Regression 2* strategies.

The strategy presenting the lower confidence values is the *3rd Guess*. This occurs because of the characteristics of the predicted player. It defines its bid depending on its production, which is based on environmental resources, while this strategy is directed to the prediction of players' actions, whose bids are defined based on intelligent decision making, rather than on natural environmental events. When confronted with other types of opponents, this strategy can prove to be very useful. This is the main advantage of considering several different approaches to predict an action, having, at all times, strategies prepared to deal with different scenarios they may come across with.

The bid power being based on natural events, much harder to predict, is also the main explanation of the Player Profile Definition mechanism's worst results when predicting a bid power, than when predicting a bid price.

As presented in these results, the Player Profile Definition mechanism proves to be an adequate tool to provide predictions of players' actions. It is able to forecast agents' bid prices, and predict players' bid powers, and therefore create effective agent profiles to be used by other ALBidS strategies which require such profiles.

The Player Profile Definition mechanism also proved to be adequate for different situations in what concerns the efficiency/effectiveness balance. Even in more exigent environments in what concerns its execution time, this mechanism is able to provide very acceptable results.

3.9. Efficiency/Effectiveness Management

Since there are many algorithms running simultaneously, some requiring higher processing times than others and some achieving a higher level of effectiveness in their forecasts, it became necessary to build a suitable mechanism to manage the algorithms efficiency in order to guarantee the minimum degradation of the previous implementation performance, *i.e.* the MASCEM simulator's processing time without considering ALBidS integration [Pinto *et al.*, 2011d].

To accomplish this goal, a methodology to manage the efficiency/effectiveness (2E) balance of ALBidS has been developed [Pinto *et al.*, 2011d], to guarantee that the degradation of the simulator processing times takes the correct measure, depending on the type of the simulation. If the simulation is intended to support a player's decision, being its market results the most important thing to look at, then the processing time of the simulation loses its importance, and the focus is given to the execution of all the techniques in the best way possible. However, if the user intends to perform a simulation to analyze some other aspect of electricity markets, such as VPP operation, or participation in other types of markets, then the system does not need to spend so much time processing the support to a player's actions in the simulations, rather provides support in a fast way, to allow the user to run the simulations more quickly, and spend its time analyzing the intended features.

3.9.1. *Mechanism for Dynamic 2E Adaptation*

To adjust the efficiency/effectiveness management mechanism to each simulation purpose, the user is able to define the relative importance that efficiency and effectiveness will have. If the user chooses 100% importance for the effectiveness, in case of the simulation being used for decision support for the market, and so the most important issue being the quality of the forecast, and not how long they take, since market bids are asked only once every 24 hours, then all the algorithms and agents will be turned on, and contribute to the simulation. In the other extreme, if the user chooses 100% importance for efficiency, in case of the simulation being with the purpose of studying other issue of the market that MASCEM offers, that not the forecasts, and so, preferring a much faster simulation, with no degradation of the simulator times, then all the strategy agents that require more time to process than the MASCEM simulation without them, are excluded, remaining only the faster ones.

Choosing anywhere in between demands a careful and smart analysis of the efficiency/effectiveness balance. This means that increasing the importance of the efficiency does not mean a direct exclusion of the agents that are taking longer to provide an answer. Rather a conjugation of this aspect with the effectiveness of each method, so an agent requiring a little more time than other but giving best responses, can be excluded later than the other. This is done using three fuzzy variables [Gorride and Pedycz, 2007] - one for the effectiveness, characterizing the difference between each forecast and the real verified value; one for the efficiency, doing the same to represent the difference between the processing times of each method and MASCEM's without considering this system integration; and a third variable, representing the user's choice of efficiency/effectiveness importance percentage. The fuzzy variables for efficiency and effectiveness are defined as in Figure 3.78.

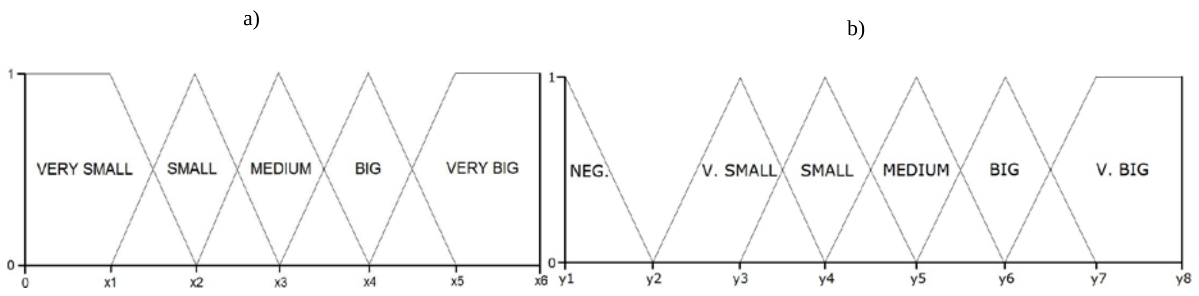


Figure 3.78 – Fuzzy Variables and respective membership functions definition, for the characterization of: a) the effectiveness, b) the efficiency [Pinto *et al.*, 2011d].

Both fuzzy variables are dynamic, being updated in every run, to guarantee the suitability of the membership functions in each situation. Defining the intervals as static values could lead to situations where all, or a large part, of the methods are being classified in the same interval of difference, and so, this methodology losing its logic. This way, with dynamic variables, the values are always adapting to the characteristics that are being observed, making sure that a Very Big value is actually a very big value considering the values that were registered so far, and that a Medium value is actually a value that lies in the average of the registered values.

To consistently adapt these intervals, in case of the effectiveness, as presented in Figure 3.78 a), the variable x6 is always the maximum value of difference between a strategies' forecast and the actual value, x3 the average value of difference between all the forecast methods, and x1, x2, x4 and x5 are defined accordingly to the other two values. The effectiveness difference is calculated through a simple comparison of the strategy's proposed price with the actual market price. However, these values pass through a penalization rule in case of the difference being positive, *i.e.* if the proposed bid is above the market price. In this case, the difference assumes the triple of the value it would get if the difference was negative, *i.e.* if the proposed bid is below the market price. This is applied because it is much more harmful to the agent if it bids above the market price, which means not being able to sell, while biding below just means a possible not achievement of the maximum income it could get (see sub-section 2.2.1.1 – spot market).

In case of the efficiency, Figure 3.78 b), y1 represents the minimum value of the processing time verified among all the methods, y2 is the processing time of the MASCEM simulator without considering the integration of ALBidS, y4 is the average of the processing times of all the methods, and y6 is the maximum time that a method took so far to achieve an answer. This way, all the values between y1 and y2 will be assigned as Negative, and the corresponding methods will never need to be excluded by this method. The methods taking all the values above the processing time of MASCEM, will depend on the percentage of importance for effectiveness/efficiency.

The fuzzy confusion matrix (can be consulted in Annex A) that joins these two fuzzy variables with the effectiveness/efficiency importance percentage fuzzy variable, will determine the action to take regarding each strategy, by combining the three fuzzy values. Besides the exclusion of an algorithm, the confusion matrix can also define that a strategy is able to stay in the system unaltered, or, in other cases, that the strategy needs to reduce its processing time, by a bigger or a smaller extent. In this last case, the strategies will adapt themselves to this request, by reducing their execution time. The way this reduction is accomplished depends on the different algorithms internal adaptation, *e.g.* the NN reduces the training limit, *i.e.* the amount of data used for its training; and the Game Theory Agent reduces the number of scenarios. Reductions in execution time are dealt with individually by strategy in the Strategy Agents section – 3.7. By allowing strategies to reduce their execution time, they are given an extra chance to adapt to the simulation circumstances. In case of their reduction being effective, they manage to continue contributing to the system without being too influential over the execution times degradation; rather than being excluded right away.

The efficiency/effectiveness management mechanism is implemented in LPA Prolog [Prolog, 2011], using the Flex environment [Flex, 2011]. The Prolog mechanism receives the requests from the JAVA program, more precisely from the Main Agent, providing the means for calculating the decision, or action to take, regarding each strategy. The mechanism receives data concerning the historic of the execution times of each strategy, and applies the described fuzzy process. By receiving the most recent updates of the execution times and effectiveness results that the strategies are presenting, the efficiency/effectiveness management mechanism also updates the intervals and values of the fuzzy sets of the fuzzy variables.

The dynamic update of the interval values of the fuzzy variables is processed through a manipulation of the fuzzy variables internal attributes. The *fuzzy_variable_range* allows changing the minimum and maximum values that the variable will process. The *fuzzy_variable_qualifier*, for each fuzzy variable, gives the chance to change all the interval values of the different fuzzy sets.

3.9.2. *Experimental Findings*

To test this mechanism's correct operation, five test cases are presented, concerning different preferences regarding the efficiency/effectiveness balance. These tests are performed using the real execution times verified by MASCEM and by the proposed strategies. Table 3.16 presents the strategies' average execution times, and for the strategies where it is applicable, the times under different efficiency/effectiveness percentage preferences (*e.g.* NN with different training limits, Game Theory strategy with different number of scenarios and possible bids).

Table 3.16 – Average execution times of the strategies, presented in milliseconds.

Strategy	Average 1	Average 2	Average 3	Regression 1	Regression 2	Economic Analysis	Simple Metalearner	Weighted Metalearner	AMES	SA Q-Learning
	AVG 1	AVG 2	AVG 3	REG 1	REG 2	ECONO	META 1	META 2	AMES	SAQL
Average Execution Time	12,46	15,59	11,25	26,95	12,36	7305,72	152,00	190,00	4587,21	5365,89
Strategy	Composed Goal Directed	Adapted Derivative Following	Market Price Following			Neural Network	Determinism Theory	Game Theory	Error Theory	Players Profiles
	CGD	ADF	MPF			NN	DETERM	GTHE	ERROR	PLAYERS
Average Execution Time	356,39	256,12	80,00	Effectiveness Percentage Preference	0	10552,88	2373,20	3223,65	10986,43	1625,87
					20	11343,61	53635,20	58825,61	11890,48	11449,29
					50	11919,50	57801,26	62386,67	12302,90	11970,10
					80	12112,63	60346,61	64388,02	12912,31	12205,85
					100	13289,64	60830,44	66043,77	13618,86	13346,92

As presented in Table 3.16, from the strategies that present no possibility of adapting its processing times, the most critical strategies from an execution time point of view are the Economic Analysis, AMES and SA Q-Learning. The other strategies present very fast execution times. Regarding the strategies for which the processing time reduction is applicable, it is notable the difference between the Determinism Theory and the Game Theory strategies and the others; both in terms of maximum execution time, in the case of 100% preference for effectiveness; and also regarding the minimum time, in the case of 100% preference for the efficiency. This is due to these methods utilization of the Players Profiles Definition mechanism, where the exclusion of the slower methods, maintaining only the Prolog based approaches, is highly noticed. The similarly between the execution times of the NN and the Error Theory strategies is easily explicable. Besides the Error Theory's utilization of the NN to forecast the expected error, the execution time of its additional processing is not very relevant.

The tests of the Efficiency/Effectiveness Management mechanism are performed by feeding this mechanism with the verified execution times, and value of difference between prediction and the verified reality, of each strategy in each execution. The mechanism then analyses the received values and generates the request for each strategy, which may be one of the following: excluding the strategy (highlighted in red in the result tables), performing a big reduction in execution times (highlighted in orange), performing a small reduction (highlighted in yellow), or ignoring it, *i.e.*, accepting the current execution time (highlighted in green). In all the test cases, the presented number of executions is dependent on the stabilization of the mechanism outputs. The reference value of MASCEM's execution time without considering ALBidS is 4530 milliseconds (the average execution time observed when running a simulation in MASCEM, using the test scenario, without using ALBidS). Table 3.17 presents the results for the case with 0% preference for effectiveness.

Table 3.17 – Results of the *2E Management* mechanism with 0% preference for effectiveness.

		Strategies															
Iteration		AVG 1	AVG 2	AVG 3	REG 1	REG 2	DETERM	ECONO	GTHE	META 1	META 2	AMES	ERROR	SAQL	CGD	ADF	MPF
1	efficiency	10,66	12,98	13,49	22,94	12,85	61774,07	6582,86	67095,82	114,38	167,76	3627,16	11527,50	5644,10	397,85	301,15	68,61
	effectiveness	0,96	1,19	0,70	1,06	0,72	0,01	0,41	0,74	0,31	0,09	1,13	0,01	1,06	2,13	2,75	3,84
2	efficiency	14,17	13,74	12,83	31,00	14,26				149,89	214,57	5111,64			380,35	281,18	70,47
	effectiveness	1,22	1,17	1,55	1,73	1,44				0,31	0,09	1,21			2,18	1,79	2,91
3	efficiency	10,67	19,45	9,06	32,94	14,43				116,68	198,71				308,66	257,62	71,16
	effectiveness	1,31	1,45	1,52	1,50	1,09				0,36	0,10				2,41	2,69	2,92

Analyzing Table 3.17 it is visible that, for a full preference on the efficiency of the simulation, the mechanism excluded all the strategies with execution times above the MASCEM reference right on the first execution. The

AMES strategy presented an execution time below the MASCEM reference on the first execution, and therefore its execution time was ignored. However, in the second execution the time was higher, what led to this strategy's exclusion as well. A total of six strategies were excluded in this case, being left only the ones with a very high performance in terms of execution times. Table 3.18 presents the results for the case with 20% preference for effectiveness.

Table 3.18 – Results of the *2E Management* mechanism with 20% preference for effectiveness.

		Strategies															
Iteration		AVG 1	AVG 2	AVG 3	REG 1	REG 2	DETERM	ECONO	GTHE	META 1	META 2	AMES	ERROR	SAQL	CGD	ADF	MPF
1	efficiency	10,66	12,98	13,49	22,94	12,85	61774,07	6582,86	67095,82	114,38	167,76	3627,16	11527,50	5644,10	397,85	301,15	68,61
	effectiveness	0,96	1,19	0,70	1,06	0,72	0,01	0,41	0,74	0,31	0,09	1,13	0,01	1,06	2,13	2,75	3,84
2	efficiency	10,24	17,39	13,70	22,59	10,93		7892,85		180,68	223,69	5028,30	10125,82	6228,96	420,99	217,62	99,24
	effectiveness	1,34	0,79	0,83	1,12	0,77		0,36		0,30	0,11	1,32	0,01	1,54	2,61	2,53	2,65
3	efficiency	14,21	16,41	9,61	23,32	10,07		9094,63		153,86	235,74	5216,47	12930,71	4851,45	325,50	246,92	98,52
	effectiveness	1,16	1,39	1,30	1,41	1,60		0,43		0,25	0,08	1,47	0,01	1,01	2,54	1,80	2,56

For a case with still very high preference for the efficiency, it is visible that the two strategies with higher execution times are excluded on their first execution. The third higher execution time, presented by the Error Theory strategy, which is still much lower than the two others, is asked to perform a big reduction in the execution time. It was not excluded because of its lower time compared to the other two, combined with its very good performance in what concerns the effectiveness of the method. Regarding the SA Q-Learning and the Economic Analysis strategies, they are asked to perform a small reduction in execution times. Both these strategies are not prepared to do so and therefore the request is ignored.

In the second execution, the Error Theory strategy is once again asked to reduce its execution time by a big value, together with the Economic Analysis, which could not perform the small reduction that was required, and presented worst results in the second execution; therefore it was now asked to perform a big reduction. The SA Q-Learning strategy received once again a small reduction request, as well as the AMES strategy, which performed worst in this execution, and surpassed the MASCEM reference time.

The third execution dictated the exclusion of the Economic Analysis and Error Theory strategies, as they both presented worst execution times than in the previous executions, being the two main contributors to the global execution time degradation. The SA Q-Learning and AMES strategies keep receiving a request to lower their executions times by a small value, but since none of them is able to do so, and being both just a bit above the MASCEM reference, this request is kept being sent over time, but does not lead to a exclusion of these strategies.

Concerning the results for the case with 50% preference for efficiency/effectiveness, they are presented in Table 3.19.

Table 3.19 – Results of the *2E Management* mechanism with 50% preference for effectiveness.

		Strategies															
Iteration		AVG 1	AVG 2	AVG 3	REG 1	REG 2	DETERM	ECONO	GTHE	META 1	META 2	AMES	ERROR	SAQL	CGD	ADF	MPF
1	efficiency	10,66	12,98	13,49	22,94	12,85	61774,07	6582,86	67095,82	114,38	167,76	3627,16	11527,50	5644,10	397,85	301,15	68,61
	effectiveness	0,96	1,19	0,70	1,06	0,72	0,01	0,41	0,74	0,31	0,09	1,13	0,01	1,06	2,13	2,75	3,84
2	efficiency	12,04	16,80	11,53	33,63	9,61	46005,48	5693,74	53605,22	155,76	144,55	5120,49	11339,75	5420,25	391,07	273,67	96,80
	effectiveness	1,36	0,90	0,83	1,10	0,72	0,05	0,36	2,15	0,32	0,10	0,95	0,01	1,24	1,91	1,88	2,48
3	efficiency	10,05	12,59	11,69	33,57	10,59	2379,61	7733,29	3329,17	183,60	207,21	3577,47	11829,33	5990,74	389,66	196,39	66,39
	effectiveness	1,13	1,15	0,88	1,24	1,52	1,51	0,50	2,12	0,26	0,11	0,92	0,01	1,09	2,09	2,82	3,50

The third test, with the same importance for efficiency and effectiveness originated, in the first execution, a reduction request for only the three strategies presenting the highest processing times. A big reduction request has been issued to the two higher: Determinism and Game Theory strategies; and a small reduction request to the third, Error Theory strategy.

