# Hierarchical Reinforcement Learning for Trading Agents

Rodrigue TALLA KUATE

*Doctor of Philosophy*

ASTON UNIVERSITY

April 2015

ASTON UNIVERSITY

# Hierarchical Reinforcement Learning for Trading Agents

Rodrigue TALLA KUATE

Doctor of Philosophy, 2015

Thesis Summary

Autonomous software agents, the use of which has increased due to the recent growth in computer power, have considerably improved electronic commerce processes by facilitating automated trading actions between the market participants (sellers, brokers and buyers). The rapidly changing market environments pose challenges to the performance of such agents, which are generally developed for specific market settings. To this end, this thesis is concerned with designing agents that can gradually adapt to variable, dynamic and uncertain markets and that are able to reuse the acquired trading skills in new markets.

This thesis proposes the use of reinforcement learning techniques to develop adaptive trading agents and puts forward a novel software architecture based on the semi-Markov decision process and on an innovative knowledge transfer framework. To evaluate my approach, the developed trading agents are tested in internationally well-known market simulations and their behaviours when buying or/and selling in the retail and wholesale markets are analysed. The proposed approach has been shown to improve the adaptation of the trading agent in a specific market as well as to enable the portability of the its knowledge in new markets.

**Keywords:** SMDP, Agent Learning and Adaptation, Knowledge Transfer, Trading Strategies, Design of Trading Agents

*Dedicated with love to all members of my family*

# Acknowledgements

Without the direct or indirect contributions and advice of many people, the completion of the work presented in this PhD thesis would not be possible. First and foremost, I am deeply grateful to my supervisors Dr. Minghua He and Dr. Maria Chli as well as to my associate supervisor Dr. Hai Wang for their heartfelt encouragement, their constructive comments and the enlightening discussions throughout this research. I would also like to extend my gratitude to Professor Ian Nabney, who gave me valuable guidance on the use of machine learning techniques at the beginning of my research. More generally, I am indebted to the School of Engineering and Applied Science of Aston University that provides me the ideal environment for innovative and ground breaking research. I would also like to acknowledge the unknown reviewers who gave me insightful feedback on the papers submitted. Finally, I would specially like to thank all the members of my family and friends for their unconditional support, encouragement and patience over the years.

# Contents

# List of Tables

# List of Figures

# Nomenclature

| | |
|---|---|
| ARHMM | Autoregressive HMM |
| BDI | Belief-Desire-Intention |
| CDA | Continuous Double Auctions |
| HAM | Hierarchy of Abstract Machines |
| HMM | Hidden Markov Model |
| HRL | Hierarchical Reinforcement Learning |
| IOHMM | Input-Output HMM |
| MC-HRL | Monte Carlo HRL |
| MDP | Markov Decision Process |
| MOMDP | Multi-Objective MDP |
| MORL | Multi-Objective Reinforcement Learning |
| NHHMM | Non Homogeneous HMM |
| POMDP | Partially Observable MDP |
| RFQs | Request For Quotes |
| SARSA | State-Action-Reward-State-Action |
| SMDP | Semi-Markov Decision Process |
| TAC | Trading Agent Competition |
| TAC SCM | TAC Supply Chain Management |
| TD | Temporal Difference |

# Chapter 1

# Introduction

Software agents are computer programs that autonomously use environment states as inputs and output actions in order to meet their delegated goals (Russell and Norvig, 2009; Wooldridge, 2009). Their use in improving business processes has been increasing, owing to the progress of information technologies (Dignum and Dignum, 2010; Müller and Fischer, 2014) and to their attractive properties such as autonomy, reactivity, proactivity and social ability that are instrumental in simplifying the implementation of distributed and automated systems. A wide range of techniques and programming languages have been devised to support the development of robust software agents (Ferber, 1999; Girardi and Leite, 2013). Architectures of such agents are generally composed of three types of components:

- *sensors* to identify the states of their environment. An environment that may be observable or hidden, static or variable, deterministic or non-deterministic, accessible or inaccessible and discrete or continuous (Russell and Norvig, 2009),
- a *reasoning engine* to aid the agent in deciding which actions to take in order to achieve its goal and
- *actuators* to execute the agent's selected actions in the environment.

This architectural setting is also widely adopted for building of software agents termed trading agents that interact autonomously in diverse market systems - real or simulated. In these systems, the agents are responsible of any relevant trading tasks or sub-tasks such as buying, selling or searching for the most advantageous service providers (Section 2.1 provides more details for trading agents).

Despite the fact that the adaptation of trading agent's sensors, reasoning component and actuators is essential for its performance in variable and non-deterministic market environments, only the adaptation of sensors using machine learning techniques has attracted many researchers (LeBaron, 2006; He et al., 2006; Pardoe, 2011; Wellman, 2011). Nevertheless, researchers have shown an increased interest in developing adaptive reasoning engines for trading agents acting in simplified market settings using flat reinforcement learning. This kind of adaptation enables the agent to hone its strategies to be as optimal as possible with respect to one decision problem, but it does not allow the agent to generalise its learning through hierarchically related decision processes (Raju et al., 2006; Nevmyvaka et al., 2006; Reddy and Veloso, 2011a; Mahvi and Ardehali, 2011; Peters et al., 2013).

Furthermore, most current techniques that are used for the development of trading agents' reasoning are tailored to a specific market setting; this is the case for the following approaches: the Belief-Desire-Intention (BDI) framework (Fasli, 2001, 2003), fuzzy logic (He et al., 2003, 2006), environment-specific and flat Markov decision process (MDP) models (Reddy and Veloso, 2011b; Peters et al., 2013) or rule-based routines (LeBaron, 2006; Benisch et al., 2009). Since an adaptive trading agent is designed to solve an environment-specific task, it is not able to reuse the acquired trading ability in new markets in order to minimise the time required by the learning. In addition to this, actuators are mainly used as the interface for action execution, with no relevant studies being reported in the software agent community to support their adaptation.

Against this background, this thesis is concerned with the development of a trading agent that can continually adapt its reasoning engine as well as its sensors and actuators to the market changes in order to maximize its overall profit and to reduce the training effort required in new markets. Specifically, this thesis focuses on the development of a reasoning engine that will enable the trading agent to acquire most of its behavioural knowledge through continuous interactions with its environments as well as to reuse the acquired trading skills in new markets without having to learn how to behave from scratch. To achieve this I consider hierarchical reinforcement learning (HRL) and semi-Markov decision processes (SMDP), which have been used in simplified settings to build adaptive robots (Barto and Mahadevan, 2003a; Kober et al., 2013) and, recently, portable agents that can reuse their reasoning engines in new settings (Taylor and Stone, 2009; Lazaric, 2012). This thesis proposes for the first time the use of hierarchical reinforcement learning techniques to build a trading agent which is characterised by

two core properties:

1. *Continuous Adaptability*: Using advanced learning techniques, such as hierarchical reinforcement learning (Barto and Mahadevan, 2003a), the agent's reasoning can learn and adapt continually to changes occurring in its environments. The SMDPs presented in this thesis have been used by the developed broker, AstonTAC to optimise diverse decision problems and to continually adapt to different market settings (see Chapters 3-6).

2. *Portability*: Just like a human trader, AstonTAC is able to reuse its trading experience in different and new market domains using the proposed transfer framework inspired by existing robotic approaches (Taylor and Stone, 2009; Lazaric, 2012) that deal with knowledge transfer in reinforcement learning. The portability of AstonTAC is presented and evaluated in Chapter 7.

The remaining of this chapter is organised as follows. Section 1.1 discusses the major contributions of this thesis. In Section 1.2, the content of the thesis is outlined. A list of publications arising from this thesis is presented in Section 1.3.

## 1.1 Research Contributions

This thesis has three main contributions: the design of an SMDP-based trading agent architecture, the implementation and evaluation of an SMDP-based electricity trader as well as a novel knowledge transfer framework to enable the portability of its trading skills.

### 1.1.1 SMDP-based Trading Agent Architecture

As part of this contribution, a trading agent architecture associates HRL algorithms with a common agent architecture to create a trading agent that can learn and adapt to markets online, while at the same time being able to reuse its trading skills in markets it has not previously encountered - the proposed architecture is introduced in Section 2.3. The market simulations used for evaluation and benchmarking purposes are environments that are assumed to be partly hidden, variable, non-deterministic, inaccessible and continuous. This design approach has been used to implement AstonTAC, which has been tested in a set of diverse trading environments. Different versions of AstonTAC architecture are described Chapters 3-6.

### 1.1.2 HRL-based Electricity Retailers

To illustrate the design approach put forward, novel SMDP-based architectures enable the AstonTAC versions, acting as electricity retailers, to trade simultaneously in wholesale and retail markets. When purchasing electricity from the wholesale markets, the AstonTAC aims to buy the electricity needed at low prices while keeping the imbalance between the energy required and the energy bought low. The challenge of the simulation environment considered, is that the clearing prices, demand and supply volume of the electricity can be very volatile at short term, as renewable energy sources used are mostly weather-dependent.

In view of this, a standard MDP is used for the first time to solve the short-term electricity procurement problem from the retailer's viewpoint. The design of the procurement MDP has focused on relevant market features that remain invariant for a wide range of wholesale market settings. This procurement MDP was evaluated in Power Trading Agent Competitions (Power TAC), a multi-agent simulation environment that supports the development and testing of electricity retailer agents (Ketter et al., 2015) and offers a configurable retail and wholesale markets with real-life features and complexity. During the Power TAC competition in 2012, AstonTAC_V1 was the only broker that could minimise its procurement cost and energy imbalance. Chapter 3 describes the procurement MDP.

Since consumers are able to produce electricity in a smart grid market using solar panels, for instance, the volume of electricity needed by the retailer can vary rapidly at short-term so that the retailer will need to have the ability to buy and sell electrical energy in the wholesale markets. To enable AstonTAC_V1 to buy its energy shortage and sell its energy excess in the wholesale market, the procurement MDP is extended into a procurement SMDP while maintaining the knowledge acquired by the procurement MDP. In order to solve the SMDP, a hierarchical reinforcement learning (HRL) approach called hierarchy of abstract machines (HAM) is applied, as it enables the use of domain knowledge that reduces the learning time needed to solve the SMDP. This work is encompassed within AstonTAC_V2. An evaluation of AstonTAC_V2 in Power TAC shows that when trading for short-term procurement the new SMDP approach outperforms the continuous double auctions (CDA) and the MDP trading techniques. Chapter 4 provides details on the design, implementation and use of the procurement SMDP.

Symmetrically to the procurement SMDP, considering the decision problem faced by a retailer agent in a retail market, a semi-Markov decision process is proposed in

Chapter 5 to optimise simultaneously the agent's market share and profit rate. In order to maximise its profit, the trading agent has two extreme alternatives: one alternative consists of using a maximal profit margin with a minimal number of customers and the other alternative is the use of a minimal profit margin with a maximal number of customers. These extreme alternatives have motivated the creation of two behavioural options for the agent in retail market: "customer-enticing" or "profit-oriented". This agent was named AstonTAC_V3. To evaluate the performance of the proposed approach, AstonTAC_V3 uses the SMDP reasoning engine to compete in Power TAC. During the Power TAC competition in 2013, AstonTAC_V3 was able to keep its profit and market share higher than the other retailers.

Finally, a novel SMDP framework is proposed in Chapter 6 to formalise all the decision problems of the retailer in the retail and wholesale markets. The novel SMDP framework optimises the retail and wholesale strategies, while also optimising the overall retailer's strategy. This SMDP framework makes use of the MDP coarticulation approach of Rohanimanesh (2006) to enable the execution of concurrent actions. This novel SMDP formalisation reuses the knowledge contained in the retail and procurement SMDPs to speed up the learning. The resulting reasoning SMDP is solved with different reinforcement learning techniques depending on the decision problem considered. Using the same SMDP to manage all its trading decisions in the wholesale and retail markets, this version of our agent, AstonTAC_V4, has a better coordination of its retail wholesale actions, which enables it to perform better than the top TAC retailers when the level of interdependence between retail and wholesale market is high. This means that the level of demand in retail market influences the wholesale prices and reciprocally the supply level in the wholesale market influences the retail price.

### 1.1.3 Building of Portable Reasoning Structure

After designing, implementing and evaluating the SMDP-based agent architecture, a novel knowledge transfer framework is put forward in order to enable an electricity retailer to reuse its SMDP reasoning engine in new, different markets. This novel framework extends existing knowledge transfer approaches by combining two knowledge transfer approaches: agent-centric framework of Konidaris et al. (2012) and inter-task mappings of Taylor et al. (2007); Taylor and Stone (2007). The proposed knowledge transfer is considered to be agent-centric, as it uses the same core reasoning system of a trading agent engine to act in different markets. At the same time, it also applies inter-

task mapping techniques to enable the retailer to act in the newly-encountered markets. The knowledge transfer framework is evaluated in the TAC environments and is shown to truly enable trading skills transfer of AstonTAC_V5 to new markets (see Chapter 7 for further details).

## 1.2 Thesis Structure

The remainder of this thesis is composed of eight chapters and an appendix.

Chapter 2 introduces the concepts that are relevant for the understanding of this work: agent architectures, semi-Markov decision processes with HRL and knowledge transfer for reinforcement learning. It also contains a critical review of previous work on development of trading agents.

Chapter 3 describes the formalisation of the short-term procurement problem as an MDP which is solved using a Monte Carlo approach. A critical literature review is presented at the beginning of the chapter. Subsequently, the MDP model and the actual architecture of the retailer agent are described, followed by the evaluation of the AstonTAC_V1's performance in 2012 Power TAC.

Chapter 4 gives details on the extension of the procurement MDP framework presented in Chapter 3 to a procurement SMDP that enables AstonTAC_V2 to alternatively buy and sell in wholesale markets for short-term electricity procurement. AstonTAC_V2 is evaluated in a series of controlled experiments.

Chapter 5 describes the SMDP framework that is proposed to optimise the retail market strategy of an electricity retailer agent. First, existing studies that discuss the development of retail strategies are reviewed. Then, the SMDP framework and the implementation of AstonTAC_V3 are described. AstonTAC_V3 is evaluated in the Power TAC 2013.

Chapter 6 describes the formalisation of the broker's decision problem as a semi-Markov decision problem. This generalised SMDP framework is built using the trading skills acquired by retail and procurement SMDPs. An evaluation compares the performance of AstonTAC_V4 with top TAC brokers.

Chapter 7 presents the proposed knowledge transfer framework and describes the learning algorithm and some insights on how to technically build portable agents. The developed agent, AstonTAC_V5, is tested in the TAC environments Power TAC and TAC SCM.

Finally, Chapter 8 summarises the main contributions of this thesis and highlights

the possible extensions of the thesis.

An appendix complements the thesis with additional information on the implementation of AstonTAC sensors.

## 1.3 Publications from this Thesis

The work contained in this thesis has been very positively received by both the Agents and Multiagent Systems and the Energy and Smart Grid communities. Parts of the work presented in this thesis have been or will be published in the following papers:

1. Chapter 3 was originally published in Kuate et al. (2013):

   Rodrigue T Kuate, Minghua He, Maria Chli, and Hai Wang. An intelligent broker agent for energy trading: an MDP approach. *In Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 234–240, 2013

2. Chapter 5 has, in part, been published in Kuate et al. (2014):

   Rodrigue T. Kuate, Maria Chli, and Hai Wang. Optimising market share and profit margin: SMDP-based tariff pricing under the smart grid paradigm. In *Innovative Smart Grid Technologies Europe, 2014 5th IEEE/PES. IEEE*, 2014.

3. Chapters 6 and 7, in part, appear in Kuate et al. (2015):

   Rodrigue T. Kuate, Maria Chli, and Hai Wang. An efficient knowledge transfer solution to a novel SMDP formalization of a broker decision problem. In *International Conference on Autonomous Agents and Multiagent Systems*, 2015.

# Chapter 2

# Background and Related Work

As this thesis is concerned with the application of HRL to design trading agents, this chapter introduces and reviews the two areas relevant to this work: trading agents in Section 2.1 and HRL in Section 2.2. Section 2.3 discusses the modelling of SMDPs for the trading agents.

## 2.1 Trading Agents

Section 2.1.1 presents background knowledge on agent architecture, whereas studies on trading agent architectures and learning are reviewed in Section 2.1.2. Section 2.1.3 introduces the simulation environments considered in this thesis for evaluation purposes.

### 2.1.1 Software Agents Architecture

Several definitions of software agents can be found in the literature. A generally accepted definition that will be used throughout this thesis is provided by Wooldridge (2009, page 21):

"*An agent is a computer system that is situated in some* ***environment****, and that is capable of* ***autonomous action*** *in this environment in order to meet its delegated* ***objectives***".

To achieve its objectives, the agent proceeds in three steps:

1. It uses its sensors to estimate the state of the environment which is defined by a combination of relevant environment features.

2. Knowing the state of the environment, the agent selects the appropriate action to take using its reasoning system.

3. Finally, using its actuators or effectors, the agent executes the selected action according to the environment rules or mechanisms.

These three components are essential parts of the agent architecture. Depending on the impact of its three core components (sensors, actuators and reasoning engine), the agent can exhibit these four key properties: autonomy, reactivity, pro-activeness and social ability (Wooldridge and Jennings, 1995; Wooldridge, 2009).

- *Autonomy*: an autonomous agent is able to act on its environment without human intervention by autonomously sensing the environment, deciding what to do, how to do it and executing the action in order to achieve the delegated objectives.

- *Reactivity*: a reactive agent responds to the changes occurring in its environment by taking the corresponding actions.

- *Pro-activeness*: a proactive agent creates new goals and takes initiatives in order to meet the delegated objectives.

- *Social ability*: a social agent is able to communicate with other agents in order to achieve its individual or the collective objectives.

These four properties are important characteristics of trading agents, as they are expected to trade autonomously without human intervention. To improve their performance, trading agents need to be able to react to environment changes that are foreseen or unforeseen at design time, while at the same time being able to pro-actively plan the actions to take in order to realise their long-term goals. Furthermore, a trading agents is expected to communicate with other traders in order to buy or sell their goods or services.

In order to enable these four properties, four groups of agent architectures have been explored (Russell and Norvig, 2009; Wooldridge and Jennings, 1995; Ferber, 1999; Wooldridge, 2009; Girardi and Leite, 2013): reactive agent architecture, deliberative agent architecture, hybrid agent architecture and learning agent architecture.

#### 2.1.1.1 Reactive Agent Architecture

A *reactive agent architecture* enables the agent to react to its environment changes by using provided rules that associate an action with each state of the environment.

Many examples of reactive architectures have been presented in the literature: the subsumption architecture of Brooks (1995), the Pengi architecture of Agre and Chapman (1987), the situated automata approach of Kaelbling and Rosenschein (1990) as well as the agent network architecture of Maes (1989). The main motivation for the adoption of a reactive architecture is the fast reaction to environment changes which is a very important feature when building trading agents that may need to react in real time. However, this approach is not appropriate for building a proactive agent that does not only use the current environment state, but also an additional source of knowledge in order to plan their actions for achieving its long-term goals.

#### 2.1.1.2 Deliberative Agent Architecture

A *deliberative agent architecture* is applied for agents that use logical reasoning to decide their goals and achieve them. Contrary to reactive agents which essentially react to their environment, the deliberative agents are proactive by using a knowledge database to plan and execute their actions in attempting to achieve their goals. A deliberative agent is assumed to have a complete knowledge of its world, which remains unchanged during the execution of a plan, and to know the effect of its action that are always successful. The building of deliberative agents that are able to manipulate logical formulae to achieve their goals turns out to be very difficult, as the designer needs to always identify the relevant symbolic representation that will support the agent reasoning. Relevant symbolic representation can be difficult to define, even for common sense reasoning problem. Moreover, the use of this logical representation to act is time-consuming and does not enable the agent to be efficient when acting under a time constraint. Very few studies have been reported on the use of pure deliberative architectures for building trading agents (Fasli, 2001, 2003).

#### 2.1.1.3 Hybrid Agent Architecture

A *hybrid agent architecture* is applied for agents that are both reactive and deliberative. To enable deductive, practical reasoning architecture to support real-time reaction, many hybrid architectures have been studied to create reasoning systems that are reactive and proactive. These architectures present two types of layered structures: horizontal layering and vertical layering. In a horizontally layered architecture each reasoning layer can fully carry out a reasoning process by using information from the agent's sensors and outputting one or more actions for the agent's actuators, whereas

in a vertically layered architecture no single layer fully carries out the reasoning, they collaborate in handling the sensory input and suggesting an output action. A hybrid architecture provides a more appropriate approach for designing trading agents that act for short- and long-term goals. However, most trading agents use pre-programmed rules to identify which actions to take and generally, no logical representation of the environment is provided to infer the logical actions to take. Therefore, most trading agents with a hybrid architecture are more reactive than proactive, as described in Section 2.1.2.2.

#### 2.1.1.4 Learning Agent Architecture

A *learning agent architecture* which characterises any architecture that is augmented with a learning component to improve the adaptation of the agent. Augmenting the agent architecture with learning capability enables the agent to adapt its behaviour to unforeseen environment changes (Ferguson, 1992; Russell and Norvig, 2009). The agent can be equipped with two types of learning capability (1) logic-based learning which enables the agent to create new rules using a set of rules provided at design time (Juba, 2013; Muggleton and Lin, 2013) or (2) machine learning techniques, which generally include regression models, classification models, decision trees models and reinforcement learning algorithms (Kazakov and Kudenko, 2001; Tan et al., 2011; Singh, 2011; Sniezynski, 2014).

The reactive, the deliberative as well as the hybrid agent architectures as presented above are based on predefined, preprogrammed logic and rules, which need to be specified at design-time and do not enable the agent to learn and adapt to unforeseen environment changes. As the trading agent's environment can be variable and dynamic, it is essential for its performance to be able to learn and adapt to its environment. As reviewed in the next subsection, many trading agents have been equipped with learning capability in order to adapt and to perform well in their environments.

### 2.1.2 Trading Agent Architecture

In electronic markets, trading agents are generally used as buyers or sellers that act in electronic markets on behalf of their owners (Chen et al., 2012; Geanakoplos et al., 2012). Seller agents enable their owners to optimise their sale strategy either through improvement of their online advertisement strategy or through computing of competitive prices for their services and goods. Analogously, buyer agents optimise their owners' purchase strategy given their preferences which can be defined by the prices or/and

the quality of the offered products or services. This thesis uses electricity trading agents (Rogers et al., 2012; Bach et al., 2012; Reddy, 2013; Amato et al., 2015) to illustrate the proposed learning approach as outlined in Section 1.1.

In order to understand or predict behaviour of the complex markets such as the financial market, new research disciplines have been created: agent-based computational economics (Tesfatsion, 2002; Tesauro and Kephart, 2002; MacKie-Mason and Wellman, 2006; Richiardi, 2012) and agent-based computational finance (Hommes, 2006; Samanidou et al., 2007). In both scenarios, a trading agent makes use of its three core components (sensors, reasoning engine and actuators) in order to act autonomously.

This thesis aims to design and evaluate a reasoning engine that enables trading agents to perform well by taking appropriate real-time trading decisions in dynamic and uncertain markets. In view of this, we use test environments that simulate dynamic and uncertain markets and that impose to the agent the need to act quickly. AstonTAC is developed and tested in well-known simulation environments of the Trading Agent Competition (TAC) community, which is an international forum whose aim is to support the development of intelligent agents through agent competitions. Since 2000, it has attracted many research groups that evaluated the performance of their agents in competitive environments such as TAC SCM, TAC Ad Auctions or Power TAC. AstonTAC was placed in two different environments: TAC SCM and Power TAC. While Power TAC simulates a smart grid electricity market which encompasses both a competitive electricity wholesale and a retail market under the smart grid paradigm (Ketter et al., 2015), TAC SCM simulates a PC market (Arunachalam and Sadeh, 2005; MacKie-Mason and Wellman, 2006). These two environments are characterised by common features that have impacted the development of AstonTAC.

1. *Two market*s: Retail and wholesale. A trading agent acting in the two environments trades simultaneously in the retail and in wholesale markets, which are interdependent markets, as the level of the retail demand influences the wholesale prices and the level of wholesale supply influences the retail prices. Despite this interdependence, the two markets present two different optimisation problems to the trading agents, which aim to attract as many clients as possible and make as much profit as possible in the retail market, while in the wholesale market, they aim to buy the products needed on time for their customers and at lowest possible price.

2. *The structure of the retail markets*: To act optimally, the trader needs to under-

stand the customer expectations and offer competitive prices in order to attract customers and make profit. However, the retail market offers limited information, therefore the agent cannot easily predict the customer expectations and demand or the behaviour of competitor agents. Agent designers generally use diverse techniques for forecasting the retail behaviour based on the limited amount of information. The trading agent uses the price as an essential tool to attract customers and aims to maintain a good reputation among customers.

3. *The structure of the wholesale markets*: In the wholesale market, the trader agent mainly acts as a buyer who needs to buy the required products in order to meet the predicted retail demand. The agents compete with others traders to buy the products offered by the suppliers at the lowest prices possible. In a manner analogous to the retail market, the agent has limited access to the market information needed to predict the market supply and the behaviour of other trading agents and supplier agents.

4. *Market dynamics*: The agents face an environment with limited access to relevant information. The market dynamic is created by the behaviour of the different agents in the environment. The customer behaviours influence the demand level in the retail market and sometimes their preferences to the offered products or services, whereas wholesale trader behaviour influences the supply level. The behaviours of other agents have an impact on the state of the markets as each agent tries to control the environment using the offered prices.

Section 2.1.3.1 provides an overview of each evaluation environment: Power TAC and TAC SCM.

#### 2.1.2.1 Trading Agent Sensors

Designers of trading agents apply diverse machine learning techniques to improve the performance of the agents' sensors in determining the state of environment which is commonly specified by the variables: projected market prices, demand and supply. As the focus of this thesis is on the reasoning engine, this section mainly outlines techniques that have been used for the implementation of the trading agents' sensors without providing cross-domain comparisons as the sensors are environment-specific components.

In the Santa Fe Artificial Stock Market (SFASM) simulation that has been created to investigate the dynamics of stock markets and to test trading strategies (Palmer et al.,

1994; LeBaron, 2002), the trading agent applies time series models to estimate the state of the market (LeBaron, 2001, 2006).

Considering the trading agent competition (TAC) environment, several simulation environments have been developed to promote research on trading agents. In the TAC Supply Chain Management (SCM) environment, which simulates the trading of personal computers, the trading agents use diverse approaches to estimate the state of the environment (Wellman et al., 2005; MacKie-Mason and Wellman, 2006). Most of the brokers employ the forecast of retail prices and retail demand as inputs for their decision making; these forecasting techniques vary from simple average of previous observations to multi-linear regression approaches. A price average is used by He et al. (2006) as input to fuzzy-based reasoning. The retail mean price is used by Ketter et al. (2006) to determine the market state. The k-nearest neighbours algorithm is applied by Kiekintveld et al. (2009) to forecast the retail prices, whereas in the wholesale market, a linear regression is performed to predict suppliers' future prices. A further technique, the distribution tree is used by Benisch et al. (2009) to predict the retail prices. Pardoe (2011) uses transfer learning to improve the performances of the linear regression models used to predict the relevant environment variables such as prices, demand and supply level in new TAC markets.

Similar to TAC SCM, trading agents in Power TAC (Ketter et al., 2015), which is a simulation environment for electricity trading, make decisions based on their estimation of the future prices, demand and supply quantities. Urieli and Stone (2014) use linear regressions to predict future customers' energy demand and energy procurement cost which are used as inputs for deciding the agent's tariff prices in the retail market. In Reddy and Veloso (2011a,b), the retail price status and the structure of the agent portfolio are used as inputs of the reasoning structure. Peters et al. (2013) propose to use of feature selection and regularisation techniques (Petrik et al., 2010) to identify the environment variables that are relevant for the trading agent's decision making.

These studies on the sensory techniques perform well in the market setting considered by the authors. The sensors are generally designed in an environment-specific manner, because there are interface components that are responsible for gathering the market data into information that is usable by the agent's reasoning. As this thesis focuses of the enhancement of the reasoning engine and not on the sensory component, the full description of the AstonTAC's sensors is provided in Appendix A.

#### 2.1.2.2 Trading Agent Reasoning Engine

A wide range of techniques have been proposed to design the reasoning engine: both static and adaptive approaches. While static approaches design rule-based and non-adaptive reasoning engine, adaptive approaches propose reasoning engine which can adapt their behaviour to environments changes.

**Static approaches** Rule-based subroutines are used to map environment states to possible market actions. In the SFASM environment, LeBaron (2001, 2006) designed agents that use a rule-based reasoning engine to trade by mapping specified trading rules to market states. Using game theoretical approaches (Binmore, 1990; Fudenberg and Tirole, 1991; Binmore, 2007), the behaviour of the trading agent is codified by game-theoretical rules that are predefined offline (Kiekintveld et al., 2004; Kiekintveld, 2008; Liefers et al., 2014).

When the agent designer does not know or has a limited knowledge of the rules required for it to perform well, the building of a rule-based reasoning engine is difficult. Moreover, in a dynamic environment as the financial market, rule-based agents, which use trading strategies that are predefined and preprogrammed at design time, cannot adapt to unforeseen market changes, whereas agents with adaptive reasoning can.

**Adaptive Approaches** In the TAC environment, trading agents use utility functions to identity their trading prices as a function of historical market prices and the levels of demand and supply (He et al., 2006; Pardoe, 2011; Urieli and Stone, 2014). The drawback of this approach is that the utility functions are tailored to the agent's environment and generally do not give any information on real trading strategy, as more powerful mathematical models can be used to design utility functions for real-life trading (Heij et al., 2004; Waters and Waters, 2008; Wooldridge, 2010).

Many trading agents use evolutionary algorithms to identify the suitable trading rules given a set of market simulations with predefined settings (LeBaron, 2001, 2006). Evolutionary algorithms are also used as tools in trading markets to identify the optimal trading rules (Lohpetch and Corne, 2009; Wilson and Banzhaf, 2010; Hassan, 2010; Lohpetch et al., 2011; Esfahanipour and Mousavi, 2011). The main drawback of evolutionary algorithms is the time required to find the suitable rules, which makes these approaches inappropriate for trading agents that act with time constraints.

**Reinforcement Learning as an Adaptive Approach** Reinforcement learning offers a much more structured technique for enabling efficient agent adaptation (Sutton and Barto, 1998). In simplified market settings, trading agents that use flat MDPs and the reinforcement learning for decision making have been presented (Raju et al., 2006; Reddy and Veloso, 2011b,a; Peters et al., 2013). Flat MDPs are not appropriate for the type of markets considered, as the resulting reasoning MDPs can be very complex and difficult to solve. This thesis remedies this limitation by applying SMDP and HRL to model and solve complex decision making problems as faced by trading agents that act in the environments considered. Furthermore, SMDP and HRL not only enable the trading agents to adapt to variable and uncertain markets, but they also enable the agent to reuse the knowledge acquired without having to learn from scratch. The background knowledge on SMDP and HRL is provided in Section 2.2. Critical reviews of the reasoning engine of trading agents are presented for each decision problem considered in the thesis. First, Section 3.3 reviews work on decision making in wholesale market. Then, in Section 5.3, the techniques used by trading agents to optimise the decision making in retail markets are presented. Finally, Section 6.2 contrasts studies on decision making for trading agents.

#### 2.1.2.3 Trading Agent Actuators

Similar to the the sensors, actuators are generally designed for a specific environment, however, far too little attention has been paid to the implementation of trading agents' actuators, whereas the implementation of robots' actuators has a large volume of published studies (Khatib and Burdick, 1986; Hunter et al., 1991; Zinn et al., 2004). When designing trading agents that need to continually adapt to environment changes, there is a need to develop intelligent actuators that can optimise the execution of the selected actions. Moreover, intelligent actuators can handle and adapt to environment-specific changes that may affect the execution of the trader's action, while the invariant trading knowledge is managed by the reasoning component.


Figure 2.1: Power TAC Overview from Ketter et al. (2015). This is an overview of the Power TAC simulation environment that simulates an electricity wholesale market with large electricity suppliers and a smart grid retail market with retail consumers and producers. The electricity retailers that act as brokers compete in wholesale and retail markets for buying and selling electrical energy. A distribution utility is responsible of electricity balancing between supply and demand.

### 2.1.3 Trading Environments

#### 2.1.3.1 Power TAC Environment

Power Trading Agent Competition platform Ketter et al. (2015) (Power TAC[1]) offers an open and competitive environment that is as close as possible to the complexity of real-life markets, encompassing both a competitive electricity wholesale and a retail market under the smart grid paradigm (see Figure 2.1). While the wholesale market mimics day-ahead energy markets such as Nord Pool in Scandinavia or FERC in North America, the retail market simulates several types of real-life energy consumers and tariffs. The broker agents offer diverse tariff contracts to the customers and provide them the required energy each hour by buying the needed volume from the wholesale market.

**Wholesale Market** The wholesale market, which is composed of generator companies (GenCos), energy brokers and transmission operators, called Independent Systems

[1]http://www.powertac.org; accessed 16-November-2015

Operator (ISO), simulates a periodic double auction and day-ahead market where the market is cleared each simulated hour. The market is cleared when there is an equilibrium price between the bids and the asks placed to the market. Successful bids are buy orders that have a limit price higher than the clearing price and successful asks are sell orders that have limit price lower than the clearing price. The broker agent can buy energy for future delivery between 1 and 24 hours ahead in the wholesale market. This gives the broker a time horizon of 1 to 24 hours ahead to satisfy the hourly demand for the contracted power consumers.

**Retail Market** The retail market provides several types of customers that can be grouped in two classes: (1) small and elemental customers, such as households, small and medium businesses, small energy producers and electric vehicles; and (2) large customers, such as greenhouse complexes and manufacturing facilities. A Distribution Utility owns the distribution network and ensures real time energy balancing between supply and demand. Moreover, it charges the broker with a penalty, if it cannot balance its electricity demand and supply. In the retail market, the agent can offer different types of consumption and production tariffs with real world tariff features: periodic payments, time-of-use tariffs, tiered rates, sign-up bonuses and early withdrawal fees, as well as dynamic pricing. Power TAC customers select the energy tariffs that are the most advantageous considering their energy requirements.

**Retailer Agent Activities** In the Power TAC environment, the retailer agent is exposed to series of activities that can be executed during the game (this is illustrated in Figure 2.2). The agent's activities encompass the recording of environment data contained in the public and private server messages and the sending of messages to the game server. The server messages are the agent financial states (market position, cash position and transactions), weather report and forecast, the balancing state between retailer's energy supply and demand as well as all tariff information (customer subscription, cancellation and tariffs' transaction). The agent needs to use this information to adapt to the environment changes by optimising the tariff pricing in retail market.

At any time in the game, the agent can take actions in the two markets. In the wholesale market, it can buy or/and sell electricity by submitting bids and asks to the market. As feedback, it is informed by the game server about the clearing prices, the order book and its successful orders. In the retail market, the trading agent deals with managing a portfolio of electricity tariffs for the consumers by creating new or updating

existing tariff contracts given the retail strategy that it follows: the broker can offer tariff contracts for consumption, production, storage and balancing. The retail customers select and sign to diverse tariffs according to their preferences and the reputation of the trading agents. These activities have influenced the development of AstonTAC versions, as AstonTAC_V1 presented in Chapter 3 and AstonTAC_V2 in Chapter 4 deal with the optimisation of the wholesale activities, whereas AstonTAC_V3 described in Chapter 5 optimises the retail strategy. The wholesale and retail markets strategies are optimised simultaneously by AstonTAC_V4, which is presented in Chapter 6.

**Tournament Settings and AstonTAC Approach** Each Power TAC game is initialised with a setup game that lasts for 15 days (384 simulated hours or timeslots) and procures initial simulation data, which is composed of customer energy consumption, customer energy production, clearing prices and weather reports. A Game generally runs for about 60 virtual days or 1440 simulated hours or timeslots - each simulated hour lasts 5 seconds in the real world. At the end of the simulation, the broker with the highest balance in the bank wins the game. Starting the game with an empty bank account, the broker agent gets a bank credit by buying energy in the wholesale market. It earns money by getting payment for the sold energy and spends it by paying for the required energy or market fees (energy distribution fee, energy imbalance fee, tariff publication fee, tariff revocation fee and the bank interest for the debt). During the game, the bank always loans the broker money to purchase energy and charges interest.