In the second execution the three accomplished the requests, but it was not enough for the 2E Management mechanism, which asked these strategies exactly the same thing as in the previous execution.

The third execution resulted on the acceptance of the Determinism and Game Theory strategies' execution times, as they were highly reduced. The Error Theory strategy kept receiving the request of small reduction, but it is not able to reduce much more, and therefore this request kept being made, with no result on exclusion. Note that if a strategy receives the same request for a repeated number of iterations and it is not capable of accomplishing that request, that does not mean that the strategy is excluded. The results are analysed for each iteration, and therefore if the 2E Management mechanism finds that the strategy's execution time is no longer suitable, it will exclude it. Otherwise, by sending a small or big reduction request, the mechanism is indicating that even though the execution time should be decreased, that time is not bad enough for it to be excluded.

The final results for this case were the big reduction in the execution times of two strategies, and a small reduction of one. Table 3.20 presents the results for the case with 80% preference for effectiveness.

Table 3.20 – Results of the *2E Management* mechanism with 80% preference for effectiveness.

		Strategies															
Iteration		AVG 1	AVG 2	AVG 3	REG 1	REG 2	DETERM	ECONO	GTHE	META 1	META 2	AMES	ERROR	SAQL	CGD	ADF	MPF
1	efficiency	10,66	12,98	13,49	22,94	12,85	61774,07	6582,86	67095,82	114,38	167,76	3627,16	11527,50	5644,10	397,85	301,15	68,61
	effectiveness	0,96	1,19	0,70	1,06	0,72	0,01	0,41	0,74	0,31	0,09	1,13	0,01	1,06	2,13	2,75	3,84
2	efficiency	10,04	12,71	8,62	26,76	12,52	58348,81	7891,23	52908,25	134,86	169,75	4776,99	8912,56	4855,49	267,98	235,67	86,94
	effectiveness	1,24	0,92	0,98	1,41	1,52	0,08	0,39	2,22	0,36	0,10	1,10	0,01	1,18	2,50	2,70	2,47
3	efficiency	10,41	16,74	13,53	30,03	15,19	56932,85	6204,50	3298,95	185,09	177,03	3577,07	10386,23	4729,49	424,31	259,55	86,74
	effectiveness	1,00	1,40	1,39	1,84	1,82	0,39	0,47	2,87	0,27	0,09	1,22	0,01	1,42	2,27	2,05	3,80

The case with 80% preference for effectiveness only affected the two highest values of execution times, which detach highly from the rest. The Determinism strategy is asked to perform a small reduction, while the Game Theory strategy is asked to perform a big one. Although the two present similar processing times, the request is different because of these two strategies' difference in terms of effectiveness results. The Determinism strategy is presenting very good results in terms of effectiveness; hence it is less penalized than the Game Theory strategy.

In the second execution, the requests are the same as the previous, even though both strategies accomplished the initial request. Only in the third execution, with an evident reduction of the Game Theory strategy's execution time, the mechanism accepted this strategy's time. The Determinism strategy keeps receiving small reduction requests. The final test, concerning the case with 100% preference for effectiveness is presented in Table 3.21.

Table 3.21 – Results of the *2E Management* mechanism with 100% preference for effectiveness.

		Strategies															
Iteration		AVG 1	AVG 2	AVG 3	REG 1	REG 2	DETERM	ECONO	GTHE	META 1	META 2	AMES	ERROR	SAQL	CGD	ADF	MPF
1	efficiency	10,66	12,98	13,49	22,94	12,85	61774,07	6582,86	67095,82	114,38	167,76	3627,16	11527,50	5644,10	397,85	301,15	68,61
	effectiveness	0,96	1,19	0,70	1,06	0,72	0,01	0,41	0,74	0,31	0,09	1,13	0,01	1,06	2,13	2,75	3,84

The case with 100% preference for effectiveness determines that all strategies' execution times are adequate. The two most relevant cases in terms of degradation of execution times present good effectiveness results, which is the most important factor for this case, therefore they present no issue to the *2E Management* mechanism.

The results presented in these tests demonstrate the performance of the 2E Management mechanism in adapting the strategies' usage to different situations in what concerns the efficiency/effectiveness balance. This is done with the purpose of degrading as less as possible the ALBidS results, while adapting the execution times. The outputs of this mechanism affect both the performance of the strategies, as well as the performance of ALBidS. Table 3.22 presents the average execution time of the MASCEM simulator for one negotiation period, considering the degradation caused by ALBidS, in the same considered cases of efficiency/effectiveness preference.

Table 3.22 – Average execution times of the MASCEM, presented in milliseconds.

	Without ALBidS	ALBidS with 0% effectiveness	ALBidS with 20% effectiveness	ALBidS with 50% effectiveness	ALBidS with 80% effectiveness	ALBidS with 100% effectiveness
Execution Time	4530	4880,26	13183,42	21156,2	69283,45	76971,04

Table 3.22 shows that, for the case of 0% preference for effectiveness, the system's degradation is almost irrelevant, and that as the percentage increases, the average execution time increases as well. These results show that the 2E Management mechanism is able to accomplish its purpose adequately.

Regarding the influence of this management in what concerns the effectiveness of the ALBidS system, a case study was performed, with ALBidS supporting a player when negotiating in the electricity market, under different efficiency/effectiveness preferences. This case study is presented in the global case studies chapter (Chapter 4), namely in section 4.3 – Contexts and Efficiency/Effectiveness balance.

3.10. Final Remarks

This chapter presented ALBidS, a multiagent system that provides decision support to electricity market negotiating players. ALBidS is integrated with the MASCEM simulator, which provided the means for testing and suiting ALBidS' implemented tools and mechanisms that were presented throughout this chapter.

ALBidS' design, architecture, and integration with the MASCEM simulator proved to be adequately implemented, as showed by the several tests presented in this chapter. This is supported by a wider range of tests, which are presented in Chapter 4.

Regarding the several different approaches to provide actions' proposals for the Main Agent to choose from, their individual presentation and testing proved their independent adequacy to their purposes, each showing to be, on its own, an important added value to the ALBidS system. The wide range of approaches (from mathematics to physics, from power systems' applications to artificial intelligence techniques) guarantees that at each moment responses based different views are suggested, offering a high variety of possible actions for the ALBidS system to consider. This provides the means for the system to increase its chances of always be provided with at least one adequate answer, even when the majority of the approaches fails in the accomplishment of its purposes.

This fact is extended to the other mechanisms which complement the system, *i.e.* the Context Analysis mechanism, the Player Profile Definition mechanism, and the Efficiency/Effectiveness Management mechanism. Once again, it is important to accentuate the importance of the case studies presented in Chapter 4, in their purpose of supporting the influence and advantage of these mechanisms, by displaying further tests with the whole ALBidS system's mechanisms fully integrated.

Finally, regarding the main entity in the ALBidS system – the Main Agent, it also proved to efficiently accomplish its purposes, showing its ability to choose the most appropriate actions for the supported player to take, by using the reinforcement learning algorithms at its disposal, and taking advantage on the complementary mechanisms that aim to enlarge ALBidS' capabilities of adaptation and providing of intelligent decision support.

The ALBidS system as a whole is tested and challenged in real data based scenarios, which are presented in Chapter 4. These case studies aim at guaranteeing the adequate coordination between the several mechanisms of this system, and proving its suitability in solving the issues it is designed to overcome.

CHAPTER 4. CASE-STUDYING

4.1. Introduction

This chapter presents three case studies, which intend to assess the adequacy of the ALBidS system. These case studies approach the methodologies and strategies presented in the scope of this work.

Although every developed methodology was tested independently, with the resulting experimental findings presented in each correspondent section, it was found essential to exhibit some final case studies, concerning the full inclusion of all the mechanisms in the ALBidS system.

These case studies regard those that were considered to be the most relevant features to be shown, *i.e.* the matters whose individual tests were not considered to be expressive enough to show their influence, or those whose effective results could not be shown apart from a full market simulation. These are, therefore, complementary tests to all those that have been presented throughout Chapter 3. The purpose is to provide a more global perspective of ALBidS as a fully integrated system.

The first case study intends to demonstrate the advantage of a market player's use of ALBidS support. This is shown by running a market simulation scenario with a player using ALBidS, and repeating the same exact simulation with the same player using other strategies. The comparison of the results shall allow taking conclusions on the advantages of using ALBidS.

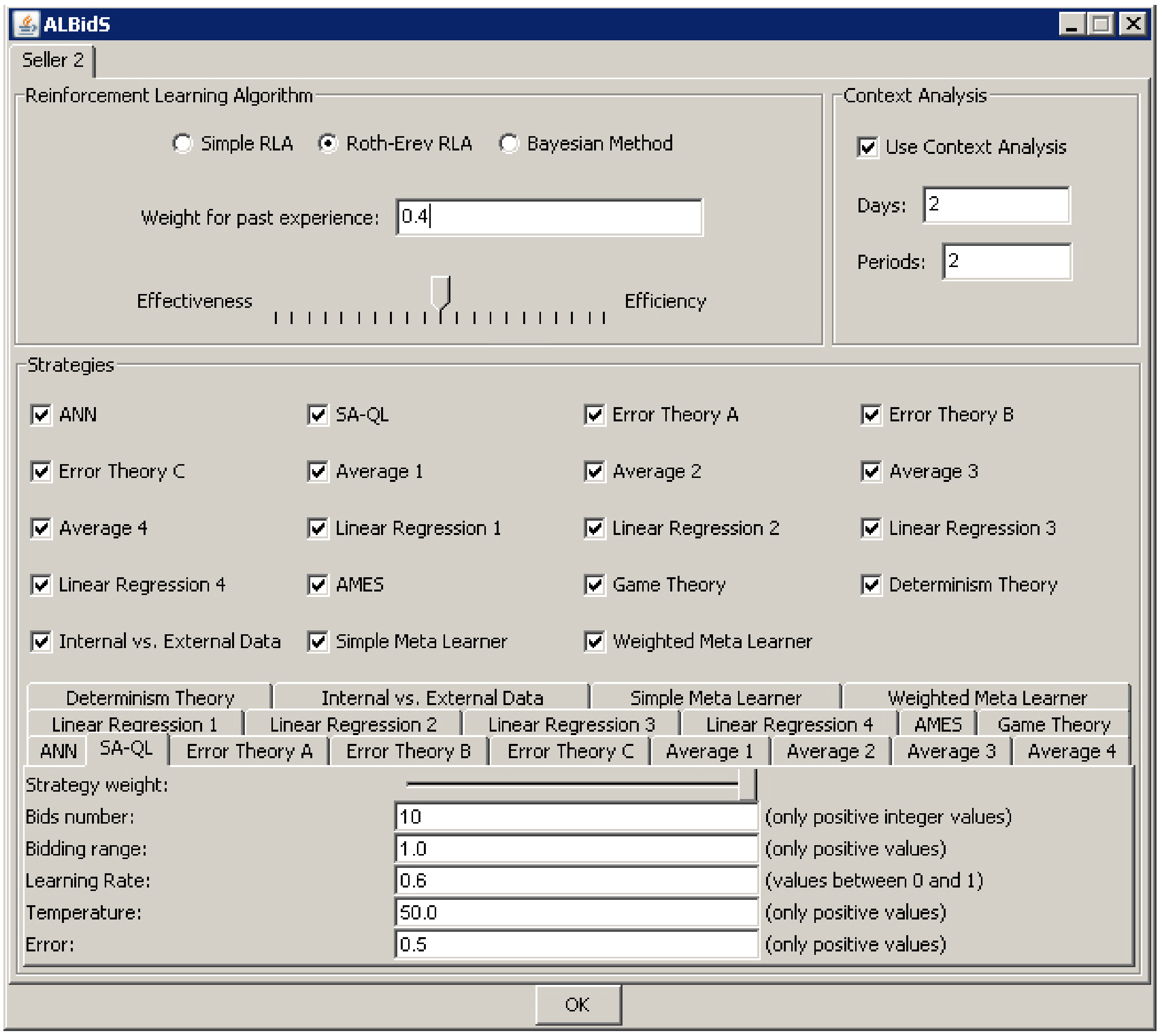
Concerning the second case study, the benefit of using contexts definition is analyzed by comparing simulations' results using ALBidS with and without the use of the Context Definition mechanism. The use of the Efficiency/Effectiveness Management mechanism is also approached in this case study, comparing the gains when using ALBidS with a 100% preference for effectiveness, with the advantage in terms of execution time when using this system with a full preference for the efficiency.

Finally, the ALBidS' performance when confronted with intelligent competition in the market is tested in the third case study. This test's results aim to support ALBidS competence in providing decision support, by adapting to different situations, not only on simpler cases, and when competing against weak opponents, but also when dealing with intelligent competition.

4.2. Case Study 1 – ALBidS performance in the market

In this case study three MASCEM simulations are presented, referring to 14 consecutive days, starting from Wednesday, 29th October, using the test scenario.

Seller 2, which is the test subject, will use a different strategy in each of the three simulations. In the first simulation it will use the Neural Network as strategy for the bid definition. In the second it will use a statistical approach, a regression on the data of the last 5 business days. Finally, in the third simulation, it will use the ALBidS system for decision support. The reinforcement learning algorithm selected for this simulation has been the revised Roth-Erev, with equal value of the algorithms weight, and a past experience weight *W* value of 0.4, a small value to grant higher influence to the most recent results, so that it can quickly learn and catch new tendencies, since the market is always changing fast. Regarding the efficiency/effectiveness management, the chosen value of preference for the efficiency is 50%, meaning equal importance for the two. Figure 4.1 presents the definitions for Seller 2's use of ALBidS (regarding the third simulation).


ALBidS

Seller 2

Reinforcement Learning Algorithm

Simple RLA [✓ Roth-Erev RLA] Bayesian Method

Weight for past experience: 0.4

Effectiveness — Efficiency

Context Analysis

✓ Use Context Analysis

Days: 2

Periods: 2

Strategies

✓ ANN	✓ SA-QL	✓ Error Theory A	✓ Error Theory B
✓ Error Theory C	✓ Average 1	✓ Average 2	✓ Average 3
✓ Average 4	✓ Linear Regression 1	✓ Linear Regression 2	✓ Linear Regression 3
✓ Linear Regression 4	✓ AMES	✓ Game Theory	✓ Determinism Theory
✓ Internal vs. External Data	✓ Simple Meta Learner	✓ Weighted Meta Learner	

Determinism Theory | Internal vs. External Data | Simple Meta Learner | Weighted Meta Learner

Linear Regression 1 | Linear Regression 2 | Linear Regression 3 | Linear Regression 4 | AMES | Game Theory

ANN | SA-QL | Error Theory A | Error Theory B | Error Theory C | Average 1 | Average 2 | Average 3 | Average 4

Strategy weight:		
Bids number:	10	(only positive integer values)
Bidding range:	1.0	(only positive values)
Learning Rate:	0.6	(values between 0 and 1)
Temperature:	50.0	(only positive values)
Error:	0.5	(only positive values)

OK

Figure 4.1 – ALBidS definitions for Seller 2.

From Figure 4.1 it is visible that all ALBidS' strategies are selected, therefore all will be used at first, being the Efficiency/Effectiveness Management mechanism responsible for managing which will continue to be used, and which will be excluded or have the execution time reduced. Additionally, the Context Analysis mechanism is also being used for this simulation.

After the simulations, the incomes obtained by Seller 2 can be compared using each of the three tested strategies. This allows to take conclusions on the use of ALBidS over individual strategies, and also to understand in what ALBidS based the choosing of its bids. Seller 2 available offer capacity is kept constant at 550MW for each period throughout the simulations.

As in the individual experimental findings of the strategies, the evolution of the performance will be analyzed for each period individually. Figure 4.2 presents the evolution of Seller 2 incomes in the first period of the day, along the 14 considered days. Figure 4.2 a) presents the results for the first simulation, Figure 4.2 b) presents the incomes of Seller 2 for the second simulation, and Figure 4.2 c) presents the incomes for the third simulation (Seller 2 using ALBidS).

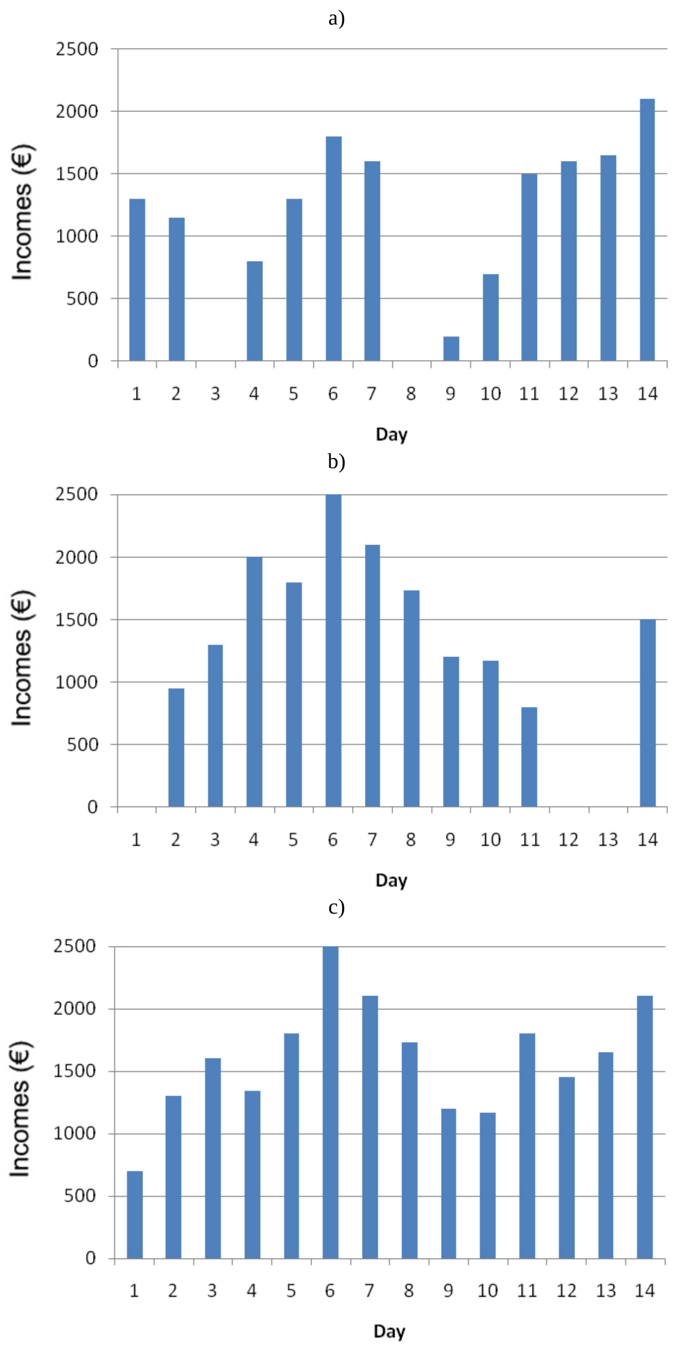


Figure 4.2 – Incomes obtained by Seller 2 in the first period of the considered 14 days, using: a) the regression on the data of the last 5 business days, b) the Neural Network, c) the ALBidS system [Vale *et al.*, 2011a].

Comparing the charts presented in Figure 4.2, it can be seen that the third simulation was clearly the most profitable for Seller 2.

In the first day the income using the ALBidS system is located below the value of the income using the regression approach. That is because the confidence values for all algorithms have been initialized with equal values, and so the selection of the answer is made by chance. The selected algorithm answer originated a low income.

In the second day, after the reinforcement learning algorithm is updated, it chooses the strategy that had the greater success in the first day. Obviously the one that got the best reward, and it is visible that the income is located above both the other comparison strategies, as well as it happens on the third day.

In the fourth day its value is below the results of the NN; the algorithm is still selecting the other algorithm that got the best results in the first 3 days.

In the fifth day the value is equal to the NN, and it follows that trend until day 10, in spite of its lower values in days 9 and 10. In day 11 the chosen algorithm is no longer the NN, as the third simulation's value is higher than the ones of the other two considered strategies. Other algorithm passed the NN as the best result of confidence as a result of the NN worst results in days 9 and 10.

In the last two days of the simulation, the selected algorithm is the regression, catching its high value tendency still on time to get two good final results. This is due to the low weight value of past experience; otherwise this algorithm would not be chosen, for its weaker performance in the first days.

Figure 4.3 presents the total amount of incomes obtained by Seller 2 in this period, using each of the three approaches.

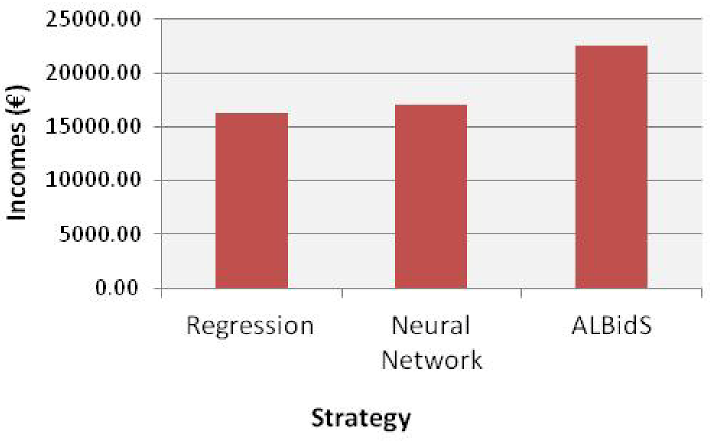
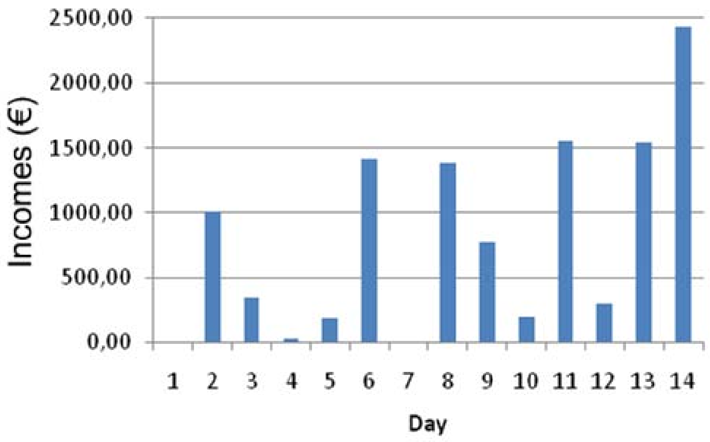


Figure 4.3 – Total incomes obtained by Seller 2 in the first period using each of the three approaches.

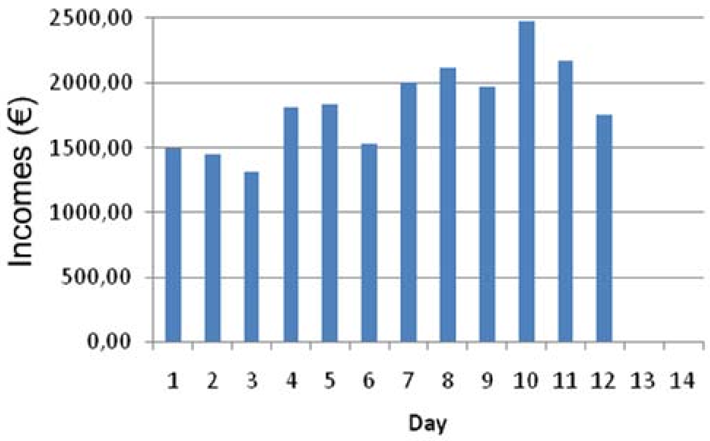
From Figure 4.3 it is evident that from the considered approaches, ALBidS was the one that, by far, was able to provide the best results for Seller 2 in the considered period.

Figure 4.4 presents the evolution of Seller 2 incomes in the thirteenth period of the day (to represent a period in the middle of the day, along the same 14 days, using each of the three considered methods.

a)



b)



c)

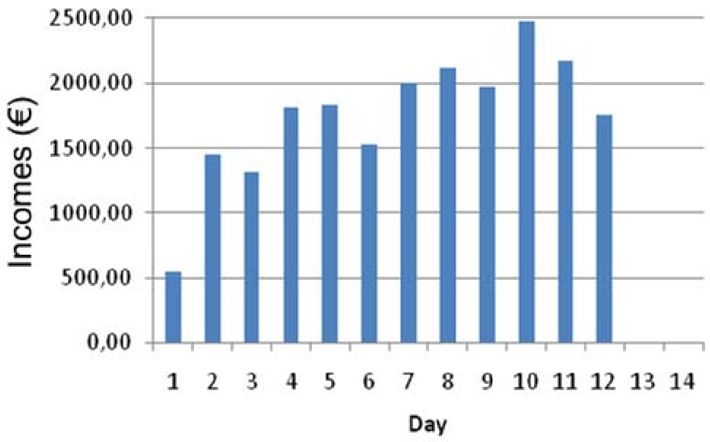


Figure 4.4 – Incomes obtained by Seller 2 in the thirteenth period of the considered 14 days, using: a) the regression on the data of the last 5 business days, b) the Neural Network, c) the ALBidS system [Pinto *et al.*, 2011c].

Comparing Figure 4.4 a) and b), it is visible that the NN approach was clearly more profitable than the regression. This suggests that ALBidS is expected to choose more often the NN results over the regression.

Analysing the chart presented in Figure 4.4 c), it is visible that in the first day the income using ALBidS is located below the value of the income using the NN, and above the regression. That is once again because of the similar initialization of the strategies' stats, and consequent random selection of the answer. The selected agent's answer originated a low income, which was not from either the NN or the regression's responses.

In the second day ALBidS chose the suggestion of the algorithm that got the best reward, and it can be seen that the income is equal to the one obtained by the NN for the same day. From this day forward the selected values remained always equal to the NN's proposals, taking advantage on this method's good results along the days for this period. This selection is verified even in the last two days, when the NN's results were null and the regression's higher. This is because of the value of confidence that the NN gained in the reinforcement learning algorithm over the passage of the days, due to its overall good performance. In spite of the regression getting good results in the last two days, it was not enough for it to overcome the NN confidence, even with the low value of the past experience weight attributed to the learning algorithm, to grant higher influence to the most recent results.

Figure 4.5 presents the total amount of incomes obtained by Seller 2 in the thirteenth period, for the fourteen considered days, using each of the three approaches.

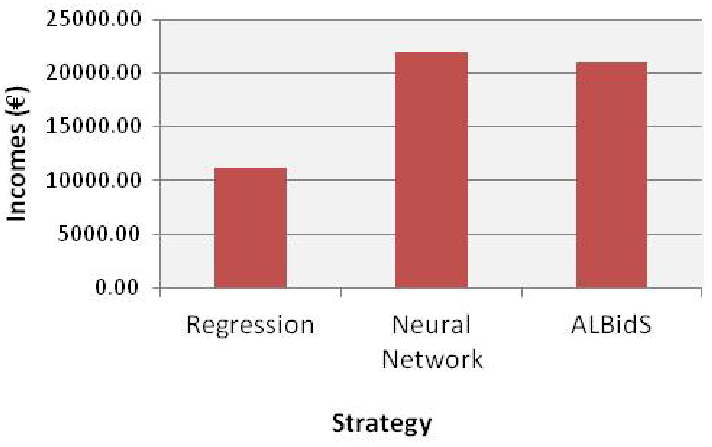


Figure 4.5 – Total incomes obtained by Seller 2 in the thirteenth period using each of the three approaches.

As presented in Figure 4.5 the simulation using ALBidS resulted on very similar results concerning the incomes of Seller 2, comparing to the one where the NN is used. This is due to what was analysed before, ALBidS having supported its decisions on the NN's suggestions throughout all days except for the first day. The first day, in which no data was still available for ALBidS' to support its choosing, was the reason behind the small discrepancy between the both simulations total results for this period.

Finally, the total amount of incomes obtained by Seller 2 considering all the 24 periods of the 14 considered days, can be compared, as presented in Figure 4.6.

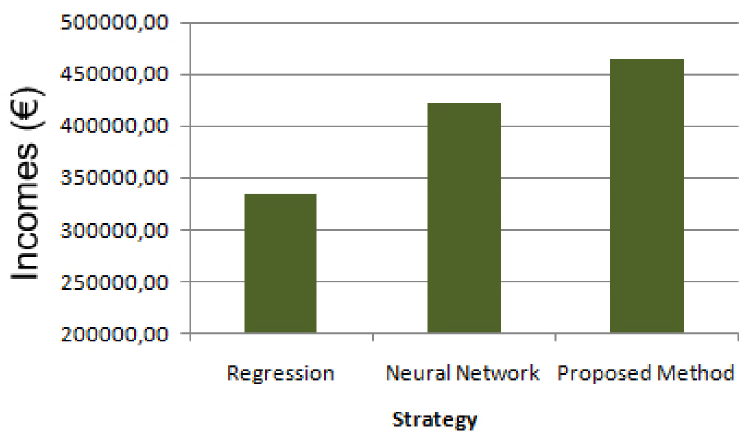


Figure 4.6 – Total incomes obtained by Seller 2 for the considered 14 days [Pinto *et al.*, 2011a].

As showed by Figure 4.6, the proposed method for strategy formulation obtained a higher amount of incomes in the total of the 14 days when compared to the other two considered strategies.

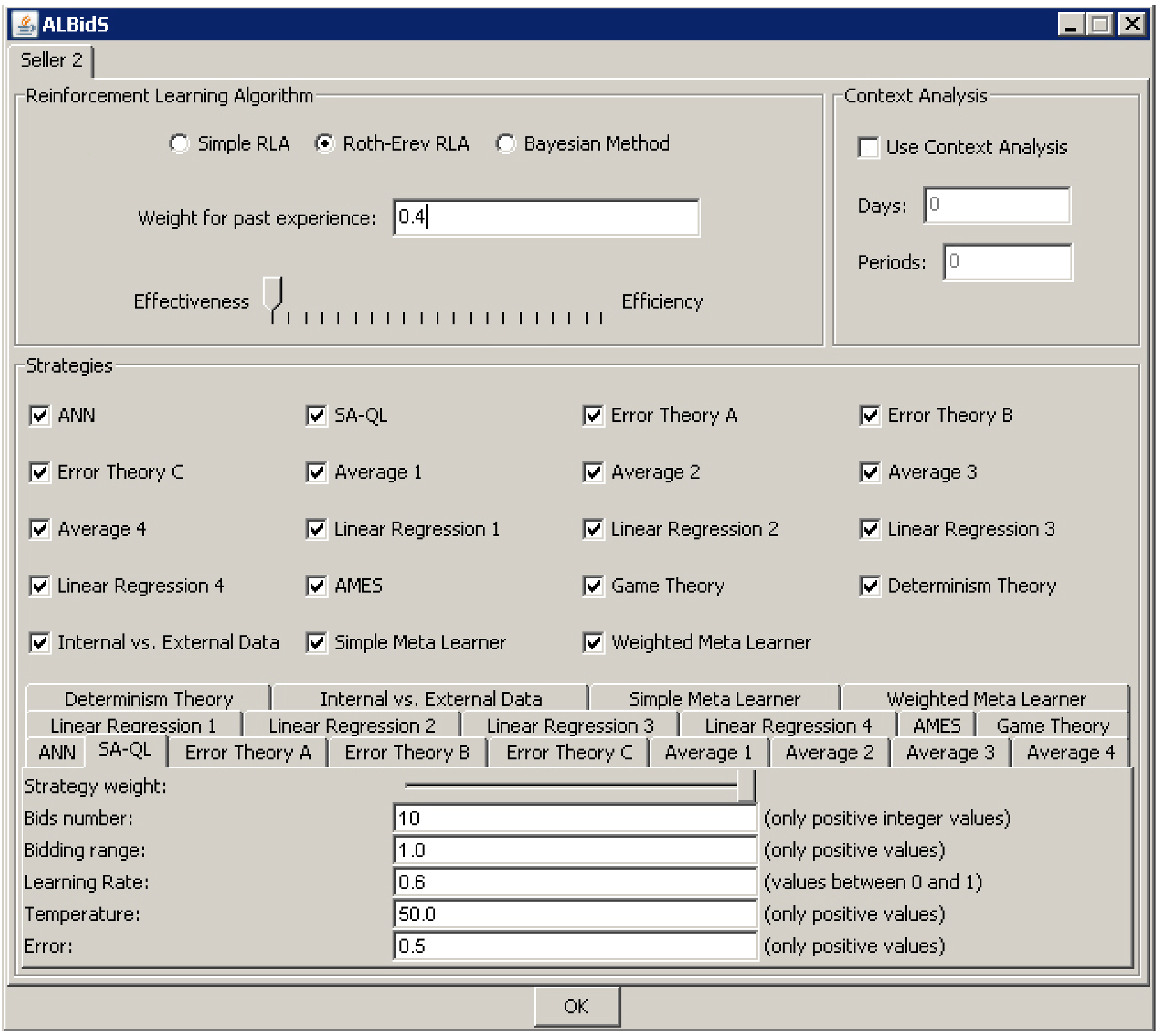
Further details of this case-study can be consulted at http://www.mascem.com/[4].