The dynamism and uncertainty of the Power TAC environment is a result of the definition of the implementation of the game server:

1. The *number of agents and the type* of agents competing in each game are variable. As result, the behaviours of wholesale and retail markets depend on the agents playing in a specific games. A basic way to develop a trading strategy is to create a profile of the competing agents and use game-theoretical approach to select a trading strategy as suggested by Vidal and Durfee (2003); Gmytrasiewicz and Durfee (2000). However, there are two main limitations to this approach. First, the game server does not offer enough information to confidently profile the agents competing in a competition. This makes the creation of agent profiles difficult to realise. Moreover, as adaptive agents adjust their behaviour during and between the tournament games, game-theoretical approaches are difficult to apply successfully. Additionally, as using a game-theoretical approaches to determine each decision making is time consuming, it is not an efficient approach for Power


Figure 2.2: Activities of the Retailer Agent from Ketter et al. (2015). This figure shows the interactions between the retailer agent and the Power TAC simulation environment. The simulation game sends at every time step diverse information to the retailer agent about the weather state, the transaction in the wholesale and retail markets, as well as its financial situation.

TAC where real-time decision making is required (for instance the agent may be required to make a wholesale market decision within 5 seconds).

2. The *wholesale market trading* as well as *retail consumption* and *production* are influenced by the weather parameters which are variable and drawn from real weather history of real location on the planet. This makes the energy production and consumption behaviour of each game to be specific and influenced by the real-world weather data of a specific location in the world. Therefore, the weather makes the dynamic of the game to be different from game to game making the prediction of the game dynamic difficult. The Appendix A explains the approach used by AstonTAC to dynamically predict the consumption and production online in each game.

3. The *initialisation of customers type and number*, which differ from game to game, is also a source of uncertainty in the environment, particularly in the retail market. Similar to real-world case, large customers prefer variable tariffs as they pay only for what they consume, while elemental customers prefer fixed tariffs as they cater for fixed and regular payments without risk of extremely high electricity bill to pay. As elemental and large customers behaviours are different both in the amount of energy they consume and their tariff preference, the trading agent needs to adapt its tariff contracts and procurement strategy according to mix of customers in each game.

These environment specifications have largely influenced the implementation of AstonTAC and strengthened the need for building the proposed SMDP architecture.

The Power TAC tournament is held every year and enables researchers around the world to build and test their retailer agents in a competitive environment. During the Power TAC tournament, the agent attends several competitions where it plays several games with different number of agents (two, four or eight). At the end of each Power TAC simulation, the retailer agent with the highest standard score ($z$-scores) of the accumulated profits wins the game. The $z$-scores use the standard normal distribution to show how well a player performs compared to all other players (Marx and Larsen, 2006).

#### 2.1.3.2 TAC Supply Chain Management

While the Power TAC has been used to test the SMDP-based architecture of the trading agent, TAC Supply Chain Management has been used to demonstrate the knowledge


Figure 2.3: The scenario of the TAC Supply Chain Management from Collins et al. (2006) . The retail agent has four main activities: (1) buying computer parts from the suppliers, (2) assembling the computers, (3) selling the computers to the customers as well as (4) scheduling their delivery.

transfer of the developed MDP modules.

TAC Supply Chain Management simulates a supply chain scenario in which six broker agents compete in selling 16 types of personal computers (PC) in a retail market, as well as in buying the computer components from suppliers and schedules PCs production and delivery Arunachalam and Sadeh (2005); Collins et al. (2006). Similar to the Power TAC, this game has a retail and a wholesale market as illustrated in Figure 2.3. In the retail market, the trading agent competes in selling different types of PCs by bidding on customers' request for quotes (RFQs) . The PCs differ by their components (CPUs, Motherboards, memory and disk drives), which are available in different brands. To manufacture the PCs, the trading agent buy the components from the wholesale market using RFQs.

**Wholesale Procurement** Using the RFQs, the trading agent sends at most five requests to each supplier individually and specifies in the RFQ the product needed, the quantity, the delivery date and the limit price. In turn, each supplier considers its future delivery commitment and capacity in order to send an offer with the quantity of the product that can be provided and its price. The trading agent can then accept or reject the suppliers' offers. The quantity of products needed by the trading agent in the future is defined by its estimation of the future retail demand and current retail commitments. Because of the late delivery penalty in the retail market, the retail agent strives to buy the components on time from the wholesale market in order to avoid the penalty.

Moreover, the agent does not only face the time constraint, it also has to buy when the prices are low.

**Retail Market** In the retail market, the trading agents compete by bidding on the customer RFQs while having to satisfy the customer orders on time given their production capacity and scheduled deliveries. The customers' RFQs are composed of the type of the PC needed, the quantity and the due date. After receiving the brokers' offers, the customers select the most suitable offers and send orders back to the corresponding broker.

**Trading Agent Activities** The trading agent needs to excel in retail market, wholesale procurement and production scheduling in order to be successful in TAC SCM. The game server provides to the broker market information such as minimum and maximum prices of PCs and of the needed components. The bank account information and the negotiation orders and offers are sent privately to each broker.

**Game Settings** The game lasts for 220 simulated days with each day representing roughly 15 seconds in real time. At the end of the game the brokers with the highest cash in the bank account wins the game.

## 2.2 Hierarchical Reinforcement Learning

This section introduces the (S)MDP formalisms that have been used to model behavioural modules as well as the techniques used for learning trading strategies in Sections 2.2.1 - 2.2.3.

### 2.2.1 Introduction to Markov Decision Process

When modelling decision problems in dynamic and uncertain environments, MDPs are the most effective and widely accepted approaches (see Bellman (1957); Witten (1977); Whittle (1982); Ross (1983); Littman (1996)). The standard discrete-time MDP with finite state and action spaces can be formally defined with the tuple $\langle S, A, P, R \rangle$ as follows:

- $S$ is the set of environment states. The state of the environment at time $t$ is noted $s_t \epsilon S$. In the standard MDP, the states of the environment are assumed to be fully

observable. This means that the decision maker knows the exact state $s_t$ of the environment.

- $A$ is the set of actions that can be taken by the agent. The action at time $t$ is noted $a_t \epsilon A$. The agent uses the action $a_t$ to control the environment and achieve its goals. At each run or episode, the agent takes a sequence of actions that optimises the control of the environment. The sequence of actions for each run is defined by the policy function:

$$\pi : S \rightarrow A.$$

The deterministic policy $\pi$ associates to each state $s_t$ the action $a_t$, so that:

$$a_t = \pi(s_t).$$

- $P$ is the state transition distribution. It defines the probability $P(s_{t+1}|s_t, a_t)$ of transition from state $s_t$ to state $s_{t+1}$ after the agent has taken the action $a_t$. The transition to state $s_{t+1}$ depends only on the current state $s_t$ and the taken action $a_t$. $P(s_{t+1}|s_t, a_t)$ does not depend on the previous actions and previous states.

- $R$ is the reward function. The decision maker receives a positive or a negative reward for being in a certain state of the environment. At each state $s_t$ of the environment, the agent receives the reward $r_t$ from the system after the action $a_{t-1}$ is taken from state $s_{t-1}$. The reward function of the MDP is used to control the aims of the agent training. While a positive reward encourages the decision maker to take in the future the same decision in the same situation, a negative reward discourages him or her to take the same decision in the same situation. Consequently, the agent gradually takes more successful decisions in order to get the maximum reward. The aim of the decision maker is to find the policy $\pi$ that maximises the return (cumulated rewards) $R$ in the long run. At each time step $t$, the cumulative discounted return $Return_t$ is defined by:

$$Return_t = r_t + \gamma^1 r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^T r_T, \tag{2.1}$$

$$Return_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{2.2}$$

- where $\gamma$ is the discount factor, $0 \leq \gamma \leq 1$ and $T$ the final time step of an episode. The end of an episode is the end of the decision process. In an infinite-horizon decision process, $T \rightarrow \infty$ and $0 \leq \gamma < 1$. $\gamma$ specifies the relationship between the future rewards and the current state-action pair $(s_{t-1}, a_{t-1})$: if $\gamma = 0$, then $R_t = r_t$, the action selection are driven by the immediate reward $r_t$ and the future rewards are irrelevant for agent's decisions. If $\gamma = 1$ for finite-horizon decision problem, then all future rewards (delayed rewards) influence the selection of action $a_{t-1}$ at state $s_{t-1}$.

Following $\pi$, at each state $s_t$, the agent selects the action $\pi(s_t) = a_t$. The expected cumulative value of the future rewards $V^{\pi}(s_t)$ is the state-value function corresponding to a deterministic policy $\pi$ defined as follows:

$$V^{\pi}(s_t) = E\left[Return_t | s_t\right]. \tag{2.3}$$

$$V^{\pi}(s_t) = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t\right]. \tag{2.4}$$

The associated Bellman equation is defined as:

$$V^{\pi}(s_t) = \sum_{s_{t+1}} P(s_{t+1}|s_t, \pi(s_t))\left[R(s_{t+1}|s_t, \pi(s_t)) + \gamma V^{\pi}(s_{t+1})\right], \tag{2.5}$$

where $R(s_{t+1}|s_t, \pi(s_t))$ is the expected reward with $R(s_{t+1}|s_t, \pi(s_t)) = E\left[r_{t+1}|s_t, \pi(s_t)\right]$.

The optimal policy $\pi^*(s_t)$ is any policy that maximises the value $V^{\pi}(s_t)$ at state $s_t$. The maximal value of $V^{\pi}(s_t)$ is denoted $V^*(s_t)$:

$$V^*(s_t) = \max_{\pi} V^{\pi}(s_t). \tag{2.6}$$

If all the parameters of the MDP $\langle S, A, P, R\rangle$ are known, $\pi^*(s_t)$ and $V^*(s_t)$ can be computed using dynamic programming techniques (Bellman, 1957; Watkins, 1989b; Bertsekas et al., 1995; Bertsekas and Tsitsiklis, 1995). However, for the simulation environments considered in this thesis, the parameters $P$ and $R$ are not known and are to be learned by agent through direct interactions with the environment using reinforcement learning techniques. Considering the complexity of the evaluation test environment, as presented in Section 2.1.3.1, it is difficult to obtain enough relevant data to calculate the parameters $P$ and $R$.

Analogue to the state-value function $V^{\pi}(s_t)$, an action-value function $Q^{\pi}(s_t, a_t)$ can be considered to model Markov problems where the $Return_t$ does not only depend on the on the state $s_t$, but also on the taken action $a_t$. Similarly, following equations can be written:

$$Q^{\pi}(s_t, a_t) = E\left[Return_t | s_t, a_t\right], \tag{2.7}$$

$$Q^{\pi}(s_t, a_t) = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t, a_t\right], \tag{2.8}$$

$$Q^{\pi}(s_t, a_t) = \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t)\left[R(s_{t+1}|s_t, a_t) + \gamma Q^{\pi}(s_{t+1}, a_{t+1})\right], \tag{2.9}$$

where $R(s_{t+1}|s_t, a_t)$ is the expected reward with $R(s_{t+1}|s_t, a_t) = E\left[r_{t+1}|s_t, a_t\right]$. The action-value function $Q^{\pi}(s_t, a_t)$ is more suitable for solving MDP through interactions with the environment. The optimal action-value function $Q^*(s_t, a_t)$ is defined as:

$$Q^*(s_t, a_t) = \max_{\pi} Q^{\pi}(s_t, a_t). \tag{2.10}$$

The definition of $Q^{\pi}(s_t, a_t)$ is more meaningful for the application of diverse reinforcement learning techniques than for the formal definition of the MDP. The following section will introduce reinforcement learning.

### 2.2.2 A Primer on Reinforcement Learning

Reinforcement learning is a computational framework for learning MDP through direct interaction with the concerned environment. A reinforcement learning algorithm tries to approximate $\pi^*$ by learning suitable mappings between environment states $s_t \epsilon S$ and the actions $a_t \epsilon A$ given the observed rewards $r_t$. Sutton and Barto (1998, page 3) defines reinforcement learning as follows:

"*Reinforcement learning is learning what to do – how to map situations to actions– so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward, but also the next situation and, through that, all subsequent rewards. These two characteristics – trial-and-error search and delayed reward – are the two most important distinguishing features of reinforcement*

*learning*."

Reinforcement learning algorithms are characterised by the two fundamental properties: the trial-and-error search and the temporal assignment of the observed rewards to the visited states and actions taken. Under the trial-and-error search approach, several techniques have been proposed to enable the agent to adjust the trade-off between exploring the possible (if not all) state-action mappings and exploiting suitable state-action mappings. While the exploration enables the agent to find better policy, the exploitation supports the maximisation of expected rewards. The most common exploration-exploitation approach is the $\varepsilon$-greedy action selection (Watkins, 1989b). At each state $s$, an agent takes the action $a$ with the highest expected return with respect to $\varepsilon$-greedy. With a probability of $\varepsilon$, a random action is chosen and with the probability of $1-\varepsilon$, the action with the maximum expected value is selected. Generally, the value of $\varepsilon$ varies according to the number of training games. At the beginning of the training it has a high value to encourage exploratory action selection and toward the end of the training, it is decreased to enables more exploitation of successful policy. The $\varepsilon$-greedy approach is generalised using the softmax action selection (Bridle, 1990) which uses a Gibbs, or Boltzmann, distribution to select the next action with a probability:

$$P(a|s_t) = \frac{\exp(Q(s_t,a)/\tau)}{\sum_{i=1}^{k}\exp(Q(s_t,a_i)/\tau)}, \tag{2.11}$$

where $k$ is the number of actions available in state $s_t$, $Q(s_t,a)$ is the expected cumulative value of the future rewards when the action $a$ is taken at state $s$, $\tau$ is called the temperature and $\tau > 0$. With a high value of $\tau$, actions are selected more randomly, whereas, with low value of $\tau$ the actions are selected greedily according to $Q(s_t,a)$.

The method of assigning observed rewards to experienced state-action pairs is the core of reinforcement learning techniques, which specifies when to update the estimate $V^{\pi}(s_t)$ of the state values. The update of $V^{\pi}(s_t)$ can be done after any action taken by the agent, at the end of the of the episode or after a certain number of actions. Most reinforcement learning techniques used to solve flat MDPs can be formalised as a temporal difference (TD) learning method or with eligibility trace called TD($\lambda$) methods. $\lambda$ is the trace-decay parameter ($0 \leq \lambda \leq 1$). With $\lambda = 0$, the values of $V^{\pi}(s_t)$ are updated after each action $a_t$ follows by a reward $r_{t+1}$, whereas $\lambda = 1$, the update of the estimation $V^{\pi}(s_t)$ of all visited state-action pairs happen at the end of the episode. The following subsections will provide more details on the TD(0), TD(1) and TD($\lambda$)

#### 2.2.2.1 TD(0)

The reinforcement learning algorithms TD($\lambda = 0$), one-step Q-learning and one-step State-Action-Reward-State-Action (SARSA) update the estimates of $V^{\pi}$ at each step of the decision decision process. One-step Q-learning (Watkins, 1989b; Watkins and Dayan, 1992) is an off-policy TD control algorithm, which updates its state-value functions $Q(s_t, a_t)$ as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right], \tag{2.12}$$

where $\alpha$ is the learning rate, which defines how much the new observation $r_{t+1}$ will influence the existing estimate of $Q(s_t, a_t)$.

Q-learning is an offline policy, because the algorithm is initialised with maximal value of $Q(s_t, a_t)$, which is provided offline. This assumes that the algorithm designer has a good estimation of the state-rewards $r_t$ that are used to compute $\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$. The SARSA algorithm (Rummery and Niranjan, 1994; Sutton, 1996), which was initially proposed as a modified Q-learning, is an online TD method, where the $Q(s_t, a_t)$ are to be learned online as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right]. \tag{2.13}$$

Algorithm 1 presents the learning protocol of one-step SARSA as described in Sutton and Barto (1998). At the beginning of the learning protocol the initial state and action are defined given the exploration-exploitation approach (e.g., $\varepsilon$-greedy). At each state $s_t$ of an episode, the next action $a_t$ is chosen and taken (Line 6) based on the exploration-exploitation approach (e.g., $\varepsilon$-greedy). Then at state $s_{t+1}$, the reward $r_{t+1}$ is observed and the estimation of the action-value function $Q(s_t, a_t)$ is updated in Line 8 following Equation 2.13. This update is carried on for each visited $s_t$ until the end of the episode.

#### 2.2.2.2 TD(1)

As a TD(1) method, Monte Carlo methods (Singh and Sutton, 1996; Barto and Duff, 1994) are appropriate to learn episodic MDP tasks, where the rewards are observed at the end of the episode. Therefore, the update of state-action values has to happen at the

**Algorithm 1** One-Step SARSA as TD(0)

1: Initialise $Q(s_t, a_t)$
2: **for** each episode **do**
3: Initialise $s_t$
4: Choose $a_t$ from $s_t$ using policy derived from Q (e.g., $\varepsilon$-greedy)
5: **for** each step $t$ of the episode **do**
6: Take action $a_t$ and observe $r_{t+1}$ and $s_{t+1}$
7: Choose $a_{t+1}$ from $s_{t+1}$ using policy derived from Q (e.g., $\varepsilon$-greedy)
8: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$
9: $s_t \leftarrow s_{t+1}$
10: $a_t \leftarrow a_{t+1}$
11: **end for**
12: **end for**

end of the episode:

$$Q(s_t, a_t) \leftarrow Average(r_t). \tag{2.14}$$

Sutton and Barto (1998) describes Monte Carlo methods leaning protocol as applied in reinforcement learning. Based on this, Algorithm 2 describes a standard Monte Carlo technique where the learning happens at the end of each episode. Before starting with the learning, the $Q$-values and the behaviour policy $\pi(s)$ are initialised. After each completed episode, the experienced delayed reward $R$ is used in Line 9 to update the state-action values $Q(s_t, a_t)$ of the visited state-action pairs $(s_t, a_t)$ according to equation 2.14. In Algorithm 2, the actions are selected greedily given the $Q$-values, although a $\varepsilon$-greedy policy can also be used.

#### 2.2.2.3 TD($0 < \lambda < 1$)

If the learning does not happen after each action and nor at the end of the episode, the TD methods can be implemented as $n$-step TD methods (Watkins, 1989b), which assigns the observed rewards $r_{t+n}$ to the last $n \epsilon \mathbb{N}$ visited state-action pairs ($s$, $a$) as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[R_t^{(n)} - Q(s_t, a_t)\right], \tag{2.15}$$

with $R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^{n-1} r_{t+n} + \gamma^n Q(s_{t+n}, a_{t+n})$.

**Algorithm 2** Monte Carlo Method as TD(1)

Initialise, for all $s \in S$, $a \in A$
$Q(s,a) \leftarrow$ arbitrary
$\pi(s) \leftarrow$ arbitrary
$Returns(s,a) \leftarrow$ empty list
**for** each episode generated using $\pi(s)$ **do**
    **for** each pair of $(s,a)$ appearing in the episode **do**
        $R \leftarrow$ return following the first occurrence of $(s,a)$
        Append $R$ to $Returns(s,a)$
        $Q(s,a) \leftarrow$ average($Returns(a,s)$)
        **for** each $s$ in the episode **do**
            $\pi(s) \leftarrow \arg\max_a Q(s,a)$
        **end for**
    **end for**
**end for**

The drawback of the $n$-step TD approach is that it is computationally expensive for large $n$. The TD($0 < \lambda < 1$) remedies to this problem by using eligibility traces.

SARSA($\lambda$) (Rummery and Niranjan, 1994) and Q($\lambda$) (Watkins, 1989b; Sutton and Barto, 1998) enable the agent to learn as with the $n$-step TD methods before the end of the episode, using the eligibility traces, annotated as $e(s,a)$ for state $s$ and action $a$. The eligibility traces associate with each state a value, called trace, which corresponds to their contribution to the current reward. Proportionally to the eligibility traces, each reward is distributed to the states that are visited during the learning period.

In this thesis different reinforcement learning algorithms are applied to solve the different decision problems faced by the trading agent. In Chapters 3 and 4 a Monte Carlo approach is adopted to optimise the wholesale procurement. SARSA($\lambda$) is applied in Chapter 5 to optimise the retail pricing, whereas an $n$-step TD approach is used in Chapter 6 to learn an overall trading strategy.

### 2.2.3 Hierarchical Reinforcement Learning Algorithms

Using MDP, the actions are selected and executed completely during the same time step; this does not enable the formalisation of a temporally extended agent's action. The SMDP, which is an extension of the MDP with a temporal abstraction (Jewell, 1963;

Howard, 1971), is used to structure the agent activities as a hierarchy of high-level and low-level MDPs.

In the field of artificial intelligence, Bertsekas and Tsitsiklis (1989); Dean and Lin (1995); Lin (1997) have presented the early studies on decomposition of a complex MDP in loosely coupled subproblems. Fikes et al. (1972); Korf (1985); Iba (1989) show that search problems such as complex MDPs that are difficult to solve when using primitive operations, can be solved faster when searching in macro-operator space. Based on this idea of macro-operator, complex MDP actions have been abstracted as a hierarchy of abstract actions (Singh, 1992a,b) or of behaviours (Brooks, 1986; Huber and Grupen, 1997), of skills (Thrun and Schwartz, 1995), of macro-actions (Sutton, 1995; McGovern et al., 1997), of options (Sutton et al., 1999a) or of abstract machines (Parr and Russell, 1998).

This subsection will introduce the MAXQ value function decomposition of Dietterich (2000a), the HAM described by Parr (1998), and the option framework of Sutton et al. (1999a).

#### 2.2.3.1 MAXQ Value Function Decomposition

The MAX approach decomposes a complex task ($M$) into an hierarchy of subtasks $M_i$. Each task is executed using the primitive actions or other subtasks. Let $N$ denote the number of time steps needed to execute an action $a$ starting in state $s$ and terminating in state $s'$, Dietterich (2000a) includes the action duration in the expression of the MDP transition probability: $P(s',N|s,a)$. Similar to MDP (see Section 2.2.1), the modified Bellman equation of the value function for a fixed policy $\pi$ is:

$$V^{\pi}(s) = \sum_{s',N} P(s', N|s, \pi(s)) \left[R(s', N|s, \pi(s)) + \gamma^N V^{\pi}(s')\right]. \tag{2.16}$$

This equation is very similar to Equation 2.5 with the difference that the MAXQ approach integrates the action duration $N$ in the value function and the action $a$ is generalised with $\pi(s)$, as several actions may be needed for the execution of the subtasks. Dietterich (2000a, page 241) puts forward the following theorem for the decomposition of the value functions:

"*Given a task graph over tasks $M_0, \ldots M_n$ and a hierarchical policy $\pi$, each subtask $M_i$ defines a semi-Markov decision process with states $S_i$, actions $A_i$, probability transition function $P_i^{\pi}(s',N|s,a)$ and expected reward function $\bar{R}(s,a) = V^{\pi}(a,s)$, where $V^{\pi}(a,s)$ is the projected value function for child task $M_a$ in state $s$. If $a$ is a primit-*

*ive action, $V^{\pi}(a,s)$ is defined as the expected immediate reward of executing a in s:* $V^{\pi}(a,s) = \sum_{s'} P(s'|s,a)R(s'|a,s)$ ."

A hierarchical policy $\pi = \{\pi_0, \cdots, \pi_n\}$is a set of subtask policies so that each subtask $M_i$ has a policy $\pi_i$, which defines for each state $s$ of $M_i$ the primitive action to take or the subroutine to execute. To explain the MAX decomposition, Equation 2.16 can be extended with the index of the SMDP subtask $i$:

$$V^{\pi}(i,s) = \sum_{s',N} P_i^{\pi}(s',N|s,\pi_i(s)) \left[R(s',N|s,\pi_i(s)) + \gamma^N V^{\pi}(i,s')\right]. \tag{2.17}$$

The following equation can be rewritten as:

$$V^{\pi}(i,s) = \left[\sum_{s',N} P_i^{\pi}(s',N|s,\pi_i(s))R(s',N|s,\pi_i(s))\right] + \left[\sum_{s',N} P_i^{\pi}(s',N|s,\pi_i(s))\gamma^N V^{\pi}(i,s')\right]. \tag{2.18}$$

The first part represents the value function of a child task $M_a$ starting at state $s$ and terminating at state $s'$ after $N$ steps. Dietterich (2000a) notes the previous equation 2.18 as:

$$V^{\pi}(i,s) = V^{\pi}(\pi_i(s),s) + \sum_{s',N} P_i^{\pi}(s',N|s,\pi(s))\gamma^N V^{\pi}(i,s'), \tag{2.19}$$

where $V^{\pi}(\pi_i(s),s)$ is the projected value function for child task $M_a$.

If the task $M_a$ is a one-step action, $V^{\pi}(\pi_i(s),s) = \sum_{s'} P(s'|s,a)R(s'|a,s)$, whereas if $M_a$ is composite, $V^{\pi}(\pi_i(s),s) = \sum_{s',N} P_i^{\pi}(s',N|s,\pi_i(s)) + R(s',N|s,\pi_i(s))$. This decomposition of MAXQ enables to calculate the value function of SMDP tasks based on the value function of subtasks.

#### 2.2.3.2 Option Framework

In the option framework, the notion of action is replaced by the concept of options (also called composite actions or abstract actions). According to Sutton et al. (1999a), a temporally extended action called option $\langle \pi_o, \beta_o, I_o \rangle$ is defined by:

- a policy $\pi_o : S \times A \rightarrow [0,1]$. $\pi_o$ defines how the actions can be selected for all the states in which the option is defined.

- a termination condition $\beta_o : S^+ \rightarrow [0,1]$. $\beta_o$ gives the probability of the option to terminate in each state.
- an initiation set $I_o \subseteq S$. At option is available in state $s_t$, if $s_t \epsilon I$.

Given this, Sutton et al. (1999a, page 10) put forward the following theorem to define an SMDP:

"*For any MDP, and any set of options defined on that MDP, the decision process that selects only among those options, executing each to termination, is an SMDP.*"

The SMDP model could be formally presented as the tuple $\langle S, O, P, R \rangle$ with

- a finite set of states $S$
- a finite set of options $O$: $O_s$ is a set of options available for state $s$.
- transition probabilities $P$: probability ($P(s'|s,o)$) that the environment transitions from one state $s \epsilon S$ to another $s' \epsilon S$ after taking the option $o \epsilon O$.
- reward function $R$: $R_s^o$ is the expected reward of the option $o$ for state $s$. The expected reward is calculated as follows:

$$R_s^o = E\left\{ r_{t+1} + \gamma^1 r_{t+2} + \cdots + \gamma^{k-1} r_{k+2} \,|\, \varepsilon(o,s,t) \right\}, \tag{2.20}$$

  where $\varepsilon(o,s,t)$ is the probability of taking $o$ in state $s$ at time $t$, $k$ is the duration of the option.

Similar to the MDP (see Section 2.2.1), the Bellman equation of the value function $V^\mu(s)$ for a fixed Markov policy $\mu : S \times O \rightarrow [0,1]$ over the option $o$ is defined as follows:

$$V^\mu(s) = \sum_{o \epsilon O_s} \mu(s,o) \left[ R_s^o + \sum_{s'} p_{ss'}^o V^\mu(s') \right], \tag{2.21}$$

where $\mu(s,o)$ is the probability distribution of selecting the option $o$ according to $\mu$ $p_{ss}^o = \sum_{k=1}^{\infty} P(s',k)\gamma^k$ is the state-prediction part of the model of $o$ for state $s$. $P(s',k)$ is the probability that the option $o$ terminates in $s'$ after $k$ steps. Consequently, the optimal value function $V_O^*(s)$ is defined as:

$$V_O^*(s) = \max_{o \epsilon O_s} \left[ R_s^o + \sum_{s'} p_{ss'}^o V_O^*(s') \right]. \tag{2.22}$$

#### 2.2.3.3 HAM

HAM simplifies the modelling of a complex MDP by using deterministic and stochastic finite state machines to specify the set of actions that the agent can take in each environment state (Parr, 1998)]. These finite-state machines speed up the learning time of the system by constraining the actions that can be taken at each state. A machine for HAM is defined as the tuple $\langle \mu, \zeta, \delta \rangle$, where $\mu$ is a finite set of machine states, $\zeta$ is the initial machine state and $\delta$ is a stochastic transition function from current machine to the next machine states. A HAM-induced MDP $H \circ M$ is formally defined by the tuple $\langle Sh, Ah, Ph, Rh \rangle$ as follows for an MDP $\langle S, A, P, R \rangle$ and a HAM $\langle \mu, \zeta, \delta \rangle$:

- $Sh$ that is the state space is the cross-product of the states of HAM $H$ with the states of the MDP $M$.
- $Ah$ that is the action space corresponds to the action states (or call states) that changes only the machine component.
- $Ph$ that is transition function corresponds to the combination of the transition functions $P$ and $\delta$.
- $Rh$ that is the reward function corresponds to the MDP reward function $R$.

The HAM-induced MDP is defined as follows in Parr (1998, page 98):

"*For any MDP M and HAM H, there exists an SMDP, called $H \circ M$, the solution of which defines an optimal choice function, choose$(s,m)$, that maximizes the expected, discounted sum of rewards received by an agent executing H in M*".

This thesis applies the three HRL approaches discussed above depending on the SMDP tasks. Since domain knowledge can be included in the HAM algorithm in order to speed up the SMDP learning, HAM is applied to learn the procurement SMDP in Chapter 4. The MAXQ framework is used in Chapter 5 to model the hierarchical decision making of the broker in the retail market, as it requires less domain knowledge. The option framework is applied to model the overall strategy of the trading agent in Chapter 6, as it offers a more robust formalism that has been adopted to express concurrency and knowledge transfer in reinforcement learning.

## 2.3 SMDP Modelling for Trading Agents

When creating a trading agent, the aim is to design an agent that performs better than a human agent by exploring the environment, competitor strategies and discovering new

behaviours. Therefore, machine learning techniques are used to build adaptive sensors, while basing the reasoning engine on HRL techniques and making use of intelligent actuators.

The environments in which AstonTAC will be placed in are considered to be complex as they are multi-agent, partially observable, stochastic and dynamic environments[2]. The AstonTAC architecture is therefore designed to deal with such environmental settings; it integrates learning mechanisms in the three core components (sensor, reasoning engine and actuators). While sensors and actuators are considered to be environment-specific components, the reasoning engine is generalised to be reusable in different market domains. The resulting architecture of AstonTAC is composed of three layers (see Figure 2.4): an environment-specific layer, interpretation layer and the reasoning layer. As the broker trades in different markets that follow different trading and communication rules, the environment-specific layer enables the agent to sense the specific market and to interact with it. The reasoning layer which remains the same in all markets is the portable layer and enables to decide which actions to take in order to fulfil the followed goals, whereas a interpretation layer maps the concepts between the environment-specific layer and the reasoning layer. This AstonTAC architecture is applied in Chapters 3-7 in order to build a portable and adaptive trading agents.

### 2.3.1 SMDP with Discrete Action and State Spaces

AstonTAC SMDP actions are discrete and not continuous (Toussaint and Storkey, 2006; Konidaris and Barreto, 2009; Degris et al., 2012; Van Hasselt, 2012), because in the real-life market mechanism, continuous actions are not realistic, as trading actions are executed in a discrete manner. However, a continuous SMDP state space could be used to model AstonTAC's SMDP. To solve continuous state (S)MDP two techniques are used:

1. Discretisation of the continuous MDP states
2. Function approximation with reinforcement learning (Irodova and Sloan, 2005; Jong and Stone, 2007; Busoniu et al., 2010; Xu et al., 2014; Tamar et al., 2014).

On the one hand, the discretisation of continuous MDPs has the limitation that some environment variables are poorly represented by discrete market states and the reinforcement can suffer of the curse of dimensionality, as the number of state-action pairs

[2]The properties of the markets considered in Section 2.1.2 are not specific to them as they are properties that recur in most types of markets.

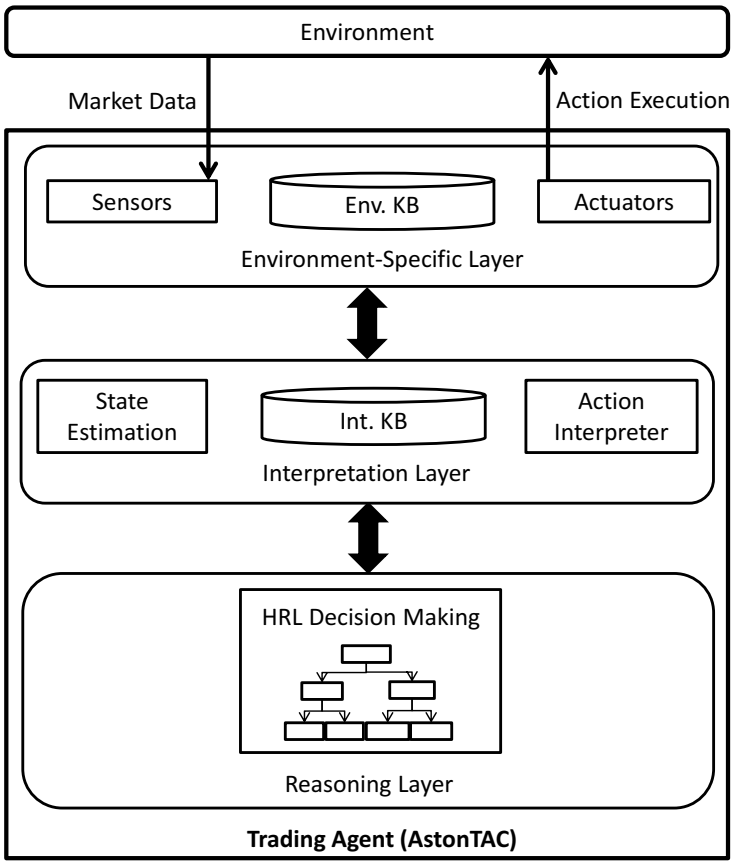



Figure 2.4: Overview of the AstonTAC Architecture. The AstonTAC architecture is composed of three layers: (1) an environment-specific layer that is composed of sensors and actuators, (2) an interpretation layer that ensures the mapping of information between the environment-specific layer and the reasoning layer, which is modelled as (3) the SMDP reasoning layer that remains invariant in all the environments the trading agent is placed in. The interpretation layer uses the state estimation component to abstract the environment information in SMDP states, whereas the action interpreter is used to convert the selected primitive actions into execution procedure that is appropriate for the environment.

can be very high. On the other hand, the use of function approximation is limited by the dynamics of the reinforcement learning problem (Thrun and Schwartz, 1993; Melo et al., 2008), so that a misuse of this technique may hinder the finding of an appropriate trading strategy.

Melo et al. (2008) prove that the function approximation works well, when immediate rewards are considered, and less so when delayed ones are considered, which means that an action does not only influence the immediate rewards, but also the expected future rewards. In the case of problems with immediate rewards solely, the initial function approximation may be different from the optimal functions to be approximated, whereas in the case of problems with delayed rewards, the initial function needs to be very close to the ideal final function. Specifically for the trading agents considered in this work, the agent's actions have a high impact on the future rewards, which means that if function approximations were used, the initial offline function must be very close to an ideal local optimum, which is not possible as the rewards have to be discovered by the agents.

Flat reinforcement learning is a commonly used approach to optimise trading decisions in real electronic markets (Moody and Saffell, 2001; Nevmyvaka et al., 2006; Moriyama et al., 2008; Corrêa et al., 2009; Bertoluzzo and Corazza, 2012). Most of the MDPs applied in real-life markets are discretised MDPs, as the MDP actions specify what to do and additional components are used to specify how to execute the action. The MDP actions can be: "*increase*", "*decrease*", "*maintain*", "*buy*" or "*sell*", which advise the trading agent about which action to take, but not how to take the actions. The trading MDPs seem to be more applicable to real-life problems, if they consider essentially high-level decision making without trying to specifically decide on how to execute the low-level decision problems which can be solved using environment-specific models derived from econometric techniques (Heij et al., 2004; Waters and Waters, 2008; Wooldridge, 2010). Additionally, real world approaches also assume a finite set of environment states (Nevmyvaka et al., 2006; Moriyama et al., 2008; Corrêa et al., 2009; Bertoluzzo and Corazza, 2012).

Therefore, discrete SMDPs with a tabular version of reinforcement learning algorithms such as SARSA($\lambda$), $n$-step MDP and Monte-Carlo models are more appropriate for the formalisation of trading decision problems. To avoid the curse of dimensionality, the number of state components is kept small, as the designed SMDPs are basically used for high-level decision making that tell the agent what to do, so that the agent's actuators decide how to accomplish the primitive actions selected by the reas-

oning engine. Moreover, knowledge transfer is proposed to reduce the time needed to train the hierarchy of MDPs.

### 2.3.2 Fully Observable SMDP

The second issue considered is the partial observability of some environment variables, as some environment information is not available to trading agents. For instance, the retail demand can be considered as hidden. One obvious solution will be to use a Partially Observable MDP (POMDP), which is a generalisation of MDPs with the assumption that the environment state is not fully observable (Sondik, 1971; Smallwood and Sondik, 1973; Kaelbling et al., 1998; Doshi-Velez, 2009).

Using the POMDP, the decision maker may have to keep track of the entire history of observations and actions in order to act optimally. For POMDPs, memoryless approaches learn a control policy by mapping the current observations to the actions without estimating the hidden states (Singh et al., 1994a,b; Loch and Singh, 1998; Littman, 1994; Singh et al., 1994a), whereas memory-based approaches learn a control policy while estimating the hidden environment states based on the past observations (Lin and Mitchell, 1993; McCallum, 1993, 1995; Meuleau et al., 1999; Doshi-Velez, 2009; Liu et al., 2011). Since, the market state can be made of several market state components, memoryless policy learning is more appropriate than memory-based learning, which is more computationally expensive. However, a memoryless approach without keeping history of any observation-action pairs will not allow credit assignment to previous actions.

Modelling the trading decision problem using POMDPs can make the problem intractable or more computationally expensive. In this work, it is assumed that it may be important to treat the estimation of the environment state and the decision making as two different problems. First, machine learning techniques are used to estimate the market states that will be the MDP states. Then, the MDP states are assumed to be fully observable; this enables to use fully observable SMDP to model the trading decision problem.

### 2.3.3 Policy Search

The reinforcement learning techniques presented in Sections 2.2.1 and 2.2.2 use the estimations of state-value or action-value functions $(V^{\pi}(s_t)$ or $Q^{\pi}(s_t, a_t))$ to search for the optimal policy $(\pi^*(s_t)$ or $\pi^*(s_t, a_t))$. However, it is also possible to search for the

optimal policy without calculating of $V^{\pi}(s_t)$ or $Q^{\pi}(s_t, a_t)$ by using the state, action and rewards as inputs with the aim being to find actions that maximise the observed rewards. The probability of taking action $a_t$ in state $s_t$ under stochastic policy $\pi$, given the vector of policy parameters $\theta$:

$$\pi_{\theta}(s_t, a_t) = Pr[a_t|s_t, \theta_t],$$

where $Pr$ is the probability density function of the action $a_t$ given $s_t$ and the vector of policy parameters $\theta$. The learning consists of identifying the set of policy parameters $\theta$ that maximise the expected return (Ng and Jordan, 2000; Peshkin, 2002).

Compared to value-based approaches, policy-based approaches enable a better and direct integration of expert knowledge to the policy search either when initialising or updating the policy parameters $\theta$. This enables the policy-based methods to generally converge better. Additionally, the search for the optimal policy is done by directly trying the different policy parametrisations without computing $V^{\pi}(s_t)$ or $Q^{\pi}(s_t, a_t)$, which reduces the search complexity and make policy-based techniques more effective for MDPs with high-dimensionality or continuous action spaces (Kober and Peters, 2012).

The first limit of the policy search approaches is a direct consequence of their advantages: because policy-based techniques are generally bootstrapped by the expert knowledge and existing sample policies, they generally converge to local optimum and are generally better applicable for MDPs, whose policies are easier to represent (Sutton et al., 1999b). Therefore, the policy-based reinforcement learning has been successfully applied to problems that require the executions of a single action such as ball-in-the-cup game (Kober and Peters, 2009), hitting a baseball (Peters and Schaal, 2008) or pan-cake flipping (Kormushev et al., 2010). Moreover, policy-based approaches are unexplored for complex decision problems that require several actions such as the decision problems faced by the trading agents. Furthermore, hierarchical MDPs have not yet been addressed by policy search approaches (Deisenroth et al., 2013). As the research presented in this thesis modelled the complex trading problems as an SMDP, policy search approaches, which rely much on the expert knowledge and are not yet explored for hierarchical decision problems, are not appropriate for this work.

### 2.3.4 Multi-Objective Reinforcement Learning (MORL)

In general, trading agents can face decision problems with competing or conflicting outcomes. For instance, when selling products in an open and competitive retail market, a

retail agent has to optimise its retail prices so that it can simultaneously sell to as many customers as possible whilst making profits. The challenge here is that the customers are generally *attracted by low prices* and the agent can only make more profit by *increasing the retail price*, if it is assumed that the total product cost remains unchanged. Another example is the optimisation of short-time procurement, where the agent must optimise the procurement cost while achieving these two objectives: (1) to buy when the procurement prices are low and (2) to meet the procurement deadlines. In order to meet the procurement deadlines, the agent may choose to just purchase the goods at any prices, however by doing so it may have to buy at very high prices. Analogously, when targeting only low prices, the agent may not meet the retail demand on time, because procurement at low prices may not be possible.

The presented problems can be modelled as multi-objective MDP (MOMDP) and solved with multi-objective reinforcement learning (MORL) approaches (Karlsson, 1997; Gábor et al., 1998; Vamplew et al., 2011). Although, MORL is still at an initial stage, it is becoming an active research area as presented recently by Roijers et al. (2013); Wang (2014); Liu et al. (2015). In contrast to the standard MDPs, MOMDPs consider more than one objective with individual rewards.

When solving MOMDP, the learning objective is either to learn a single policy that best satisfies all the objectives or learn multiple preference-driven policies that are optimal given the preference set between the objectives. To achieve the search for a single policy, the objectives reward signals are aggregated in the form of linear or non-linear scalar (Natarajan and Tadepalli, 2005; Mannor and Shimkin, 2004). A simple scalarisation approach is a weighted-sum of the objective rewards. At each time $t$, the aggregated rewards $r\_agg_t$ of the observed rewards $r_{i_t}$ of objective $i$ can be calculated as follows:

$$r\_agg_t = \sum_{i=1}^{d} w_i \, r_{i_t}, \tag{2.23}$$

where $w_i$ is the weight assigned to the reward signals $r_i$. When applying linear scalarisation to multiple policy approach, the values of $w_i$ can vary according to the states considered. To specify the preferences of the objectives, multiple-policy MORLs make use of several techniques among which the scalarisation (Lizotte et al., 2012; Barrett and Narayanan, 2008) and the population-based approaches (Van Moffaert et al., 2013).

While single-policy approaches are more suitable for online policy search than multiple policy ones, which are more computationally expensive and required offline turn-

ing of the preferences parameters. Moreover, as multi-policy approaches rely more on expert knowledge than single policy approach, they are more appropriate when the characteristics of the MOMDP are well known, otherwise the parameters tuning process can be time consuming.

The single policy approach is used in this thesis to solve the MOMDPs because the aim of the research is to enable the agent to continually learn online. Moreover, the objective rewards can be converted to monetary values, which eases the linear scalarisation of the objectives rewards using as a weighted sum as presented in equation 2.23. In Chapters 3 and 4, the linearisation approach is used to simultaneously optimise the wholesale procurement time and cost. Analogously, a single policy approach is applied in Chapter 5 to learn a policy that simultaneous facilitates a high number of customers and a high profit.

### 2.3.5 Markov Property

In reinforcement learning, an environment has the Markov property, if the transition to the next state $s_{t+1} \epsilon S$ of the environment only depends on the current state $s_t \epsilon S$ and the current action $a_t \epsilon A$:

$$P(s_{t+1}|s_0, a_0, \cdots s_t, a_t) = P(s_{t+1}|s, a_t).$$

It is the responsibility of the model designer to define the states of $S$ so that each state contains enough relevant information to enable the prediction of the next state without considering previously visited states. However, this may be difficult to achieve for some environment. Therefore, Sutton and Barto (1998) advised to still consider states that do not fully satisfy Markov property to be an approximation of the Markov state as reinforcement learning techniques can still be successfully applied. Given this, in this thesis it is assumed that the environment defined by the MDP states abstraction are Markovian or approximation of Markov states.

### 2.3.6 Learning to Execute MDP Actions Concurrently

Many artificial intelligence techniques can be used to deterministically select and execute concurrent actions: logic-based and automated planning approaches. Considering logic-based approach, a declarative language is proposed by Gelfond and Lifschitz (1992) to logically represent actions and their effects. Baral and Gelfond (1997) later

expand this declarative language to describe concurrent actions. Reiter (1996) proposes a formalisation of concurrent actions building on previous work of Pinto (1994) on situation calculus. The language of situation calculus is used by De Giacomo et al. (2009, 2000) to describe programming languages that enable concurrent actions execution by an autonomous agent.

Many works on automated planning and scheduling support generation of concurrent plans and actions using planning. Knoblock (1994); Boutilier and Brafman (2001) use partial-order planner to generate plans than can be executed in parallel. Planning of concurrent execution of MDP actions has been studied by Younes and Simmons (2004); Mausam and Weld (2005); Little and Thiebaux (2006); Aberdeen and Buffet (2007); Mausam and Weld (2008). To optimise simultaneously multiple MDP tasks, Singh et al. (1998) define a cross-product MDP (called composite MDP) whose state space is the cross product of individual MDPs state and action spaces. If the simultaneous tasks have different time step duration, these approaches are not applicable, as they assume that all concurrent actions start and end at the same time. Moreover, as the planning algorithms require the parameters of the concurrent MDPs to be known.

As logic-based and automated planning approaches do not consider online learning and adaptation to concurrent actions, only learning approaches are considered for this work. Rohanimanesh and Mahadevan (2001, 2002) propose a novel SMDP-based framework to model and learn with concurrent actions[3]. Their framework uses the formalism of the option framework (see Section 2.2.3) and models concurrent actions that have different durations and that do not terminate at the same time. Each option that is executed by multiple parallel actions or options is called a multi-option $\overrightarrow{o} \epsilon O$. Similar to equation 2.21 the associated Bellman equation of the value function is:

$$V^{\mu}(s) = \sum_{\overrightarrow{o} \epsilon O_s} \mu(s, \overrightarrow{o}) \left[ R_s^{\overrightarrow{o}} + \sum_{s'} p_{ss'}^{\overrightarrow{o}} V^{\mu}(s') \right], \tag{2.24}$$

where $R_s^{\overrightarrow{o}}$ is the expected reward when $\overrightarrow{o}$ is completely executed after a duration of $k$.

Three termination modes can be defined for $\beta_{\overrightarrow{o}}$. With the termination event *any* ($T_{any}$ mode), $\overrightarrow{o}$ terminates according to $\beta_{\overrightarrow{o}}$, if any of the parallel actions (or options) terminate, whereas with termination event *all* ($T_{all}$ mode), $\overrightarrow{o}$ terminates when all the parallel actions (or options) terminate. A third scheme, $T_{continue}$ is later defined in Rohanimanesh (2006). $T_{continue}$ is an extension of $T_{any}$ that enables to continue the execu-

[3]This framework has been extended to enable planning of concurrent decision making (Rohanimanesh and Mahadevan, 2005; Rohanimanesh, 2006).

tion of running actions (or options) after $\overrightarrow{o}$ terminates.

When modelling the execution of AstonTAC's concurrent actions, the $T_{any}$ mode is followed to conserve the semi-Markov property of the SMDP environment. $T_{all}$ is not appropriate for designing trading agent actions, as it requires the agent not to take any further actions until all actions terminate, which could lead to a under-performance of the trader in a competitive market. A $T_{continue}$ mode seems to be a more appropriate approach, as it enables the trading agents to continue with successfully policy, however the action selection by the SMDP is not semi-Markov. The action selection by the SMDP is deterministically influenced by an external subroutine in order to enable successful options that are being executed to continue.

### 2.3.7 Transfer Learning

The core idea of transfer learning is to boost the learning in new domain after the agent has been trained in a similar domain. For instance, transfer learning will enable a robot to lean how to navigate better in room B after it has been trained to navigate in room A. In this work, transfer learning is used to speed up the learning process in a personal computer market, after the trading agent has been trained to trade in an electricity market and vice versa.

Supervised learning algorithms are generally developed with the assumption that historical and future data have the same distribution and are from the same domain (Bishop, 2006). However, in reality, future data may not have an identical distribution and domain (Raina et al., 2006; Dai et al., 2007; Raykar et al., 2008). Pan and Yang (2010) review transfer learning approaches for machine learning. When designing a trading agent architecture, these approaches are applied to improve the performance of the agent's sensors in different markets (Pardoe, 2011).

Compared to machine learning transfer, reinforcement learning transfer, which is a well-developed research field, is still at an initial stage. Research on transfer learning can be grouped based on the level of similarity between the tasks considered (Taylor and Stone, 2009; Lazaric, 2012). When the set of MDP actions and the set of MDP state components are considered to be same, multi-task learning approaches are considered for enabling the knowledge transfer (Snel and Whiteson, 2014; Mehta et al., 2008a; Lazaric et al., 2008; Bernstein, 1999; Lazaric, 2008). If the set of actions and the set of state components are totally different, inter-task mapping approaches are more appropriate for the learning transfer (Torrey et al., 2006; Taylor et al., 2007; Taylor and

Stone, 2007; Torrey et al., 2005). However, if the set of actions and the state components are only partly identical, the learning transfer use task-invariant state features to enable transfer (Konidaris et al., 2012; Ravindran and Barto, 2003; Soni and Singh, 2006; Konidaris and Barto, 2007; Banerjee and Stone, 2007; Croonenborghs et al., 2008).

As the knowledge transfer problem considered in this thesis addresses transfer between domains with different action and state sets, inter-task mappings seems to be more appropriate. However, with an increasing number of tasks, the number of mappings required can increase exponentially making a general optimisation of the transfer problem difficult to realise. For a generalisation of the transfer problem, approaches such as the agent-centric approach of Konidaris et al. (2012) seems to be more suitable. This work combines these two approaches (inter-task mapping and agent-centric) to propose a new knowledge-transfer approach that is more appropriate for trading agents and that enables their reasoning systems to be portable across markets.

## 2.4 Summary of the Chapter

This chapter has introduced and critically reviewed the concepts and techniques that are key to this thesis: trading agents, HRL and SMDP. Many approaches that discuss the designing of agent architecture show that the agent acts using three components: the sensors, the reasoning component and the actuators. A large number of trading agent architectures have been proposed in the recent years to develop agents that adapt to a specific market by using adaptive sensors and rule-based reasoning systems. However, no previous study has investigated the use of SMDP and HRL to design cross-domain reasoning systems for trading agents. The next chapter presents the implementation of an MDP-based reasoning system that aims to optimise short-term procurement problems in wholesale markets.

# Chapter 3

# Optimising the Wholesale Bidding Strategy for Short-Term Procurement

The previous chapter presents the relevant background knowledge and reviews the work related to the focus of this thesis while justifying the techniques used to design the reasoning engine of the trading agent. This chapter describes the first version of AstonTAC called AstonTAC_V1, that utilises an MDP-based reasoning engine to optimise its bidding strategy in electricity wholesale markets.

In general, in order to cover short-term, middle-term and long-term retail demand, electricity retailers can make use of different provision sources, which are self-production, production contracts with end-consumers, pool-based markets and bilateral contract markets (Conejo et al., 2005a). While self-production and bilateral contract markets are used to cover middle-term and long-term retail demand, end-consumers' production and pool-based markets are suitable for short-term procurement. The procurement strategy proposed in this chapter is designed to support a retailer buying for short-term electricity procurement in pool-based markets such as the Power TAC wholesale markets which are introduced in Section 2.1.2 and described in Section 3.2.

## 3.1 Introduction

Large-scale integration of sustainable electricity through renewable sources, such as wind and solar power, implies important changes in the deregulated and decentralised electricity markets (Jónsson et al., 2010; Zugno et al., 2013). This forces the key market participants such as generator companies (GenCos), electricity retailers and consumers to adapt in order to remain viable. This particularly affects electricity retailers as they

buy most of the energy they need from wholesale markets, in order to satisfy the demand of the contracted customers in the retail markets. They have the difficult task of optimising their trading strategy in order to balance demand and supply while keeping their costs low (Kirschen, 2003; Zare et al., 2011). Since storing electricity may be very expensive, developing techniques which allow to successfully maintain a low electricity balance is paramount to the retailers and everyone else involved, to ensure the efficiency of the electricity market. The minimisation of the procurement cost and real-time electricity imbalance relies on the ability of the retailers to optimally manage the short-term procurement. Since both electricity generation and consumption heavily depend on the weather, alongside other uncertain parameters, effective optimisation of short-term procurement is challenging to realise. The increased influence of weather-dependent, renewable electricity sources creates a very dynamic and variable wholesale environment in which a human retailer cannot optimally make trading decisions for real-time and short-term electricity procurement without the support of an automated decision tool.

To counter this difficulty, a framework is designed to support automated decision-making for wholesale markets. The focus of this chapter is on the optimisation of the short-term procurement strategy. Given the market state, the optimisation model put forward enables a retailer to buy at low prices and keep the imbalance between the electricity purchased and needed low. The decision-making framework takes as inputs market information such as demand and clearing price forecasts to determine the suitable volume of electricity to buy in order to satisfy the demand. At the core of the proposed approach is an MDP that enables the optimisation of the bidding strategy, based on the underlying assumption that the wholesale market environment that is defined by the MDP state space is Markovian. The state space considered for this work is described in Section 3.4. A reinforcement learning technique, called the Monte Carlo approach and introduced in Section 2.2.2, is used to solve the proposed MDP model. While the reasons for using a discrete MDP and the reinforcement learning to solve this decision problem are presented Sections 2.1.2 and 2.3, the adoption of a Monte Carlo approach is justified by the nature of the decision problem as described in Section 3.2 - the agent observes the relevant rewards only at the end of the decision process composed of 24 decision steps.

This chapter mainly details the development and evaluation of AstonTAC_V1 in the 2012 Power TAC. An evaluation and analysis of the 2012 Power TAC finals show that AstonTAC_V1 is the only agent that can buy energy at low price in the wholesale

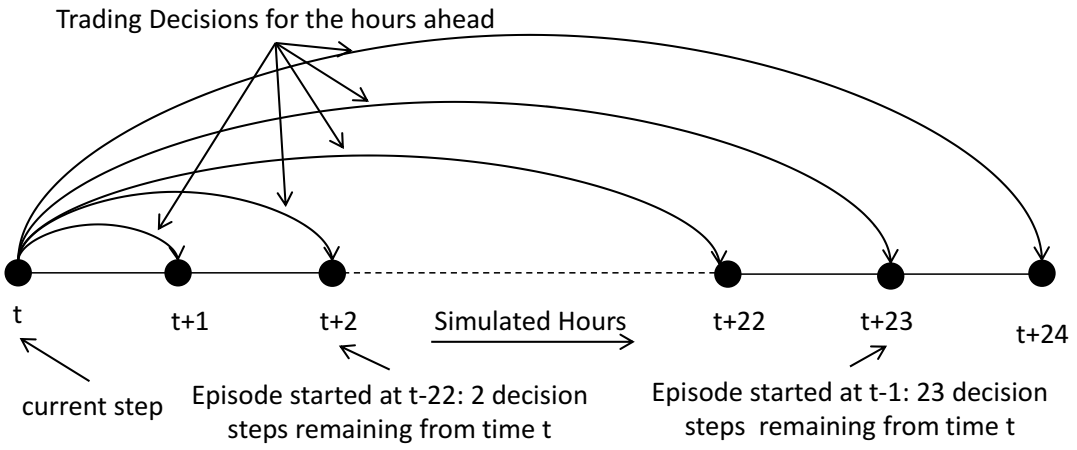



Figure 3.1: Trading Process for the Hours-Ahead in Power TAC. At each simulated hour of the game, the next 24 hours ahead are open for wholesale trading. To acquire enough energy for a specific hour, the retailer agent has 24 bidding periods to buy the energy required for the retail market. The 24 periods available represent the duration of a procurement episode. As an illustration, at time $t$, the retailer may buy electricity for time $t+2$, which belongs to an episode that started at $t-22$ and is due to end at $t+2$ when the electrical power needs to be delivered in the retail market.

market and keep energy imbalance low. The remainder of the chapter is organised as follows. In Section 3.2 a brief overview of Power TAC wholesale market is presented. Section 3.3 presents related work. In Section 3.4 the architecture of AstonTAC_V1 is described. The evaluation in the Power TAC 2012 is discussed in Section 3.6. Finally, Section 3.7 summarises the chapter.

## 3.2 Wholesale Market in Power TAC

As introduced in Section 2.1.3.1, the evaluation environment utilised, Power TAC, has a wholesale market that mimics energy markets such as Nord Pool in Scandinavia or FERC in North America. The wholesale market is modelled as an hours-ahead market, where the market is cleared every hour following a uniform pricing process. In an hours-ahead market, contracts are made between seller and buyer for the delivery of power within hours. The Power TAC simulation makes use of real-life data such as the wholesale market clearing prices as well as information on the weather to bootstrap the market changes.

Figure 3.1 presents the sequence of trading decisions that a retailer agent has to make every hour in order to trade for the next 24 hours. Each simulated hour of the

game, the retailer can make 24 trading decisions for the 24 hours ahead. Each hour represents the end of a procurement episode, when the electricity should be delivered to the retail market. Since the retailer agent may buy the electricity that needs to be delivered at a specific hour up to 24 hours ahead, each procurement episode is composed of 24 decision steps that are available to the retailer agent to satisfy the retail demand of that hour.

Furthermore, the Power TAC server provides two types of information to each retailer agent: public and private information. The public information is available to all the retailer agents. During each time slot, each retailer agent receives market clearing prices and the cleared volume, current weather and the forecast for the next 24 simulated hours. The private information is available only to the retailer in question. This includes successful bids and asks in the wholesale market, the cash position, and electricity distribution and imbalance.

## 3.3 Related Work

A large number of recent publications in the field of developing agents to act in energy markets focus on formulating suitable bidding strategies for wholesale markets. The majority of these studies observes the bidding strategy from the GenCos' point of view (Tellidou and Bakirtzis, 2006; Bach et al., 2012), while the energy procurement decision problem as faced by the retailers has not attracted as much attention.

Considering the generators' viewpoint, game-theoretic approaches as well as linear and non-linear programming models have been proposed to optimise their bidding strategies. A linear programming model is used in Morales et al. (2010) to optimise the short-term bidding strategy for a wind power producer. A stochastic linear programming model is introduced by De Ladurantaye et al. (2007) to maximise the profits of price-taker producers. Particle-swarm optimisation models are proposed by Yucekaya et al. (2009); Boonchuay and Ongsakul (2011) to maximise the profit of GenCos. Supply function models that GenCos can apply to develop an optimal bidding strategy are discussed in Gao and Sheble (2010); Bompard et al. (2010). In Song et al. (2000); Gajjar et al. (2003), MDPs are used to optimise the profit expected by a supplier over a planning horizon. Contrary to the energy generators that have a certain control on the volume of energy they produce, the retailers have less control over the aggregated retail demand. Moreover, GenCos can either be price-makers or price-takers, while retailers are generally price-takers that cannot influence the future wholesale prices. These dif-

ferences make the optimisation models of GenCos less relevant for the retailer decision problem considered in this work. Furthermore, the MDP presented in Song et al. (2000) assumes to have complete information about the competing agents. However, similar to real-life trading regulation, the Power TAC environment considers any successful bids and asks as private information.

Other studies focus on the consumers' perspective, studying strategies large consumers may adopt to deal with electricity procurement when trading in pool markets and bilateral contract market settings (Conejo et al., 2005a; Carrión et al., 2007; Zare et al., 2010, 2011). In contrast to electricity retailers, large consumers have considerably more control on planning their long-term, mid-term and short-term electricity requirements. Using game theoretical analysis, Philpott and Pettersen (2006) studied conditions under which buyers should bid for their short-term electricity demand in a Norwegian pool market. However, due to their reliance on the assumption of only having market participants with rational and constant behaviours, the game-theoretical approaches are not suitable or realistic to use when optimising decision making sequences that are dependent on the time ahead, the forecast electricity prices and demand. In reality, the behaviour of wholesale market participants is not constant and is heavily influenced by the market state.

Regarding research work on models for retailer bidding strategies, a stochastic linear programming model has been proposed in Fleten and Pettersen (2005) to construct linear bidding curves for a retailer to use in determining the price and volume of the bids to submit to the Nord Pool day-ahead market. The authors concluded that there is a trade-off between buying at low prices and reducing the imbalance and acknowledged that their model does not support the multi-period aspect of the retailer's bidding problem.

Based on the CDA algorithm of Tesauro and Bredin (2002), Urieli and Stone (2014) design a bidding strategy for smart-grid retailers. Since the clearing prices are assumed not to be predictable in CDA markets (Tesauro and Bredin, 2002; He et al., 2003; Cliff, 2005; Vytelingum et al., 2008), the bidding strategy of Urieli and Stone (2014) determines the best limit price for a buy order. However, in the environment considered by Urieli and Stone (2014) as well as in real life, the clearing price of the electricity wholesale pool market is evidently predictable (Conejo et al., 2005b; Garcia et al., 2005; Liu and Shi, 2013; Kristiansen, 2014). Furthermore, the success of the approach proposed in Urieli and Stone (2014) can be attributed to the static clearing price pattern of the considered wholesale market settings. The approach proposed in this chapter is

designed to be applicable in dynamic market environments which are characterised by uncertain clearing price and demand patterns.

In a similar setting, the Spanish day-ahead market, Herranz et al. (2012) proposed a genetic algorithm (GA) to minimise the retailer's procurement cost by deciding the right amount of electricity volume to buy given the computed clearing prices. The GA presented in Herranz et al. (2012) assumes the forecast demand to be invariant for a fixed number of days ahead. In a dynamic market setting, the GA of Herranz et al. (2012) needs to be run at each decision step for each daily bid. Since this process is very time-consuming, such a GA-based bidding strategy is not suitable for very short-term (hour-ahead) procurement.

As stated in Section 2.2, MDP and reinforcement learning are appropriate when dealing with decision making in dynamic and uncertain trading environments. Section 2.3 justifies the design choices made when designing the MDP model and implementing the learning algorithm. This chapter puts forward a standard MDP framework to maximise a retailer's bidding profit and to reduce the cost of imbalance for short-term electricity procurement (assuming variable demand and supply volume as well as prices). To increase the bidding profit, immediate price rewards are used, whereas the reduction of the imbalance cost relies on the imbalance rewards which are only observed at the end of the trading episode. As the initial target is to balance the energy needed, the proposed reinforcement learning algorithm learns at the end of the trading episode, when the balancing reward is observed. One of the robust reinforcement learning techniques that enables learning to happen at the end of the episode is the Monte Carlo, which is introduced in Section 2.2.2.

## 3.4 Proposed Wholesale Architecture

Given the complexity, and the dynamic and uncertain nature of the Power TAC environment, AstonTAC_V1 is built to adapt to environmental changes and to act autonomously during the simulation. For every time slot, AstonTAC_V1 employs MDP models to determine the bids (price, volume and time slot) submitted to the wholesale market. In order to calculate the volume to buy, AstonTAC_V1 needs to know the energy consumption of the contracted customers and the energy production of the contracted customers. In the Power TAC environment, some customers are able to produce electrical energy using renewable energy sources and to sell their energy production to the trading agent. Thus, the net customer demand is the difference between the energy con-

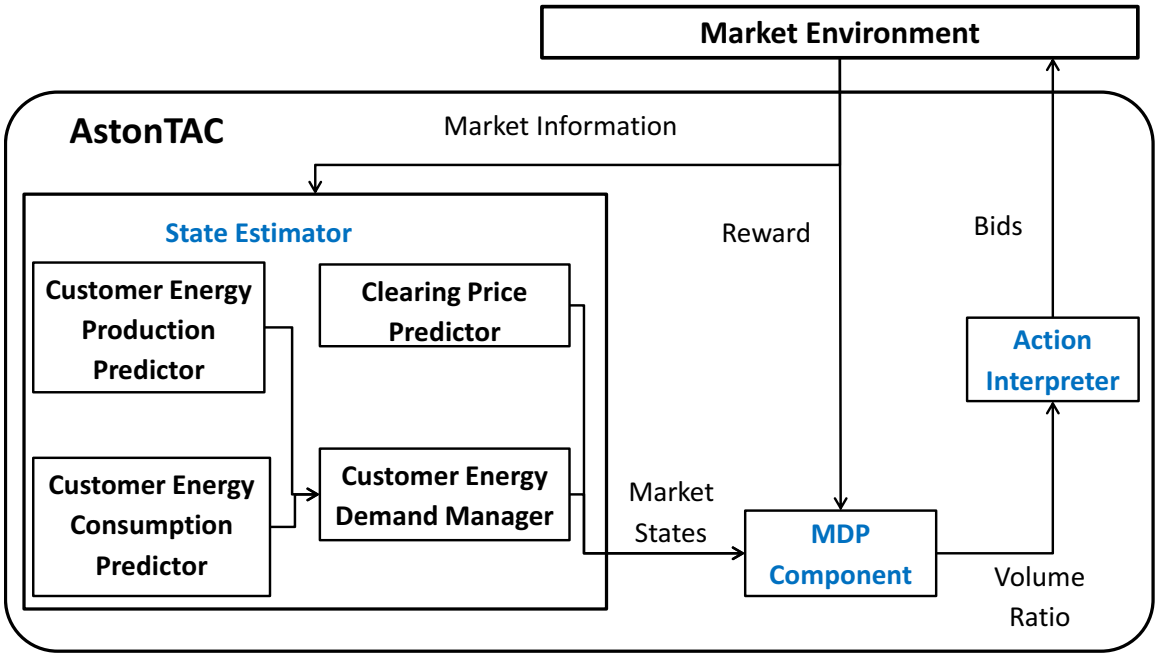



Figure 3.2: Wholesale Architecture of AstonTAC_V1 using the wholesale MDP. Following the illustration in Figure 2.4, this view of the ActonTAC's architecture is composed of three main components: the State Estimator, the MDP Reasoning Component and the Action Interpreter. The State Estimator uses the environment information to determine the market states which are the inputs of the MDP component. The MDP Reasoning Component determines the volume of electricity to buy. Using the provided electricity volume, the action interpreter submits a buy order (bid) to the wholesale market.

sumed and the energy produced. In order to set a bidding price to buy energy and the time slot for delivery, it is vital to predict the clearing price, so that the agent attempts to buy the energy when the clearing price is low. Figure 3.2 illustrates the architecture of AstonTAC_V1's architecture for procurement from the wholesale market. As introduced in Section 2.3, this architecture is composed of three key components: an MDP Reasoning Component, a State Estimator for translating the environment state to useful input, and an Action Interpreter to execute the action that the reasoning component has generated.

### 3.4.1 AstonTAC_V1 MDP Model

The aim of the MDP Reasoning Component is to aid the agent in buying energy in a way that can reduce the buying and the imbalance cost. This can be achieved only by optimising the buy orders that the trading agent sends to the wholesale market. In order to optimise a buy order, the following parameters need to be considered:

- *delivery time*: the trader agent needs to satisfy the contracted consumers on a set time in the future which is the delivery time. This time is known.
- the *clearing price* which is not known, but can be predicted as discussed in Section 3.3.
- the *volume of electricity needed*, which is the volume of electricity that a retailer agent still needs to buy. This volume is the difference between the retail demand and the the volume of electricity already acquired.
- The *volume of electricity to buy* every hour. This final parameter is the objective of the optimisation model. Thus, optimising the buy orders means optimising the volume of electricity to buy in each of the 24 hours ahead. Upon receiving the predicted energy price for each time slot and the remaining energy to buy in the wholesale market, the AstonTAC_V1 MDP decides the electricity volume of the bids to place for each of the hours ahead.

The parameters of the AstonTAC_V1 MDP are defined by the states, actions, state transition probabilities and reward function (see Section 2.2.1 for MDP formalism). Details on these parameters are given below.

#### 3.4.1.1 State Space ($S$)

The state space ($S$) is composed of the three state components, which are the remaining time until the end of the episode (either days, or hours depending on whether operating in a day-ahead or hours-ahead market), the procurement imbalance between acquired energy and the level of demand, and the projected clearing price:

**Time Ahead** Depending on the market rules, pool markets have different numbers of bidding periods per day. Considering the specific case of Power TAC, the wholesale market is an hours-ahead market with 24 periods open for trading. At every simulated hour, the agent can buy energy for the next 24 hours. This means that the agent has 24 decision steps to ensure the balancing of the retail demand (see Figure 3.1). A definition of the set of time ahead states can be defined as $S\_a = \{\text{sufficient, critical, very critical}\}$ with:

$$s\_a = \begin{cases} \text{sufficient} & \text{if } 16 < ta \leq 24 \\ \text{critical} & \text{if } 8 < ta \leq 16 \\ \text{very critical} & \text{if } 0 < ta \leq 8, \end{cases}$$

where $s\_a \epsilon S\_a$, $ta$ is the number of bidding periods ahead. The selection of the hours-ahead to consider for the definition of the states of the remaining time is dependent on the bidding problem considered. This state definition can be appropriate for a market that is regulated similarly to the Power TAC environment.

**Procurement Imbalance** This state component codifies the volume of the electricity that needs to be bought or sold. This state information is calculated as a ratio $y$ between the volume of the imbalance $i_t$ and the volume of the demand $d_t$ at time $t$:

$$y_t = i_t / d_t, \tag{3.1}$$

in which $i_t$ is defined as:

$$i_t = x_t - d_t, \tag{3.2}$$

where $x_t$ is the volume of electricity acquired to be delivered within the time ahead considered. $i_t$ represents the current shortage ($i_t < 0$) volume of electricity that a retailer needs to buy, to meet the retail demand. For example, three states can be considered to

define a set of imbalance $S_i = \{\text{shortage, balanced, surplus}\}$with:

$$s\_i = \begin{cases} \text{shortage} & \text{if } y_t < -0.05 \\ \text{balanced} & \text{if } -0.05 \leq y_t \leq 0 \\ \text{surplus} & \text{if } 0.05 < y_t, \end{cases}$$

where $s\_i \epsilon S\_i$, an imbalance of up to 5% either way is allowed. As a total balancing of supply and demand is impossible, the modeller should consider a certain imbalance margin.

**Projected Clearing Price** The projected clearing prices can be classified in a set of predefined states. A simple classification can be low, middle and high price states. Let *P_min* be the minimum clearing price ever experienced, *P_max* the maximum clearing price ever experienced and *pa* the predicted clearing price for the hour ahead *ta* using the Clearing Price Predictor. The set of price states $S\_p = \{\text{low, middle, high}\}$ can be defined as follows:

$$s\_p = \begin{cases} \text{low} & if\ P\_min \leq pa < \frac{2P\_min+P\_max}{3} \\ \text{middle} & if\ \frac{2P\_min+P\_max}{3} \leq pa \leq \frac{2P\_max+P\_min}{3} \\ \text{high} & if\ \frac{2P\_max+P\_min}{3} < pa \leq P\_max, \end{cases}$$

with $s\_p \epsilon S\_p$.

The state variables $s\_a \epsilon S\_a$, $s\_i \epsilon S\_i$ and $s\_p \epsilon S\_p$ are calculated based on the information provided by the simulation environment, Power TAC as described in Section 2.1.3.1. *s_a* is defined based on the simulation time defined by the game server. *s_i* is predicted by the State Estimator components: customer energy consumption predictor and customer energy production predictor. The predictions are done based on the retail consumption and production data provided during each game by the game server. Appendix A describes the prediction approach used to dynamically forecast the energy consumption and production. Analogue to *s_i*, environment data provided by the game server are used to predict the price state *s_p* using the same Markov approach implemented by the clearing price predictor.