4.3. Case Study 2 – Contexts and efficiency/effectiveness balance

This case study presents two main goals: testing the influence of the efficiency/effectiveness management in ALBidS, and demonstrating the importance of the context definition for the system’s performance. To do so, three simulations using the test scenario are presented. In the first one, Seller 2 uses ALBidS with no contexts and a 100% preference for effectiveness. In the second, the preference for effectiveness is maintained at its maximum, but this time using context definition. To ease the understanding of the results, the contexts are defined using only two clusters for the day analysis. This way the only separation of contexts is week days, and weekends. In the third simulation, the context definition is equal to the second, but this time the preference goes 100% for efficiency.

Figures 4.7, 4.8 and 4.9 present the definitions for Seller 2’s use of ALBidS for the first, second, and third simulations, respectively.

[4] MASCEM simulator’s home page



Reinforcement Learning Algorithm

Simple RLA [✓ Roth-Erev RLA] Bayesian Method

Weight for past experience: 0.4

Effectiveness — Efficiency

Context Analysis

Use Context Analysis

Days: 0

Periods: 0

Strategies

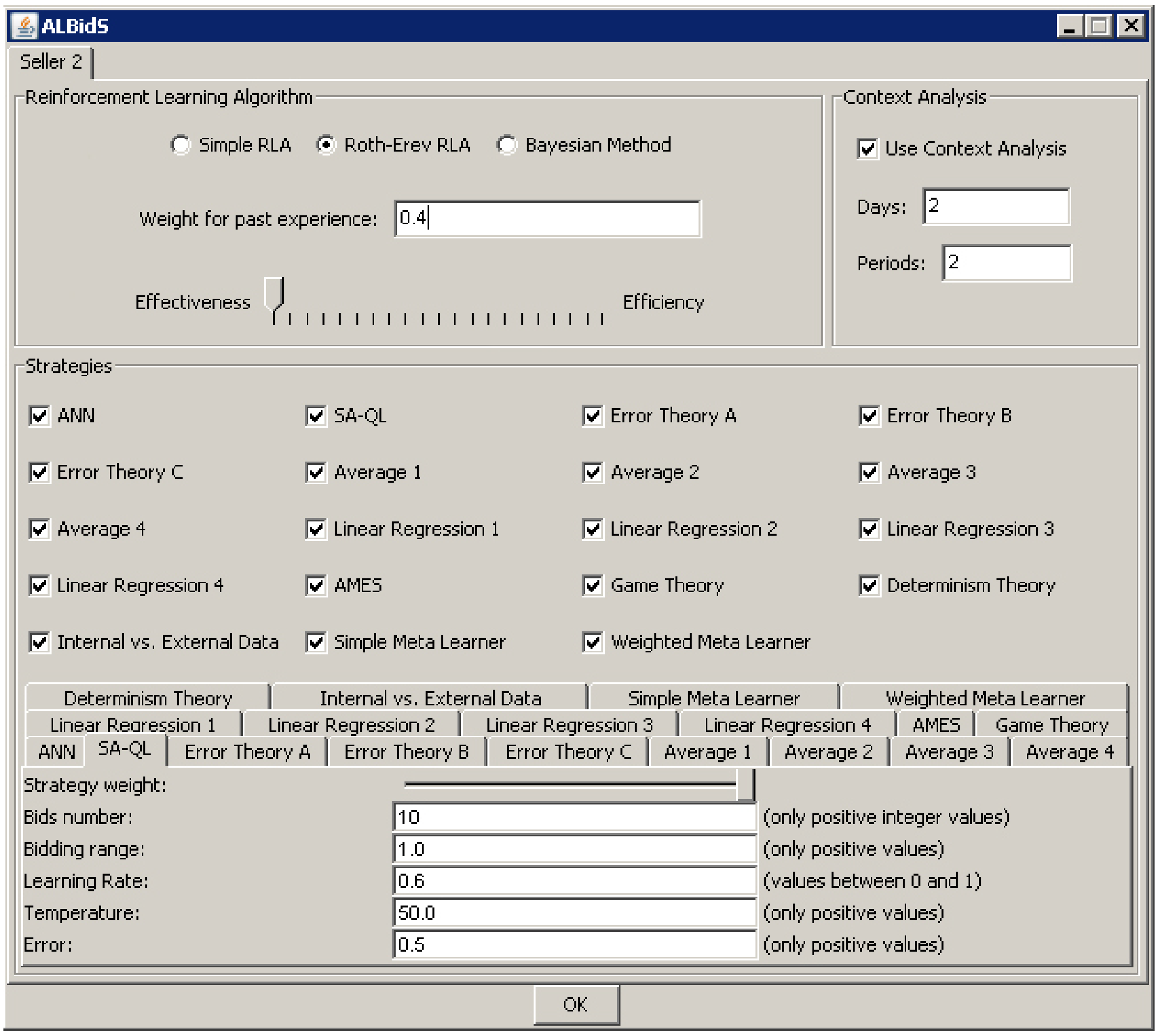
✓ ANN	✓ SA-QL	✓ Error Theory A	✓ Error Theory B
✓ Error Theory C	✓ Average 1	✓ Average 2	✓ Average 3
✓ Average 4	✓ Linear Regression 1	✓ Linear Regression 2	✓ Linear Regression 3
✓ Linear Regression 4	✓ AMES	✓ Game Theory	✓ Determinism Theory
✓ Internal vs. External Data	✓ Simple Meta Learner	✓ Weighted Meta Learner	

Determinism Theory | Internal vs. External Data | Simple Meta Learner | Weighted Meta Learner
Linear Regression 1 | Linear Regression 2 | Linear Regression 3 | Linear Regression 4 | AMES | Game Theory
ANN | SA-QL | Error Theory A | Error Theory B | Error Theory C | Average 1 | Average 2 | Average 3 | Average 4

Strategy weight:		
Bids number:	10	(only positive integer values)
Bidding range:	1.0	(only positive values)
Learning Rate:	0.6	(values between 0 and 1)
Temperature:	50.0	(only positive values)
Error:	0.5	(only positive values)

Figure 4.7 – ALBidS definitions for Seller 2, for the first simulation.

As mentioned, the first simulation does not include the Context Analysis mechanism. However, it includes all supported strategies, with the efficiency/effectiveness balance at the higher preference for effectiveness.



Reinforcement Learning Algorithm

Simple RLA [✓ Roth-Erev RLA] Bayesian Method

Weight for past experience: 0.4

Effectiveness — Efficiency

Context Analysis

[✓ Use Context Analysis]

Days: 2

Periods: 2

Strategies

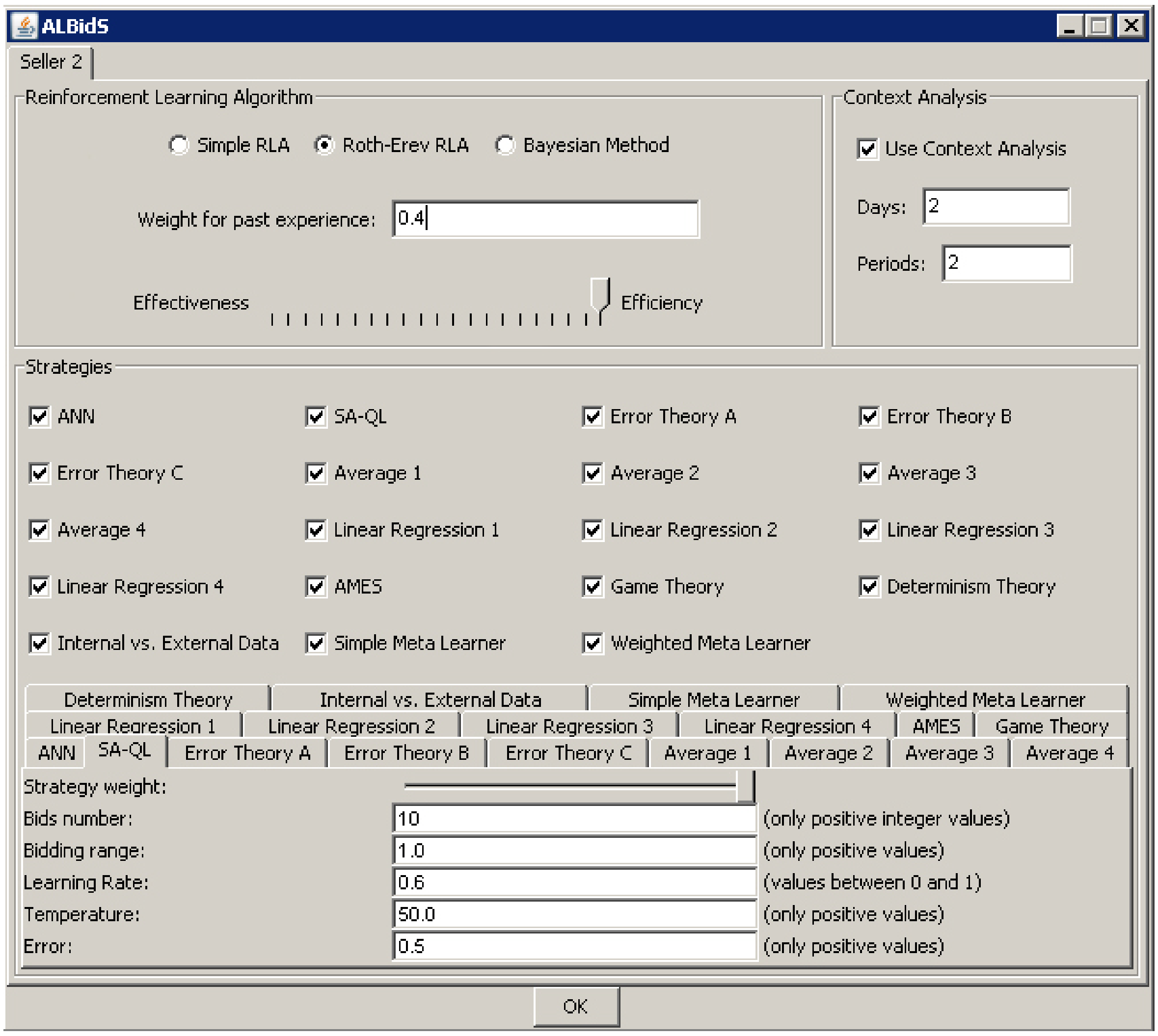
✓ ANN	✓ SA-QL	✓ Error Theory A	✓ Error Theory B
✓ Error Theory C	✓ Average 1	✓ Average 2	✓ Average 3
✓ Average 4	✓ Linear Regression 1	✓ Linear Regression 2	✓ Linear Regression 3
✓ Linear Regression 4	✓ AMES	✓ Game Theory	✓ Determinism Theory
✓ Internal vs. External Data	✓ Simple Meta Learner	✓ Weighted Meta Learner	

Determinism Theory | Internal vs. External Data | Simple Meta Learner | Weighted Meta Learner
Linear Regression 1 | Linear Regression 2 | Linear Regression 3 | Linear Regression 4 | AMES | Game Theory
ANN | SA-QL | Error Theory A | Error Theory B | Error Theory C | Average 1 | Average 2 | Average 3 | Average 4

Strategy weight:		
Bids number:	10	(only positive integer values)
Bidding range:	1.0	(only positive values)
Learning Rate:	0.6	(values between 0 and 1)
Temperature:	50.0	(only positive values)
Error:	0.5	(only positive values)

Figure 4.8 – ALBidS definitions for Seller 2, for the second simulation.

Regarding the second simulation, the ALBidS definitions are exactly the same as in the first simulation, except for the use of the Context Analysis mechanism. This mechanism considers, for this case, only the clustering by two periods and two days.



Reinforcement Learning Algorithm

Simple RLA [✓ Roth-Erev RLA] Bayesian Method

Weight for past experience: 0.4

Effectiveness — Efficiency

Context Analysis

✓ Use Context Analysis

Days: 2

Periods: 2

Strategies

✓ ANN	✓ SA-QL	✓ Error Theory A	✓ Error Theory B
✓ Error Theory C	✓ Average 1	✓ Average 2	✓ Average 3
✓ Average 4	✓ Linear Regression 1	✓ Linear Regression 2	✓ Linear Regression 3
✓ Linear Regression 4	✓ AMES	✓ Game Theory	✓ Determinism Theory
✓ Internal vs. External Data	✓ Simple Meta Learner	✓ Weighted Meta Learner	

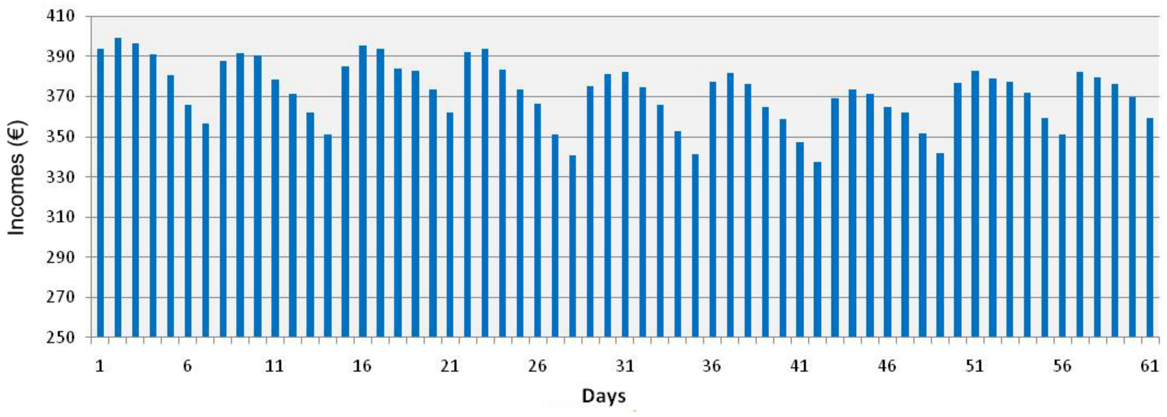
Strategy weight:		
Bids number:	10	(only positive integer values)
Bidding range:	1.0	(only positive values)
Learning Rate:	0.6	(values between 0 and 1)
Temperature:	50.0	(only positive values)
Error:	0.5	(only positive values)

Figure 4.9 – ALBidS definitions for Seller 2, for the third simulation.

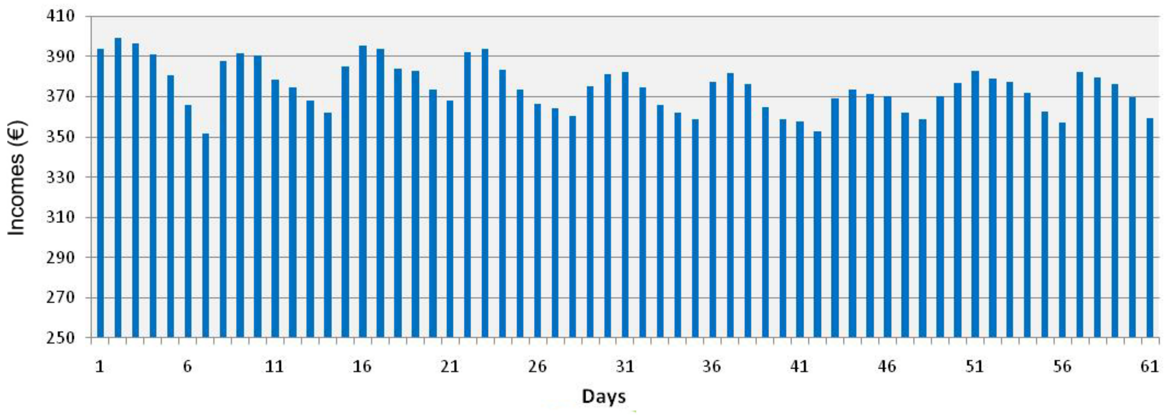
In this case the Context Analysis mechanism is active, but the efficiency/effectiveness balance preference is attributed entirely to the efficiency.

Figure 4.10 presents the incomes obtained in each of the three simulations, along 61 simulated days, for the twelfth period of the day. The starting day for these simulations is Monday October 4th, to facilitate the spotting of both, working days, and weekend days, in the graphs.

a)



b)



c)

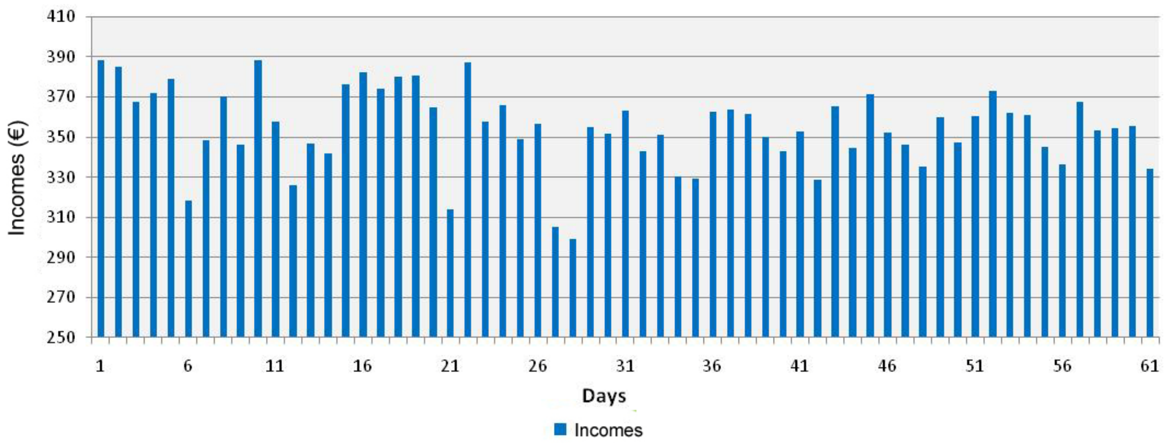


Figure 4.10 – Incomes obtained by Seller 2 in the twelfth period of the considered 61 days, in: a) the first simulation, b) the second simulation, c) the third simulation.