The definitions of $s\_a \epsilon S\_a$, $s\_i \epsilon S\_i$ and $s\_p \epsilon S\_p$ presented in this section are for illustration purposes. Section 3.5 will present more details on the actual settings used in implementing this work. While defining and fine tuning the state space for this problem, we studied the diverse set of techniques that have been suggested in past to

improve MDP state space definitions (Giunchiglia and Walsh, 1992; Givan et al., 2003; Li et al., 2006; Peters et al., 2013). The size of a state space naturally affects the training time: a large state space for Power TAC does not only require long training time, but also training settings that facilitate the exploration of most of the MDP states.

#### 3.4.1.2 Action Space ($A$)

The agent's action is a bid that is placed in the wholesale market. The bid is a combination of the bidding price and the bidding volume for a time slot in the game. For each time slot, the agent decides the bid to place for the future 24 time slots that are available for trading in the wholesale market. Bid prices are the predicted clearing prices provided by the Clearing Price Predictor. The energy volume of the bids are decided by the MDP. The bid energy volume is represented as the ratio of the remaining energy needed that should be ordered. For example, the agent's actions could be defined according to the percentage of electricity volume to buy given the states of the projected clearing price. For instance, if the states of projected clearing price are high, middle and low, the corresponding possible actions can be defined as 0%, 50% and 100% with so that:

$$a = \begin{cases} 100\% & \text{for} sp = low \\ 50\% & \text{for} sp = middle \\ 0\% & \text{for} sp = high, \end{cases}$$

where $a \epsilon A$. This example only shows how to specify the actions needed and not how the agent will select the actions. For instance, if $sa = verycritical$ and $sp = high$, the agent may learn to execute $a = 100\%$ and not 0%, in order to balance its retail demand.

As presented in Section 3.2, the trading agents can place 24 buy orders in every trading time slots. Each order requires an action selection from the MDP component, which takes the state components *s_a*, *s_i* and *s_p* as inputs and outputs the action $a$. Based on the selected $a$, the exact order volume is calculated, then associated with the predicted order price and submitted to the market to buy electrical energy for a specific delivery time. Although the same MDP is used for all decision making, each trading decision is taken independently. The MDP has been built with state and action spaces that support any of the 24 hourly trading decisions.

#### 3.4.1.3 Reward Function ($R$)

Concretely, the trading agent needs to buy more of the needed energy when the prices are low and less of it when the prices are high while meeting the target electricity volume for the delivery time. To realise this multi-objective optimisation, different type of rewards are assigned to each objective. The agent receives two types of reward: an immediate reward *r_buy* for buying at a low price and a delayed reward *r_bal* for the energy balancing after 24 decision steps.

The immediate reward is given to the agent according to the bid setting: bidding price and bidding volume. For example, if the predicted clearing price is high, the agent will receive a positive reward for buying a low volume of energy and negative reward for buying a high volume. The value of the delayed reward is based on the imbalance ratio at the end of the 24 steps. Formally the reward function *R_buy* for *r_buy* can be defined as follows:

$$R\_buy : S\_p \times A \rightarrow \mathbb{R},$$

as the mapping function that associates to each pair of price state and action combination a reward $r\_buy \epsilon \mathbb{R}$.

The simulation environment informs the agent each hour about the value of the energy imbalance. In order to guarantee a low energy supply and demand imbalance, the highest positive reward is received when the imbalance is the lowest (i.e., between to 0% and 5%). Formally, the reward function *R_bal* for *r_bal* can be defined:

$$R\_bal : s\_i_{T+1} \rightarrow \mathbb{R},$$

as the mapping function that associates to each imbalance state after the end of the episode $t = T$, a reward $r\_bal \epsilon \mathbb{R}$. The calculation of the state and action value function is presented in Section 3.4.1.5. More details on the reward function are presented in Section 3.5.

#### 3.4.1.4 Transition Probability ($P$)

$P(s'|s,a)$ is the probability that the AstonTAC_V1 MDP transitions from one state $s \epsilon S$ to another $s' \epsilon S$ after taking the action $a \epsilon A$.

#### 3.4.1.5 Optimal Value and Policy

AstonTAC_V1 MDP is a combination of a state-action rewards MDP (Littman, 1996; Singh, 1993) and a finite horizon MDP (Watkins, 1989a; Li and Littman, 2005). The overall strategy is the sequence of the decisions that can be made in order to maximise the total rewards. This overall strategy is denoted by the policy function $\pi$ where $\pi : S \rightarrow A$. Following the policy $\pi$, at each state $s$ of a run, the agent takes the decision and transitions the system with a probability of $P(s'|s,a)$ to the next state $s'$ as follows. Let $n_d$ be the total number of states to visit in the run.

- At state $s_0$ and time $t = 0$, take action $a_0$.
- Go to $s_1$, with a transition probability of $P(s_1|s_0,a_0)$.
- At state $s_1$ and time $t = 1$, take action $a_1$.
- Go to $s_i$, at time $t = i$ with a transition probability of $P(s_i|s_{i-1},a_{i-1})$, where $2 \leqslant i \leq n_d$ and $i \epsilon \mathbb{N}$.

Let $V_\pi(s)$ denote the cumulative expected reward function that starts from state $s = s_0$ at time $t = 0$ and uses a policy $\pi(s)$. The best policy $\pi(s)$ at state $s$ is therefore the one that has the maximal value of $V_\pi(s)$. The maximal value of $V_\pi(s)$ is denoted by $V^*(s)$. The AstonTAC_V1 MDP is defined as follows given $\pi(s)$ and $0 \leq t \leq T-1$ with $T = 24$. Section 2.2.1 introduces the computation of $\pi^*$ and $V^*$. As the model parameters (reward function, expected total pay-off and transition probabilities) are not available and have to be learned, AstonTAC_V1 solves the procurement MDP using the Monte Carlo Methods introduced in Section 2.2.2.

As introduced in Section 2.2.1, the aim is to maximise the expected return $Return_t$ which is defined with two types of rewards: the immediate price rewards *r_buy* and the delayed balancing rewards *r_bal*. For the 24 decision steps, the agent observes *r_buy* after every buy action and *r_bal* is observed only at the end of the episode. Based on the equation 2.1, the return can be formulated as follows:

$$Return_t = r\_buy_t + \gamma^1 r\_buy_{t+1} + \gamma^2 r\_buy_{t+2} + \cdots + \gamma^{T-t} r\_buy_T + r\_bal_{T+1}, \quad (3.3)$$

where $T = 24$ and $t < 22$. The equation can be written as follows:

$$Return_t = \left( \sum_{k=0}^{T-t} \gamma^k r\_buy_{t+k} \right) + r\_bal_{T+1}. \tag{3.4}$$

The resulting value function $V^{\pi}(s_t)$ is defined as follows:

$$V^{\pi}(s_t) = w_{buy} E \left[ \sum_{k=0}^{T-t} \gamma^k r\_buy_{t+k} | s_t \right] + w_{bal} E \left[ r\_bal_{T+1} | s_t \right], \tag{3.5}$$

where $w_{buy}$ and $w_{bal}$ are the weights assigned to the expected returns of the immediate rewards and delayed rewards. The weights are used to adjust the learning preference of the agent for immediate buy rewards or delayed reward. With $\gamma = 0$, the value function $V^{\pi}(s_t)$ depends only on the expected immediate reward $r\_buy_{t+1}|s_t$ and not on the future immediate rewards:

$$V^{\pi}(s_t) = w_{buy} E \left[ r\_buy_{t+1} | s_t \right] + w_{bal} E \left[ r\_bal_T | s_t \right] \tag{3.6}$$

Due to the volatility of the trading prices, $\gamma = 0$ seems more efficient for learning as the agent does not need to consider the future price rewards, but only the immediate rewards prices. Moreover, the immediate reward is independent of previously taken actions, it only depends on the current action. The following equations can be written:

$$V^{\pi}(s_t) = w_{buy} V\_buy(s_t) + w_{bal} R\_bal(s_t), \tag{3.7}$$

where $V\_buy$ is the expected rewards for the immediate rewards and $R\_bal(s_t)$ is the average of the $r\_bal_T$ observed each time $s_t$ has been visited. Similarly, for the reinforcement learning, an action value $Q^{\pi}(s_t, a_t)$ can be defined as follows:

$$Q^{\pi}(s_t, a_t) = w_{buy} Q\_buy(s, a_t) + w_{bal} R\_bal(s_t), \tag{3.8}$$

Algorithm 3 details the implementation of the reinforcement learning technique. At the beginning of the training, the expected reward of each state-action is initialised with zero (Line 1). In each game, a backup list with length $T = 24$ is used to record the visited state-action pairs. For each step of the game, the agent makes 24 trading decisions for the 24 hours ahead using the expected reward $R(s, a)$ (Lines 6-9). $R(s, a)$ is the sum of the immediate reward $r\_buy$ and the expected reward $R\_bal$ of the energy balance. $R\_bal$ is an average of the delayed reward $r\_bal$ observed at the end of the episode (Line 14).

**Algorithm 3** Learning of a Suitable Trading Strategy with Monte-Carlo approach

1: Initialise: $w_{buy}$, $w_{bal}$, $Q\_buy(s_t)$ and $R\_bal(s_t)$
2: **for** each simulation **do**
3: $Backup(s,a) \leftarrow$ empty backup lists with length $T$
4: **for** each bidding step $t$ of the simulation **do**
5: **for** each time ahead **do**
6: Choose a primitive action $a$ for $s$
7: using $Q^{\pi}(s,a)$ and the $\varepsilon$-greedy policy
8: Take action $a$ in state $s$
9: Add ($s$, $a$) to the corresponding $Backup(s,a)$
10: observe $r_{t+1}$ and $s_{t+1}$
11: update immediat reward expectation
12: $Q\_buy(s_t, a_t) \leftarrow Q\_buy(s_t, a_t) + \alpha\left[r\_buy_{t+1} - Q\_buy(s_t, a_t)\right]$.
13: **end for**
14: **if** Final bidding step **then**
15: Observe the delayed reward $r\_bal$
16: **for** each ($s$, $a$) in $Backup$(s,a) **do**
17: Update $R\_bal(s,a)$ with $r\_bal$
18: $R\_bal(s,a) \leftarrow (23/24)R\_bal(s,a) + r\_bal/24$
19: **end for**
20: $Backup$(s,a) $\leftarrow$ an empty backup list
21: **end if**
22: **end for**
23: **end for**

In Line 18, the update of $R\_bal$ is presented as follows:

$$R\_bal(s,a) \leftarrow (23/24)R\_bal(s,a) + r\_bal/24.$$

This update can be rewritten as $R\_bal(s,a) \leftarrow R\_bal(s,a) + (1/24)\left[r\_bal - R\_bal(s,a)\right]$, where $1/24$ is the learning rate. The agent learns from on-line experience and decides by comparing the average of experienced returns at each state. During the simulation, AstonTAC_V1 uses the pay-off average to decide about the optimal action to take instead of computing the transition probabilities and the expected reward values. The learning occurs at the end of each episode (24 simulated hours) after the delayed reward for the energy balancing is defined. At the beginning of the training, the initial

policy that will be evaluated follows the immediate reward setting: buy the maximum of the needed energy, if the predicted clearing price of the time slot is the lowest and lowest energy volume, if the price is predicted to be the highest.

### 3.4.2 State Estimator

The State Estimator determines the values of the MDP state components: time ahead, corresponding procurement imbalance and the projected prices. Therefore, it has a Clearing Price Predictor; a customer energy consumption, a customer Energy Production Predictor, and a customer energy demand manager. The Clearing Price Predictor forecasts the clearing prices of the wholesale market. The Customer Energy Production Predictor forecasts the energy consumption of the contracted customers and the Customer Energy Consumption Predictor forecasts the energy production of the contracted customers. The predictors use the Power TAC Server information (both public and private information described in Section A). The customer energy consumption and production predictors are used by the Customer Energy Demand Manager to determine the volume of the net energy demand to buy each hour (the procurement imbalance). Details on the forecasting models used by the State Estimator components are described in Appendix A.

## 3.5 Implementation of the Wholesale MDP

### 3.5.1 MDP Design

The design of the wholesale MDP for the 2012 Power TAC games was mainly influenced by the structure of the environment as presented in Section 2.1.3.1. Because the game settings were configured to express various dynamics during the tournament (see Section 2.1.3.1), the wholesale MDP is designed using invariant environment features in order to be applicable across all the games.

#### 3.5.1.1 MDP States

As introduced in Section 3.4.1.1, the MDP states are defined by the state variables: time ahead, procurement imbalance and projected clearing price. For this specific competition the time ahead was defined by 24 states with each state representing a time ahead. This means the agent is learning a specific trading behaviour for each hour ahead.

Given the level of imbalance targeted, the number of states defining imbalance can be specified accordingly. For Power TAC 2012, 25 imbalance states were defined with 24 states specifying the shortage and one state the surplus. The 24 surplus states were defined as follows: the imbalance ratio $y_t$ belongs to state $n$ if $(n-1)/24 \leq |y_t| < n/24$ with $1 < n \leq 24$. The $25^{\text{th}}$ state is defined with $y_t > 0$.

The clearing price states are composed of 24 states, where the best price belongs to the state 1 and the worst price to state 24. The states were defined around state centroids, which are the prices observed during the previous market clearings. To identify to which state the predicted price belongs, it is associated with the state of closest price centroid.

#### 3.5.1.2 MDP Actions

As introduced in Section 3.4.1.1, the AstonTAC_V1 was set to use 24 actions which define the percentage of electricity volume to buy. The actions were multiples of 4% moving from 0%, 4%, 8%, 12% until 96%. This approach enables the agent to learn when to buy low volume or high volume of electricity given the market state. As described in Section 3.4.1.2, a smaller number of states can also be considered; this has the advantage that the learning time is reduced, but the actions are not precise enough.

#### 3.5.1.3 MDP Rewards

The weighting of the rewards were set as follows $w\_bal = 1.67$ and $w\_price = 1$, which were applied to the reward per unit of electricity (*KwH* in Power TAC). The weights were defined based on the objective preference: the balancing of the energy was given a higher priority than making immediate profit price.

The AstonTAC_V1's MDP was trained before and during the tournament. Before the tournament, the agent was trained in the Power TAC environment with dummy agents using a $\varepsilon = 0.1$. During the tournament $\varepsilon$ was reduced to 0.01.

## 3.6 Evaluation in 2012 Power TAC

The evaluation is composed of two parts: the results from the 2012 Power TAC and an analysis of the tournament games.

| Agent Name | Size 1 | Size 2 | Size 3 | Accumulated Profits | *Z*-score |
|---|---|---|---|---|---|
| Crocodile Agent | 1.26E+10 | 6.31E+10 | 2.74E+10 | 1.03E+11 | 7.348 |
| **Aston TAC_V1** | **3.11E+06** | **9.58E+07** | **5.00E+07** | **1.49E+08** | **-1.217** |
| Soton Power | 1.03E+07 | 6.68E+07 | 6.22E+07 | 1.39E+08 | -1.215 |
| MinerTA | 2.33E+07 | 6.48E+07 | 1.83E+07 | 1.06E+08 | -1.217 |
| Mertacor | -5.11E+05 | -1.13E+07 | 4.98E+06 | -6.79E+06 | -1.227 |
| LARGE power | -4.16E+07 | -5.25E+07 | 1.01E+07 | -8.40E+07 | -1.238 |
| UTest | 1.55E+06 | -6.67E+07 | -4.68E+07 | -1.12E+08 | -1.235 |

Table 3.1: Results of the Power TAC Tournament. This table shows the results from the 2012 Power TAC by presenting the profit accumulated by each broker in different game sizes: Size 1 for seven-player games, Size 2 for four-player games and Size 3 for two-player games. According to the amount of cash accumulated during the competition (accumulated profits ), the first version of AstonTAC is second after CrocodileAgent, whereas it is third considering the *Z*-score.

### 3.6.1 Competition Results

The 2012 Power TAC finals Power TAC Community (2013) consisted of 184 games with three individual competitions: 63 games with two players, 105 games with four players and 16 games with seven players.[1] The teams participating to this competition were:

- CrocodileAgent team from the University of Zagreb,
- Mertacor team from the Aristotle University of Thessaloniki,
- Utest team from the University of Texas at Austin,
- MinerTA team from the University of Texas at El Paso,
- SotonPower team from the University of Southampton,
- LARGEpower team from the Erasmus University Rotterdam.

| | Agent Name | Energy Demand | Energy Bought | Imbalance (%) | Buy Price | Average Cash |
|---|---|---|---|---|---|---|
| (a) | **Aston TAC_V1** | **78478.84** | **79821.74** | **10.40** | **34.25** | **3.63E+06** |
| | SotonPower | 6810.82 | 14212.93 | 128.86 | 27.17 | 1.06E+06 |
| (b) | **Aston TAC_V1** | **85492.29** | **84661.96** | **10.43** | **33.98** | **3.66E+06** |
| | MinerTA | 1025.30 | 1662.22 | 80.10 | 26.99 | 3.43E+05 |

Table 3.2: Results of Two-Player Games with (a) SotonPower and (b) MinerTA. This table compares the performance of the first version of AstonTAC termed AstonTAC_V1 when playing against top TAC brokers (SotonPower and MinerTA) in two-player games. The data in column "Energy Demand" are the average energy demand (in MWh) of the broker agent in all the two-player games. In column "Energy Bought", the average energy volume (in MWh) bought by the broker from the wholesale market is represented. The average hourly supply demand imbalance ratio (in %) is presented in "Imbalance". AstonTAC_V1 outperforms SotonPower (3.2(a)) and MinerTA (3.2(b)) by winning most of the games, having the highest average cash in game and by having the best average of hourly imbalance in each game.

The winner of the competition is the player with the highest $z$-score of all games. Table 3.1 shows the results of the competition. This table presents the accumulated profit for each broker in seven-player (annotated Size 1), four-player (annotated Size 2) and two-player (annotated Size 3) games. Considering the total of the accumulated profits, AstonTAC_V1 is second after CrocodileAgent. According to the $z$-score, AstonTAC_V1 is third with a score of $-1.217$ after CrocodileAgent (with a score of 7.348) and SotonPower (with a score of $-1.215$). Although CrocodileAgent achieved the highest score, its success seems to have taken advantage of the weaknesses of the Power TAC server and uses an unrealistic trading strategy. For instance, when the CrocodileAgent offered tariffs with very high rates, it still attracted a lot of customers for a long period in the game.

In fact, among all the games played with SotonPower, AstonTAC_V1 outperforms SotonPower in 63% of the games (27 out of 43) and outperforms MinerTA in 60% of the games (28 out of 47). Moreover, AstonTAC_V1 won all two-player games against SotonPower or MinerTA (Ketter et al., 2013). Table 3.2 compares the performance of AstonTAC_V1 with SotonPower (3.2(a)) and MinerTA (3.2(b)) in two-

[1]Further details of the Power TAC are described in Section 2.1.3.1.

| Agent Name | Energy Demand | Energy Bought | Imbalance (%) | Buy Price | No. of Games |
|---|---|---|---|---|---|
| **Aston TAC_V1** | **29009.72** | **34731.37** | **21.52** | **29.22** | **67** |
| Mertacor | 40530.53 | 40350.92 | 18.16 | 51.84 | 69 |
| LARGE power | 22094.77 | 23434.50 | 35.21 | 37.99 | 71 |
| MinerTA | 3477.17 | 4356.54 | 75.08 | 17.44 | 74 |
| SotonPower | 9934.13 | 17683.05 | 146.29 | 19.26 | 68 |
| Crocodile Agent | 25099.90 | 63609.87 | 425.70 | 32.46 | 68 |

Table 3.3: Brokers' Performance in the wholesale market. This table shows the results of the wholesale performance of AstonTAC_V1 in wholesale market. AstonTAC_V1 is the only agent that can buy high amount of energy at low prices while keeping the imbalance low.

player games."Energy Demand" represents the average energy demand (in MWh) of the broker agent in all the two-player games. This is the sum of the net energy usage for all customer consumption. "Energy Bought" represents the average energy volume (in MWh) the agent bought from the wholesale market. "Imbalance" shows the average hourly supply demand imbalance ratio (in %). The hourly imbalance ratio is computed by dividing the absolute hourly imbalance for each broker by the hourly customer energy consumption. "Buy Price" is the average price (in EURO/MWh) of the successful bids of the broker agent. "Average Cash" represents the average profit (in EURO) the broker has at the end of the game. According to Table 3.2, AstonTAC_V1 has a bigger market share than its opponents: an average of 92.01% in the games against SotonPower and 98.81% in the games against MinerTA. These two-player games demonstrate that although AstonTAC_V1 has more than 92% of the market share, it is able to keep the energy imbalance lower than 10.5%. SotonPower and MinerTA are able to buy energy at lower price in the wholesale market but are not able to control the market or to keep the energy imbalance lower than 80%.

### 3.6.2 Competition Game Analysis

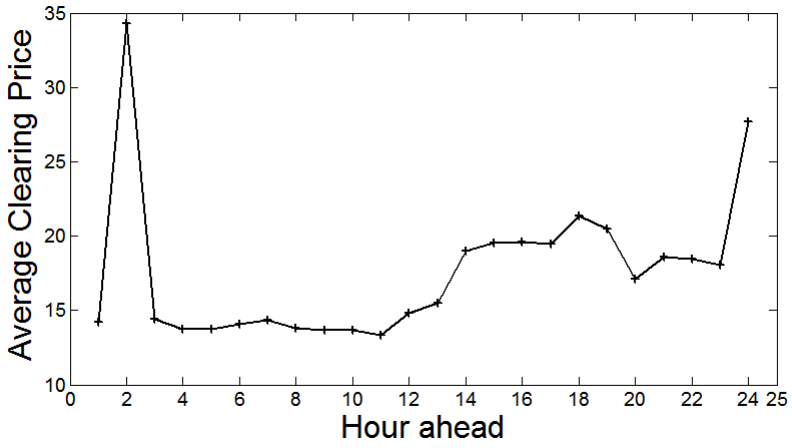
To further investigate the performance of AstonTAC_V1 in the wholesale market, this section analysed the games of the Power TAC finals by mainly focusing on the performance of AstonTAC_V1 in terms of the bidding in wholesale market and balancing supply and demand. Table 3.3 shows the brokers' performance in all successful games.[2] Mertacor did well in keeping the energy imbalance low. MinerTA and SotonPower managed to buy at low prices. AstonTAC_V1 performs well; both in energy balance and in buying at low price. As shown in Table 3.3, AstonTAC_V1 is the only agent that can buy energy at low price in the wholesale market and keep energy imbalance low. The fact that using the MDP, AstonTAC_V1 has an energy imbalance of 21.52% indirectly shows that the techniques used by AstonTAC_V1 for energy production and consumption provide acceptable prediction values (these forecast techniques are described in Appendix A).

In order to evaluate AstonTAC_V1 MDP and the clearing price prediction, the wholesale market performance of several agents is analysed in an arbitrarily chosen game (game 418) and the result is shown in Figure 3.3. Figure 3.3(a) shows the average clearing price in each hour ahead, Figures 3.3(b) and 3.3(c) illustrate the average volume of energy bought by each broker.

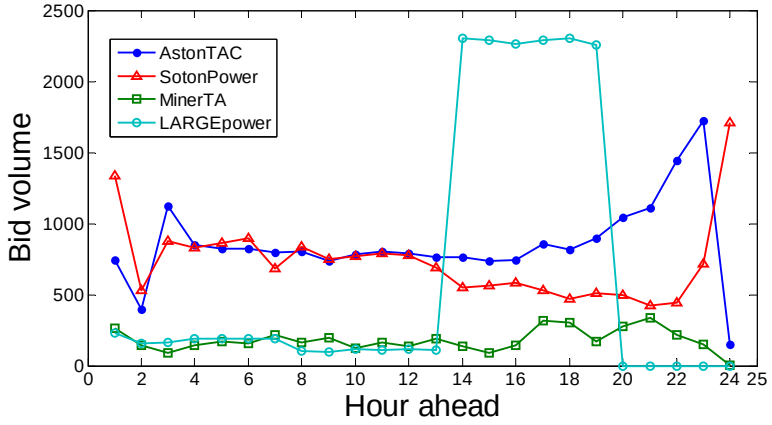
According to Figure 3.3(a), the average energy clearing prices are the highest at time slots two and twenty-four hours ahead. The lowest clearing prices are observed at time slots one, and between three and eleven hours ahead. Figures 3.3(b) and 3.3(c) show that Mertacor, CrocodileAgent and LARGEpower do not adapt their bidding behaviour to the market clearing price. In contrast, AstonTAC_V1, SotonPower and MinerTA try to adapt their bidding behaviour to the clearing price. Thus, agents that are not adapting to the wholesale market may end up buying energy at high price and adaptive agents can manage to buy less energy at high price. This is the case for AstonTAC_V1 and MinerTA which buy less energy volume at time two and twenty-four hours ahead. Although SotonPower adapts its bidding behaviour to the market, it ends up buying more energy twenty-four hours ahead when the energy is high. Using the MDP to balance the retail market, AstonTAC_V1 is able to buy less energy at high price and buy a constant volume of energy otherwise. The fact that AstonTAC_V1 buys less energy at time two and twenty-four hours ahead, demonstrates that the price prediction provides a reliable clearing price prediction.

[2] A successful game is a game that lasts over 1320 time slots which is the minimum duration of a game.

(a)




(b)




(c)

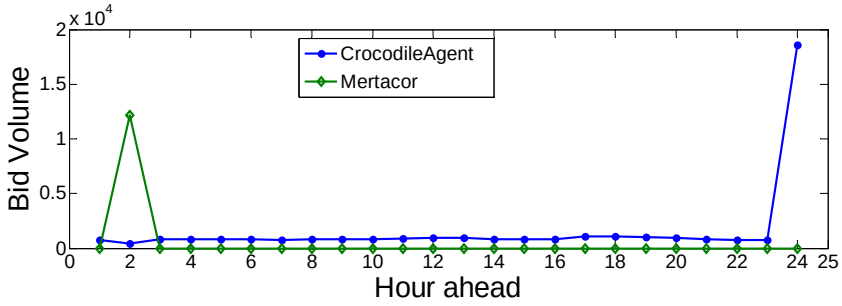



Figure 3.3: Wholesale Market Performance in Game 418. This figure shows the performance of AstonTAC_V1 in an arbitrary selected game. 3.3(a) shows the average clearing price in each hour ahead, 3.3(b) and 3.3(c) illustrate the average volume of energy bought by each broker. In this game, the wholesale prices were influenced by the volume of the energy bought by the brokers. AstonTAC_V1 is able to adapt to the market changes and buys most of the electricity needed when the prices are low and less when the prices are high.

## 3.7 Summary of the Chapter

This chapter details the design, implementation and evaluation of a novel MDP framework that is used to buy electricity at low prices in the wholesale market while keeping the imbalance between the retail demand and wholesale supply low. The analysis of the 2012 Power TAC tournament shows that AstonTAC_V1 is the only agent that can buy energy at low price in the wholesale market and keep energy imbalance low. AstonTAC_V1 MDP is designed independently from the Power TAC simulation environment. Moreover, the AstonTAC_V1 MDP provides a concrete bidding approach for the future energy market where the energy demand will be more satisfied by renewable energy sources. The increased adoption of renewable energy sources as an alternative source of electricity makes the need for automation of short-term electricity procurement more relevant than before. The real-world electricity markets generally address the problem of electricity imbalance by using energy storage facilities (spinning reserves). Real-world retailers also make used of diverse tariff contracts such as long-term and middle-term contracts to balance their electricity demand. Additionally, the imbalance level for the retailer agent depends on the proportion of electricity produced by the renewable source. The Power TAC 2012 did not offer a storage feature in the simulation. Moreover, Power TAC purely simulates a smart grid market which is not yet fully implemented in the real world.

The reinforcement learning approach proposed successfully enables the trading agent to learn how to buy more electricity when the price is low and to meet the energy needed. Although this chapter presents a successful MDP framework that deals with short-term buying, it does not considering the fact that the retailers may want to sell their energy surplus in the wholesale. As the retailers also buy electricity from consumers, it is possible that they have more energy generated from the retail market than projected. In order to improve the optimisation of the wholesale strategy, it is therefore important to extend the MDP model so that the retailer agents can monetise their electricity surplus. The next chapter describes a novel SMDP-based framework that a retailer agent can use to buy and sell in wholesale markets while keeping the electricity imbalance low.

# Chapter 4

# Optimising Bids and Asks in Electricity Pool Markets

The approach presented in the previous chapter laid out the initial steps in handling the trade-off between maximising the wholesale profit and minimising the balancing cost in a dynamic and uncertain wholesale market. The work presented in this chapter significantly improves on the results of Chapter 3 by enabling, for the first time, a retailer agent (AstonTAC_V2) to decide on the right volume of energy to bid and ask for in the wholesale market using a novel SMDP framework.

## 4.1 Introduction

Due to the privatisation and decentralisation of the electricity provision system in many countries, electricity markets have undergone several restructuring processes in order to improve the market efficiency. Given the fact that the storage of electricity is very expensive, one of the key indicators of the electricity market efficiency is the imbalance between the energy demand and energy supply. The electricity retailers' main aim is to satisfy the retail demand, which varies with time, while, at the same time, minimising their spend in wholesale markets. Since the increased uncertainty that characterises renewable electricity sources augments the volatility of the energy supply, demand and prices, it makes the retailers' aim difficult to achieve.

In view of this, the trading decision problem faced by the retailers is formulated using a discrete stochastic optimisation technique, the SMDP. Given the market clearing price and the time remaining to satisfy the end-consumers' needs, the SMDP framework developed enables the retailers to maximise their profit when buying and selling in a

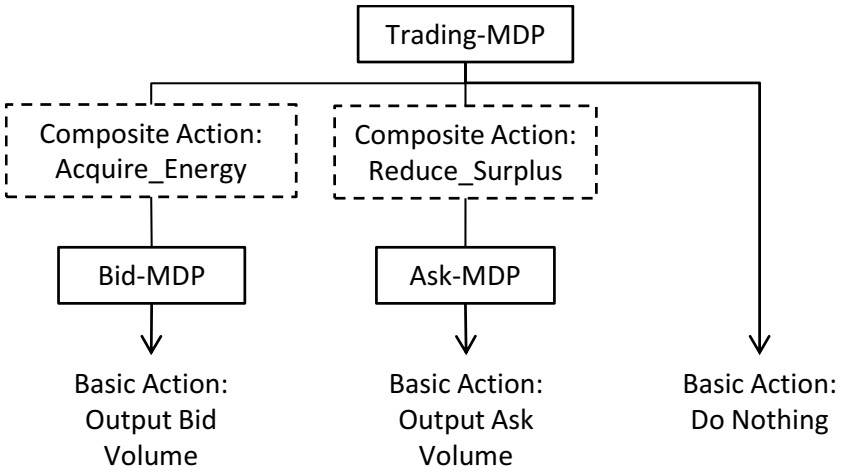



Figure 4.1: MDP Hierarchy of the SMDP. The SMDP model is composed of three MDPs: *Trading_MDP*, *Bid_MDP* and *Ask_MDP*. The *Trading_MDP* decides the strategy that the retailer should follow: to buy or to sell electricity. The *Bid_MDP* supports the procurement by defining the parameters of the bids to place in the market. The *Ask_MDP* defines the parameters of the asks. All three MDPs use the same market state to decide which action to take. The *Trading_MDP* can select between three actions. *Acquire_Energy* and *Reduce_Surplus* which are both composite actions, and Do Nothing which is a basic action. In order to execute the composite action *Acquire_Energy*, the *Bid_MDP* is used. The *Bid_MDP* selects the volume to buy given the market state. The execution of the composite action *Reduce_Surplus* is supported by the *Ask_MDP* which selects the appropriate volume of energy to sell.

wholesale electricity pool market. Using reinforcement learning to solve the SMDP, a retailer agent learns how to sell at high prices and to buy at low prices while keeping the imbalance between expected demand and supply low. To evaluate the proposed SMDP framework, it has been incorporated within a retailer agent, which operates in an open multi-agent simulation environment, the Power TAC. A thorough evaluation of AstonTAC_V2 against other Power TAC agents shows that it consistently maintains the lowest electricity imbalance while achieving the highest profit when tested in a variety of market set-ups, even those with extremely volatile prices and supply.

The remainder of the chapter is organised as follows. Section 4.2 describes in detail the proposed SMDP framework. The evaluation of the SMDP framework is in Section 4.3. Finally, Section 4.4 presents the summary of the chapter.

## 4.2 Proposed Procurement SMDP

As in the previous chapter, this chapter's focus is on a market in which a high penetration of renewable energy sources causes the electricity consumption, production and clearing prices to be volatile. In order to trade optimally in the wholesale market, the retailers have to strike the right balance between maximising their trading profit and minimising their supply and demand imbalance costs. Using the proposed SMDP model, the automated retailer agent is able to simultaneously optimise the bids and asks in order to reduce the imbalance cost for hours-ahead and days-ahead procurements. Figure 4.1 presents the hierarchy of the SMDP. At the higher level, the *Trading_MDP* supports the selection of the trading strategy to follow. It may select between the trading strategies: sell, buy or no order. The sell and buy strategies are supported by the MDPs, *Bid_MDP* that decides about the bid volume and *Ask_MDP* that decides about the ask volume. All the MDPs in the hierarchy consider the same market state as MDP inputs. SMDP is solved through direct interactions with the Power TAC using the HRL approach called HAM.

The main motivation for utilising the HAM algorithm is the flexibility offered by its implementation, as it enables the designer to add expert knowledge to the learning algorithm in order to reduce the time needed for learning. The AstonTAC HRL is enhanced with hand-coded rules that enables the agents to reconsider when to buy, sell or just wait without extra learning. The agent essentially learns how to buy and how to sell as well as when to start buying or selling.

Following the architecture presented in Section 2.3, a simplified view of the AstonTAC_V2's architecture that uses the proposed wholesale SMDP to trade in the Power TAC wholesale market is illustrated in Figure 4.2. The Sensor component enables the AstonTAC_V2 to map the market information into the market state components described in Section 4.2.1. Based on the provided market state, the SMDP Decision Engine selects the next action which is either do nothing or the volume of electricity to buy or to sell. The selection of action during the training phase is executed as presented in Section 4.2.4. After defining the volume of electricity, AstonTAC_V2 uses the Actuator component to place the order in the market. During the training simulations, AstonTAC_V2 uses its Sensor component to interpret the rewards from the market information. Using the rewards, the update of the reward expectation is done as described in Section 4.2.4. The following section provides more details on the approach put forward.

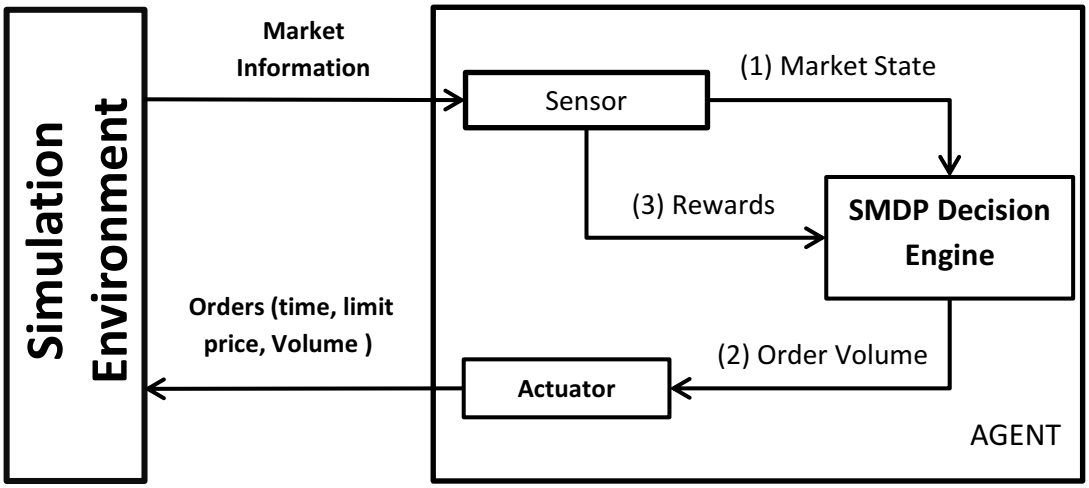



Figure 4.2: AstonTAC_V2's Architecture using the procurement SMDP. Based on the illustration in Figure 2.4, this simplified view of the agent's architecture is composed of three components: 1) the Sensor component which is able to harvest at each step of the simulation, the market information and translate it into market states, 2) the SMDP Decision Engine which uses the market state to define the appropriate volume of energy to order for a specific hour ahead, and 3) the Actuator component, which places orders in the market using information about the order volume provided by the SMDP Decision Engine. The SMDP Decision Engine learns the value of an action after it was taken using feedback from the environment, interpreted by the Sensor component as rewards.

This section describes the abstraction of the bidding problem as a semi-Markov decision problem. An SMDP is defined by the tuple $\langle S, A, P, R \rangle$, where $S$ is the set of environment states, $A$ is the set of basic or composite actions, $P$ is the transition probability from one state to another and $R$ is the reward function. Section 2.2 provides more details on the formalism of MDP and SMDP. The optimisation of the bidding strategy using the SMDP model implies the optimisation of the three MDPs of the hierarchy. The state components considered by the three MDPs are described in Section 4.2.1. The MDPs' actions are described in Section 4.2.2. Section 4.2.3 presents the reward functions of the MDPs.

### 4.2.1 State Space

All MDPs of the procurement SMDP use the same state space $S$ as described in Chapter 3, Section 3.4.1.1. Although the definition of the state components is initially influenced by the Power TAC environment, the state space can easily be adapted for any specific pool market.

### 4.2.2 Composite Actions and Primitive Actions

While the state space is the same for the three MDPs, they use different action spaces. These action spaces are represented in Figure 4.1. The *Trading_MDP* has one basic action *Do_Nothing* that executes no buy or sell order and is appropriate when the energy acquired and the demand are balanced, and two composite actions *Acquire_Energy* and *Reduce_Surplus*. The composite actions *Acquire_Energy* and *Reduce_Surplus* use the MDPs *Bid_MDP* and *Ask_MDP*, respectively, to optimise the orders that will be placed in the market in order to realise the composite actions. As their respective names suggest, *Bid_MDP* supports the selection of buy orders while *Ask_MDP* supports the selection of sell orders. In more detail, an order is composed of the parameters delivery time, limit/order price and electricity volume. Since it is assumed that the delivery time and the limit price are known, the *Bid_MDP* and *Ask_MDP* define the appropriate volume of electricity of the order to place. The appropriate volume is defined as a percentage of the imbalance $i_t$ (as explained in Section 3.4.1.2). For example, the *Bid_MDP* can have five actions, to buy electricity amounting to 25%, 50%, 75% or 100% of $i_t$.

### 4.2.3 Reward Functions

The reward functions define the goal of the optimisation process. The aim of the optimisation process is to optimise the volume in the bids and asks and to balance the electricity. Therefore, reward signals are defined to support these different goals.

#### 4.2.3.1 Optimising the Bids

A retailer can make profit by buying most of the electricity at prices that are lower than the average clearing price of the wholesale market. Let $\bar{p}$ be the average clearing price of the 24 observed clearing prices and $pb$ be the clearing price of the buy order, the procurement profit is calculated as follows:

$$r\_buy = \bar{p} - pb. \tag{4.1}$$

This is an immediate reward that is observed at each state of the market after a bid has been submitted and cleared. $r\_buy$ is applicable to both the $Bid\_MDP$ and $Trading\_MDP$.

#### 4.2.3.2 Optimising the Asks

Considering $ps$ to be the clearing price after a cleared sell order, the sell reward is defined as follows:

$$r\_sell = ps - \bar{p}. \tag{4.2}$$

This reward is immediately observable, after the sell action is taken. Similarly to $r\_buy$, the reward $r\_sell$ is applicable to both the $Ask\_MDP$ and $Trading\_MDP$

#### 4.2.3.3 Minimising the Imbalance Cost

At the end of each bidding period, market information required to compute the balancing reward $r\_bal$ is available. This information is:

- $y_T$, the ratio of the imbalance (see Equation 3.1) at the end of the bidding process. $T$ is the number of time steps of the trading episode. In Power TAC, $T = 24$.
- $f$, the imbalance fee to pay per electricity unit.

- *pb_so* the buy price of the system operator[1] in case of surplus.
- *ps_so*, the sell price of the system operator[2] in case of shortage.

Using this information, *r_bal* can be calculated according to relevance given to each parameter: more details have been provided.

$$r\_bal = m(y_T, f, pb\_so, ps\_so, \bar{p}), \tag{4.3}$$

where $m$ is a function that maps the parameters available to the *r_bal*. For instance, in the Power TAC, *r_bal* is computed as follows:

$$r\_bal = \begin{cases} (1+|y_T|)\,[(\bar{p} - pb\_so) - f] & if\ Shortage \\ 0 & if\ Balanced \\ (1+|y_T|)\,[(ps\_so - \bar{p}) - f] & if\ Surplus. \end{cases}$$

With this reward setting, an electricity shortage is advantageous for a retailer, if $\bar{p} > (pb + f)$, whereas having an electricity surplus is suitable when $ps\_so > (\bar{p} + f)$. To avoid such a situation and encourage retailers to balance their electricity, the market regulators should aim to set *pb_so* and *ps_so* accordingly. Therefore, optimising the imbalance implies minimising the retailers' imbalance costs. The delayed reward is applicable to all the MDPs, since they are all responsible for the imbalance. The next section describes the reinforcement learning techniques proposed to solve the hierarchy of MDPs.

### 4.2.4 Hierarchical Reinforcement Learning

Section 2.2.3 has presented the three main HRL techniques and briefly justified their application in this thesis. HAM is used here to learn the procurement SMDP, because it enables to speed up the learning by making use of the domain knowledge available. In Figure 4.3, the domain knowledge is represented by the conditions set on some of the arrows.

[1] Since storing excess energy is not possible for retailers due to the associated cost, the system operator buys any surplus back at a (usually dissuading) price computed by the system as a function of previously observed prices.

[2] Symmetrically to the surplus case, the system operator is able to provide energy to fulfil the contract in case of shortage at the retailers' side, for a price calculated based on observed prices.

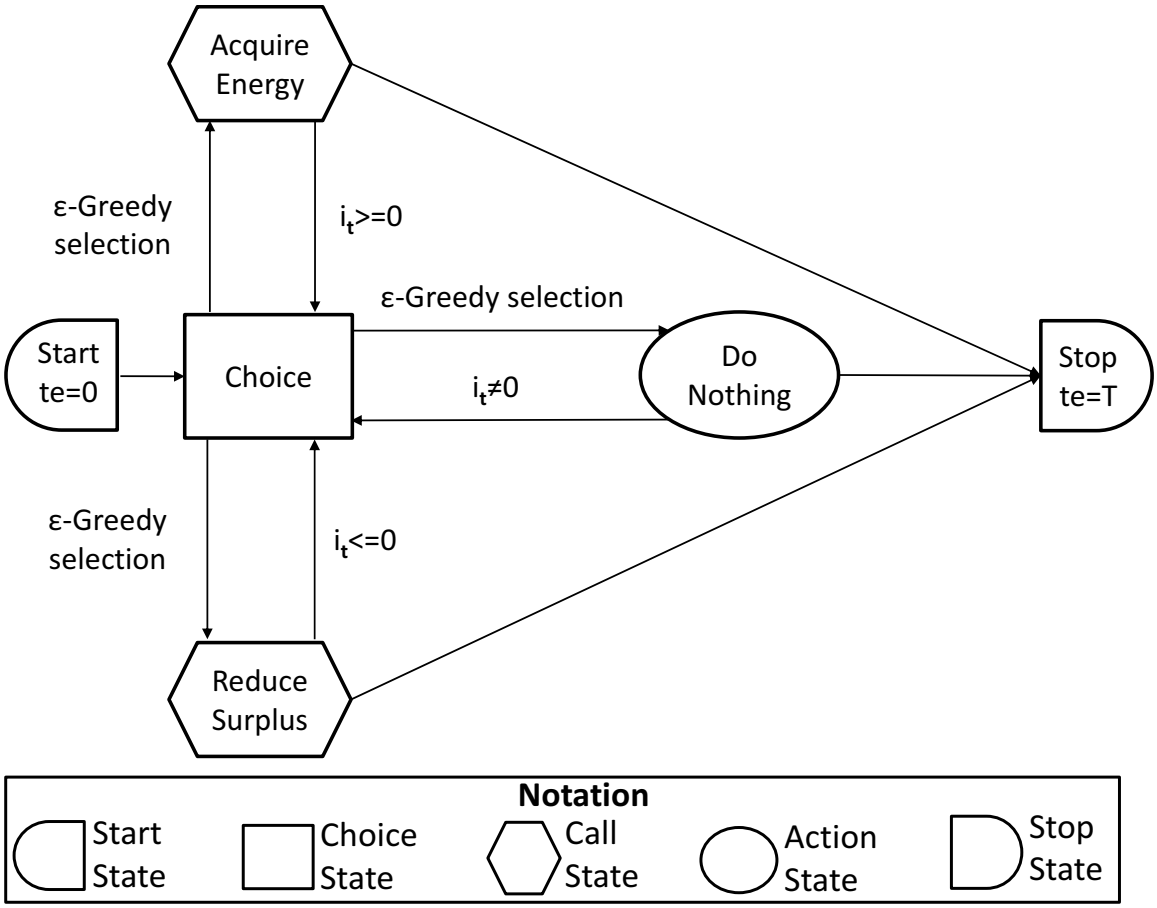



Figure 4.3: HAM Model. In the choice state, the HAM uses the *Trading_MDP* to select the next state. The next state could either be a call state or an action state. A call state executes a composite action. The call states *Acquire_Energy* and *Reduce_Surplus* use the MDPs *Bid_MDP* and *Ask_MDP* to place orders in the market. The action state *Do_Nothing* executes no action. The level of imbalance $i_t$ is used to decide when to invoke the choice state.

#### 4.2.4.1 HAM Model

Figure 4.3 illustrates the transitions between the states of the HAM. Several approaches could be used to illustrate the states of the HAM (Parr, 1998; Andre and Russell, 2001; Cuayahuitl, 2009). In Figure 4.3, the representation of Andre and Russell (2001), which is more descriptive, is adopted. HAM defines five types of machine states: *start*, *choice*, *call*, *action* and *stop* state.

**Start State** This starts the execution of the current state at the beginning of each run at $te = 0$ with $te$, the time step of the episode.

**Choice State** At this state, the agent non-deterministically selects the next state by using the SMDP framework to decide which call state or action state to select. The choice state uses an $\varepsilon - greedy$ policy to select the next state. This policy is used to manage the trade-off between exploration and exploitation in reinforcement learning. A random state is chosen with a probability of $\varepsilon$ and the state with the maximum value of expected rewards is selected with probability $1 - \varepsilon$. It is always possible to move from the action state back to the choice state.

**Call States** These execute composite actions in the current environment state. The call states are represented by the composite actions *Acquire_Energy* and *Reduce_Surplus*. While the *Acquire_Energy* uses the *Ask_MDP* to decide on the primitive ask actions, the *Reduce_Surplus* uses the *Bid_MDP* for the primitive bid actions.

**Action States** These execute actions (primitive actions) in the current environment state. In Figure 4.3, the state *Do_Nothing* is an action state.

**Stop State** This state halts the execution of the HAM. The HAM execution ends with the end of bidding periods ahead when $te = T$.

The choice and stop states are reachable from all action and call states. In the HAM presented in Figure 4.3, the conditions used by the call and action states to transition back to the choice state are prior knowledge or partial description of the environment provided by the designer. This reduces time required by the learning process to find the optimal policy. Each action state has information about the state of the imbalance $i_t$. To simplify the understanding of the Figure 4.3, it is assumed that $i = 0$ is balanced. However, an imbalance interval could be assumed to be balanced ($-0.05 \leq i \leq 0.05$).

**Algorithm 4** Learning of a Suitable Trading Strategy

```
1: Initialise: w_buy, Q_buy(s_t), w_sell, Q_sell(s_t), w_bal and R_bal(s_t)
2: for all Simulations do
3:     Backup(s,a) ← empty backup lists with length T
4:     for each bidding step t of the simulation do
5:         for each time step ahead do
6:             Read HAM state, H
7:             if H ≠ start state then
8:                 call Update-Expected-Reward-Value(H, MDP)
9:             end if
10:            if H ≠ stop state then
11:                a ← Call Select-Primitive-Action(H, MDP)
12:            end if
13:            Take action a in state s
14:            Add (s, a) to Backup(s,a)
15:        end for
16:    end for
17: end for
```

#### 4.2.4.2 HRL Algorithm

In general, the HRL algorithm is similar to the HAM Q-Learning algorithm (Parr and Russell, 1998; Parr, 1998) and the Hierarchical Semi-Markov Q-Learning algorithm (Dietterich, 2000a,b). The key difference is that the proposed approach uses a Monte-Carlo method as discussed in Chapter 3.

**Monte Carlo HRL (MC-HRL)** Algorithm 4 presents the MC-HRL algorithm for learning a suitable bidding strategy.

Similar to Algorithm 3, at the beginning of the HAM training the weights $w_{buy}$ and $w_{sell}$, as well as the state-value functions $Q\_buy(s_t)$, $Q\_sell(s_t)$ and $R\_bal(s_t)$ are initialised. The parameters $w_{buy}$, $Q\_buy(s_t)$ and $R\_bal(s_t)$ have been presented in Chapter 3. $w_{sell}$ is the weight assigned to expected immediate profit $Q\_sell(s_t)$ of a sell action. In Line 3, the backup list $Backup(s,a)$ that is initialised enables to record the visited the pairs $(s,a)$ independently of the MDP used (*Ask_MDP* or *Bid_MDP* ) for decision making. Several simulations can be used to train the system until convergence to a good trading policy. The key steps in the learning algorithm are the update of the

**Algorithm 5** Update-Expected-Reward-Value($H$, $j$)

1: **if** $H \neq stopstate$ **then**
2: Observe the price reward $r_t$
3: Update $Q\_j(s_t, a_t)$ with $r_t$
4: $Q\_j(s_t, a_t) \leftarrow Q\_j(s_t, a_t) + \alpha \left[ r_{j_{t+1}} - Q\_j(s_t, a_t) \right]$
5: **end if**
6: **if** $H = stopstate$ **then**
7: Observe the delayed reward *r_bal*
8: **for** each ($s$, $a$) in $Backup$(s,a) **do**
9: Update $R\_bal(s_t)$ with *r_bal*
10: $R\_bal(s,a) \leftarrow (23/24)R\_bal(s,a) + r\_bal/24$
11: **end for**
12: $Backup$(s,a) $\leftarrow$ an empty backup list
13: **end if**

expected rewards value using the immediate rewards and the delayed rewards (Line 8), and the selection of the actions (Line 11). The update function, *Update-Expected-Reward-Value(H, MDP)*, which is called with the HAM state and the MDP(*Ask_MDP* or *Bid_MDP* ) being executed, is the policy evaluation step and is described in Algorithm 5. Algorithm 6 describes the action selection algorithm, *Select-Primitive-Action(H, MDP),* which is the policy improvement step initialised with the HAM and the MDP. While the policy evaluation is concerned with the update of the expected rewards value, the policy improvement is concerned with the selection of a suitable action at each step.

**Policy Evaluation** Algorithm 5 describes the update of the expected reward using the observed rewards. For each of MDP $j$ (*Ask_MDP* or *Bid_MDP* ) of the MDP hierarchy (see Figure 4.1), the action-value function is calculated as in Equation 3.8:

$$Q^{\pi}_{sum\_j}(s_t, a_t) = w_j Q\_j(s_t, a_t) + w_{bal} R\_bal(s_t), \tag{4.4}$$

with $s_t \epsilon S_j$ and $a_t \epsilon A_j$, where $S_j$ is the state of MDP $j$ and $A_j$ the action space of MDP $j$. The state-value $Q_h^\pi(s_t, a_j)$ of the *Trading_MDP* is expressed as follows:

$$Q_h^\pi(s_t, a_h) = \begin{cases} w_j Q\_j(s_t, a_t) + w_{bal} R\_bal(s_t) & \text{if } a_h \text{ is call state} \\ w_{bal} R\_bal(s_t) & \text{if } a_h \text{ is action state} \end{cases}, \tag{4.5}$$

where $s_t \epsilon S$ , with $S = \cup S_j$, $a_h \epsilon Ah$, with $Ah$ the set of call (*Acquire_Energy*, *Reduce_Surplus*) and action (*Do_Nothing*) states available to the *Trading_MDP*.

**Policy Improvement** Algorithm 6 describes *Select-Primitive-Action(H)* that selects the next primitive action depending on the current HAM state. If the HAM state is the start state, it is initialised with the choice state and *Select-Primitive-Action(choice state)* is called (Line 1-4). If the HAM state is the choice state, the next state is selected with the *Trading_MDP* using the $\varepsilon$-*greedy* method with respect to $Q_h^\pi(s_t, a_h)$ where $j$ is the *Trading_MDP* (Line 5-14). If the selected state is an action state, the corresponding action is selected as the basic action. *Select-Primitive-Action(call state)* is called, if the selected state is a call state. If the HAM state is a call state, the primitive action is selected using *Bid_MDP* or *Ask_MDP* and following the $\varepsilon$-*greedy* policy with respect to $Q_h^\pi(s_t, a_h)$ where $j$ is the *Bid_MDP* or the *Ask_MDP* (Line 15-19).

The proposed SMDP-HRL-based approach is evaluated in different Power TAC settings. The evaluation is done in two steps: first AstonTAC_V2 learns the parameters of the SMDP through interactions with the Power TAC wholesale market using the HRL technique presented, then the performance of the SMDP bidding strategy is compared with other bidding strategies in the same environment.

## 4.3 Evaluation of AstonTAC_V2

To illustrate the performance of the SMDP approach, AstonTAC_V2 was developed and then tested in the Power TAC wholesale market. The evaluation methodology consists of comparing the wholesale market performance of AstonTAC_V2 to other Power TAC retailer agents in a series of controlled experiments. To do this, the retailer agents' trading profit and hourly imbalance volume are compared in four market scenarios. Each market scenario is specified by the volatility level of the retail demand and the wholesale market clearing prices. While the variation of the hourly retail demand

**Algorithm 6** Select-Primitive-Action($H$, $j$)

1: **if** $H = start\,state$ **then**
2: $H \leftarrow choice\,state$
3: return Select-Primitive-Action($H$)
4: **end if**
5: **if** $H = choice\,state$ **then**
6: Choose the next state $H$ from $s_t$
7: using $Q_h^\pi(s_t, a_h)$ and the $\varepsilon$-greedy policy
8: **if** $H = call\,state$ **then**
9: return Select-Primitive-Action($H$)
10: **end if**
11: **if** $H = action\,state$ **then**
12: return primitive action $a_h$ for $s_t$
13: **end if**
14: **end if**
15: **if** $H = call\,state$ **then**
16: Choose a primitive action $a_t$ for $s_t$
17: using $Q_j^\pi(s_t, a_t)$ and the $\varepsilon$-greedy policy
18: return $a_t$
19: **end if**

mainly affects the ability of the retailer agents to balance their energy, the variation of the hourly wholesale market prices affects their ability to maximise their bidding profit. The first market scenario considers a stable market, which is characterised from a retailer's viewpoint by a stable retail demand and wholesale clearing prices. The second scenario evaluates the performance of the agents when the demand is volatile and the clearing price stable. In the third scenario a stable demand and a volatile clearing price is used. The fourth market scenario evaluates the performance of the agents in a volatile market by making both the demand and the clearing price volatile. Figures 4.4a and 4.4b illustrate the behaviour of the environment for Scenario 1. Figures 4.4a and 4.5b illustrate Scenario 2, Figures 4.5a and 4.4b illustrate Scenario 3 and Figures 4.5a and 4.5b illustrate Scenario 4.

### 4.3.1 Implementation of HAM

The HAM is implemented as presented in Algorithm 4 Section 4.2. The training of the individual MDPs *Ask_MDP* and *Bid_MDP* are implemented following the HRL-MC explained in algorithms 5 and 6. Similar to AstonTAC_V1, AstonTAC_V2 was trained first in series of controlled experiments with decreasing value of $\varepsilon$. The algorithm hyperparameters used to implement individual MDPs *Ask_MDP* and *Bid_MDP* are the same as used for AstonTAC_V1 in Chapter 3. Moreover, the weighting of the rewards remain the same as in the experiments with AstonTAC_V1 (presented in Section 3.5.1.3): $w\_bal = 1.67$ and $w\_price = 1$.

### 4.3.2 Environment Settings

This section describes the environment settings of Power TAC. AstonTAC_V2 has been compared to two established retailer agents that use the bidding strategies presented in Section 4.1. The first agent considered is TacTex, the winner of the Power TAC 2013 that relies on a CDA approach to trade in the wholesale market (Urieli and Stone, 2014). The second agent is AstonTAC_V1. For clarity, TacTex is termed CDA Retailer and AstonTAC_V1 is the MDP Retailer.

As described in Section 3.2, the Power TAC environment provides a simulated environment that is very close to a real-life power market (Ketter et al., 2015). The Power TAC game server offers the ability to configure the agents (customers, big buyers and suppliers) that are present in the environment. The scenarios considered in this evaluation result from four different configurations of the game server. The retailer agent is configured to be a price taker - this is the default setting of the simulation environment (Ketter et al., 2015). Since the volume of energy traded by the retailer in the wholesale market is not high enough to make a significant impact on the clearing prices, the retailer agent has to adapt their bidding strategy to the wholesale market clearing prices in order to optimise its profit.

Before evaluating the performance of AstonTAC_V2 in the different scenarios, the SMDP Decision Engine is trained according to the MC-HRL presented in Section 4.2.4. Each hour represents the end of an episode when the retail demand should be satisfied. The delayed reward, which is available at the end of each episode, is therefore available each hour. A Power TAC game lasts approximately 1500 simulated hours (the end of the game is randomised). Since the retailer agent can take 24 actions each simulated hour, it uses approximately 36000 actions to learn the expected return of the immediate rewards

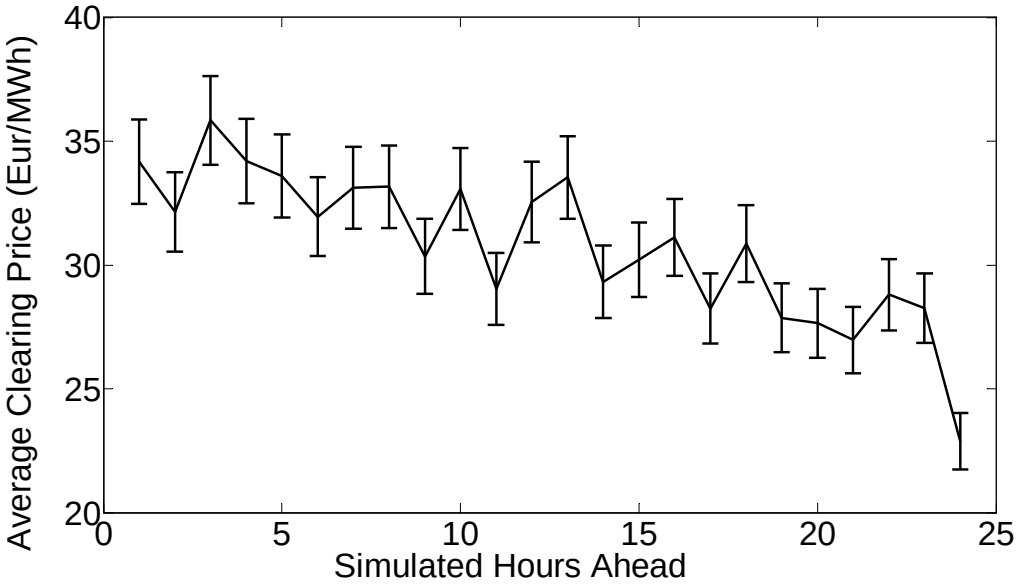
(*r_sell* and *r_buy*) in one game. In each game, AstonTAC_V2 observes approximately 1500 delayed rewards (*r_bal*). To train AstonTAC_V2, a total of 200 games was enough to observe a convergence of the retailer policy to a suitable policy.

#### 4.3.2.1 Scenario 1: Stable Market

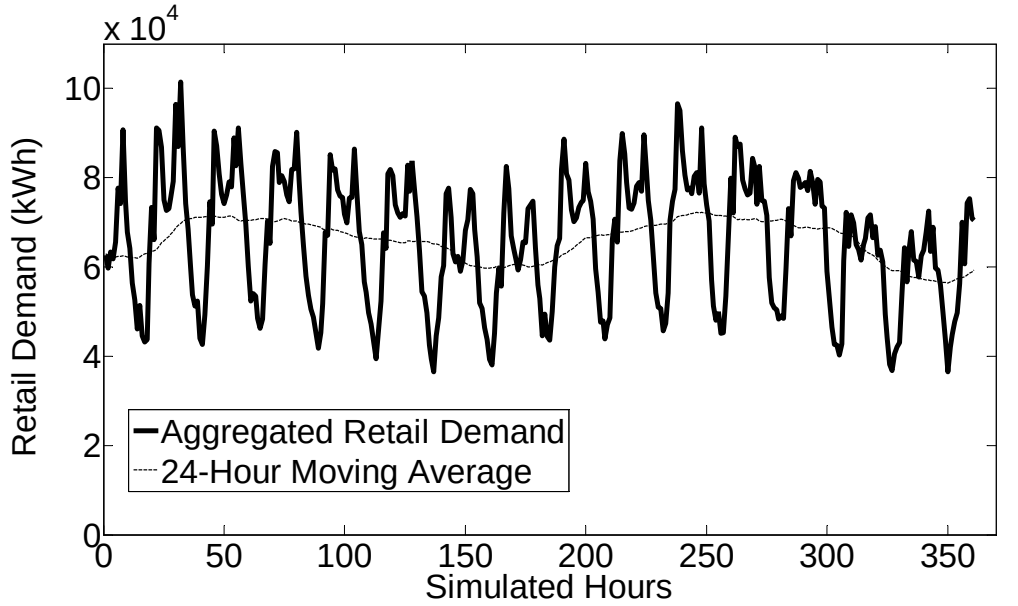
This scenario encompasses a stable market environment where the aggregated energy needed by the retail and the wholesale clearing prices are not volatile. In this case, less than 10% of the volume of electricity produced in the environment is from renewable energy sources. This is the default setting of Power TAC 2014 and 2013. The stable environment setting is mainly characterised by two market aspects from the retailers' viewpoint: a less variable retail demand and a less variable wholesale clearing price. Figures 4.4a and 4.4b show the two aspects. In Figure 4.4a, the average clearing prices for the 24 hours ahead are illustrated. This shows that in a stable market, where retailers are price takers, the clearing price is higher when the time remaining for the delivery is short. Figure 4.4b shows the aggregated retail demand which follows a non-volatile demand pattern. Weekly, daily and intra-daily patterns can be observed in such a stable retail market.

#### 4.3.2.2 Scenario 2: Volatile Electricity Demand

In this scenario, the clearing prices are less variable and the number of end-customers that can partly produce their required energy in the retail market is increased. These customers produce approximately 50% of the energy needed in the retail market. This makes the aggregated demand estimation of the retailer more volatile. The aggregated electricity demand estimated by the retailer is very volatile when the percentage of electricity from renewable sources is high. Figure 4.5b shows the aggregated demand for power that AstonTAC_V2 faces during a certain period of the simulation. This scenario is a realistic and very relevant one for the future modernised electricity grid. In order to support the production of electricity from the demand side, a retailer needs to provide production tariffs that can help the end-consumers to easily re-inject their electricity surplus in the electricity grid and to use the electricity grid in case of an electricity shortage. Enabling production by end-consumers is one of the key motivations of the smart grid.

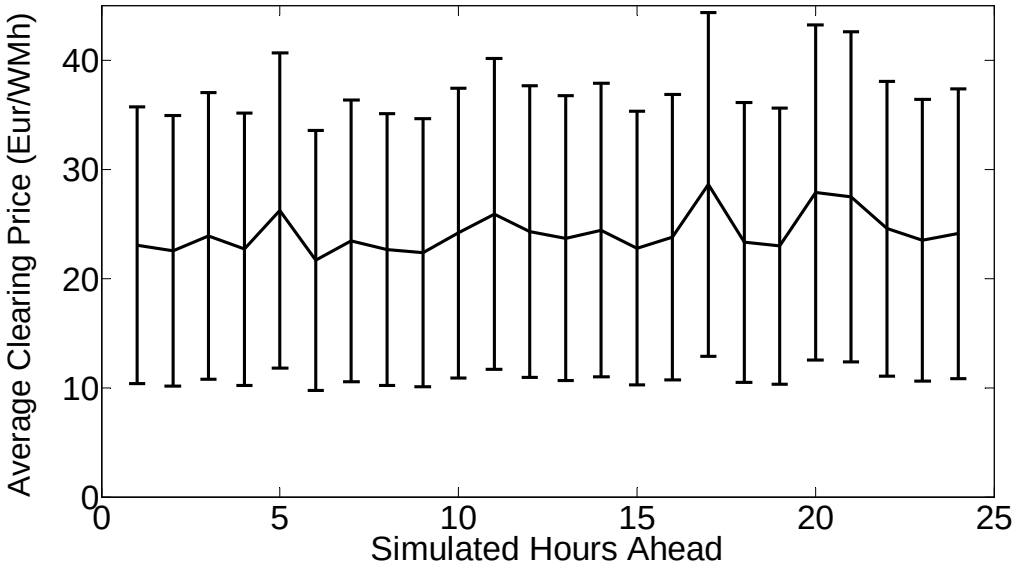



(a) Average Market Clearing Price for 24 Hours Ahead When the Wholesale Supply is Stable. This figure shows the average clearing price for each of the 24 hours ahead when the electricity supply in the wholesale market is stable. The error bars report the standard deviations. The lower the time remaining (time ahead) for electricity delivery, the higher the clearing price could be.

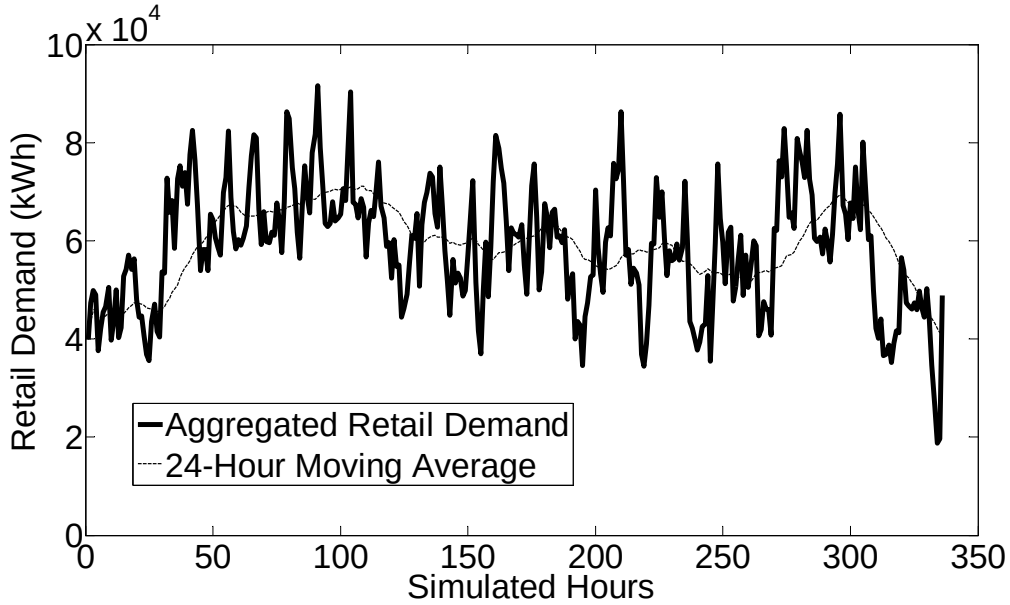



(b) Stable Retail Demand. This figure shows the average of hourly retail demand for more than ten days in an arbitrarily selected game. The daily and intra-daily patterns of the average aggregated retail demand are stable. The curve of the 24-hour moving average shows that the weekly pattern of retail demand is stable.

Figure 4.4: Stable Market Clearing Price and Retail Demand. Figures 4.4a and 4.4b illustrate the behaviour of the environment for Scenario 1.




(a) Average Market Clearing Price for 24 Hours Ahead When the Wholesale Supply is Volatile. This figure illustrates the average clearing price when the volume of electricity supplied by the wholesale market is volatile. The standard deviations are illustrated by the error bars. The wholesale market prices are variable depending on the volume of electricity supplied, which is influenced by the weather situation.




(b) Volatile Retail Demand. This figure shows the retail demand in a volatile environment for more than ten days in an arbitrarily selected game. The aggregated retail demand and the 24-hour Moving average are variable. There is no weekly, daily or intra-daily pattern that is stable. The variable retail demand is largely influenced by the retail electricity production which is dependent on the weather state.

Figure 4.5: Volatile Market Clearing Price and Retail Demand. Figures 4.5a and 4.5b illustrate Scenario 4.

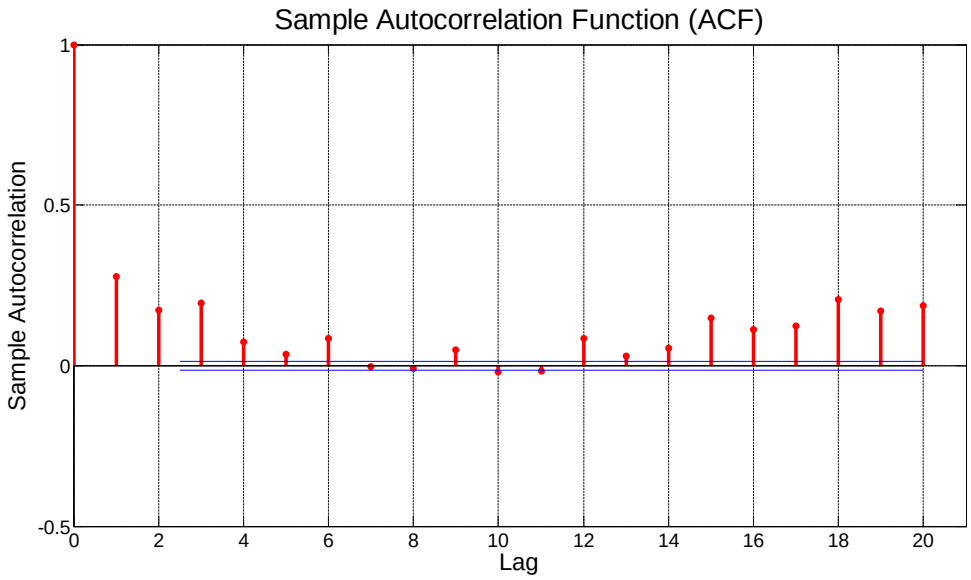



Figure 4.6: Autocorrelation Function for Market Clearing Price with 95% of confidence interval. As all the autocorrelations (with the exception of lag 0) are near-zero, it can be concluded that the wholesale data in Scenario 3 are random.

#### 4.3.2.3 Scenario 3: Volatile Clearing Prices

In this scenario, the retail demand is not volatile and the number of wholesale suppliers that use wind farms for the electricity production is increased. Here 50% of the electricity provided by the GenCos is from renewable energy sources. A volatile electricity supplied by the GenCos implies volatile wholesale market clearing prices. Figure 4.5a illustrates the clearing price pattern in this scenario for a random game with more than 1500 clearing processes. To measure the stability of the wholesale market data, Figure 4.6 shows the autocorrelation plot for market clearing prices generated using the sample autocorrelation function "autocorr" of MATLAB 2011 which computes the sample autocorrelation function of a univariate, stochastic time series with 95% confidence. In the presented figure, all autocorrelations fall within the 95% confidence limits, with the exception of lag 0, whose lag is always 1. Therefore, for 26769 observed clearing prices in a game of scenario 3, the sample autocorrelation function shows that the data are random.

#### 4.3.2.4 Scenario 4: Volatile Market

Figures 4.5a and 4.5b illustrate the market behaviour in Scenario 4. Figure 4.5a shows the average clearing prices for each of the hours ahead. Figure 4.5b shows the aggreg-

ated retail demand for AstonTAC_V2 during a period of an arbitrarily selected game.