Analyzing the chart presented in Figure 4.10 a), the pattern concerning the weekend days results is clearly visible. The achieved incomes are always much lower on weekends than during working days. This tendency is softened slowly over the days, but never completely. This is due to the consideration of all days in the same way, not taking into account the different characteristics that weekend days present.

This is visible by analyzing Figure 4.10 b), which by considering weekend days as a different context, and consequently treating these days independently from the rest, manages to get better results on weekends, with no degradation of the results of the business days. Looking closely at this graph, it is visible that the results improvement on weekends starts on the second weekend and continues improving until the end of the simulation days. Regarding the first weekend (days 6 and 7) presents weak results, since the strategies stats for this context are still in an initial point, *i.e.*, as initialized. This is easily visible by Figure 4.11, which presents the comparison of the incomes of Seller 2 in the first and second simulations.

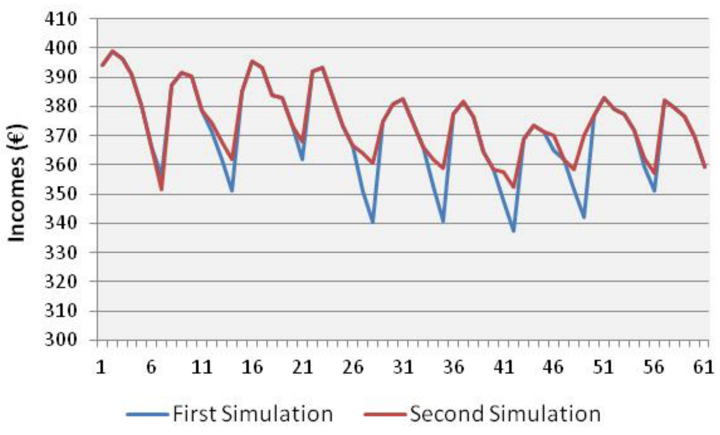


Figure 4.11 – Incomes obtained by Seller 2 in the first and second simulations, for the twelfth period of the considered 61 days.

Concerning the third simulation, since the full preference is given to the efficiency of the method and not on its effectiveness, the results are much more unstable. However, it is still possible to see in Figure 4.10, the weekends' worst results pattern in the beginning of the simulation, which is softened throughout the days, to the point that at the end it is almost imperceptible. This is due to the context definition, even with a low preference for effectiveness.

The comparison of the total incomes achieved by Seller 2 in the three simulations, in the total of the 61 days for the considered period is presented in Figure 4.12.

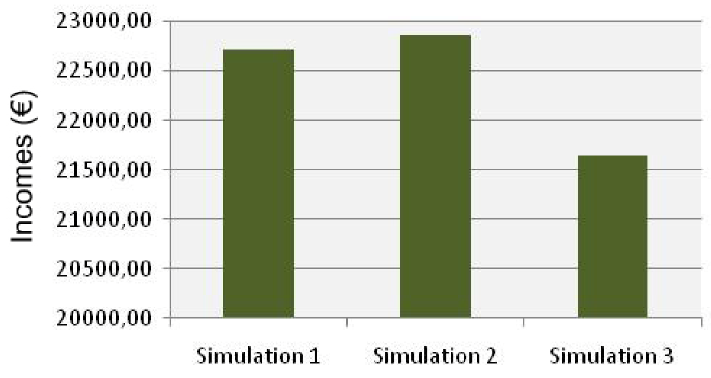


Figure 4.12 – Total Incomes obtained by Seller 2 in the twelfth period of the considered 61 days.

These results show the advantage of using context analysis to define the negotiating behaviour, since the simulation that used it – Simulation 2 is the one that achieved higher incomes. Regarding the third simulation, its preference for efficiency over effectiveness originated the achievement of worst incomes. However, this is compensated by the advantage in processing time, as presented in Table 4.1.

Table 4.1 – Average execution times of the three simulations for one negotiation period, presented in milliseconds.

	Simulation 1	Simulation 2	Simulation 3
Execution Time	80944,21	81374,63	4030,28

The results presented in Table 4.1 are clear in what concerns the difference of execution times when using ALBidS with full preference for efficiency or effectiveness, as supported by the difference between Simulation 3 and the other two.

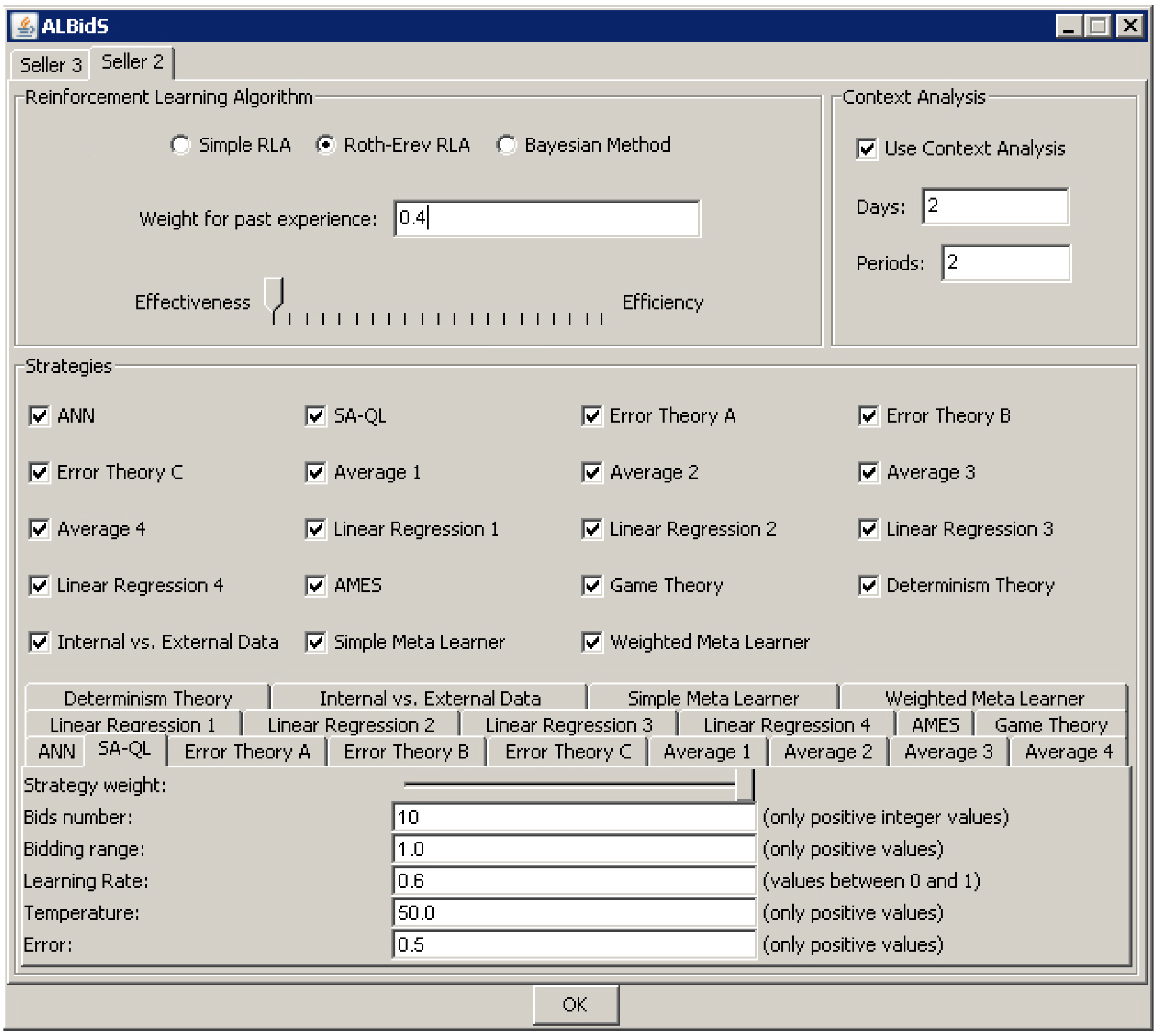
Regarding the difference between the first and second simulations, which refers to the use of the context analysis, it is minimal, taking into account the amount of the values.

This case study allowed showing the advantage of using the context definition, through the demonstration of the achievement of best incomes using this tool. It also presented the adequacy of using the efficiency/effectiveness management, to drastically decrease the simulation executing times, while still achieving good results.

4.4. Case Study 3 – Intelligent competition

This case study presents one simulation using the test scenario, with the purpose of analyzing the performance of ALBidS when competing with players with actual intelligent behaviour, since the majority of the tests done in the scope of this work use a scenario where the competing negotiating players do not present much intelligence in the definition of their bids.

An interesting way to analyze this is to watch a player using ALBidS competing with another player also supported by ALBidS. In order to do this, the test scenario is used, but this time both Seller 2 and Seller 3 use the ALBidS support, with the same characteristics and strategy parameters, and the same amount of power to sell. This way they both start in equal ground, and it is possible to see how they anticipate each other and react to the higher success of their competitor. Both players' definition is presented in Figure 4.13.


ALBidS

Seller 3 | Seller 2

Reinforcement Learning Algorithm

Simple RLA [✓ Roth-Erev RLA] Bayesian Method

Weight for past experience: 0.4

Effectiveness — Efficiency

Context Analysis

[✓] Use Context Analysis

Days: 2

Periods: 2

Strategies

[✓] ANN	[✓] SA-QL	[✓] Error Theory A	[✓] Error Theory B
[✓] Error Theory C	[✓] Average 1	[✓] Average 2	[✓] Average 3
[✓] Average 4	[✓] Linear Regression 1	[✓] Linear Regression 2	[✓] Linear Regression 3
[✓] Linear Regression 4	[✓] AMES	[✓] Game Theory	[✓] Determinism Theory
[✓] Internal vs. External Data	[✓] Simple Meta Learner	[✓] Weighted Meta Learner	

Determinism Theory | Internal vs. External Data | Simple Meta Learner | Weighted Meta Learner
Linear Regression 1 | Linear Regression 2 | Linear Regression 3 | Linear Regression 4 | AMES | Game Theory
ANN | SA-QL | Error Theory A | Error Theory B | Error Theory C | Average 1 | Average 2 | Average 3 | Average 4

Strategy weight:		
Bids number:	10	(only positive integer values)
Bidding range:	1.0	(only positive values)
Learning Rate:	0.6	(values between 0 and 1)
Temperature:	50.0	(only positive values)
Error:	0.5	(only positive values)

OK

Figure 4.13 – ALBidS definitions for Seller 2 and Seller 3.

Both players use the Roth-Erev reinforcement learning algorithm, with a weight value for past experience of 0.4. They are both supported by the Context Analysis mechanism, and by all the available strategies, with full preference for the effectiveness of the method. The simulation is performed for 61 days, and both players' bids for the twelfth period of each day are presented in Figure 4.14.

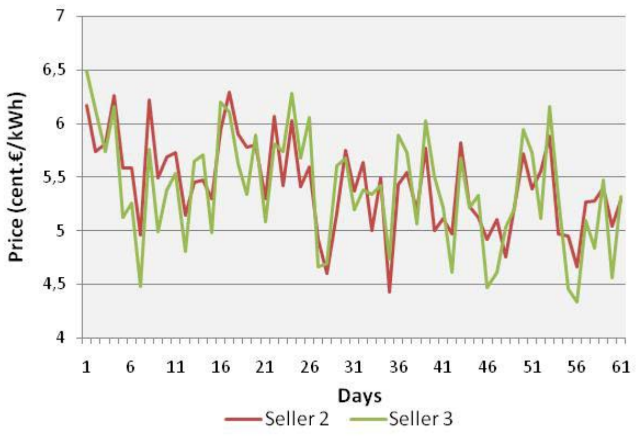


Figure 4.14 – Seller 2 and Seller 3's bids in the twelfth period of the considered 61 days.

As presented in Figure 4.14, the two players' bids are always very close, and constantly changing whose is above and whose is below. It is even difficult to analyze their performance, so many are the changes in their behaviour, to make it unpredictable, anticipating the other's actions, and reacting to the environment. In fact, in order to understand which of the two performed better, it is necessary to take a look at their obtained incomes, which are presented in Figure 4.15. Note that the range of the graph may be misleading, since the axes' range had to be small, in order to be able to properly show the differences.

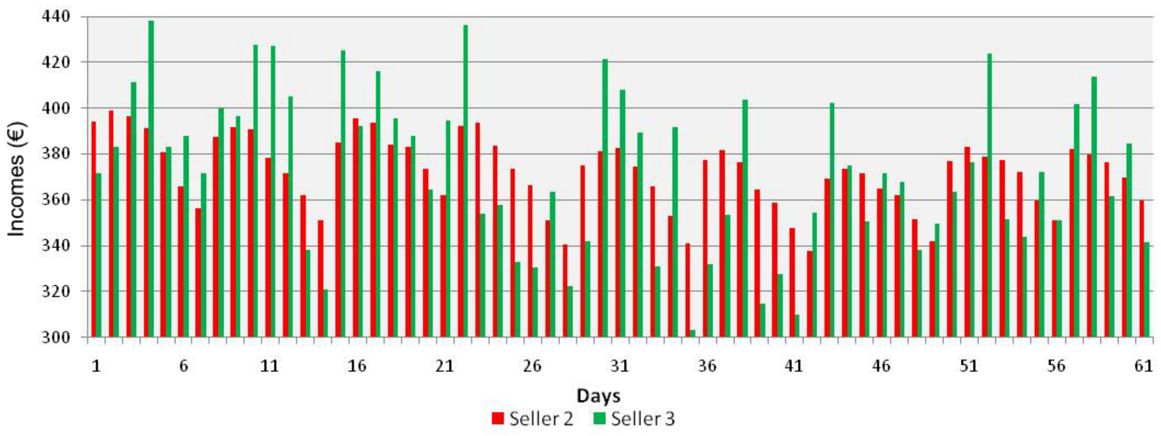


Figure 4.15 – Seller 2 and Seller 3's incomes in the twelfth period of the considered 61 days.

A better visualization of this equilibrium can be achieved by analysing Figure 4.16.

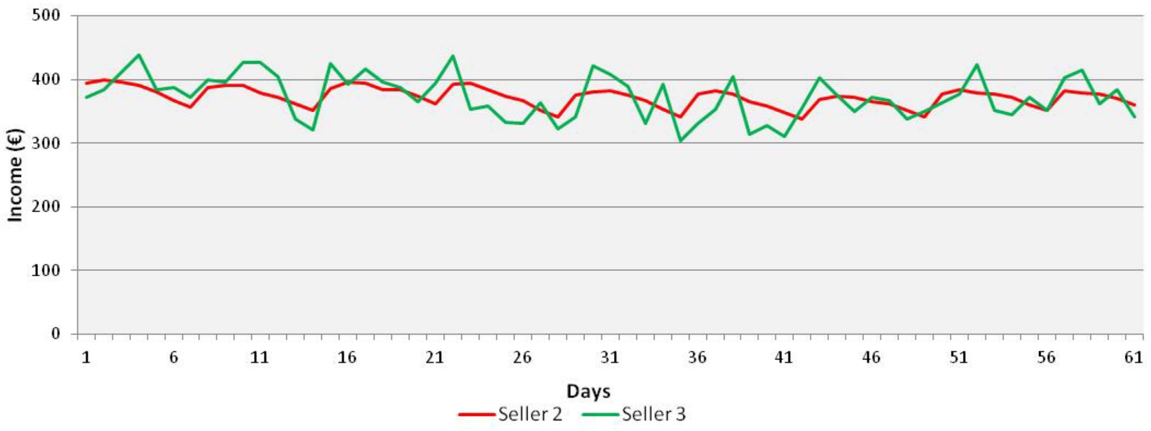


Figure 4.16 – Comparison between Seller 2 and Seller 3's incomes in the twelfth period of the considered 61 days.

As can be seen by Figure 4.16, the total incomes of both considered players concerning this period are very close, resulting from the constant variations and reaction in both players' actions. The closeness in both players' results is supported by Figure 4.17.

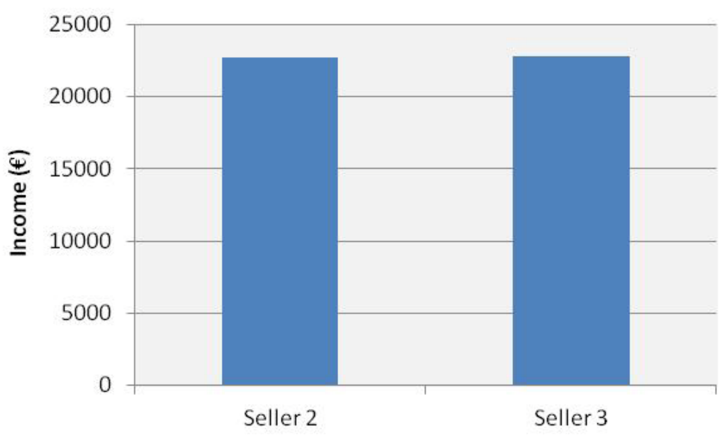


Figure 4.17 – Seller 2 and Seller 3's total incomes in the twelfth period of the considered 61 days.

From Figure 4.17 the difference between both players' achieved incomes is almost imperceptible. In fact the results for this period are as close as: Seller 2: 22,710.24 €; Seller 3: 22,756.36 €.

The results equilibrium is a constant factor, not only for the period in matter, but for all. Figure 4.18 presents the total incomes achieved by both players in the total of the 24 hourly periods of the 61 considered days.

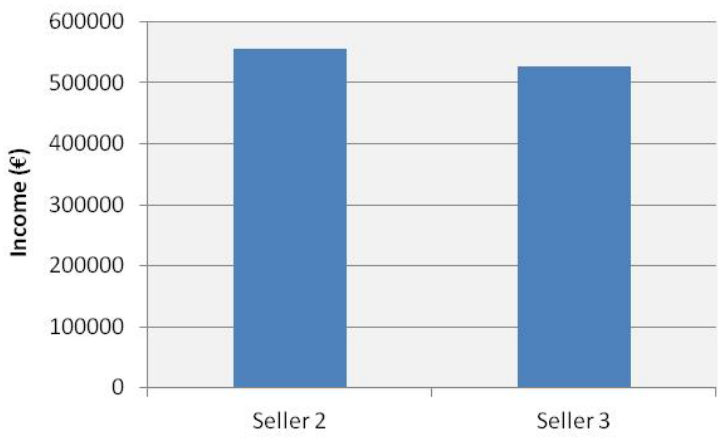


Figure 4.18 – Seller 2 and Seller 3's incomes in the total of the 24 periods of the considered 61 days.

Analyzing Figure 4.18 the remark goes for the equilibrium of both players' achieved incomes. The total income results for all periods are: Seller 2: 555,040.21 €; Seller 3: 526,152.74 €. The two players' equilibrium through competition was expected. However, such close results can be surprising, even though the two players were equipped with the same exact tools.

The most relevant conclusion to take from this test is that a player using ALBidS can, in fact, deal with the competition of intelligent opposition, and still be able to adequately achieve high incomes and adapt its behaviour to anticipate changes, react to other's plays, and make its own behaviour as unpredictable as possible.

4.5. Final Remarks

The case studies presented in this chapter supported the demonstration of the ALBidS system's advantages in supporting the decisions of an electricity market's negotiating player.

As showed by the first case study, the use of ALBidS' support resulted in the achievement of higher incomes by the test subject, when compared to the results of the same player when using other negotiating strategies. The system also proved to be able to make the adequate choices in what concerns the selection of the most appropriate supporting strategies, as showed by ALBidS ability to change its selected negotiating approach depending on the expectable results of each considered strategy throughout time, depending on the circumstances.

The advantage of using the Context Definition mechanism was showed by the results improvement on weekends. This proved to be a simple, but effective case study in what concerns the demonstration of the required feature. This demonstration also included the evaluation of the Efficiency/Effectiveness Management mechanism's suitability in managing the balance between the system's execution time and its achieved results. This mechanism proved to be able to guarantee high incomes when the preference for effectiveness is more accentuated. It also showed that it can drastically reduce the ALBidS system's execution time, when such reduction is required, nevertheless leading to the consequent smaller achievement of incomes.

Regarding the competition against players provided with intelligent behaviour, ALBidS proved to be able to appropriately adapt to such environment, while still being able to provide the supported player with high incomes.

CHAPTER 5. CONCLUSIONS

This dissertation presented ALBidS – Adaptive Learning strategic Bidding System, a multiagent system that provides intelligent decision support to electricity market negotiating players.

ALBidS combines several different strategic approaches, which are used as tools at the reach of the Main Agent, so that this central entity can use them when it finds them to be useful for each context and simulation circumstance. These alternative approaches were presented and tested individually in order to guarantee their independent adequacy to their purposes, and prove their importance to the ALBidS system. The wide range of developed approaches guarantees that at each moment responses based on different views are suggested, offering a high variety of possible actions for the ALBidS system to consider. This provides the means for the system to increase its chances of always be provided with at least one adequate answer, even when the majority of the approaches fail in the accomplishment of its purposes.

Additionally, other mechanisms were developed in order to complement the system. The Context Analysis mechanism was developed so that it could provide the Main Agent with the ability to choose the most adequate actions depending on each distinct market negotiation context. The Player Profile Definition mechanism was built to provide support to several strategies which require adequate predictions on what is expected for the market competitor players to perform next, based on their recent history of actions and reactions to the observed events. The Efficiency/Effectiveness Management mechanism came to support ALBidS adequacy to the simulation circumstances, ensuring that the system adapts its execution time to the simulation requirements. This was accomplished by filtering the approaches that should be executed at each time, depending on their results both in terms of the achieved results and the time they require to provide such results. Additionally, the Efficiency/Effectiveness Management mechanism is able to manipulate the strategies internally, so that they can individually adapt to their execution time, hence avoiding their immediate exclusion when not fulfilling the requirements.

Regarding the main entity in the ALBidS system – the Main Agent, it also proved to efficiently accomplish its purpose, showing its ability to choose the most appropriate actions for the supported player to take, by using the reinforcement learning algorithms at its disposal, and taking advantage on the complementary mechanisms that aim to enlarge ALBidS' capabilities of adaptation and providing of intelligent decision support. In addition, the Main Agent also provides the connection between ALBidS and the MASCEM electricity market simulator. The ALBidS integration with MASCEM provided the means for testing and suiting ALBidS' implemented tools and mechanisms that were presented throughout this dissertation.

Most importantly, the integration with MASCEM provided the means to perform market simulations based on real data. Taking advantage on the developed case studies based on real data, it was possible to validate the developed methodology, guaranteeing the adequate coordination between the several mechanisms of this system, and its adequacy to achieving its purpose in real cases.

Analyzing ALBidS characteristics and performance it is adequate to state that this system is a real reflection of true artificial intelligence. It is actually able to learn and adapt its behaviour, by understanding the surroundings, and the reasons behind the behaviours of others, and therefore act accordingly. It is able to adapt to what is being observed; to contexts of negotiation; and of the situation and time, by understanding the circumstances of the simulation.

This system is able to make effective decisions by itself with no interaction since the time it is launched, according to the best way of accomplishing its goals. These decisions are performed so that the system can achieve the best profits, and also by adapting its behaviour to what that behaviour is desired to be. ALBidS learns with its errors and adapts itself by observing the behaviour of others. It learns autonomously so that it can achieve the best accomplishments. This system acts like it has a life of its own, deciding with no interaction, and with full awareness of its purpose and what it expected of it.

From a scientific perspective, the achieved results show that ALBidS achieved considerably better results than all the other negotiation methods to which it was compared, being able to obtain higher profits. From the presented results, this system showed its ability to learn, showing evidence of a clear improvement in its performance throughout the time, due to learning and adaptation to the change of the circumstances, by using the available resources. These resources are the distinct approaches that the system has at its reach; the strategies, and learning and data analysis algorithms, which provide the system with action proposals, hence allowing it to choose at each time the best decisions to make. This allows ALBidS to learn what to perform, so that it can continually improve the best way of achieving its objectives. The findings with this system's development contributed to relevant advances in the artificial intelligence area, namely in the fields of machine learning and data analysis.

From an economic perspective, being this system integrated with a real market data simulator, it has been demonstrated that, trained with this data, ALBidS is capable of guaranteeing high profits with an increasing tendency for the market player it represents (*i.e.* the player to which it provides decision support). By adapting and observing changes in the market and in competitor's strategies, ALBidS is able to react rapidly in order to guarantee the best profits in each and any circumstance. The relevance of this work and its importance concerning the support of real market players is enhanced by the interest demonstrated by the Argentinean and Colombian governmental electricity market companies in using the MASCEM simulator, equipped with ALBidS, for supporting their decisions in market negotiations.

Concluding, this work has proven to be highly successful, as supported by the achieved results and enhanced by the scientific publications it has originated. It is as well a starting point for many equally relevant future findings in what concerns the system's development, *e.g.* the possibility of inclusion of many further strategies. ALBidS is not only an adequate decision support for market players, but also presents the possibility of being applied to provide decision support in many different scopes, such as in Virtual Power Players and Smart Grid operation.

The continuous development of this work is a relevant part of the core of some recently approved FCT projects, which come to give continuity to the projects already mentioned in the introductory chapter; namely the following:

- IMaDER – Intelligent Short Term Management of Distributed Energy Resources in a Multi-Player Competitive Environment (PTDC/SEN-ENR/122174/2010);
- MAN-REM – Multi-Agent Negotiation and Risk Management in Electricity Markets (PTDC/EEA-EEI/122988/2010). This project will be developed in joint endeavour between GECAD – Knowledge

Engineering and Decision Support Research Centre, LNEG – National Laboratory of Energy and Geology, and ISEL – Institute of Engineering of the Polytechnic of Lisbon.

Among the many developments that the ALBidS creation potentiated, allowing future works and scientific findings, some can be referred:

- Include the solar intensity verified in each day and period in the context definition, analogously to the actual consideration of the wind intensity;
- Create a new reinforcement learning algorithm, which attributes higher reinforcements to the strategies that present decreasing errors, *i.e.* not only favouring strategies that present low errors, but also encouraging the ones that are showing to be evolving towards the required direction;
- Enhance and improve the choosing process of the action proposals, by creating a new learning module for this purpose. Running all the reinforcement learning algorithms (RLA) in parallel, and using a final RLA to choose the most appropriate among all, *i.e.* the one that is providing the best results. This way the user will no longer have to choose the RLA he/she wants or that he/she thinks to be the best, rather letting the system autonomously perform this decision as well, by analyzing the results of each RLA. This also allows comparing performances of RLAs, including the possible inclusion of further versions of RLAs, and also testing different performing ways of the existing ones, *e.g.* by using the same RLA with different context definitions (considering a different number of contexts, not considering contexts, or considering only differentiation by period or day); or by endowing them with alternative ways of defining the reinforcements, *e.g.* by utilizing the profit achievement of each strategy, instead of the comparison between the proposed bid and the actual market price. This suggestion takes into deliberation the fact that several strategies do not perform forecasts or point at predicting the market price, rather using other basis for their acting. Considering achieved profits to define the reinforcements may lead to interesting conclusions, as these strategies will no longer be penalized if proposing values distant to the market price, if these proposals present good results nevertheless;
- Improve the Economic Analysis Agent, by including more data in the companies' external analysis, *i.e.* considering companies' types of technology, and their reach of investments in other sectors;
- Create an evolutionary approach to automatically combine parameters of an optimization algorithm (used in the Determinism Theory strategy), to create new combinations, so that they can be used, and possibly give origin to better combinations than the initially considered ones;
- Develop a new metalearner that uses a neural network. This neural network considers as inputs the strategies' outputs, and generates as its output, the predicted action;
- Consider the clustering of competitor players, in order to group them by resemblance, and spare time in profiles definition when required;
- Develop a neural network which includes each period's economic situation, wind and solar intensity predictions, and the clustering of days and periods, in its training;
- Include a mechanism to support the usage of ALBidS when a small amount of data is available. When the amount of data is not enough for the strategies to be used adequately, a backwards prediction is suggested, in which instead of predicting what is to come, the predictions are made on what happened

before the timing of the available data. This way these predictions can be used to enlarge the amount of data for the actual predictions used by the strategies, while the amount of data does not reach a point in which only the real observed data is sufficient for the strategies' adequate use (for example, in the case of the dynamic neural network, the minimum training limit considered is 60 days. When the available number of days is below that number, the strategy cannot be used until the gathered data reaches 60 days. By using the suggested approach, if the available data considers 50 days, a prediction is done backwards, to predict the missing required 10 days, and these days are used with the original 50 in order to form the required 60. This way the strategy is able to be used, even if with weaker results. As each day is over, one of the predicted historic values is discarded, and the new observed data included replacing it);

- Allow the saving and loading of historic statistics. This way, when starting a new scenario, the user can choose to load the strategies' stats of a similar scenario run before, therefore starting with possible higher advantage;
- Endow the ALBidS system with the capability of loading strategies' statistics autonomously, by analyzing the characteristics of a new scenario, and comparing them to the characteristics of past considered scenarios;
- Include in the data and context analysis, not only the historic market data, but also information from external data sources, *e.g.* stock markets, raw material prices, environmental data, emission market data, macroeconomic data, and also the competitiveness analysis, by studying the participation of competitor entities;
- Complement the ALBidS system with a new mechanism directed to the optimization of market players' investments in alternative markets. This can be done by an analysis on the characteristics and particularities of the several existing markets, including complementary markets such as the derivatives market. This way, adequate predictions of the expected incomes resulting from the investments in each of those markets can be achieved, depending on each circumstance and context. This context considers as fundamental part the various types of players, allowing an adaptation of the investments depending on their characteristics, objectives, and particular needs.

The majority of these suggestions has been considered not only for the future development of this dissertation work in the scope of the previously mentioned projects, but also as basis for the already granted PhD scholarship (reference SFRH/BD/81848/2011) in the scope of the "*Doutoramento e Pós-Doutoramento 2011*" (Phd and post-doctoral 2011) programme of FCT.

As a final remark, this document has been developed so that it could combine technical formality and strictness under a scientific view, with the possible simplicity, so that a larger range of readers can properly understand this work and take advantage of its progresses. This combination aims to instigate this work's contribution to the scientific development of the area.

REFERENCES

[Amjady *et al.*, 2010] Amjady, N. *et al*., "Day-ahead electricity price forecasting by modified relief algorithm and hybrid neural network", *IET Generation, Transmission & Distribution*, vol. 4, no. 3, pp. 432-444, 2010

[Auer *et al*., 2010] Auer, P. *et al*., "Near-optimal regret bounds for reinforcement learning", *Journal of Machine Learning Research*, vol. 11, pp. 1563-1600, 2010

[Azevedo *et al.*, 2007] Azevedo, F. *et al*., "A Decision-Support System Based on Particle Swarm Optimization for Multi-Period Hedging in Electricity Markets", *IEEE Transactions on Power Systems*, vol. 22, no. 3, pp. 995-1003, 2007

[Bandyopadhyay, 2011] Bandyopadhyay, S., "Multiobjective Simulated Annealing for Fuzzy Clustering With Stability and Validity," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 41, no. 5, pp.682-691, 2011

[Bayro and Daniel, 2010] Bayro, E. and Daniel, N., "Clifford Support Vector Machines for Classification, Regression, and Recurrence", *IEEE Transactions on Neural Networks*, vol. 21, no. 11, pp. 1731 – 1746, 2010

[Berofsky, 1971] Berofsky, B., "Determinism", *Princeton University Press*, 1971

[Biggs, 1985] Biggs, J.,"The role of meta-learning in study process", *British Journal of Educational Psychology*, vol. 55, pp. 185-212, 1985

[Bompard *et al.*, 2005] Bompard, E. *et al*., "A game theory simulator for assessing the performances of competitive electricity markets", *IEEE St. Petersburg Power Tech*, 2005

[Boyarshinov *et al.*, 2007] Boyarshinov, V. *et al*., "Efficient Optimal Linear Boosting of a Pair of Classifiers", *IEEE Transactions on Neural Networks*, vol. 18, no. 2, pp. 317-328, 2007

[Brealey and Myers, 2003] Brealey, R. and Myers, S., "Principles of Corporate Finance", *McGraw-Hill*, New York, 2003

[Bruha, 2004] Bruha, I., "Meta-Learner for Unknown Attribute Values Processing: Dealing with Inconsistency of Meta-Databases", *Journal of Intelligent Information Systems*, vol. 22, no. 1, 2004

[Brustolini, 1991] Brustolini, J., "Autonomous Agents: characterization and requirements", *Carnegie Mellon Technical Report CMU-CS-91-204*, Pittsburgh, 1991

[Bunge, 1959] Bunge, M., "Causality: The Place of the Causal Principle in Modern Science", *Harvard University Press*, 1959

[Cao *et al*., 2009] Cao, L. *et al*., "Agent Mining: The Synergy of Agents and Data Mining", *IEEE Intelligent Systems*, vol. 24, no. 3, pp. 64-72, 2009

[Church and Ware, 2000] Church, J. and Ware, R., "Industrial Organization: A Strategic Approach", *McGraw-Hill*, New York, 2000

[Coelho, 1994] Coelho, H., "Inteligência Artificial em 25 lições", *Fundação Calouste Gulbenkian*, 1994