### 4.3.3 Results

| | Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 |
|---|---|---|---|---|
| **Aston TAC_V2** | **7.73** (±0.5) | **9.25** (±0.9) | **10.97** (±1.0) | **11.53** (±1.1) |
| MDP Retailer | 10.11 (±0.7) | 12.41 (±1.1) | 14.91 (±1.2) | 15.59 (±1.2) |
| CDA Retailer | 10.16 (±0.6) | Not Applicable | 18.49 (±1.8) | 18.57 (±2.5) |

Table 4.1: Retailers' Imbalance. This table presents the average of the electricity imbalance in percentage for each retailer. The numbers in brackets represent the standard deviations of the arithmetic means in percentage. AstonTAC_V2 performs well across the four scenarios by having the lowest imbalance. The imbalance of the CDA retailer is the highest in each of the four scenarios. MDP approaches utilised by MDP retailer and AstonTAC_V2 achieve a lower imbalance in volatile electricity markets than the CDA approach of the CDA retailer.

Tables 4.1 and 4.2 present the results of the evaluation. While Table 4.1 presents the imbalance ratio $y_T$ (in percentage) at the end of each bidding episode, Table 4.2 represents the percentage of the ratio between average hourly profit $r_t$ (as described in Section 4.2.4) and the average of wholesale market clearing prices $\bar{p}$ made by the retailers.

#### 4.3.3.1 Imbalance Level

**Scenario 1** When the aggregated demand is stable (renewable sources supply less than 10% of retail consumption), there are no major changes in the procurement volume planned for each hour by the retailer agent. In this case all three approaches perform well and the imbalance between supply and demand is low for all retailer agents. AstonTAC_V2 has the lowest imbalance, 7.73% (Table 4.1), whereas the MDP Retailer has an imbalance of 10.11% and the CDA Retailer 10.16%. The advantage of AstonTAC_V2 is that it is the only agent to simultaneously optimise the bids and asks in order to balance its retail demand. The two other agents optimise only the buying process and not the selling of energy surplus.

**Scenario 2** With a volatile demand, the retailer risk of paying a high imbalance fee increases. The CDA retailer avoids to take this risk in the Power TAC environment by not providing tariffs that support energy production by end-consumers. The imbalance ratio has increased for AstonTAC_V2 and the MDP Retailer, in comparison to Scenario 1. The MDP retailer which supports the end-consumers' production cannot sell the energy surplus when the aggregated demand is volatile. This justifies the increase of its imbalance to 12.41%. AstonTAC_V2 can better balance its energy in such situation with an imbalance of 9.25%.

**Scenario 3** The volatility of the wholesale market prices affect the imbalance of the retailers that want to make hourly price profit. It is more probable in this setting that the orders placed by the retailers may not be cleared. This forces the retailers to buy at high prices or to risk the imbalance of the retail demand. In this case the imbalance experienced by the CDA broker is high 18.49%. AstonTAC_V2 has a better balance of energy with 10.97% imbalance. The advantage of MDP and SMDP approaches is that they empower the agents to learn when to buy at high prices in order to satisfy the demand.

**Scenario 4** In this scenario, optimising the balancing of the energy needed is difficult, since the retail demand and the wholesale clearing prices are volatile. AstonTAC_V2 achieves the best performance with an imbalance of 11.53%. The MDP Retailer with an imbalance of 15.59% performs better than the CDA Retailer that has an imbalance of 18.57%. For the CDA Retailer, there is no difference between Scenario 3 and Scenario 4, since it chooses not to offer any production tariffs in the retail market.

#### 4.3.3.2 Profit Level

**Scenario 1** When the electricity generation from renewable sources is less than 10% of the overall wholesale electricity supplied, then the wholesale market clearing prices are less volatile. In this case, the retailer agents as price takers can make more profit by buying during the bidding periods with the lowest clearing price. These bidding periods with the lowest price are the same across the set of simulations. In such a setting, the CDA Retailer that tries to buy all the energy needed at the lowest price made the most profit (9.53%). Since the clearing prices for the bidding periods ahead are not variable, buying all the energy needed at the lowest price is the best approach. The

| | Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 |
|---|---|---|---|---|
| **Aston TAC_V2** | **9.46** (±1.8) | **8.03** (±0.23) | **5.10** (±2.3) | **4.40** (±2.5) |
| MDP Retailer | 9.43 (±2.4) | 6.57 (±0.26) | 3.83 (±2.6) | 3.27 (±2.8) |
| CDA Retailer | 9.53 (±2.3) | Not Applicable | 2.07 (±1.3) | 2.03 (±1.2) |

Table 4.2: Retailers' Profit. This table presents the average of trading profit made by each retailer in percentage. The standard deviations are provided in brackets. AstonTAC_V2 has a high profit margin in all environment settings and the highest average profit in volatile markets. In the stable environment setting (Scenario 1), the CDA retailer performs well. The MDP approach performs well in all environment settings.

MDP Retailer and AstonTAC_V2 can also perform well in this case with respectively 9.43% and 9.46% profit margin.

**Scenario 2** The volatile retail demand also affects the price profit made by the retailers, since the volume of energy needed can change at any time, the retailers may be forced to buy when the clearing prices are high in order to reduce the imbalance cost. In this case, AstonTAC_V2 performs better with an hourly profit of 8.03% compared to the MDP retailer that has an hourly profit of 6.57%.

**Scenario 3** In this setting, the CDA Retailer approach does not perform well. The hourly price profit is the lowest, at 2.07%. Since the CDA approach does not predict the clearing prices of the bidding periods ahead, its ability to make hourly price profit is reduced, whereas the MDP and SMDP approach can perform better with a profit of 5.10% for AstonTAC_V2 and 3.83% for MDP Retailer.

**Scenario 4** The profit margins of the retailer agents are generally lower in this scenario. For the CDA Retailer, a profit margin of 2.03% remains as low as in Scenario 3. AstonTAC_V2 has the highest profit margin of 4.40% in a volatile market. The MDP Retailer that only buys the energy needed and does not sell the surplus has a profit margin of 3.27% higher than CDA Retailer's profit margin and lower than the AstonTAC_V2's profit margin.

This section has evaluated the retailer AstonTAC_V2 in four different market settings. In a stable market, the SMDP-approach performs as well as the other approaches in optimising the trading strategy. When the market is volatile, AstonTAC_V2 outperforms the MDP Retailer that does not consider the selling of the surplus, as well as outperforms the CDA Retailer that does not use the clearing price forecast. As key findings for optimising the short-term procurement in a pool market, the following can be concluded:

1. To bid optimally, the retailer needs good forecasting techniques to predict the clearing prices for the future hours. This enables the retailer to better optimise the volume of energy to order in order to reduce the imbalance risk.

2. Retailers should make use of their ability to sell energy in the wholesale market in order to sell the energy excess and keep supply and demand balanced.

3. In volatile and uncertain environments, SMDP and HAM are appropriate to optimise simultaneously buying and selling in wholesale markets, as well as balancing retail demand and supply.

## 4.4 Summary of the Chapter

This chapter proposes a trading strategy that electricity retailers can use to support short-term procurement when the electricity demand, supply and prices are variable. The strategy principally enables the retailers to simultaneously optimise the trading profit and the balancing cost in different market environments. This optimisation problem is modelled as an SMDP solved through interactions with a simulation environment using the proposed reinforcement learning algorithm MC-HRL. MC-HRL is based on the HAMs that supports the implementation of hierarchy of MDPs and the integration of prior expert knowledge in the HRL. A comparison with other retailers that utilise MDP and CDA approaches for short-term procurement has demonstrated that the SMDP agent is effective in maintaining the lowest electricity imbalance, while achieving the highest margin of the trading profit in a wide range of smart grid market set-ups. The results presented testify to the ability of the presented automated retailer agent to successfully participate in a realistic power trading market and yield profits, outperforming competitors. The next chapter presents the retail SMDP framework, which is used to optimise the retail strategy.

# Chapter 5

# Optimising Retail Market Share and Profit Margin

The two previous chapters present (S)MDP frameworks for optimising the wholesale strategy of a retailer agent. Despite the success of these frameworks, the overall maximisation of the agent's profit also requires an optimisation of the retail strategy. This chapter describes a novel SMDP-based framework that enables the trading agent, AstonTAC_V3, to simultaneously optimise its retail market share and profit margin when trading in a smart grid retail market. These optimisation objectives are motivated by the fact that it is essential for a retailer to make profit in the market without compromising its market share (Wangenheim and Bayón, 2004; Cheema, 2008; Lambrecht et al., 2012).

## 5.1 Introduction

Governments around the world are increasingly turning to sustainable energy sources in order to meet the rising demand in energy while avoiding to exacerbate the global warming problem (Rummery and Niranjan, 2014). As a result, the increased proportion of renewable production in the energy mix requires substantial changes to the technologies applied in electricity wholesale and retail markets to control the electricity distribution. This introduces challenges relating to achieving a good balance between demand and supply, since both electricity generation and consumption heavily depend on the weather alongside other uncertain parameters. While smart grid technologies support both electricity and information flow in the new modernised electricity grid, the dependence of the system on a multitude of uncertain attributes makes both short-

and long-term planning especially difficult for all stakeholders involved. Very few approaches have been proposed that can enable policy makers and market participants to better understand the end-consumer response to retailer strategies such as tariff pricing in a competitive retail market and most are at a primitive stage (Rautiainen et al., 2013).

Against this background, a retailer agent is developed to deal with the uncertainity and the changes imposed by multi-agent retail markets, such as the ones within the Power TAC simulations (Power TAC is introduced in Section 2.1.2). The performance of an electricity retailer is largely dependent on the strategy adopted for pricing of the offered electricity tariffs. The tariff-pricing strategy considers market information as its input, in order to learn the tariff price that is most suitable for the current market. A suitable tariff is at the same time priced to be profitable to the retailer and attractive to the largest possible proportion of the consumers. The focus of this chapter is on a pricing algorithm that empowers an electricity retailer to simultaneously maximise its profit level and market share. This is achieved by using a mechanism that senses the market state to obtain information about the suitability of the current prices and adapting the tariff price to market state accordingly. The purpose of the proposed mechanism is to maximise the expected return of the selected tariff prices over a time horizon.

At the core of the pricing framework is an SMDP framework. The SMDP is an extension of an MDP with abstract actions (see Section 2.2). The advantage of the SMDP and reinforcement learning is that it enables the retailer agent to adapt to the strategy changes of game opponents using the environment states. The main contribution of this work is the implementation of an SMDP decision framework aiding an electricity retailer to define the tariff prices it offers, in order to maximise its profit and market share. During the Power TAC competition in 2013, the agent performed stably and successfully. Moreover, it was the only agent able to perform well in all retail market settings.

The remainder of this chapter is organised as follows. Section 5.2 presents a brief overview of Power TAC environment. Section 5.3 describes related work. In Section 5.4, the SMDP pricing framework is described. Section 5.5 evaluates the retail strategy of AstonTAC_V3. Finally, Section 5.6 concludes the chapter.

## 5.2 Retail Market Simulation

In the Power TAC environments, the retail market provides several types of customers that could be grouped in two classes: (1) small and elemental customers, such as owners

of electric vehicles; and (2) large customers, such as manufacturing facilities. Some of the customers are able to generate part of the electricity that they consume using solar or wind energy sources. The customer behaviour in Power TAC reflects real-life scenarios. While the customers generally try to minimise the cost of their energy bill, they do not always: (1) evaluate newly published tariffs, (2) evaluate all the available tariffs in the market before deciding which tariff to select, (3) select the most suitable tariff, or (4) have the same risk estimation for a tariff contract. These behaviours are influenced by the customer types and customer classes. Additionally, the Power TAC environment enables the design of tariffs with real-world tariff features (e.g., periodic payments, tiered rates, sign-up bonuses, dynamic pricing). Further information on the Power TAC environment is provided in Section 2.1.3.1.

## 5.3 Related Work

As the retail market can be modelled as a series of non-cooperative games with multiple players, the game theoretical approaches provide tools to calculate the optimal behaviour (Dutta, 1999; Binmore, 2007; Fudenberg and Tirole, 1991). However, these approaches make the assumption that the opponents have static strategies. This assumption is not realistic in real life scenarios and in the Power TAC tournament. In Power TAC, two reasons make it difficult to apply game-theoretical approaches. First, the consumers do not always act rationally when taking decisions on electricity tariffs. Second, most of the retailers competing in TAC adapt their trading strategy to the specific market situation (see Section 2.1.1). Thus, the behaviour of market participants is too variable to be captured by game theoretical analysis (Shoham and Leyton-Brown, 2008).

Within the TAC communities, several studies that attempt to create trading agents have been reported. However, reports on trading agents that act as electricity retailers are scarce. The first designs of electricity retailers using MDP and reinforcement learning were presented recently (Reddy and Veloso, 2011b; Peters et al., 2013). They used very simplified, non-realistic settings that include only fixed tariffs and a fixed customer load. Reddy and Veloso (2011b) suggested an environment-specific MDP model that can enable decision making only in the retail market environment presented. Environment-specific MDPs cannot be deployed in markets with different settings where the agent may need a whole new set of variables to estimate the state of the environment and a fresh action set. To address this limitation, (Peters et al., 2013) pro-

posed to use feature selection to identify in each environment the relevant sets of state components and actions to use as MDP parameters. The drawback of this approach is that for each new market setting the MDP model needs to be designed and solved from scratch. Moreover, the approach presented by Peters et al. (2013) cannot be used to model MDPs for volatile environments, as the relevant environment features that are identified for the MDP can change over time. The model put forward in this thesis focuses on applying environment-invariant features to model the MDP. The adaptation of the trading agent in each environment is supported by the environment-specific components sensors and actuators (see Chapters 6 and 7).

Peters et al. (2013); Reddy and Veloso (2011b) considered discrete and specified tariff price changes as resulting actions. This is a common approach when learning a retail strategy (Han et al., 2008). This design approach restrains the adaptation of the retailer agent to market situations, as the price can only be increased or decreased by a fixed margin or to a predefined maximum or minimum price. Thus, the trader agent is not able to offer prices that are adaptable to the market situation. To remedy this, the agent's actuators interpret the MDP actions according to the market situation (see Section 5.4.4).

A more recent approach is evaluated in the Power TAC environment by Urieli and Stone (2014) and proposes a utility optimisation algorithm to decide on the tariff prices. This algorithm optimises the future energy-selling pricing and the future total energy demand given the prediction of the energy-procurement costs. However, in many Power TAC game settings with more than three retailer agents in the same market, this algorithm maximises the market share but generally fails to do the same for the profit level (see Section 5.5).

## 5.4 SMDP Pricing Framework

The aim of the pricing framework is to support the retailers' decision making by determining a tariff price that will simultaneously maximise the number of contracted customers and profit margin. The SMDP methodology enables the modelling of such decision process. Figure 5.1 presents the hierarchy of the SMDP framework. Using this framework at each time step, the decision maker first uses the top-level decision model (SMDP) and information it holds on the current market state to decide which strategy to follow: customer-enticing or profit-oriented pricing. After the selection of the strategy, an MDP is used to decide on the concrete actions to take. Three actions are available

to decide on the price setting: increase, decrease or maintain the current price. Discrete prices are deliberately not used as primitive actions. The interpretation of the primitive actions can be tailored to specific market requirements. For example, a fixed amount can be added to the price every time that the primitive action is *Increase_Price.* The action *Maintain_Price* is used to continue exploiting the current price. The modelling of the MDPs and the primitive actions can easily be adapted to the type of electricity tariff considered. The non-deterministic selection of an abstract action is motivated by the rewards resulting from the execution of the primitive actions of each one of the smaller MDPs. The SMDP mechanism is explained in more detail in Section 5.4.1 and 5.4.2 below, while the specifics of its solution and implementation are given in 5.4.3 and 5.4.4, respectively.

### 5.4.1 Higher Level Decision Making

Using this framework, given the market state, the decision maker can adjust the tariff price while choosing to follow a customer-enticing or profit-oriented policy. Formally, at each step of the simulation, the retailer selects an abstract action to follow using the SMDP (see Figure 5.1). After the completion of the abstract action, the environment transitions from the current state to the next state. The SMDP model can be formally presented as the tuple $\langle S, O, P, R \rangle$ (see Section 2.2 for an introduction to SMDP and HRL):

- *Finite set of states (S)*: the state components are defined by the number of retailers in the market, the market share and the profit level. While the market share enables the retailer agent to indirectly sense the customers' tariff preferences, the profit level enables it to sense the state of its profit margin. Moreover, the defined state components also enable the indirect sensing of the strategies of the market competitors.

- *Finite set of abstract actions, also called options (O)*: there are two options: a customer-enticing option and a profit-oriented option. The agent uses the option $o_t \epsilon O$ to stochastically control the environment and achieves its goals. The agent takes a sequence of options that maximises the control of the environment. The sequence of options for each simulation is defined by the policy function:

$$\pi^o : S \rightarrow O.$$

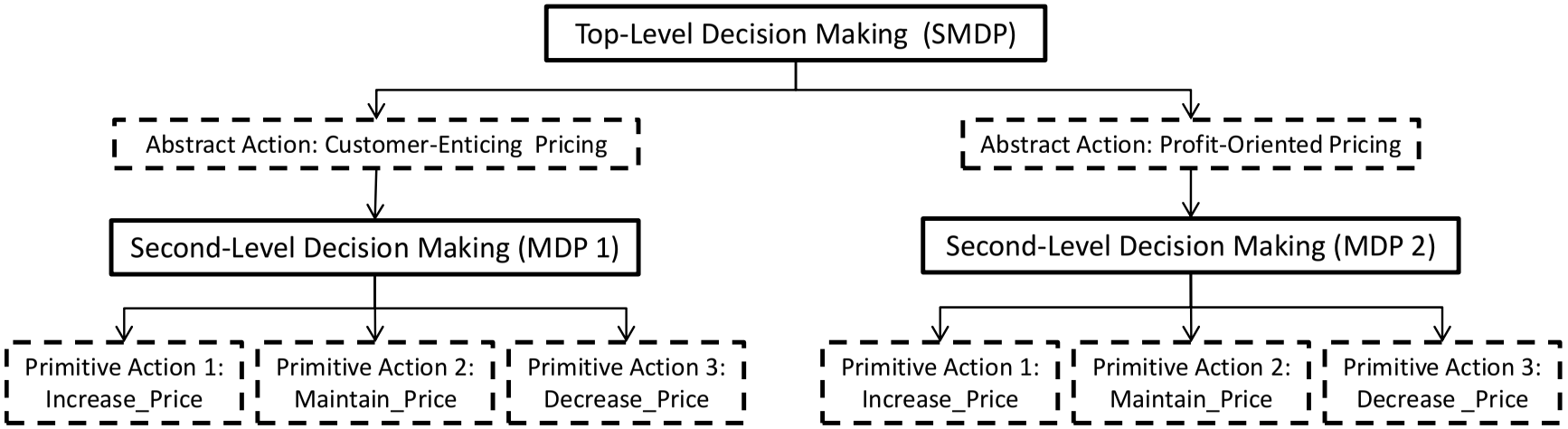



Figure 5.1: SMDP Pricing Framework. This figure illustrates the hierarchical structure of the SMDP framework. The structure is composed of two levels of decision making: a top-level decision making to determine stochastically the selling policy (customer-enticing or profit-oriented pricing) using the SMDP and a second-level decision making to select stochastically a primitive action ( *Increase_Price*, *Maintain_Price* or *Decrease_Price*) to be taken in the retail market using the flat MDPs.

- *Transition probabilities (P)*: The probability $P(s_{t+1}|s_t, o_t)$ of transition from state $s_t \epsilon S$ to state $s_{t+1} \epsilon S$ after the agent has taken the option $o_t \epsilon O$.

- *Reward function (R)*: At each state $s_t \epsilon S$ of the environment, the agent receives the reward $r_t^o$ from the system after the abstract action $o_{t-1} \epsilon O$ is taken from state $s_{t-1} \epsilon S$. The aim of the decision maker is to find the policy $\pi^o$ that maximises the return (cumulated rewards) $R$ in the long run. At each time step $t$, the cumulative discounted reward $R_t$ is defined by:

$$R_t = r_t^o + \gamma^1 r_{t+1}^o + \gamma^2 r_{t+2}^o + \cdots + \gamma^T r_T^o, \tag{5.1}$$

  where $\gamma$ is the discount factor, $0 \leq \gamma \leq 1$ and $T$ the final time step. In infinite-horizon decision process, $T \to \infty$ and $0 \leq \gamma < 1$.

The expected cumulative value of the future rewards $V^{\pi^o}(s_t)$ is the value function corresponding to a deterministic policy $\pi^o$ defined as follows:

$$V^{\pi^o}(s_t) = E\left[R_t|s_t, o_t\right]. \tag{5.2}$$

### 5.4.2 Lower Level Decision Making

The MDPs used for the lower level of decision making are similar to one another. They mainly differ in the setting of their reward functions. The MDPs are defined as follows:

- *Finite set of states (S)*: the state of the environment considered for the lower level decision making is the same as for the SMDP.

- *Finite set of primitive actions ($A_a$)*: MDP actions are *increase*, *decrease* and *maintain* the current price.

- *Transition probabilities ($P_a$)*: Probability ($P_a(s'|s,a)$ ) that the environment transitions from one state $s \epsilon S$ to another $s' \epsilon S$ after taking action $a \epsilon A$. The transition to state $s' \epsilon S$ depends only on the current state $s' \epsilon S$ and the taken action $a$. $P_a(s'|s,a)$ does not depend on the previous actions and previous states.

- *Reward function ($R_a$)*: The reward is defined by the profit margin and the fluctuation in the number of customers. At initialisation of the customer-enticing policy, the primitive action *decrease_price* carries higher rewards. Analogously, the profit-oriented policy assigns a higher reward to the primitive action *Increase_Price*

at initialisation. The higher the reward, the more motivated the decision maker will be to prioritise a specific primitive action for the the selected abstract action. The MDPs are solved through direct interactions with environment. This enables the retailer agent to learn the individual $R$ of each of its primitive actions in the environment. Let $r$ be the reward observed when executing a lower-level MDP. As the purpose of the learning is to simultaneously optimise the profit and the number of customers, the reward $r$ is an aggregation of the shaped profit reward $r_p$ and shaped customer reward $r_c$, calculated as follows:

$$r = w_p r_p + w_c r_c, \tag{5.3}$$

where $w_p$ is the weight assigned to $r_p$ and $w_c$ is the weight of $r_c$.

Given the set of rewards $\{r_1, r_2 \cdots r_N\}$ observed when executing a lower-level MDP for $N$ steps, the higher-level reward $r_o$ is calculated as the sum of $r$ :

$$r_o = \sum_{j=1}^{N} r_j. \tag{5.4}$$

### 5.4.3 Solving the SMDP

To efficiently solve the retail SMDP, the agent needs to learn about the outcome of their actions during the game and not only at the end of the game. This is particularly important because the duration of a decision episode is the length of the whole game, as the agent needs to attract customers and make profit throughout the game. Algorithm 7 implements the HRL algorithm for the SMDP is illustrated in Figure 5.1 and described in Sections 5.4.1 and 5.4.2. As presented in Chapter 2, an appropriate learning approach for the retail decision problem is a TD approach augmented with the eligibility traces -annotated as $e(s,a)$.

Contrary to the wholesale optimisation described in Chapter 4, where the HAM approach facilitates the reduction of the learning time through the use of the expert knowledge, there was no relevant expert knowledge available for the optimisation of the retail strategy. It is not really obvious when to select a profit-enticing or customer-oriented strategy, nor it is clear when to drop or increase the retail price. Because of this knowledge gap, HAM was not the suitable HRL for this problem. A more suitable HRL approach that can be used to solve this problem is the MAXQ approach.

Furthermore, given the complexity of the retail market, it is important to assign

**Algorithm 7** Hierarchical Learning of retail strategy

Initialise:
$Q(s,a) \leftarrow 0$, $Q(s_{top},o) \leftarrow 0$
$totalReward \leftarrow 0$
**for all** Simulations **do**
    Initialise the eligibility traces
    **for** each time step **do**
        **if** subMDP $i$ is not terminated **then**
            observe $r$ for the pair ($s$, $a$) in state $s'$
            choose $a'$ from $s'$ using $\varepsilon$-greedy policy
            $\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$
            $e(s,a) \leftarrow 1$
            **for** all ($s_i$, $a_i$) **do**
                $Q(s_i,a_i) \leftarrow Q(s_i,a_i) + \alpha\delta e(s_i,a_i)$
                $e(s_i,a_i) \leftarrow \gamma\lambda e(s_i,a_i)$
            **end for**
            $totalReward \leftarrow totalReward + r$
            take action $a'$,
            $s \leftarrow s'$; $a \leftarrow a'$
        **else if** subMDP $i$ is terminated **then**
            observe totalReward from the excution of $i$ in state $s'_{top}$
            choose $o'$ from $s'_{top}$ using $\varepsilon$-greedy policy
            $\delta \leftarrow totalReward + \gamma Q(s'_{top},o') - Q(s_{top},o)$
            $e(s_{top},o) \leftarrow 1$
            **for** all ($s_{top}$, $o$) **do**
                $Q(s_{top},o) \leftarrow Q(s_{top},o) + \alpha\delta e(s_{top},o)$
                $e(s_{top},o) \leftarrow \gamma\lambda e(s_{top},o)$
            **end for**
            take option $o'$ and start the corresponding subMDP $i$
            $s \leftarrow s'_{top}$; $o \leftarrow o'$
            $totalReward \leftarrow 0$
        **end if**
    **end for**
**end for**

observed rewards appropriately to all actions taken in the past. A robust way of solving this credit assignment problem is the use eligibility trace, which is more convenient with SARSA than with Q-Learning approach. [1] Thus, the robust and simple HRL approach for retail market optimisation is the MAXQ approach, used with SARSA($\lambda$).

As introduced in Section 2.2.2, two main reasons have motivated the use of SARSA ($\lambda$) in this work. First, because Watkin's Q($\lambda$) requires $e(s,a)$ to be set to zero every time there is an exploratory action, the learning with Q($\lambda$) is slower than the learning with SARSA ($\lambda$), which does not have this requirement. Second, as an off-policy TD control algorithm, Q($\lambda$) requires to start the learning with an estimation policy, which is not only difficult to define appropriately, but also not relevant for the decision problem considered in this chapter. As an on-policy method, SARSA ($\lambda$) provides a more suitable learning mechanism, which is also used by Peters et al. (2013) to solve retail MDPs. Algorithm 7 presents the proposed learning algorithm inspired by the MAXQ value decomposition and using SARSA($\lambda$) to learn each MDP in the hierarchy.

At the beginning of the training (Lines 1-3), the learning parameters are initialised. As required for the SARSA($\lambda$), in Line 5, the eligibility trace $e(s,a)$ is initialised at the beginning of each episode, which is represented by a game. For each time step, the agent uses the SMDP to decide which action to take. Action selections with the lower-level MDPs are described from Line 7 to Line 18, which implement the tabular SARSA($\lambda$) as introduced in Section 2.2.2 and described in Sutton and Barto (1998). Lines 19-31 describe the implementation of SARSA($\lambda$) for the higher-level MDP. In this algorithm, $r_o$ is termed *totalreward,* as it is the sum of lower-level reward $r$.

### 5.4.4 Implementation of an SMDP-based Retailer Agent

AstonTAC_V3's architecture, as illustrated, in Figure 5.2 is essentially composed of three key components: the SMDP Framework, the State Estimator and the Action Interpreter. The State Estimator determines the current environment state, which forms the input to the SMDP reasoning engine. The Action Interpreter aims to interpret the SMDP outputs (*decrease_price*, *increase_price* and *maintain_price*) into concrete price values according to market state.

[1] see Section 2.2.2 for more information on the eligibility trace.

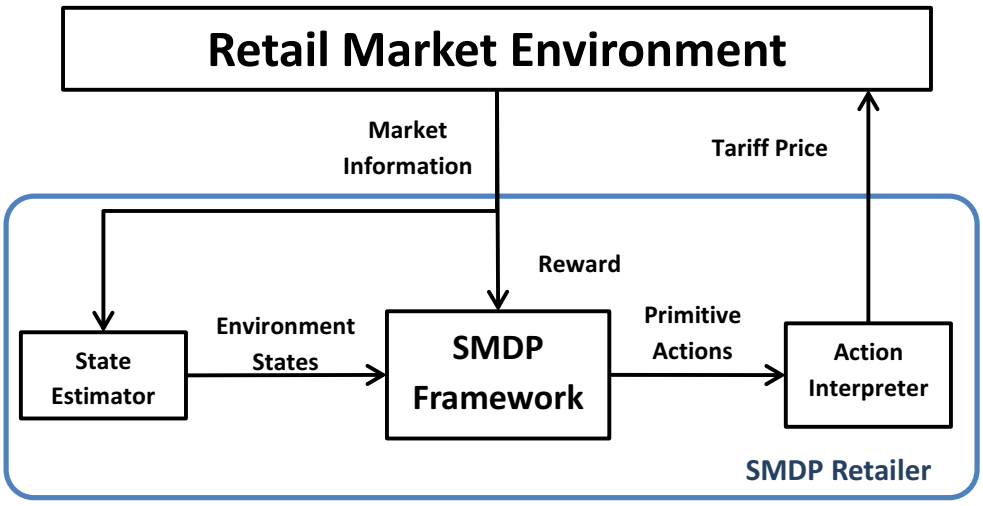



Figure 5.2: Retail view of AstonTAC_V3's Architecture using the retail SMDP. Similar to the AstonTAC's architecture presented in Section 2.4, the retail view is composed of the three core components: the SMDP Framework, the State Estimator and the Action Interpreter. The SMDP Framework is the reasoning engine of AstonTAC_V3. It uses the environment states provided by the State Estimator as inputs to select the action that need to be taken. The Action Interpreter is responsible for executing the action in a environment-specific manner.

#### 5.4.4.1 State Estimator

The State Estimator determines the state of the market share and profit level based on the information in the environment. The market share $m_t$ is computed as follows:

$$m_t = \frac{s_t}{n}, \tag{5.5}$$

where $s_t$ is the number of subscriptions to the agent's tariffs at time $t$ and $n$ is the total number of customers in the market. The profit level $p_{level_t}$ is defined to be the ratio between the hourly profit $p_t$ and the expected hourly profit $\bar{p}$.

$$p_{level_t} = \frac{p_t}{\bar{p}}, \tag{5.6}$$

where $\bar{p}$ is the average of the hourly profit in a similar game. In this research, $\bar{p}$ is specified by the designer given the observed retail market features, which are the number of agents in market and the profiles of the competitor agents. At the beginning of each game, a look up table is used to identify the value $\bar{p}$ to consider for the started game. The hourly profit $p_t$ is considered to be the hourly difference between the amount of cash received from the energy consumers and the cost of energy supplied. In case of

losses, $p_t$ is negative. The profit levels are classified in predefined intervals to form the profit level state. In the experiments, there are three profit level states: *high*, *average* and *low*. The market share is represented with 10 states where each state is represented by 10%, 20%, …, 100% of the market share.

#### 5.4.4.2 Action Interpreter

The outputs of lower level MDP are environment-invariant actions, which tell the trader agent what to do, but not how to execute the actions. For each actions, the agent is provided with a set of activities that could be executed according to the state of the market. In the experiments, AstonTAC_V3 increases or decreases the tariff price using environment-specific steps:

- *Step 1*: After selecting the action $a$, the Action Interpreter defines the new tariff prices $\rho$ by using an environment-specific function $\chi$ which depends on the published retail market prices $\rho_m$, the agent's profit level $p_{level_t}$ and market share $m_t$ so that:

  $$\rho = \chi(\rho_m, p_{level_t}, m_t, a).$$

  In the Power TAC environment, $\rho$ was represented by the look-up table, which maps the market price interval of $\rho_m$, the agent's profit level $p_{level_t}$ and market share $m_t$ and the selected action $a$ to a new tariff price. However, for real-world studies, more appropriate techniques are used to determine the new price given the selected action $a$ (Nevmyvaka et al., 2006; Moriyama et al., 2008; Bertoluzzo and Corazza, 2012).

- *Step 2*: After determining $\rho$, the Action Interpreter chooses between publishing new tariff contracts or modifying the existing tariff contracts. Since customers in Power TAC never evaluate all the tariffs at the same time but a subset of the tariffs that are available in the market, it is important to consider the number of tariffs that are published by the other retailer agents. If the number of tariffs published by other traders is high ($> 20$), AstonTAC_V3 publishes a new tariff, and if this number is low, it updates existing tariff prices.

Although the action to take is specified by the SMDP reasoning component, the execution has to consider the specific-environment requirements.

### 5.4.5 Implementation of the HRL algorithm

The HRL is implemented following Algorithm 7. In the Power TAC environment, the agent can publish or update a tariff contract every six hours. However, in the simulation environment, six hours are not enough for the estimation of the effect a tariff change. Therefore, the retail decisions were considered every 24 hours. The learning rate $\alpha$ is set to 0.042 for a learning window of 24 decision steps ($\alpha = 1/24$).

The discount factor $\gamma$ is set to 0.99 to enable each agent action to have a long-term impact on its success. Moreover, in the Power TAC, the customers behave according to the reputation of the brokers. As the number of daily number decisions is approximately 63 given the duration of the game, the setting $\lambda = 55/63 = 0.87$ makes sure than the agent actions influence its behaviour for a long period.

The training of the agent with these settings was done before and during the Power TAC tournament. Before the tournament, the agents is trained in controlled experiments with profiled competition agents with $\varepsilon > 0.009$, whereas during the tournament with $\varepsilon < 0.005$, the agents learns with tournament agents. The weights $w_p$ and $w_c$ needed to calculate $r$ were set as follows: $w_p = 0.9$ and $w_c = 0.1$.

## 5.5 Evaluation

This section discusses the evaluation of the SMDP Framework that is used by AstonTAC_V3 to trade in an electricity retail market. The performance of AstonTAC_V3 is compared to other electricity retailer agents. The evaluation consists of two parts. (1) The results and analysis of 2013 Power TAC final, which demonstrate the ability of AstonTAC_V3 to use the SMDP Framework to take suitable pricing decisions. (2) The analysis of two games in the actual competition to showcase the ability of AstonTAC_V3 to use the primitive actions to control the maximisation of the profit and customer numbers.

### 5.5.1 PowerTAC 2013 Final

The 2013 Power TAC finals consisted of 60 games with three individual competitions: 21 games with two players, 35 games with four players and 4 games with seven players. The teams participating to the competition were:

- cwiBroker team from Centrum Wiskunde & Informatica, Amsterdam,

- MLLBroker team from the University of Freiburg,
- CrocodileAgent team from the University of Zagreb,
- Mertacor team from the Aristotle University of Thessaloniki,
- TacTex team from the University of Texas at Austin,
- INAORBroker02 team from the National Institute of Astrophysics, Optics and Electronics, Mexico.

The agents competed in different game settings. In order to evaluate AstonTAC_V3's strategy in the retail market, the performance of the three best agents (in addition to AstonTAC_V3) are compared in the retail market for each game size. Table 5.1 presents the results of the retailers' performance in each game size. These tables compare the cash and energy transfers between customers and retailers based on the published tariffs. Thus, these tables present the performance of the retailers in providing competitive tariffs and in making profit. The retailers are classified in each game size according to their ability to receive a high amount of cash for the transferred volume of energy.

The analysis shows that although AstonTAC_V3 did not have the lowest sell price, it could constantly transfer a high volume of energy to the customers. This suggests that AstonTAC_V3 could handle profit and market share maximisation very well. While the profit maximisation is demonstrated by a high sell price and a high percentage of the average cash transferred, the maximisation of the market share is demonstrated by a high percentage of the average volume of energy transferred to the customers. This is particularly the case in 4-player games, where AstonTAC_V3 had the highest average sell price and 33.63% of the energy transferred to customers, whereas TacTex had 41.20% of the energy sold with the lowest average sell price. Consequently, there is just a slight difference between their tariff returns: 37.81% for TacTex and 37.11% for AstonTAC_V3. Compared to the other competing retailer agents, this approach maximises the tariff price and the volume of energy transferred to the retail market.