[Dang *et al*., 2004] Dang, V. *et al*., "Generating coalition structures with finite bound from the

	optimal guarantees", *Proceedings of the 3rd International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 564-571, 2004
[Davidson, 1980]	Davidson, D., "Mental Events in his Essays on Actions and Events", *Oxford University Press*, 1980
[Davis, 1980]	Davis, R., "Report on the Workshop on Distributed Artificial Intelligence", *SIGART Newsletter*, 1980
[Dekel *et al.*, 2009]	Dekel, O. *et al.*, "Individual Sequence Prediction Using Memory-Efficient Context Trees", *IEEE Transactions on Information Theory*, vol. 55, no. 11, pp. 5251-5262, 2009
[Dietterich, 1991]	Dietterich, T., "Bridging the gap between specification and implementation", *IEEE Expert* , vol. 6, no. 2, pp. 80-82, 1991
[Dore and Regazzoni, 2010]	Dore, A. and Regazzoni, C., "Interaction Analysis with a Bayesian Trajectory Model", *IEEE Intelligent Systems*, vol. 25, no. 3, pp. 32–40, 2010
[Einstein, 1905]	Einstein, A., "Zur Elektrodynamik bewegter Koerper", *Annalen der Physik,* vol. 17, pp. 891-921, 1905
[Erev and Roth, 1998]	Erev, I. and Roth, A., "Predicting how people play games with unique, mixed-strategy equilibria", American Economic Review, vol. 88, pp. 848–881, 1998
[FERC, 2003]	Federal Energy Regulatory Commission, "Notice of White Paper", *U.S. Federal Energy Regulatory Commission*, vol. 4, no. 28, 2003
[FERC, 2011]	Federal Energy Regulatory Commission, "Report to Congress on competition in the wholesale and retail markets for electric energy", *U.S. Federal Energy Regulatory Commission*. Available: http://www.ferc.gov/legal/maj-ord-reg/fed-sta/ene-pol-act/epact-final-rpt.pdf
[Figueiredo *et al.*, 2005]	Figueiredo, V. *et al.*, "An electric energy consumer characterization framework based on data mining techniques", *IEEE Transactions on Power Systems*, vol. 20, no. 2, 2005
[Fisher, 1922]	Fisher, R., "On the mathematical foundations of theoretical statistics", *Philosophical Transactions of the Royal Society*, vol. 222, pp. 309-368, 1922
[Ginsberg, 1993]	Ginsberg, A., "A unified approach to automatic indexing and information retrieval", *IEEE Expert* , vol. 8, no. 5, pp. 46-56, 1993
[Gintis, 2000]	Gintis, H., "Game theory evolving: a problem-centered introduction to modeling strategic behavior", *Princeton University Press*, 2000
[Glover, 1989]	Glover, F., "Tabu search - part I", *ORSA Journal on Computing*, 1989
[Goldkuhl *et al.*, 1998]	Goldkuhl, G. *et al.*, "Method integration: the need for a learning perspective", *IEEE Proceedings on Software*, vol. 145, no. 4, pp. 113-118, 1998
[Gorride and Pedycz, 2007]	Gomide, F and Pedrycz, W., "Notions and Concepts of Fuzzy Sets", *Fuzzy Systems Engineering: Toward Human-Centric Computing*, 2007
[Green¸ 2003]	Green, C., "The Lost Cause: Causation and the Mind-Body Problem", *Oxford Forum Publisher: Oxford Forum*, vol. 1, 2003
[Greenwald *et al.*, 1999]	Greenwald, A., *et al.*, "Shopbots and Pricebots", *Proceedings of the Sixteenth*

	International Joint Conference on Artificial Intelligence - IJCAI, Stockholm, 1999
[Harp *et al.*, 2000]	Harp, S. *et al.*, "Sepia: A Simulator for Electric Power Industry Agents", *IEEE Control Systems Magazine*, vol. 20, no. 4, pp. 53 – 69, 2000
[Hartley, 1928]	Hartley, H., "Transmission of Information", *Bell System Technical Journal*, vol. 7, no. 3, pp. 535 - 563, 1928
[Hatziargyrious *et al.*, 2002]	Hatziargyrious, N. *et al.*, "Distributed energy sources: technical challenges" *IEEE Power Engineering Society Winter Meeting*, vol. 2, pp. 1017-1022, 2002
[Heisenberg, 1962]	Heisenberg, W., "Physics and Philosophy: The revolution in modern science", *HarperCollins Publishers*, 1962
[Hill and Westbrook, 1997]	Hill, T. and Westbrook, R., "SWOT Analysis: It's Time for a Product Recall", *Long Range Planning*, vol. 30, no. 1, pp. 46-52, 1997
[Hortaçsu and Puller, 2008]	Hortaçsu, A. and Puller, L., "Understanding strategic bidding in multi-unit auctions: a case study of the Texas electricity spot market" *The RAND Journal of Economics*, vol. 39 no. 1, pp. 86-114, Wiley InterScience, 2008
[Houaiss, 2001]	*Dicionário Houaiss da língua Portuguesa*, Temas e debates (ed.), Instituto António Houaiss, 2001
[Ilic and Galiana, 1998]	Ilic, M. and Galiana, F., "Power Syst Restructuring: Engineering and Economics", *Kluwer Academic Publishers Group*, 1998
[Jing *et al.*, 2009]	Jing, Z. *et al*., "Study on the convergence property of re learning model in electricity market simulation", *Advances in Power System Control, Operation and Management*, 2009
[Juang and Lu, 2009]	Juang, C. and Lu, C., "Ant Colony Optimization Incorporated With Fuzzy Q-Learning for Reinforcement Fuzzy Control", *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 39, no. 3, pp. 597-608, 2009
[Kaelbling *et al*., 1996]	Kaelbling, L. *et al*., "Reinforcement Learning: A Survey", *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996
[Kennedy and Eberhat, 1948]	Kennedy, J. and Eberhat, R., "Particle Swarm Optimization", *Proceedings of the 1995 IEEE Internacional Conference on Neural Networks*, pp.1942-1948, Perth, Australia, 1995
[Klemperer, 1999]	Klemperer, P., "Auction Theory: A Guide to the Literature," *Journal of Economic Surveys*, vol. 13, no. 3, pp. 227–286, 1999
[Korb and Nicholson, 2003]	Korb, K. and Nicholson, A., "Bayesian Artificial Intelligence", *Chapman & Hall/CRC*, 2003
[Koritarov, 2004]	Koritarov, V., "Real-World Market Representation with Agents: Modeling the Electricity Market as a Complex Adaptive System with an Agent-Based Approach", *IEEE Power & Energy magazine*, pp. 39-46, 2004
[Kullback and Leibler, 1951]	Kullback, S. and Leibler, "On Information and Sufficiency", *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79-86, 1951
[Li and Tesfatsion, 2009]	Li, H. and Tesfatsion, L., "Development of Open Source Software for Power

	Market Research: The AMES Test Bed", *Journal of Energy Markets*, vol. 2, no. 2, pp. 111-128, 2009
[Liu *et al.*, 2005]	Liu, Z. *et al.*, "Electricity Price Forecasting Model Based on Chaos Theory", *The 7th International Power Engineering Conference*, 2005
[Liu *et al.*, 2008]	Liu, W. *et al*., "An Information Theoretic Approach of Designing Sparse Kernel Adaptive Filters", *Transactions on Neural Networks*, vol. 20, no. 12, pp. 1950-1961, 2008
[Luck *et al.*, 2005]	Luck, M. *et al.*, "Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)", *University of Southampton*, 2005
[Madureira and Pereira, 2010]	Madureira, A. and Pereira, I., "Self-Optimization for Dynamic Scheduling in Manufacturing Systems", *Technological Developments in Networking, Education and Automation*, pp. 421-426, Springer Netherlands, 2010
[Matos, 2007]	Matos, F., "Mercados Ambientais e Certificados Verdes", *Workshop Sistemas de Energia - Instituto Superior de Engenharia de Lisboa*, 2007
[Meeus *et al.*, 2005]	Meeus, L. *et al.*, "Development of the Internal Electricity Market in Europe", *The Electricity Journal*, vol. 18, no. 6, pp. 25-35, 2005
[Mellor, 1986]	Mellor, D., "Fixed Past, Unfixed Future", *Contributions to Philosophy: Michael Dummett*, pp. 166-86, 1986
[Menon *et al.*, 1999]	Menon, A. *et al.*, "Antecedents and Consequences of Marketing Strategy Making", *Journal of Marketing - American Marketing Association*, vol. 63, no. 2, pp. 18-40, 1999
[Michalski, 1980]	Michalski, R., "Pattern Recognition as Rule-Guided Inductive Inference", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 4, pp. 349-361, 1980
[Migliavacca, 2007]	Migliavacca, G., "SREMS-electricity market simulator based Game Theory and incorporating network constraints", *IEEE Power Tech*, Lausanne, Swiss, 2007
[Minsky, 1961]	Minsky, M., "A Selected Descriptor-Indexed Bibliography to the Literature on Artificial Intelligence", *IRE Transactions on Human Factors in Electronics*, vol. 2, no. 1, pp. 39-55, 1961
[Minsky, 1986]	Minsky, M., "The Society of Mind", *Simon and Schuster*, 1986
[Neumann *et al.*, 2003]	Neumann, J. *et al.* "The essence of game theory", *IEEE Potentials*, vol. 22, no. 2, 2003
[Nicolaisen *et al.*, 2001]	Nicolaisen, J. *et al.*, "Market power and efficiency in a computational electricity market with discriminatory double-auction pricing", *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 5, pp. 504-523, 2001
[Oliveira *et al.*, 2009]	Oliveira, P. *et al*., "MASCEM – An Electricity Market Simulator providing Coalition Support for Virtual Power Players", *International Conference on Intelligent System Application to Power Systems*, 2009
[Olsson and Soder, 2008]	Olsson, M. and Söder, L., "Modeling Real-Time Balancing Power Market Prices Using Combined SARIMA and Markov Processes", *IEEE Transactions on Power*

	Systems, vol. 23, no. 2, 2008
[Osborne and Martin, 2004]	Osborne, A. and Martin, J., “An introduction to game theory”, *Oxford University Press,* 2004
[Owen, 1995]	Owen, G., “Game Theory”, *Academic Press*, 1995
[Oxford, 2011]	*Oxford dictionary,*New York Oxford Press
[Padovitz *et al.*, 2010]	Padovitz, A. *et al.*, “Multiple-Agent Perspectives in Reasoning About Situations for Context-Aware Pervasive Computing Systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 4, pp. 729-742, 2008
[Pal, 2010]	Pal, R., "Context-Sensitive Probabilistic Boolean Networks: Steady-State Properties, Reduction, and Steady-State Approximation", *IEEE Transactions on Signal Processing*, vol. 58, no. 2, pp. 879-890, 2010
[Panait and Luke, 2005]	Panait, L. and Luke, S., “Cooperative Multi-Agent Learning: The State of the Art”, *Autonomous Agents and Multi-Agent Systems*, pp.387-434, 2005
[Pereira, 2010]	Pereira, I., “Aspectos de Aprendizagem em Optimização”, *Master of Science thesis in Computer Science – Knowledge-based and Decision Support Technologies,* Polytechnic of Porto, Portugal, 2010
[Pinto *et al.*, 2009]	Pinto, T. *et al.*, “Multi-Agent Based Electricity Market Simulator with VPP: Conceptual and Implementation Issues”, *IEEE PES General Meeting*, 2009
[Pinto *et al.*, 2011a]	Pinto, T. *et al.*, “Multiagent System for Adaptive Strategy Formulation in Electricity Markets”, *IEEE Symposium Series on Computational Intelligence - IEEE Symposium on Intelligent Agents - SSCI-IA*, 2011
[Pinto *et al.*, 2011b]	Pinto, T. *et al.*, “Cost Dependent Strategy for Electricity Markets Bidding Based on Adaptive Reinforcement Learning”, *International Conference on Intelligent System Application on Power Systems - ISAP*, 2011
[Pinto *et al.*, 2011c]	Pinto, T. *et al.*, “Strategic Bidding Methodology for Electricity Markets using Adaptive Learning”, *Modern Approaches in Applied Intelligence*, vol. 6704, pp. 490-500, 2011
[Pinto *et al.*, 2011d]	Pinto, T. *et al.*, “Bid Definition Method for Electricity Markets Based on an Adaptive Multiagent System", *Advances on Practical Applications of Agents and Multiagent Systems*, Springer Berlin / Heidelberg, vol. 88, pp. 309-316, 2011
[Pinto *et al.*, 2011e]	Pinto, T. *et al.*, “A new approach for multi-agent coalition formation and management in the scope of electricity markets”, *Energy*, vol. 36, no. 8, pp. 5004-5015, 2011
[Pinto *et al.*, 2011f]	Pinto, T. *et al.*, “Multi-Agent Negotiation for Coalition Formation and Management in Electricity Markets”, *Negotiation and Argumentation in MAS*, Bentham Science Publishers Ltd., 2011
[Porter, 1979]	Porter, M., “How Competitive Forces Shape Strategy”, *Harvard business Review*, 1979
[Porter, 2008]	Porter, M., "The Five Competitive Forces that Shape Strategy", *Harvard Business*

	Review, pp. 86-104, 2008
[Praça *et al.*, 2003]	Praça, I. *et al*., “MASCEM: A Multi-Agent System that Simulates Competitive Electricity Markets”. *IEEE Intelligent Systems*, vol. 18, no. 6, pp. 54-60, Special Issue on Agents and Markets, 2003
[Praça *et al.*, 2007]	Praça, I. *et al*., “Virtual Power Producers Integration Into Mascem”, *Establishing The Foundation Of Collaborative Network Book Series IFIP*, pp. 291-298, Springer, 2007
[Principe, 2010]	Principe, J., “Information Theoretic Learning”, *Springer, Information Science and Statistics series*, 2010
[PSI, 2007]	Paul Scherrer Institut - PSI, “CO2 -Emissionen aus dem nuklearen Kreislauf”, *PSI Fachinfoszu Energiefragen*, Villigen, Switzerland, 2007
[Rahimi-Kian *et al.*, 2005]	Rahimi-Kian, A. *et al*., “Q-learning based supplier-agents for electricity markets”, *IEEE Power Engineering Society General Meeting*, vol. 1, pp. 420 - 427, 2005
[Rahwan *et al*.,2008]	Rahwan, T. *et al*., “Coalition structure generation: dynamic programming meets anytime optimization”, *Proceedings of the 23rd Conference on AI – AAAI*, pp. 156-161, 2008
[Rao and Principe, 2008]	Rao, S. and Principe, J., “Mean Shift: An Information Theoretic Perspective”. *Transanctions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, 2008
[Rényi, 1970]	Rényi, A., “Probability Theory”, *American Elsevier Publishing Company*, New York, 1970
[Rich and Knight]	Rich, E. and Knight, K., “Artificial Intelligence*”, McGraw-Hill*, 1991
[Roth and Erev, 1995]	Roth, A. and Erev, I., “Learning in extensive form games: Experimental data and simple dynamic models in the intermediate term”, *Games and Economic Behavior*, vol. 8, pp. 164–212, 1995
[Sama *et al.*, 2010]	Sama, M. *et al*., “Context-Aware Adaptive Applications: Fault Patterns and Their Automated Identification", *IEEE Transactions on Software Engineering*, vol. 36, no. 5, pp. 644-661, 2010
[Santos *et al.*, 2011]	Santos, G. *et al*., “Complex Market integration in MASCEM electricity market simulator”, *International Conference on the European Energy Market 11 – EEM*, 2011
[Schaefer *et al.*, 2001]	Schaefer, A. *et al*., “A Recurrent Control Neural Network for Data Efficient Reinforcement Learning", *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, vol. 1, no. 5, pp. 151-157, 2007
[Shahidehpour *et al*., 2002]	Shahidehpour, M. *et al*., “Market Operations in Electric Power Systems: Forecasting, Scheduling, and Risk Management”, *Wiley-IEEE Press*, pp. 233-274, 2002
[Shannon, 1948]	Shannon, C., “A Mathematical Theory of Communication”, *Bell System Technical Journal*, vol. 27, pp. 379-423, 623 - 656, 1948
[Sheblé, 1999]	Sheblé, G., “Computational Auction Mechanisms for Restructured Power

Industry Operation”, *Kluwer Academic Publishers*, 1999

[Shoham *et al.*, 2009] Shoham, B. *et al.* “Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations”, *New York: Cambridge University Press*, 2009

[Silverman, 1986] Silverman, J. “The Arithmetic of Elliptic Curves”, *Graduate Texts in Mathematics*, Springer, 1986

[Simon, 1973] Simon, H., "Administrative Decision Making", *IEEE Engineering Management Review*, vol. 1, no. 1, pp. 60-66, 1973

[Skinner, 1971] Skinner, B., “Beyond Freedom and Dignity”, *New York: Knopf*, 1971

[Sousa, 2011] Sousa, T., “Sistema de aprendizagem para agentes na Simulação de Mercados de Electricidade”, *BSc thesis in Computer Science*, Polytechnic of Porto, Portugal, 2011

[Stone and Veloso, 1997] Stone, P. and Veloso, M., “Towards collaborative and adversarial learning: A case study for robotic soccer”, *International Journal of Human Computer Studies*, vol. 48, 1997