### 5.5.2 Competition Game Analysis

To evaluate how well AstonTAC_V3 can use primitive actions to control the environment so that it could obtain and maintain a high profit while continuing to attract more customers, two arbitrarily selected games are analysed: one 4-player game (Game 136) and one 7-player game (Game 110). While Game 136 presents the retailer behaviour in

7-player Games

| Broker | Energy Sold | Energy Sold (%) | Returns | Returns (%) | Sell Price |
|---|---|---|---|---|---|
| **AstonTAC** | **1.36E+07** | 39.67 | **8.11E+05** | 45.74 | **0.060** |
| cwiBroker | 1.67E+07 | 48.65 | 7.31E+05 | 41.18 | 0.044 |
| Crocodile | 3.62E+06 | 10.54 | 2.03E+05 | 11.44 | 0.056 |
| MLLBroker | 3.91E+05 | 1.14 | 2.90E+04 | 1.64 | 0.074 |

4-player Games

| Broker | Energy Sold | Energy Sold (%) | Returns | Returns (%) | Sell Price |
|---|---|---|---|---|---|
| TacTex | 5.55E+07 | 41.20 | 3.28E+06 | 37.81 | 0.059 |
| **AstonTAC** | **4.53E+07** | 33.63 | **3.22E+06** | 37.11 | **0.071** |
| cwiBroker | 2.63E+07 | 19.51 | 1.64E+06 | 18.84 | 0.062 |
| MLLBroker | 7.63E+06 | 5.66 | 5.41E+05 | 6.23 | 0.071 |

2-player Games

| Broker | Energy Sold | Energy Sold (%) | Returns | Returns (%) | Sell Price |
|---|---|---|---|---|---|
| TacTex | 6.83E+07 | 30.80 | 6.03E+06 | 32.44 | 0.088 |
| **AstonTAC** | **6.85E+07** | 30.90 | **5.39E+06** | 28.99 | **0.079** |
| cwiBroker | 4.53E+07 | 20.46 | 4.34E+06 | 23.34 | 0.096 |
| MLLBroker | 3.96E+07 | 17.85 | 2.83E+06 | 15.22 | 0.071 |

Table 5.1: Power TAC Final Results of AstonTAC_V3, Retail Market. Column "Energy Sold" represents the average energy volume (in kWh) transferred to the customers. In column "Returns", the average amount of cash (in EUR) transferred from customers to retailers. "Energy Sold (%)" normalises the values in column "Sold Energy". Similarly, "Returns (%)" normalises the values in "Returns". The column "Sell Price" is the corresponding average unit price (EUR/kWh). In a wide range of retail markets, AstonTAC_V3 is able to constantly transfer a high volume of energy to the customers and receives a high amount of cash using the same SMDP Framework.

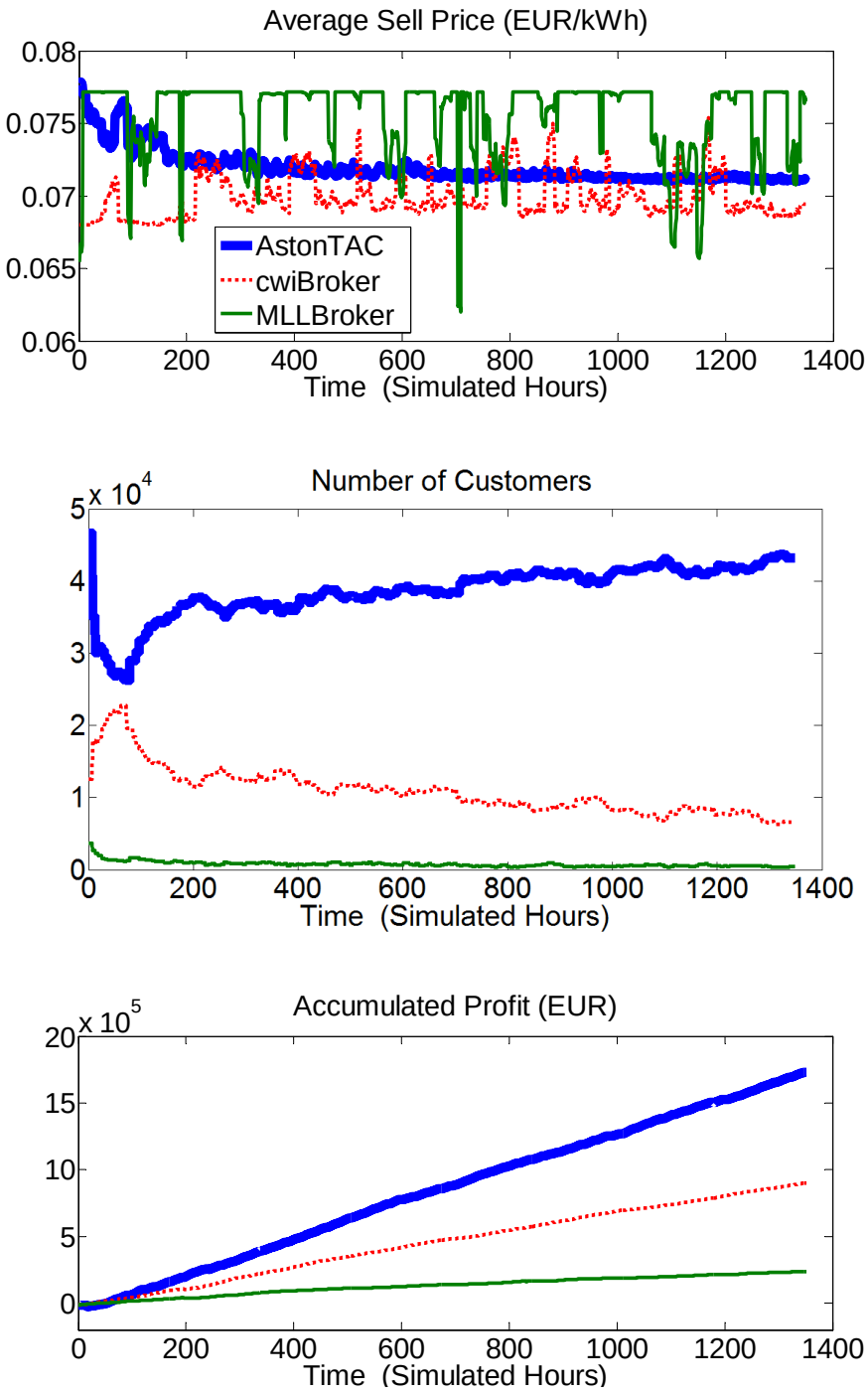



Figure 5.3: Game with Four Retailers (Game 136). Using the primitive actions, AstonTAC_V3 was successful in changing its tariff prices in order to maximise the number of customers and the accumulated profit. In this game with cwiBroker, AstonTAC_V3 has the highest number of customers and the highest profit.

a game where the opponents are less competitive, Game 110 presents AstonTAC_V3's behaviour in a challenging environment. The focus is on the analysis of those retailers that were able to play the game until the end.[2] For each game, the evaluation results show the hourly values of the average sell price, the hourly number of customers subscribed to each retailer's tariffs, and the hourly accumulated profit, which is represented by the amount of cash in the retailer account.

#### 5.5.2.1 Game 136

Figure 5.3 shows the data pertaining to the agents AstonTAC_V3, cwiBroker and MLLBroker in Game 136. In this game, AstonTAC_V3 adapts its tariff prices until time slot 193. Due to the increasing number of customers and hourly accumulated profit, AstonTAC_V3 made few changes to the tariff thereafter. The agent cwiBroker generally appears to wait for other agents to publish their first tariffs and adapts its prices accordingly. This justifies the fact that AstonTAC_V3 has the largest number of customers at the beginning of the game. While MLLBroker and cwiBroker increased their prices after the first tariff publications, leading to a decrease in the number of customer subscriptions, AstonTAC_V3 decreased its prices in order to respond to the market changes. Since the motivation of the customers to evaluate the electricity tariffs available in the market decreases with the increasing number of tariffs, the constant changes in cwiBroker's and MLLbroker's tariff price cause the customers to evaluate their tariffs less often.

#### 5.5.2.2 Game 110

Figure 5.4 shows the results for a 7-player game, Game 110. In this game, AstonTAC_V3 publishes tariffs with lower prices. cwiBroker responds to this and adapts its prices shortly afterwards. This enables cwiBroker to increase its customer subscriptions. TaxTex gradually drops the prices to respond to cwiBroker's price level. Although TacTax, cwiBroker and CrocodileAgent have similar average tariff prices, customers appear locked in to TacTex's tariffs. In this game, AstonTAC_V3 tries to get the targeted market share and a positive increasing profit. In contrast, TacTex targets a big market share, but its profit is lower than that of AstonTAC_V3 and cwiBroker. Overall, as is apparent in the aggregate results shown in Table 5.1, AstonTAC_V3 thrives in competitive

[2]The analysis therefore does not present the results for Mertacor and INAOBroker02. Mertacor could play only for approximatively 300 time slots in each game. Very few customers signed up to INAOBroker02's tariffs.

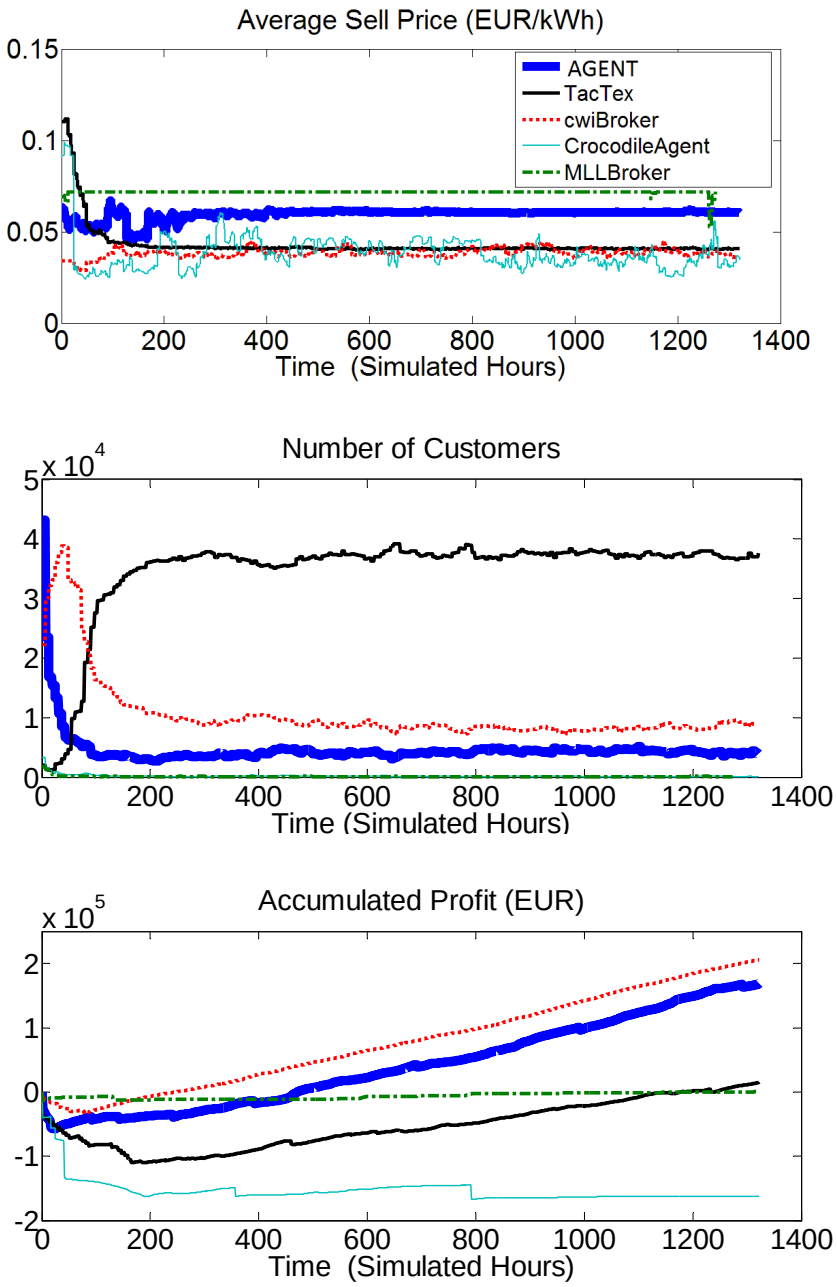



Figure 5.4: Game with Seven Retailers (Game 110). In this game, AstonTAC_V3 attempts to obtain the targeted market share and a positive increasing profit. TacTex seems to target a big market share, but its profit is lower than that of AstonTAC_V3 and cwiBroker.

multiple player environments, accumulating the highest (or nearly highest) volume of returns.

The evaluation of this framework shows that the SMDP-based retailer proposed, AstonTAC_V3, simultaneously optimises its market share and its profit level. The high values market shares represent the readiness of customers to select the tariff contracts provided by AstonTAC_V3. Additionally, AstonTAC_V3 keeps the tariff price as high as possible to maximise its profit. Compared to other retailer agents such as TacTex (presented in Section 5.3), AstonTAC_V3 performs better in diverse competitive smart grid market simulations.

## 5.6 Summary of the Chapter

This chapter describes a selling strategy that a retailer can use to simultaneously maximise its profit margin and its market share. This optimisation problem is modelled as an SMDP, which enables the retailer agent to change the retail price based on the market state which is determined by the state of the number of trading agents in the market, the state of the market share and the state of the retail profit. The SMDP is solved by interacting with the simulation environment using the HRL framework of Dietterich (2000a) and SARSA($\lambda$) as learning algorithm.

A comparison with other retailer agents has demonstrated that my SMDP agent is effective in attracting as many customers as possible while maximising its profits in diverse retail market settings. The success of AstonTAC_V3 in the retail market is attributed to the fact it can change its retail price as to maintain a high number of customers while makes sure that the profit level remains positive and if possible increasing.

The (S)MDP frameworks described in Chapter 5 (optimisation of the retail strategy), 3 (optimisation a wholesale buying strategy) and 4 (optimisation of a wholesale trading) optimise individually the retail and the wholesale strategies of the retailer agents. These separate optimisations of the two strategies neglect their interdependence. This disconnected optimisation approach does not allow for the exploration of alternative wholesale strategy based on the agent's retail situation. The next chapter proposes an optimisation framework that supports the simultaneous optimisation of retail and wholesale strategies.

# Chapter 6

# Novel SMDP Formalisation of a Trader's Decision Problem

Chapters 3, 4 and 5 focus on independently optimising the wholesale and retail strategies. The aim of this chapter is to model all trading agent activities using the MDPs described in the previous Chapters 3, 4 and 5. Furthermore, the described MDPs are based on common environment features which facilitate the transfer of knowledge as formalised in Chapter 7.

While an agent's aim is to minimise the procurement cost by buying the target product quantity on time and at low prices in the wholesale market, in the retail market its aim is to maximise its market share and retail revenue. Many studies separately optimise the wholesale and the retail strategies (He et al., 2006; Benisch et al., 2009) and consider the global optimisation of the trading strategy as intractable (Kiekintveld et al., 2004; Urieli and Stone, 2014). Therefore, each of these strategies is optimised separately and their interdependence is generally ignored, with resulting trading agents not aiming for a globally optimal retail and wholesale strategy. This chapter describes a novel formalisation of the trading problem as faced by a trader agent that essentially purchases the product needed from wholesale trader and sells the final products to customers in the retail market. The proposed SMDP framework is used by AstonTAC_V4 to simultaneously optimise its overall strategy as well as the market-specific strategies.

## 6.1 Introduction

Agent-based simulation is largely used to study the behaviour of complex systems such as market environments. To this end, the TAC community offers a number of

multi-agent simulation environments to promote the development of autonomous trading agents. Among the TAC environments provided, many support the development of retailer agents that make profit by buying in a wholesale market and selling in a retail market. While a broker's aim is to minimise the procurement cost by buying the target product quantity on time and at low prices in the wholesale market, in the retail market its aim is to maximise its market share and retail revenue. Studies on broker agent development have optimised the wholesale and retail strategies individually (He et al., 2006; Benisch et al., 2009), while considering global optimisation of the broker activities as insoluble. Against this background, an approach is proposed to globally optimise its strategy while optimising each activity. It designs individual optimisation modules to model each decision faced by the retailer agent and it defines a structure of an hierarchy of sequential and parallel decision units, where each decision unit is modelled as an MDP-based decision support entity. The whole optimisation of all trading decisions is modelled as a hierarchy of (S)MDPs.

Using the SMDP, which enables the decomposition of a complex MDP in a hierarchy of smaller MDPs, a trading agent can simultaneously optimise the overall strategy, and the retail and procurement strategies. Furthermore, SMDP approaches reduce the computational resources required to solve the original complex MDPs that may be very computationally expensive to solve. This SMDP is learned using the option framework (Sutton et al., 1999a; Barto et al., 2013), which is a well-established HRL technique (Barto and Mahadevan, 2003b). An extension of the option framework as presented in Rohanimanesh and Mahadevan (2001) enables us to model the parallel tasks executed by the trading agent.

The main contribution of this chapter is an SMDP formalisation of the broker's decision problem which enables the simultaneous optimisation of its main goal (to maximise profit) and sub-goals such as minimising procurement costs and maximising retail returns. An evaluation and analysis of a systematic range of controlled experiments show that the SMDP approach enables the new broker termed AstonTAC_V4 to outperform the selected brokers (AstonTAC_V3 and TacTex).

The remainder of this chapter is organised as follows. First, the related work is discussed in Section 6.2. The formalisation of the broker's decision problem is described in Sections 6.3. Section 6.4 presents technical details on the implementation of an SMDP-based broker. Section 6.5 contains a thorough evaluation of the SMDP approach. Finally, Section 6.6 concludes this chapter.

## 6.2 Related Work

Architectures of broker agents are driven by the actions that agents need to execute in the environment. Generally, the architecture of a trading agent has two key components: one to manage a retail market and one to manage the wholesale market (He et al., 2006; Pardoe and Stone, 2006; Benisch et al., 2009). Managing the retail market consists of identifying the retail prices that could be accepted by most of the customers and of forecasting the short- and long-term retail demand. Knowing the projected retail demand, the broker can better plan its procurement in the wholesale market.

In Power TAC, an SMDP approach is proposed to simultaneously optimise the market share and the broker's profit on the retail side (Kuate et al., 2014). MDPs have been proposed to optimise the wholesale strategy (Buffett and Scott, 2004; Kuate et al., 2013; Urieli and Stone, 2014). Combinations of heuristic techniques and game theoretical approaches are used to separately optimise retail and wholesale decisions (Kiekintveld et al., 2004; Wellman et al., 2005). In order to be successful, the brokers generally apply diverse machine learning techniques to predict retail and wholesale prices, and forecast the demand and supply quantities (see Chapter 2).

Simultaneously optimising wholesale and retail market strategies has been considered to be intractable in TAC environments because of the complexity of the environments (Kiekintveld et al., 2004; Urieli and Stone, 2014). In this work, SMDP and HRL are applied to simultaneously optimise the retail and wholesale strategies. SMDP methodology enables us to decompose the complex decision problem of the broker in sub-problems. Furthermore, the SMDP approach enables the development of portable trading agents that can reuse the MDP-modules in new and different simulation environments. Using HRL to model broker's reasoning as an hierarchy of MDP modules reduces the computation and facilitates the transfer of trading knowledge and skills (Barto et al., 2013).

HRL frameworks (Sutton et al., 1999a; Parr, 1998; Dietterich, 2000a) have been used in the past to solve hierarchies of MDPs, SMDPs (Howard, 1971). The option framework of Sutton et al. (1999a) is used in this work, because it provides more suitable formalisms to abstract the decision problems faced by a trader agent. An SMDP model inspired by the work on coarticulation framework (Rohanimanesh and Mahadevan, 2001; Rohanimanesh, 2006) is applied to simultaneously optimise the retail and wholesale strategies. HAM is appropriate when expert knowledge and external component needs to be integrated in the learning system whereas the MAXQ is more

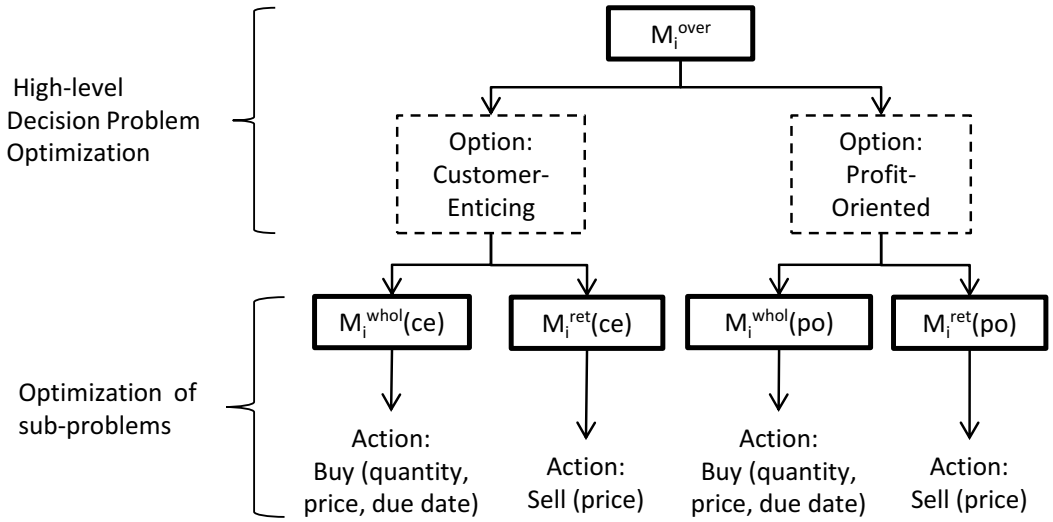



Figure 6.1: Actual SMDP Hierarchy of a Proposed Broker. Notation $M_{i,}^{j}$ stands for the MDP task that optimises the decision problem $j$ of the hierarchy, in simulation environment $i$. This figure extends the retail SMDP illustrated in Figure 5.1 with wholesale MDPs.

appropriate for simplified and straightforward SMDP problem where some option or action selection can be made deterministically. The option framework is a generalisation of the HAM and MAXQ approach which offers a broader set of theoretical tools to manipulate complex SMDPs as the one considered in this research. None of the two techniques HAM (Parr, 1998) or MAXQ (Dietterich, 2000a,b) offer mathematical fundamentals that backup the use of parallel options. Moreover, the option framework can be applied to justify the use of different reinforcement learning techniques to solve the standard MDPs of the SMDP. The SMDP presented in this chapter is learned by using two different RL techniques to solve the standard MDPs of the SMDP hierarchy. Further information on the implementation of option framework is provided in Section 6.4.2.

## 6.3 SMDP Formalisation

In order to perform well, the trading agent needs to optimise all its decision making problems both at a global and an individual level. Each decision problem can be modeled as an (S)MDP so that a hierarchy of (S)MDPs can be structured as illustrated in Figure 6.1. The high-level SMDP is used to decide the overall strategy of the broker. Depending on the option/strategy it selects, MDPs for retail or wholesale markets sup-

port the optimisation of market-specific strategies. While the retail MDP uses the retail price to simultaneously maximise the agent's market share and retail profit, the wholesale MDP uses the order parameters to minimise the procurement cost. This abstraction of the broker's decision problem is applicable to a wide range of decision problems as faced by the type of trading agent specified in Chapter 2.

In the (S)MDP hierarchy, each (S)MDP addresses a specific optimisation problem. Consider $M_i^j$, the MDP task for optimising the decision problem $j$ of the hierarchy, in simulation environment $i$. Specifically, the overall SMDP, $M_i^{over}$, decides the hierarchical, concurrent option to follow: customer-enticing (ce) or profit-oriented (po) option. Based on the selected option, the wholesale MDP, $M_i^{whol}$, decides the quantity of product to buy, given the projected wholesale price. Concurrently, based on the same option information, the retail MDP $M_i^{ret}$ decides the retail price that can simultaneously increase the profit and the market share. Each of the standard MDPs ($M_i^{whol}$ and $M_i^{ret}$) can stochastically select many primitive actions before the option of $M_i^{over}$ that is being executed terminates. The options of $M_i^{over}$ are defined by the tuple $\langle I, \mu, \beta \rangle$ (Precup, 2000):

- an initiation set $I \subseteq S_i^{over}$,
- a policy over SMDP options $\mu : S_i^{over} \times O_i^{over} \rightarrow [0, 1]$,
- a termination condition $\beta : S_i^{over} \rightarrow [0, 1]$.

The option $\langle I, \mu, \beta \rangle$ is available in state $s$, if $s \epsilon I$ . $I$ denotes the set of states $s \epsilon S_i^{over}$ in which the option can be initiated. $I$ and $\beta$ are provided by the SMDP designer in order to facilitate the learning. As the options of $M_i^{over}$ trigger concurrent MDP tasks ($M_i^{over}$ and $M_i^{ret}$), they are the so-called multi-options.

The any-termination condition (see Section 2.3.6) is applied for the $M_i^{over}$ multi-options, as it is convenient to implement and preserves the Markov (or semi-Markov) property of the model. Using the any-termination mode, the selected multi-option terminates when one of the parallel options terminates; all other concurrent options are interrupted. Moreover, the concurrent options of the hierarchical multi-options are executed independently without competing for shared resources. This is the case in the formalisation since the concurrent options are applied in different markets (retail and wholesale), as there is no shared state component between the retail and wholesale MDPs.

## 6.4 Implementation of AstonTAC_V4

To illustrate the implementation of the SMDP approach, this section describes the architecture of the broker (Section 6.4.1) and the implementation of this HRL method (Section 6.4.2).

### 6.4.1 AstonTAC_V4's Architecture

Based on the AstonTAC architecture described in Section 2.1.2, the architecture of AstonTAC_V4 is composed of a reasoning, mapping and environment-specific components. The reasoning component (SMDP Component) uses the hierarchy of (S)MDPs to decide how to act. The mapping components (State Estimator, Action Interpreter) enable information mapping between the reasoning component and the environment-specific components (market sensor and actuator) which, in turn, enables the trading agent to act on and sense a specific environment $i$ (Figure 6.2).

#### 6.4.1.1 SMDP Component

The SMDP Component is the decision making engine of the trading agent. It uses the market states provided by the State Estimator as inputs in order to select the next primitive action, which is then executed by the Action Interpreter. The SMDP Component uses the reward signals provided by the market sensor to learn which action to select according to the market state. As presented in Figure 6.1, this component uses $M_c^{over}$, $M_c^{whol}$ and $M_c^{ret}$ modules to make trading decisions.

**$\mathbf{M_c^{over}}$ Module** It has a set $O_c^{over}$ of two multi-options $\overrightarrow{o_c^{over}} \epsilon O_c^{over}$ (a *customer-enticing option* and a *profit-oriented option*) that defines the overall strategy. Its state space $S_c^{over}$ is determined by three state components: number of retailers in the market, market share and profit level. Let $N_{\overrightarrow{o}}^{ret}$ be the number of time steps that was needed in the retail market during the execution $\overrightarrow{o_c^{over}}$ and $N_{\overrightarrow{o}}^{whol}$ the number of time steps in the wholesale market. At time $t_{over}^*$, the aggregated reward $r_{\overrightarrow{o}}^{over}$ of $M_c^{over}$ when the option $\overrightarrow{o_c^{over}}$ is terminated is:

$$r_{\overrightarrow{o}}^{over} = w_{whol} \sum_{b=0}^{N_{\overrightarrow{o}}^{ret}} r_{c,b}^{whol} + w_{Ret} \sum_{d=0}^{N_{\overrightarrow{o}}^{whol}} r_{c,d}^{ret}, \tag{6.1}$$

where $r_{c,b}^{whol}$ and $r_{c,d}^{ret}$ represent the rewards received at each time step in each market, $w_{whol}$ or $w_{ret}$ is the weight that the model designer will attribute to wholesale or retail reward. The definitive influence of the options on the wholesale and retail MDPs is

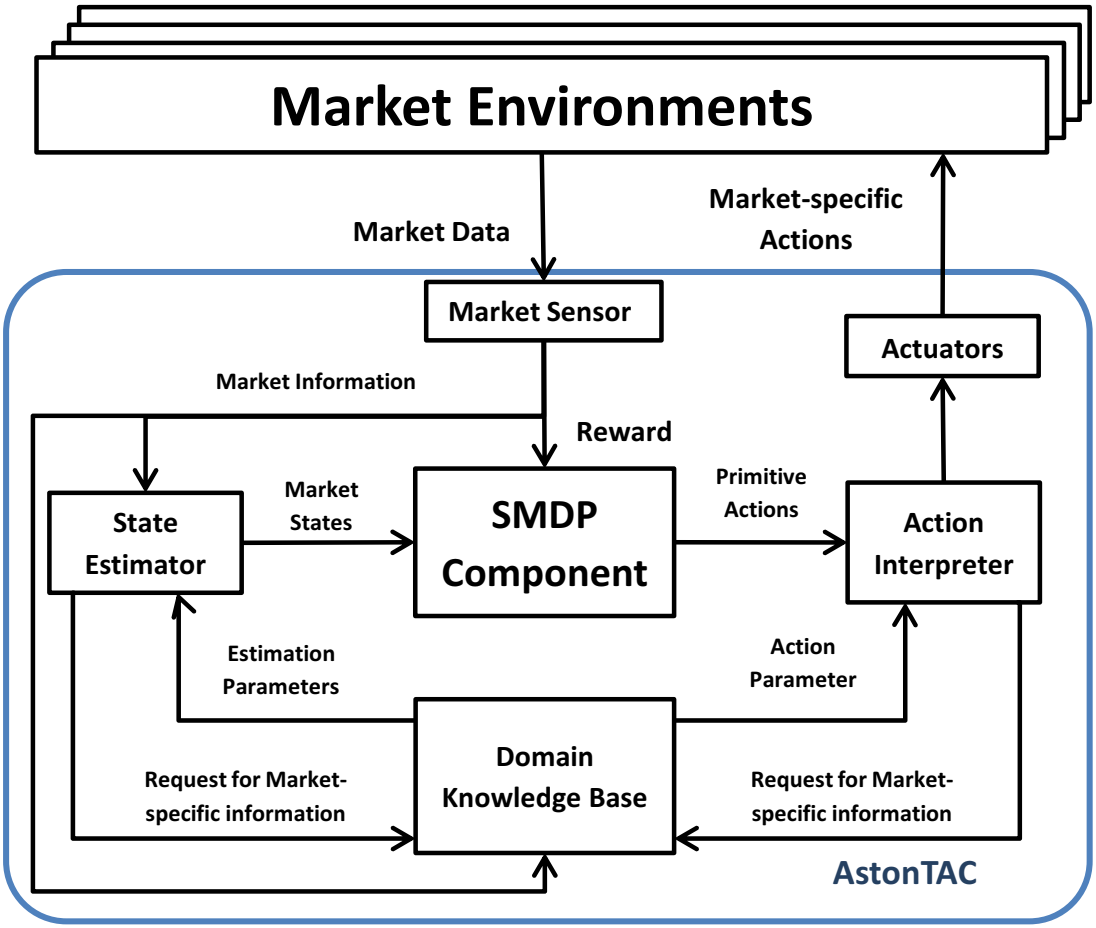



Figure 6.2: AstonTAC_V4's Architecture using the generalised SMDP. Following the architecture illustrated in Figure 2.4, the SMDP component, which is at the core of the AstonTAC_V4's architecture, specifies the behaviour of the agent. The State Estimator determines the environment states, which are the inputs to the SMDP component. The Action Interpreter interprets the SMDP outputs into environment-specific actions, which are executed by the actuators.

learned by the broker. However, the definition of the reward function is used to influence the initial behavior of the agent:

1. When the option "customer-enticing" is selected, an additional reward is given for selecting the action "*decrease*" (retail MDP) and for purchasing all the products needed on time (procurement MDP).

2. When "profit-oriented" is followed, an additional reward is given for selecting "*increase*" (retail MDP) and for buying at lower prices (procurement MDP)

**$M_c^{whol}$ Module** Its action determines the quantity of product to buy given the projected price. Its state space $S_c^{whol}$ is composed of three state components: remaining time until the product is due in the retail market, current procurement imbalance between acquired quantity and needed, and projected wholesale price for a day or an hour in the future. The set of actions $A_c^{whol}$ is defined by a set of percentages. For instance, the set of actions may be to purchase 25%, 50%, 75% or 100% of the remaining quantity of the needed products. The reward values $r_c^{whol}$ of $M_c^{whol}$ are generic in order to capture the reward signals of the different environments. The calculation of the rewards is specified in the mapping layer. The reward depends on the price of the procured product, storage cost, factory cost, late-delivery penalties and cost of the imbalance between retail supply-demand. Depending on the environment the agent is acting in, the weights of these reward components may vary. Chapter 3 details the implementation of a standard wholesale MDP such as $M_c^{whol}$. Since AstonTAC_V4 is modelled to be reused in TAC SCM environment described in Section 2.1.3.2, its wholesale optimisation is influenced by the environment specification of TAC SMC. As the TAC SCM offers only the ability to buy PC components from the wholesale markets and does not enable the wholesale selling, a sell-MDP was not defined for AstonTAC_V4. Chapter 7 describes the knowledge transfer framework needed to reuse AstonTAC_V4 in TAC SCM.

**$M_c^{ret}$ Module** It uses the same state components as $M_c^{over}$ module to form the state space $M_c^{ret}$. Its set of actions $A_c^{ret}$ are *increase*, *decrease* or *maintain* the current retail prices. In Peters et al. (2013), the authors discuss modelling of MDPs for retail electricity trading; they use an environment-specific approach with discrete price changes as MDP actions which are not reusable in new target markets. Generalised MDP actions such as ours are more appropriate to adapt to new markets. The design, implementation and evaluation of a standard retail MDP such as $M_c^{ret}$ is discussed in Chapter 5.

#### 6.4.1.2 Mapping Components

**State Estimator Component** The State Estimator is responsible for identifying the market states needed by the SMDP framework. To do this, it makes use of specific market information provided by the Market Sensor and the parameter provider by the Domain Knowledge Base. Its role is to map environment information into MDP states and reward signals that can be interpreted by the market MDPs $M_c^{whol}$ and $M_c^{ret}$ modules of the reasoning component. Its maps the market information according to the domain knowledge to a discrete market state. The State Estimator does not store data. It processes the data received and send it to the SMDP framework.

**Action Interpreter Component** The Action Interpreter is responsible for mapping the primitive actions selected by the SMDP framework into environment-specific actions that are executed by the agent's actuators.

#### 6.4.1.3 Environment-Specific Components

**Market Sensor** The market sensor acts as a filter that selects the information needed by the broker. It provides the SMDP components with the reward signal; and the State Estimator with the relevant market information

**Actuators** The Action Interpreter is responsible for communication with the specific market according to the communication protocols. Each market follows a set of rules and mechanisms that the agent needs to follow when acting in the environment. For instance, the market rules a retailer agent has to follow when dealing with personal computer trading are different from the ones follow when trading electricity commodity.

**Domain Knowledge Base** The domain knowledge base provides the state estimate and the actuator environment-specific knowledge that can improve their calculations. This component contains information such as maximum or average market price, as well as parameter needed for the forecasting models.

### 6.4.2 Hierarchical Reinforcement Learning

$N$-step TD methods are applied to learn $M_c^{over}$ (Section 2.2.2 provides more details on the $n$-step TD method). Monte Carlo (MC) methods, which have been shown in Chapter 3 to be appropriate for learning this model of the wholesale market MDP, is used to

**Algorithm 8** Hierarchical Learning of a Suitable Trading Strategy

```
1: for all Simulations do
2:     for any time step of the wholesale or retail MDP do
3:         Call βCheck
4:         if β == true then
5:             Call OverallMDP-RL
6:         end if
7:         if β == false then
8:             Call StandardMDP-RL
9:         end if
10:    end for
11: end for
```

solve $M_c^{whol}$. $M_c^{ret}$ is solved using SARSA($\lambda$) as presented in Section 5.4.3. An $\varepsilon$-greedy policy was applied for action selection. To balance exploration and exploitation, $\varepsilon$ varies according to the number of training games. The implementations of $n$-step TD methods, MC methods and SARSA($\lambda$) are introduced in Section 2.2.2.