[Sun and Tesfatsion, 2007] Sun, J. and Tesfatsion, L., “Dynamic Testing of Wholesale Power Market Designs: An Open-Source Agent-Based Framework”, *Computational Economics*, vol. 30, no. 3, pp. 291-327, 2007

[Sutton *et al*., 1998] Sutton, R. *et al*., “Reinforcement Learning: An Introduction”, *MIT Press*, 1998

[Sutton, 1988] Sutton, R., "Learning to predict by the method of temporal differences", *Machine Learning*, vol. 3, pp. 9-44, Springer, 1988

[Tellidou and Bakirtzis, 2007] Tellidou, A. and Bakirtzis, A., “Agent-Based Analysis of Monopoly Power in Electricity Markets”, *International Conference on Intelligent Systems Applications to Power Systems*, 2007

[Tokic, 2010] Tokic, M., "Adaptive e-Greedy Exploration in Reinforcement Learning Based on Value Differences", *Advances in Artificial Intelligence. Lecture Notes in Computer Science*, pp. 203-210, Springer Berlin / Heidelberg, 2010

[Vale *et al*., 2008] Vale, Z. *et al*., “Distributed Generation Producers’ Reserve Management” *IEEE PES General Meeting*, 2008

[Vale *et al*., 2011a] Vale, Z. *et al*., “MASCEM - Electricity markets simulation with strategically acting players”, *IEEE Intelligent Systems*, vol. 26, no. 2, pp. 54-60 Special Issue on AI in Power Systems and Energy Markets, 2011

[Vale *et al*., 2011b] Vale, Z. *et al*., “Electricity Markets Simulation: MASCEM contributions to the challenging reality”, *Handbook of Networks in Power Systems*, Springer-Verlag, 2011

[Vale *et al*., 2011c] Vale, Z. *et al*., “VPP’s Multi-Level Negotiation in Smart Grids and Competitive Electricity Markets”, *IEEE PES General Meeting - PES-GM*, 2011

[Veen and Vries, 2008] Veen, R. van der, and Vries, L. De, “Balancing market design for a decentralized electricity system: Case of the Netherlands”, *First International Conference on Infrastructure Systems and Services: Building Networks for a Brighter Future, INFRA*, 2008

[Wand *et al.*, 2008]	Wang, B. *et al.*, "Prediction of Power System Marginal Price Based on Chaos Characteristics", *IEEE International Conference on Industrial Technology*, pp. 21-24, 2008
[Weiss, 2010]	Weiss, G., "Multiagent Systems: a modern approach to distributed artificial intelligence", *MIT Press*, 2010
[Wellman *et al.,* 2007]	Wellman, P. *et al*., "Autonomous Bidding Agents: Strategies and Lessons from the Trading Agent Competition", *MIT Press*, 2007
[Wilamowski and Yu, 2010]	Wilamowski, B. and Yu, H., "Neural Network Learning Without Backpropagation", IEEE Transactions on Neural Networks, vol. 21, no. 11, pp. 1793-1803, 2010
[Winnie, 1996]	Winnie, J., "Deterministic Chaos and the Nature of Chance", *The Cosmos of Science - Essays of Exploration*, University of Pitsburgh Press, pp. 299–324, 1996
[Winston, 1992]	Winston, P., "Artificial Intelligence", *Addison- Wesley*, pp. 15-228, 1992
[Wooldridge, 2002]	Wooldridge, J., "Econometric Analysis of Cross Section and Panel Data", *Cambridge MIT Press*, Cambridge, 2002
[Wooldridge, 2002]	Wooldridge, M., "An Introduction to Multiagent Systems", *Wiley*, New York, 2002
[Zambonelli *et al.*, 2004]	Zambonelli, F. *et al*., "Towards a Paradigm Change in Computer Science and Software Engineering: A Synthesis", *The Knowledge Engineering Review*, vol. 18, no. 4, 2004
[Zhao *et al.*, 2008]	Zhao, J. *et al*., "A Statistical Approach for Interval Forecasting of the Electricity Price", *IEEE Transactions on Power Systems*, vol. 23, no. 2, pp. 267-276, 2008
[Zhifeng *et al.*, 2009]	Zhifeng, Q. *et al*., "Comparison of Two Learning Algorithms in Modelling the Generator's Learning Abilities", *15th International Conference on Intelligent System Applications to Power Systems – ISAP*, vol. 1, no. 6, pp.8-12, 2009
[Zimmermann *et al.*, 2001]	Zimmermann, H. *et al*., "Multi-agent modeling of multiple FX-markets by neural networks", *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 735-743, 2001
[Zimmermann *et al.,* 2004]	Zimmermann, R. *et al.,* "PowerWeb: a tool for evaluating economic and reliability impacts of electric power market designs", *IEEE PES Power Systems Conference and Exposition*, vol. 3, pp. 1562-1567, 2004
[Zimmermann *et al.*, 2005]	Zimmermann, H. *et al*., "Dynamical consistent recurrent neural networks", *IEEE International Joint Conference on Neural Networks*, vol. 3, pp. 1537- 1541, 2005

URL REFERENCES

[BANJO, 2011]	BANJO – homepage, http://www.cs.duke.edu/~amink/software/banjo/, accessed on March 2011
[BNJ3, 2011]	BNJ3 – homepage, http://www.kddresearch.org/Groups/Probabilistic-reasoning/BNJ/index.html, accessed on March 2011
[EAC, 2009]	Electricity Advisory Committee, "Keeping the Lights On in a New World", January 2009. Available: http://www.oe.energy.gov/eac.htm
[Flex, 2011]	Flex expert system toolkit – homepage, http://www.lpa.co.uk/flx.htm, accessed on August 2011
[IS, 2011]	LPA Intelligent Server – homepage, http://www.lpa.co.uk/int.htm, accessed on August 2011
[JAVA, 2011]	Java programming language – homepage, http://www.java.com, accessed on August 2011
[JAVABAYES, 2011]	JavaBayes – homepage, http://www.cs.cmu.edu/~javabayes/, accessed on March 2011
[JBNC, 2011]	jBNC – homepage, http://jbnc.sourceforge.net/, accessed on March 2011
[JMatLink, 2011]	JMatLink – homepage, http://www.held-mueller.de/JMatLink, accessed on March 2011
[MatLab, 2011]	MatLab programming language – homepage, http://www.mathworks.com/products/matlab/, accessed on August 2011
[OAA, 2011]	Open Agent Architecture – homepage, http://www.ai.sri.com/~oaa/, accessed on August 2011
[OMEL, 2011]	*Operador del Mercado Ibérico de Energia* – homepage, http://www.omel.es, accessed on August 2011
[Prolog, 2011]	LPA-Prolog programming language – homepage, http://www.lpa.co.uk/, accessed on August 2011
[Siemens, 2011]	Siemens AG, Corporate Technology – homepage, http://www.siemens.com/, accessed on August 2011
[Wiki, 2011a]	Wikipedia – homepage, http://en.wikipedia.org/, accessed on August 2011
[Wiki, 2011b]	Wikipedia – definition of context page, http://en.wikipedia.org/Context, accessed on August 2011

Annex A. – Fuzzy Confusion Matrix

The table presented in this annex represents the fuzzy confusion matrix used by the Efficiency/Effectiveness Management mechanism (presented in section 3.9) to interconnect the considered fuzzy variables. This matrix contains all the possible combinations between the values that the fuzzy variables can assume, and the consequent outcome that each combination produces.

The first column of Table A.1 contains the possible values of the Efficiency difference variable, which represents the difference between a strategy's execution time, and MASCEM's execution time without the use of ALBidS. Remember that, according to the explanation in section 3.9, this difference value assumes the *negative* value when the execution time of a strategy is smaller than MASCEM's.

The second column of the table represents the possible values of the Effectiveness difference variable. These values represent the difference between a forecasted value from behalf of a strategy, and the actual value, observed in the market after the simulation.

The values representing the user's preference for the efficiency/effectiveness balance are presented in the third column, and the consequent outcomes that result from the combination of the three variables are presented in the fourth column. These outcomes are the requests that are sent to each strategy so that it can adapt to the current circumstances.

Table A.1 – Fuzzy confusion matrix.

Efficiency Diference		Effectiveness Diference		2E Preference		Action to Take
negative	*	very_small	*	big_effectiveness	>	ignore
negative	*	very_small	*	small_effectiveness	>	ignore
negative	*	very_small	*	medium	>	ignore
negative	*	very_small	*	small_efficiency	>	ignore
negative	*	very_small	*	big_efficiency	>	ignore
negative	*	small	*	big_effectiveness	>	ignore
negative	*	small	*	small_effectiveness	>	ignore
negative	*	small	*	medium	>	ignore
negative	*	small	*	small_efficiency	>	ignore
negative	*	small	*	big_efficiency	>	ignore
negative	*	medium	*	big_effectiveness	>	ignore
negative	*	medium	*	small_effectiveness	>	ignore
negative	*	medium	*	medium	>	ignore
negative	*	medium	*	small_efficiency	>	ignore
negative	*	medium	*	big_efficiency	>	ignore
negative	*	big	*	big_effectiveness	>	ignore
negative	*	big	*	small_effectiveness	>	ignore
negative	*	big	*	medium	>	ignore
negative	*	big	*	small_efficiency	>	ignore
negative	*	big	*	big_efficiency	>	ignore
negative	*	very_big	*	big_effectiveness	>	ignore
negative	*	very_big	*	small_effectiveness	>	ignore
negative	*	very_big	*	medium	>	ignore
negative	*	very_big	*	small_efficiency	>	ignore
negative	*	very_big	*	big_efficiency	>	ignore

Efficiency Diference		Effectiveness Diference		2E Preference		Action to Take
very_small	*	very_small	*	big_effectiveness	>	ignore
very_small	*	very_small	*	small_effectiveness	>	ignore
very_small	*	very_small	*	medium	>	ignore
very_small	*	very_small	*	small_efficiency	>	ignore
very_small	*	very_small	*	big_efficiency	>	small_reduction
very_small	*	small	*	big_effectiveness	>	ignore
very_small	*	small	*	small_effectiveness	>	ignore
very_small	*	small	*	medium	>	ignore
very_small	*	small	*	small_efficiency	>	ignore
very_small	*	small	*	big_efficiency	>	small_reduction
very_small	*	medium	*	big_effectiveness	>	ignore
very_small	*	medium	*	small_effectiveness	>	ignore
very_small	*	medium	*	medium	>	ignore
very_small	*	medium	*	small_efficiency	>	ignore
very_small	*	medium	*	big_efficiency	>	small_reduction
very_small	*	big	*	big_effectiveness	>	ignore
very_small	*	big	*	small_effectiveness	>	ignore
very_small	*	big	*	medium	>	ignore
very_small	*	big	*	small_efficiency	>	ignore
very_small	*	big	*	big_efficiency	>	small_reduction
very_small	*	very_big	*	big_effectiveness	>	ignore
very_small	*	very_big	*	small_effectiveness	>	ignore
very_small	*	very_big	*	medium	>	ignore
very_small	*	very_big	*	small_efficiency	>	small_reduction
very_small	*	very_big	*	big_efficiency	>	small_reduction
small	*	very_small	*	big_effectiveness	>	ignore
small	*	very_small	*	small_effectiveness	>	ignore
small	*	very_small	*	medium	>	ignore
small	*	very_small	*	small_efficiency	>	ignore
small	*	very_small	*	big_efficiency	>	small_reduction
small	*	small	*	big_effectiveness	>	ignore
small	*	small	*	small_effectiveness	>	ignore
small	*	small	*	medium	>	ignore
small	*	small	*	small_efficiency	>	ignore
small	*	small	*	big_efficiency	>	small_reduction
small	*	medium	*	big_effectiveness	>	ignore
small	*	medium	*	small_effectiveness	>	ignore
small	*	medium	*	medium	>	ignore
small	*	medium	*	small_efficiency	>	ignore
small	*	medium	*	big_efficiency	>	small_reduction
small	*	big	*	big_effectiveness	>	ignore
small	*	big	*	small_effectiveness	>	ignore
small	*	big	*	medium	>	ignore
small	*	big	*	small_efficiency	>	small_reduction
small	*	big	*	big_efficiency	>	small_reduction
small	*	very_big	*	big_effectiveness	>	ignore
small	*	very_big	*	small_effectiveness	>	ignore
small	*	very_big	*	medium	>	ignore
small	*	very_big	*	small_efficiency	>	small_reduction
small	*	very_big	*	big_efficiency	>	small_reduction
medium	*	very_small	*	big_effectiveness	>	ignore
medium	*	very_small	*	small_effectiveness	>	ignore
medium	*	very_small	*	medium	>	ignore
medium	*	very_small	*	small_efficiency	>	small_reduction
medium	*	very_small	*	big_efficiency	>	small_reduction
medium	*	small	*	big_effectiveness	>	ignore
medium	*	small	*	small_effectiveness	>	ignore
medium	*	small	*	medium	>	ignore
medium	*	small	*	small_efficiency	>	small_reduction
medium	*	small	*	big_efficiency	>	small_reduction
medium	*	medium	*	big_effectiveness	>	ignore
medium	*	medium	*	small_effectiveness	>	ignore
medium	*	medium	*	medium	>	ignore

Efficiency Diference		Effectiveness Diference		2E Preference		Action to Take
medium	*	medium	*	small_efficiency	>	small_reduction
medium	*	medium	*	big_efficiency	>	big_reduction
medium	*	big	*	big_effectiveness	>	ignore
medium	*	big	*	small_effectiveness	>	ignore
medium	*	big	*	medium	>	small_reduction
medium	*	big	*	small_efficiency	>	small_reduction
medium	*	big	*	big_efficiency	>	big_reduction
medium	*	very_big	*	big_effectiveness	>	ignore
medium	*	very_big	*	small_effectiveness	>	ignore
medium	*	very_big	*	medium	>	small_reduction
medium	*	very_big	*	small_efficiency	>	small_reduction
medium	*	very_big	*	big_efficiency	>	big_reduction
big	*	very_small	*	big_effectiveness	>	ignore
big	*	very_small	*	small_effectiveness	>	ignore
big	*	very_small	*	medium	>	small_reduction
big	*	very_small	*	small_efficiency	>	big_reduction
big	*	very_small	*	big_efficiency	>	big_reduction
big	*	small	*	big_effectiveness	>	ignore
big	*	small	*	small_effectiveness	>	ignore
big	*	small	*	medium	>	small_reduction
big	*	small	*	small_efficiency	>	big_reduction
big	*	small	*	big_efficiency	>	big_reduction
big	*	medium	*	big_effectiveness	>	ignore
big	*	medium	*	small_effectiveness	>	ignore
big	*	medium	*	medium	>	small_reduction
big	*	medium	*	small_efficiency	>	big_reduction
big	*	medium	*	big_efficiency	>	exclude_algorithm
big	*	big	*	big_effectiveness	>	ignore
big	*	big	*	small_effectiveness	>	small_reduction
big	*	big	*	medium	>	big_reduction
big	*	big	*	small_efficiency	>	big_reduction
big	*	big	*	big_efficiency	>	exclude_algorithm
big	*	very_big	*	big_effectiveness	>	ignore
big	*	very_big	*	small_effectiveness	>	small_reduction
big	*	very_big	*	medium	>	big_reduction
big	*	very_big	*	small_efficiency	>	exclude_algorithm
big	*	very_big	*	big_efficiency	>	exclude_algorithm
very_big	*	very_small	*	big_effectiveness	>	ignore
very_big	*	very_small	*	small_effectiveness	>	ignore
very_big	*	very_small	*	medium	>	small_reduction
very_big	*	very_small	*	small_efficiency	>	big_reduction
very_big	*	very_small	*	big_efficiency	>	exclude_algorithm
very_big	*	small	*	big_effectiveness	>	ignore
very_big	*	small	*	small_effectiveness	>	ignore
very_big	*	small	*	medium	>	big_reduction
very_big	*	small	*	small_efficiency	>	exclude_algorithm
very_big	*	small	*	big_efficiency	>	exclude_algorithm
very_big	*	medium	*	big_effectiveness	>	ignore
very_big	*	medium	*	small_effectiveness	>	ignore
very_big	*	medium	*	medium	>	big_reduction
very_big	*	medium	*	small_efficiency	>	exclude_algorithm
very_big	*	medium	*	big_efficiency	>	exclude_algorithm
very_big	*	big	*	big_effectiveness	>	ignore
very_big	*	big	*	small_effectiveness	>	small_reduction
very_big	*	big	*	medium	>	exclude_algorithm
very_big	*	big	*	small_efficiency	>	exclude_algorithm
very_big	*	big	*	big_efficiency	>	exclude_algorithm
very_big	*	very_big	*	big_effectiveness	>	ignore
very_big	*	very_big	*	small_effectiveness	>	big_reduction
very_big	*	very_big	*	medium	>	exclude_algorithm
very_big	*	very_big	*	small_efficiency	>	exclude_algorithm
very_big	*	very_big	*	big_efficiency	>	exclude_algorithm