To implement the HRL algorithm, the MDPs $M_c^{whol}$ and $M_c^{ret}$ are assumed to be Markov, whereas the multi-options $\overrightarrow{o}$ (a *customer-enticing option* and a *profit-oriented option*) are semi-Markov. As the any-termination scheme is used, the termination condition of an option $\overrightarrow{o}$ is governed by the termination condition of each of the lower-level MDPs. Each lower-level MDP considers individual duration of its time step and executes their actions at individual time interval. Algorithm 8 summarises the implementation of the hierarchy of (S)MDPs presented in Figure 6.1. Having a sequence of games to be played, the learning of the MDP architecture is carried out in each game (Line 2). For any time step of $M_c^{whol}$ or $M_c^{ret}$, the learning performs the following operations: evaluate the termination condition (Line 3); according to the termination condition, it runs $M_c^{over}$ module using *OverallMDP_RL* or call *StandardMDPs_RL* to execute the actions $M_c^{whol}$ or/and $M_c^{ret}$ depending on the state of their time step.

#### 6.4.2.1 Termination Condition $\beta$-Check

The termination condition is evaluated for each of the MDPs $M_c^{whol}$ and $M_c^{ret}$. For instance, the level of retail profit or market share can be used to specify the termination condition of the retail MDP, whereas the level of imbalance between supply and demand can be considered to determine the termination of the procurement strategy. In Line 4,

$\beta$-*Check* is called to determine, if one of the termination conditions has occurred. If the termination condition has occurred, the option terminates. *OverallMDP_RL* is used to stochastically select the next options.

#### 6.4.2.2 Learning the Overall Strategy with OverallMDP_RL

The execution of the *OverallMDP_RL* updates the accumulated rewards of $M_c^{over}$. Using $r_{\overrightarrow{o}}^{over}$ and given a $h$-step TD method (with a backup length of $h$), the update of the expected $h$-step return for each visited state-action pairs ($s_c^{over}$, $\overrightarrow{o_c^{over}}$) at time step $t_{over}$ is calculated as follows:

$$R(s_c^{over}, \overrightarrow{o_c^{over}}) \leftarrow h^{-1}\left((h-1)R(s_c^{over}, \overrightarrow{o_c^{over}}) + r_{\overrightarrow{o}}^{over}\right). \tag{6.2}$$

After updating the expected returns of the visited state-action pairs ($s_c^{over}$, $\overrightarrow{o_c^{over}}$), *OverallMDP_RL* selects the next multi-option $\overrightarrow{o}$ using $M_c^{over}$ and call *StandardMDPs_RL* for the selection of the next primitive actions in each market.

#### 6.4.2.3 Learning of the Market MDPs with StandardMDPs_RL

For the $M_c^{whol}$ or $M_c^{ret}$ Module, *StandardMDPs_RL* executes the update of the expected cumulative future discounted reward and the selection of market-specific primitive actions. In the retail market, using the tabular SARSA($\lambda$) as described in Sutton and Barto (1998), the action values $Q(s_c^{ret}, a_c^{ret})$ are updated for all pairs ($s_c^{ret}$, $a_c^{ret}$) as follows:

$$Q(s_c^{ret}, a_c^{ret}) \leftarrow Q(s_c^{ret}, a_c^{ret}) + \alpha\delta e(s_c^{ret}, a_c^{ret}), \tag{6.3}$$

where $\alpha$ is step-size parameter or the learning rate, $e(s_c^{ret}, a_c^{ret})$ is the eligibility trace and $\delta$ the TD error (Sutton and Barto, 1998). For the experimentation, the values of $\alpha$, $\delta$, and $e$ are set as presented in Section 5.4.4.

In the wholesale market, the MC method is used. The end of an episode is to the due time of the required product for the retail market. The parametrisation of $M_c^{ret}$ follows the description made in Sections 3.4 and 3.5.

## 6.5 Evaluation

The performance of AstonTAC_V4, is compared with the performance of the top TAC-brokers, AstonTAC_V3 and TacTex in the Power TAC environment. Urieli and Stone

(2014) describes the implementation of TacTex, which applies a utility optimisation algorithm to specify the retail strategy and applies the CDA algorithm of Tesauro and Bredin (2002) to trade in the wholesale market. This section compares the overall performance as well as the retail performance and wholesale performance of the three agents AstonTAC_V4, AstonTAC_V3 and TaxTex. AstonTAC_V3 is essentially considered here to evaluate the retail approach of AstonTAC_V4.

### 6.5.1 Experiment Setup

A test broker termed AstonTAC_V4 that is implemented as described Section 6.4 is trained with 800 training games in the same environment as AstonTAC and TacTex. 100 test games were used to compare their performances in three scenarios. The implementation of AstonTAC_V4 SMDP follows the description given in Section 6.4.2. As in previous contribution chapters, the training starts with exploratory games using high values of $0.09 < \varepsilon < 0.15$ and the test games are played with $0.01 < \varepsilon < 0.04$. The learning parameters for the wholesale and retail MDPs stay the same as presented in the previous Chapters 3 and 5, which have SMDP based on the common environment features in order to facilitate the knowledge transfer presented in Chapter 7.

By changing the wholesale market settings, the energy demand of the retailer agent influences the wholesale clearing process. In this case the retail demand is used to influence the wholesale prices, which in turn influence the retail prices:

- In scenario 1, the retailer agents are price takers, i.e. their retail demand will have no impact on the wholesale prices. Therefore, for scenario 1, the interdependence between wholesale and retail markets is 0%.
- In scenario 2, the broker orders can partly influence the wholesale clearing process: the market interdependence is 50%.
- In scenario 3, the brokers are set to be price makers. This means that the brokers fully influence the wholesale prices through their orders (100% interdependence).

### 6.5.2 Results

The results presented in Tables 6.1 and 6.2 show that AstonTAC_V4 outperforms AstonTAC_V3 and TacTex across the scenarios. Considering the average total cash received by each agent, the more the markets are interdependent, the better is the As-

tonTAC_V4 performance and the worse the performance of AstonTAC_V3 and TacTex. In the wholesale market, using $M_c^{whol}$, AstonTAC_V4 outperforms TacTex in optimising the order price and the energy imbalance by having the lowest average order price and lowest energy imbalance. The imbalance is the difference between the energy purchased by the broker and the broker's retail demand. In the retail market, AstonTAC_V4 outperforms AstonTAC_V3 and TacTex in optimising the retail price by having a higher average retail price and a higher average revenue. AstonTAC_V3 and TacTex do well in optimising the market share; however, by selling the energy at a lower price, they have a lower average revenue in the retail market. These results show that the SMDP approach outperforms well-performing brokers that optimise each sub-problem individually. This approach performs well irrespective of the level of interdependence between the two markets.

| | | Overall Result | Retail Market | | |
|---|---|---|---|---|---|
| Brokers | Market Inter dependence | Average Total Cash (EUR/game) | Average Market Share | Average Price (EUR/kWh) | Average Revenue (EUR/game) |
| **Aston TAC _V4** | **0%** | **6.82E+06** | **48.70%** | **0.084** | **10.93E+06** |
| Aston TAC_V3 | 0% | 6.15.E+06 | 51.30% | 0.074 | 10.17E+06 |
| **Aston TAC _V4** | **50%** | **6.45E+06** | **48.64%** | **0.081** | **10.74E+06** |
| Aston TAC_V3 | 50% | 5.18E+06 | 51.36% | 0.072 | 9.37E+06 |
| **Aston TAC _V4** | **100%** | **6.05E+06** | **47.09%** | **0.078** | **9.85E+06** |
| Aston TAC_V3 | 100% | 4.26E+06 | 52.91% | 0.063 | 9.03E+06 |

Table 6.1: Performance of the HRL Approach in Power TAC environment. AstonTAC_V4, is compared to AstonTAC_V3 in the Power TAC simulation environment. This table shows the results of their overall, wholesale and retail performances in different market settings.

| | | Overall Result | Wholesale Market | | | Retail Market | | |
|---|---|---|---|---|---|---|---|---|
| Brokers | Market Inter dependence | Average Total Cash (EUR/game) | Average Price (EUR/MWh) | Average Imbalance | Average cost (EUR/game) | Average Market Share | Average Price (EUR/kWh) | Average Revenue (EUR/game) |
| **AstonTAC _V4** | **0%** | **6.49E+06** | **25.91** | **8.3%** | **3.51 E+06** | **47.47%** | **0.077** | **10.79E+06** |
| TacTex | 0% | 5.71.E+06 | 27.17 | 10.3% | 4.23 E+06 | 52.53% | 0.064 | 9.97E+06 |
| **AstonTAC _V4** | **50%** | **5.16E+06** | **26.24** | **8.8%** | **3.56 E+06** | **47.69%** | **0.075** | **9.94E+06** |
| TacTex | 50% | 4.79E+06 | 30.84 | 19.2% | 4.11 E+06 | 52.31% | 0.062 | 9.08E+06 |
| **AstonTAC _V4** | **100%** | **6.05E+06** | **27.10** | **9.5%** | **3.70 E+06** | **48.29%** | **0.071** | **10.17E+06** |
| TacTex | 100% | 4.26E+06 | 34.25 | 20.4% | 4.71 E+06 | 51.71% | 0.060 | 9.19E+06 |

Table 6.2: Performance of the HRL Approach in the Power TAC environment. AstonTAC_V4, is compared to TacTex in the Power TAC simulation environment. This table shows the results of their overall, wholesale and retail performances in different market settings.

## 6.6 Summary of the Chapter

This chapter addresses the broker's decision-making problem by proposing a novel formalisation of it as an SMDP. This enables the broker to simultaneously optimise its retail and wholesale strategies without compromising its global strategy. SMDP described is composed of three MDPs: and overall strategyMDP, a retail MDP and a procurement MDP. The SMDP described in this chapter is a based on the MDPs presented in previous Chapters 3, 4, 5 which use the common environment features to model the MDPs. As it will be presented in Chapter 7, the use of the MDPs with common features facilitates the transfer of knowledge.

An evaluation and analysis of the formalisation show that the broker outperforms the top TAC-broker. Despite the success of the SMDP approach in reducing the complexity of the broker's decision problem, solving the resulting SMDP still turns out to be a relatively intense computation task. To further reduce the computational effort required to acquire an efficient solution, a novel knowledge transfer framework is proposed in Chapter 7.

# Chapter 7

# Transfer of Trading Skills

Chapter 6 describes the formalisation of the broker's problem as an SMDP and presents the design of the broker architecture which is composed of three components: the SMDP reasoning component, the mapping components and the environment-specific components. This chapter puts forward a knowledge transfer framework that enables AstonTAC_V5[1] to reuse the SMDP reasoning component in new market. Enabling the trading agent to transfer trading skill speeds up its SMDP learning and boosts its initial performance in new markets.

## 7.1 Introduction

In a manner analogous to human brokers that use the same type of reasoning (aided by decision support systems) to trade and adapt to different markets, the aim is to build brokers possessing an invariant reasoning component that is able to profitably trade and adapt to individual market using market-specific sensors, actuators and mapping components. This ability to transfer knowledge has been recently explored in robotics for simplistic environment settings. This chapter extends the existing work to create a new knowledge transfer framework that is applied to a complex trading environment. The transfer of trading skills yields two key advantages: it reduces the learning necessary when entering a new market and boosts the initial performance of the agents. Moreover, in similar markets the agent may perform well without extra training.

The proposed skill transfer framework is driven by the design of a hierarchical reasoning structure for the agent, which is based on market-independent features as presen-

[1] AstonTAC_V5 is an implementation of AstonTAC_V4 (see Chapter 6) that transferred the trading knowledge from Power TAC markets to TAC SCM markets.

ted in Chapter 6. The proposed transfer framework has been thoroughly evaluated in two well-established multi-agent simulation environments within the Trading Agent Competition (TAC) community. Analysis of controlled experiments shows that AstonTAC_V5, the portable version of AstonTAC, is able to perform well in a wide range of environments by re-using knowledge acquired in previously experienced settings. The remainder of the chapter is organised as follows. First, the research background is briefly presented in Section 7.2. Then, the skill transfer approach is described in Section 7.3. Subsequently, the proposed approach is evaluated in Section 7.4. Finally, Section 7.5 concludes the chapter.

## 7.2 Related Work

In general, the purpose of transfer learning (TL) is to reuse knowledge acquired from learned source tasks in order to facilitate the learning of a new target task. This is particularly useful when the target task is complex, or difficult/impractical to use for training and the learning is time-consuming. TL is applied in machine learning for classification, regression, clustering (Pan and Yang, 2010) and for reinforcement learning (Lazaric, 2012). This study focuses on transfer in reinforcement learning.

In this work, $M_i = \langle S_i, A_i, P_i, R_i \rangle$ denotes the MDP task in a simulation environment $i \epsilon \mathbb{N}$. The *task domain*, $D_i = \langle S_i, A_i \rangle$, is defined by the state and action spaces of $M_i$. At the beginning of the learning $T_i = \langle P_i, R_i \rangle$ is defined as the *task objectives* of $M_i$ and at the end of the learning it represents the *task skills*.

Most of the studies that have been conducted in reinforcement learning in order to enable knowledge transfer can be grouped based on the differences allowed between the MDP tasks $M_i$ (Taylor and Stone, 2009; Lazaric, 2012). Many studies have been reported on transfer learning between two tasks ($M_s$ as source tasks and $M_t$ as target task) that have the same domain $D = \langle S, A \rangle$ (state components or variables and set of actions remain invariant for the two tasks) and different task objectives $T_i$ (Sherstov and Stone, 2005; Asadi and Huber, 2007; Mahadevan and Maggioni, 2007; Ferrante et al., 2008; Lazaric, 2008), whereas other studies have been reported on multi-task learning of different task-specific objectives $T_i$ in the same domain $D$ (Wilson et al., 2007; Lazaric, 2008; Mehta et al., 2008a; Lazaric et al., 2010; Frommberger and Wolter, 2010; Snel and Whiteson, 2014). When the task domains are different, two approaches are used in the literature to enable transfer: domain-invariant features (Konidaris and Barto, 2007; Banerjee and Stone, 2007; Sharma et al., 2007; Croonenborghs et al.,

2008; Konidaris et al., 2012) or inter-task mappings (Torrey et al., 2006; Taylor et al., 2007; Taylor and Stone, 2007).

Inter-task mappings have been proposed to support information transfer through hand-coded mappings between a source and a target task that are related but have different states and actions (Torrey et al., 2006; Taylor et al., 2007; Taylor and Stone, 2007). However, inter-task mapping approaches are not suited for dealing with transfer across multiple tasks, as they do not cater for merging knowledge from potentially numerous source tasks before transferring the knowledge to the target task.

Contrary to inter-task mappings, an agent-centric approach based on task-invariant features has been proposed (Konidaris and Barto, 2007; Konidaris et al., 2012). It modelled an MDP task $M_i$ as the tuple $\langle S_i, A_i, P_i, R_i, D \rangle$ where $D$ is the space of features shared across all related tasks and called the agent-space. The agent-space approach enables the knowledge transfer across related tasks that are reward-linked. Related tasks are reward-linked, if their reward functions individually allocate rewards the same way. Furthermore, this approach assumes that tasks must have the same set of actions for the skills transfer to be possible through the portable options.

This chapter is essentially concerned with knowledge transfer across MDP tasks with different domains $D_i$ and different objectives $T_i$. On this instance, a trading agent is able to transfer trading knowledge between electricity and PC markets. In this case, explicit mappings between tasks are needed to support knowledge transfer across tasks. By using explicit mappings, the agent-centric approach is extended to enable transfer knowledge and skill transfer across related tasks that may not be reward-linked or have the same set of actions. The key idea of the proposed approach is the use of an invariant abstract task representation $M_c$ that is common to all tasks $M_i$. During the learning, all the source tasks $M_s$ are mapped to $M_c$ and learning happens at the level of $M_c$. In new environments, to support the knowledge transfer, target tasks $M_t$ are also mapped to $M_c$. This enables the agent to transfer previously acquired knowledge to a new market and subsequently hone its trading skills to the characteristics of that specific market.

## 7.3 Knowledge Transfer Framework

The knowledge transfer framework can be split in two phases: a transfer phase and learning phase. During the transfer phase, the portable trading skills are transferred to improve the initial agent performance. The learning phase enables the trading agent to improve the portable trading skills in order to adapt better to a new environment.

### 7.3.1 Transfer Phase

A common abstract MDP task $M_c^j = \langle S_c^j, A_c^j, P_k^j, R_k^j \rangle$ which is characterised by an invariant domain $D_c = \langle S_c, A_c \rangle$ and the portable skills $T_k = \langle P_k, R_k \rangle$ is defined in order to enable skills transfer. When starting to solve the task $M_i^j$ in a new environment $i$, a mapping $h_i^j$ is provided by the designer and is used to map the task domain $D_i$ to the common domain $D_c$. Mapping $h_i^j$ is defined by the tuple $\langle f_i^j, g_i^j, z_i^j \rangle$ of surjective functions so that:

- $f_i^j : S_i^j \rightarrow S_c^j$ maps the state space (or state components) of $M_i^j$ to that (or those) of $M_c^j$.
- $g_i^j : A_i^j \rightarrow A_c^j$ maps the action space of $M_i^j$ to the action space of $M_c^j$.
- $z_i^j : y_i^j \rightarrow y_c^j$ maps the reward intervals defined by $y_i^j$ of $M_i^j$ to common reward intervals defined by $y_c^j$ of $M_c^j$. Function $y_i^j$ maps each reward $r_i^j$ in the range of the reward function $R_i^j$ to a unique interval $[a, b] = \{x \epsilon \mathbb{R} | a \leq x \leq b\}$.

Mapping $h_i^j$ makes it possible for the different MDP domains $D_{i,}^j = \langle S_i^j, A_i^j \rangle$ to seem the same to the agent. Within the knowledge transfer framework an MDP task $M_i^j$ are defined as composed of task domain $D_i$ and task objectives $T_i$. When solving a new MDP task $M_i^j$, which has a task-specific domain and task-specific objectives, the former is mapped to the invariant domain $D_c$ to enable transfer of the portable task skills, while the latter are still to be solved. This results in a new reduced task model $M_{i'}^j = \langle S_c^j, A_c^j, P_i^j, R_i^j \rangle$ that has an invariant task domain $D_c \langle S_c, A_c \rangle$ and task-specific objectives $T_i \langle P_i, R_i \rangle$. When entering a new simulation environment $i$, $h_i^j$ is used to transform the specific task $M_i^j$ in a reduced model $M_{i'}^j$ $\langle S_c^j, A_c^j, P_i^j, R_i^j \rangle$ with a common domain $D_c$ $\langle S_c, A_c \rangle$ and task-specific objectives $T_i \langle P_i, R_i \rangle$.

### 7.3.2 Learning Phase

Having defined $M_{i'}^j$ for each task, transferred knowledge $T_k^j = \langle P_k^j, R_k^j \rangle$ is provided to the agent at the beginning of the training in environment $i$. The aim is to improve the transferred skills $T_k^j$ to solve $M_{i'}^j$. Let $L_j$ be the learning algorithm used to solve $M_{i'}^j$:

$$L_j : T_k^j \rightarrow T_i^j. \tag{7.1}$$

In each new environment, $T_k^j$ is initially used and subsequently improved to approximate the skill required for achieving the task objectives $T_i^j$. Consider $\Omega$ the space of all

tasks $M_i^j$. $\Omega$ is the space of all tasks that the trading agent has experienced in the past and is about to experience in the current environment; it does not refer to tasks to be solved further in the future. The aim of the approach is to get a $T_k^j$ to be as close as possible to all $T_i^j$ of all $M_i^j$ in $\Omega$. Formally, at each step of the decision making, the aim of the training algorithm $L_j$ is to minimise the difference between $T_k^j$ (portable skills) and $T_i^j$ (task-specific skills).

The state-value function $V^\pi$ or action-value function $Q^\pi$ and policy $\pi$ are the transferred knowledge, as the abstract task parameters $\langle S_{c,}^j, A_c^j, P_k^j, R_k^j \rangle$ are the transferred knowledge.

In this work, it is assumed that the broker designer provides $h_{i,j}$ and $\Omega$ so that the transfer of skill is still beneficial for the execution of a new task. The transfer is not beneficial, if the agent performs worse with transfer than without transfer. Moreover, as the agent aims to continually learn, the learning of new tasks should continually improve the acquired skill. In general, the acquired skills should not deteriorate with new tasks, otherwise continuous learning across different tasks may not be really possible. Under-performing because of transfer and deterioration of the acquired skills is called in this thesis negative transfer. In this work, it is assumed that the broker designer provides $h_{i,j}$ and $\Omega$ so that a negative transfer is avoided. Negative transfer has not been hindering information transfer in the experiments presented, and thus has not been extensively studied as part of this work.

The proposed approach makes it possible for a trading agent to use previously acquired trading skills to perform well across a wide range of new settings with little or no environment-specific training. Such a portable broker has some initial trading knowledge that will guide its actions while learning to adapt to a new environment's specific characteristics.

## 7.4 Evaluation

The evaluation of the transfer approach proposed in this chapter consists of evaluation of the novel SMDP formalisation described in Chapter 6. The SMDP components are reused in a new environment by applying the mapping presented in Section 7.3. In this evaluation, we transfer the Q-table from the Power TAC environment to TAC SCM environment. The simulation environments used are outlined in Section 7.4.1. Section 7.4.2 presents the experiment setup and Section 7.4.3 analyses the results.

### 7.4.1 Simulation Environments

Two challenging TAC environments TAC SCM (Collins et al., 2006) and Power TAC (Ketter et al., 2015) are used in order to evaluate the performance of the SMDP approach. Section 2.1.3.1 describes TAC SCM and Power TAC in more detail.

### 7.4.2 Experiment Setup

The SMDP presented in Chapter 6, is built and tested within the Power TAC environment. Fundamentally, the SMDP is designed in a generic manner to be adaptable to wide range of markets including TAC SCM. Similar to the Power TAC, the SCM TAC has retail and wholesale markets. Analogously to Power TAC, in the retail markets, the brokers need to optimise the retail profit by setting competitive and profitable PC prices. Similarly, in the wholesale market, the broker needs to buy the PC components on time to satisfy the retail market and try buy when the prices are lower. TAC SCM game server offers different set of data to the agents. As presented in Figure 6.2, the state estimator enables the agent to translate at run-time the environment signals $s_i \epsilon S_i^j$ in common state $s_c \epsilon S_c^j$. Similarly, actions $a_c \epsilon A_c^j$ selected by the $M_c^j$ (MDPs presented in Chapter 6) are translated by the action interpreter into environment specific actions $a_i \epsilon A_i^j$. While the state estimator implements the state space mapping $f_i^j$, the action interpreter implements the action mapping $g_i^j$ and the reward mapping $z_i^j$ is defined offline by the modeller. Chapters 3, 4 and 5 describe how the environment-specific signal can be mapped to shaped rewards in the Power TAC.

Two series of controlled experiments were run to evaluate the performance of this knowledge transfer approach. In each test environment a baseline broker was defined to facilitate cross-environment comparison of test brokers in different scenarios. The baseline brokers are implemented as presented in Section 6.4 and have 50 initial training games in the environment in which they are situated. For the test brokers, 100 of the games were used for training and 20 as test for all experiments. To evaluate the transfer approach, two groups of controlled experiments are run: one to evaluate the overall performance of the knowledge transfer approach and one to evaluate the knowledge transfer ability of each MDP in the reasoning layer.

### 7.4.3 Experiment Results

**Overall Performance** Table 7.1 shows the results of the knowledge transfer by comparing the performance of AstonTAC_V5 to the baseline agent. The comparison is based on the metrics used in previous transfer learning work (Taylor and Stone, 2009; Lazaric, 2012). The column *Training-SCM-Power* refers to training in TAC SCM then in Power TAC and *Training-Power-SCM* to training in the inverse order. The knowledge transfer enables the broker to increase its performance with less training in the target environment. As expected, when training and testing happen in the same environment, the broker performs better than when the source and target environments differ. However, if the broker receives training in both environments, it outperforms the broker that is tested and trained in the same environment. This presents a strong incentive for broker agents to transfer knowledge across simulation environments.

**Performance of each MDP module** Table 7.2 shows the performance of each MDP module by comparing the performance of AstonTAC_V5 to the baseline agent. Using the baseline broker to initialise AstonTAC_V5, each MDP module is trained individually and measured the jumpstart in the target environments. These results show that a transfer of the high-level skills ($M_c^{over}$) is more advantageous than the transfer of lower level ($M_c^{whol}$ and $M_c^{ret}$) as the highest jumpstarts are obtained when training only $M_c^{over}$. Finally, transferring the $M_c^{ret}$ skills is slightly more beneficial than the transfer of $M_c^{whol}$ skills. This is probably due to the fact that the retail profit is generally higher than the procurement profit.

| | Training in TAC SCM | | | Training in Power TAC | | | Training in SCM->Power | | | Training in Power->SCM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Total Reward | Jump start | Training Time | Total Reward | Jump start | Training Time | Total Reward | Jump start | Training Time | Total Reward | Jump start | Training Time |
| Test in SCM | 130.9% | 0% | 100 games | 127.7% | 27.7% | 0 | 144.3% | 44.3% | 100 games | 141.4% | 0% | 200 games |
| Test in Power | 123.6% | 23.6% | 0 | 131.6% | 0% | 100 games | 144.8% | 0% | 200 games | 146.2% | 46.2% | 100 games |

Table 7.1: Overall Performance of the Knowledge Transfer Framework. Using the metrics total rewards, jumpstart and training time Taylor and Stone (2009); Lazaric (2012), this table presents the comparison of the performance of different test brokers according to the types training received. The test brokers' performances are compared to a baseline broker that is defined in each test environment.

| | Test in TAC SCM | | | Test in Power TAC | | |
|---|---|---|---|---|---|---|
| | $M_c^{over}$ | $M_c^{whol}$ | $M_c^{ret}$ | $M_c^{over}$ | $M_c^{whol}$ | $M_c^{ret}$ |
| Training in TAC SCM | 118.8% | 113.6% | 115.7% | 114.4% | 106.9% | 107.6% |
| Training in Power TAC | 117.6% | 108.1% | 108.2% | 120.0% | 115.2% | 119.1% |

Table 7.2: Jumpstart Performance of each MDP Module. This table presents the jumpstart performance of the test brokers when only one MDP of the brokers' reasoning component is trained at a time. The overall jumpstart performance of the test brokers is compared to performance of the defined baseline broker.

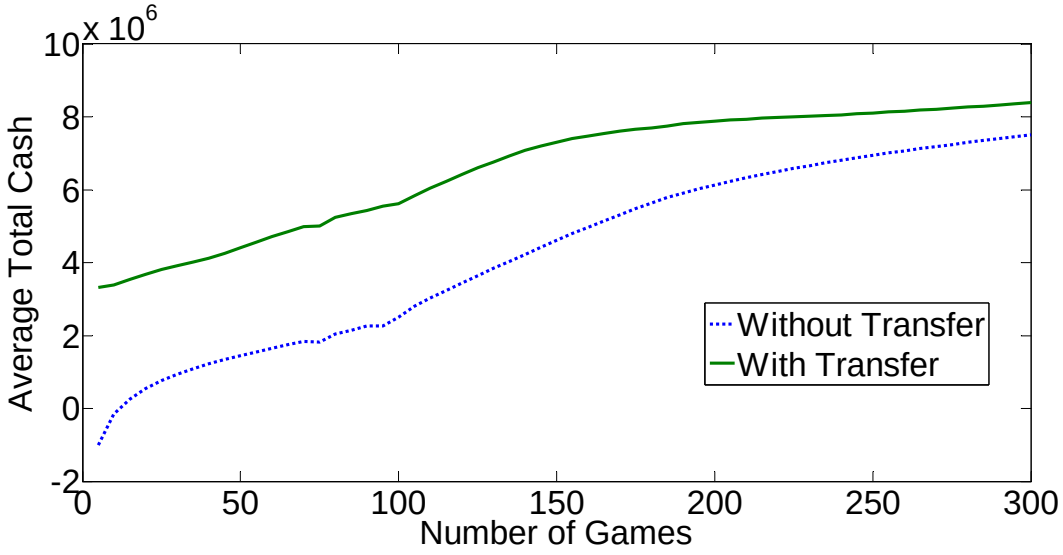



Figure 7.1: Learning Curves for AstonTAC_V5. This figure shows the learning curves of the broker agents in Power TAC. It compares the performance of the agent with and without transfer over 300 games.

**Advantage of the Transfer** Figure 7.1 shows the average total cash gained by the broker over 300 short Power TAC games (1080 time steps). The curve annotated *with transfer* illustrates the performance in Power TAC of a test broker agent when trained in TAC SCM with 100 games and placed in Power TAC for further training. The curve termed *without transfer* shows the performance of the same test broker when trained in Power TAC without transfer. The performance of the broker with knowledge transfer is better to its performance when no transfer is considered, throughout with the difference

in performance being more significant the less games the agent has experienced.

## 7.5 Summary of the Chapter

This chapter proposes an efficient agent-centric knowledge transfer approach, which enables information transfer between MDP tasks with different state spaces, different action spaces, different reward functions and different state transition models. This proposed transfer learning approach enables SMDP traders to reuse the acquired trading skills in new and different markets. Basically, the knowledge transfer technique enables agent designers to reuse (if not the whole reasoning SMDP) relevant MDPs of the reasoning engine, when building new trading agents. The current approach relies on the designer to provide cross-domain mappings between the environment-specific MDPs and the invariant MDPs.

An evaluation and analysis of the formalisation show that AstonTAC_V5 can outperform the top TAC-brokers. Moreover, the results of the evaluation show that this knowledge transfer approach truly enables the agent to transfer trading skills to new markets. The evaluation was based on two challenging and realistic TAC environments with configurable settings. In the future, the proposed AstonTAC_V5 can be tested in other environments, in order to further evaluate the performance of the knowledge transfer approach. Moreover, there is a need for theoretical analysis of possible negative transfers that may occur when selecting source tasks or specifying mapping functions. The next chapter discusses the research directions that can be followed to realise the theoretical analysis in order to provide more insights on predicting and avoiding negative transfers.

# Chapter 8

# Conclusion and Future Perspectives

As automated trading applications are increasingly used in businesses, it is important to develop robust agents that can continually learn and adapt their behaviours to markets. To achieve this, software agent design needs to consider frequent environment changes that may or may not be foreseen at design time and to enable portability of the learned skills in new environments. This implies that similarly to a human trader, the trading agent should have the ability to continually integrate new information in its decision making process, without having to learn the trading skills from scratch and at the same time has the ability to take suitable trading decisions in new markets without need for specific initial training.

To achieve this, this thesis proposes an agent architecture that supports continuous learning and adaptation to market changes and to new markets. Therefore, the focus has been on designing, implementing and evaluating an adaptive, scalable and portable reasoning structure for trading agents using a HRL and SMDPs. These techniques have, in the past, been extensively used to model and solve decision making problems in robotics and have heretofore not been applied to solve trading agents' decision problems. This latter problem type has generally been dealt with in the past using rule-based approaches and only recently using techniques such as flat MDPs and reinforcement learning. Since SMDP and HRL techniques enable modelling and learning of hierarchical and parallel tasks, as well as providing skills transfer when facing new tasks, this thesis demonstrates that these techniques offer the fundamental direction for modelling a new generation of trading agents that will strive for lifelong learning and adaptation. Using these techniques, this thesis puts forward a number of SMDP models and HRL frameworks to address the key decision problems faced by a trading agent when buying and selling in retail and wholesale markets. This concluding chapter summarises

the thesis by consolidating the research contributions in Section 8.1, and suggests key research directions for future work in Section 8.2.

## 8.1 Summary of the Research Contributions

The main contribution of the thesis is the use of SMDP formalisms to design the reasoning engine of trading agents which trade in a multi-agent market environment. In order to illustrate the development of adaptive, scalable and portable SMDP reasoning structure, Chapter 2 proposes an overall architecture of trading agents based on an SMDP formalism and discusses the motivation of the reinforcement learning techniques employed in this work. The new architecture presents a certain number of advantages for the trading agent such as autonomy, reactivity, pro-activeness, adaptation, scalability and portability.

The second main contribution is the design of several (S)MDP-based reasoning engines that have been implemented for AstonTAC when acting in Power TAC environment:

- In Chapter 3, a novel MDP-based framework is put forward to enable an electricity trading agent to minimise its short-term procurement cost and the imbalance between the aggregated retail demand and electricity bought. When buying electricity to cover short-term needs in a smart grid market with a high penetration of renewable electricity sources, the main challenges faced by a retailer are the volatile retail demand, wholesale supply and prices, which are influenced by the behaviour of market participants and the weather state. To address this challenge, the decision problem is modelled as an MDP and uses a Monte Carlo approach to learn the optimal policy. This procurement MDP has been used by AstonTAC to compete in TAC tournaments with great success. An analysis of the tournament results shows that AstonTAC is able to minimise its procurement cost and imbalance by adapting its trading behaviour to wholesale market prices fluctuations for the next 24 hours. This approach works well, if the retailer has a constant number of customers in its portfolio and therefore having a low prediction error of the retail demand. However, if this is not the case, in the considered simulation environment, the retail demand becomes difficult to predict when the number of contracted customers is constantly variable and therefore causing the retailers to sometimes have high electricity excess.

- To remedy the electricity excess problem, this thesis also proposes an SMDP-based framework that extends the previous procurement MDP with the ability to sell electricity surplus in the wholesale market (this contribution is described in Chapter 4). To learn the procurement SMDP, the HAM approach was used, as it enables to incorporate explicitly domain knowledge in the SMDP model in order to speed up the learning process. This new HAM-based procurement approach is evaluated in a series of controlled experiments and is shown to improve the retailer performance in very volatile conditions. Compared to other retailers, AstonTAC had the lowest procurement cost and imbalance level in all market scenarios and did particularly well in market with very volatile retail demand and wholesale prices.

- In Chapter 5, a novel retail SMDP framework is proposed to address the agent's decision making problem optimising simultaneously its market share and profit. The key challenges when developing a retail strategy are the consideration of the diversified behaviours of the market participants (customers and other retail traders) and the trade-off existing between making profit and maintaining a high market share. The behaviours of the market participants are automatically considered in the defined Markov state of the SMDP, whereas the multi-optimisation problem is dealt with by defining a reward function that considers the level of profit and the impact on the number of customers. The learning of the SMDP is achieved with the MAXQ approach and SARSA($\lambda$) is used to learn the individual MDPs of the hierarchy. An evaluation of AstonTAC in the Power TAC tournament 2013 shows that it can maintain a positive profit level and high number of customers in different environment settings.

- In Chapter 6, an SMDP framework is put forward to optimise the whole retailer's decision problem by simultaneously optimising the wholesale and retail strategies of the trading broker. While designing this SMDP, the core challenges were the modelling and learning of concurrent decision processes. The option framework is used to model the trader's SMDP, which can continually be expanded to support optimisation of lower level decision problem. An analysis of the controlled experiments results shows that using the SMDP-based retailer, AstonTAC, performs better that other traders by continually improving its overall profit while optimising simultaneously retail and wholesale strategies.

The third contribution is the modelling of a novel knowledge transfer framework that

enables the trading agent to reuse the acquired skills in new markets without having to learn afresh how to trade. The knowledge transfer approach put forward combines two transfer approaches (inter-task mappings and agent-centric) to define an agent's reasoning that can be reused for decision making in new and different environments. An evaluation of the transfer framework shows that AstonTAC can use the same core reasoning system to act in electricity and personal computer markets.

## 8.2 Future Work

A number of possible research avenues can be followed in order to enhance the work presented in this thesis. This thesis can directly be improved based on the following extensions:

- Analysis of the non-stationary property of the Power TAC. As presented in Chapter 2, Power TAC provides configurable key features to create a dynamic, variable and uncertain trading environment. In order to more deeply assert the ability of the agent to continually learn in a dynamic environment, it is relevant to evaluate the non-stationary property of the environment considered.

- Influence of weather. The weather has a major influence on the behaviour of Power TAC environment. It can be interesting to know how far the cyclic behaviour of the weather can be exploited to improve the optimisation of the agent trading strategy.

- The environment-specific MDPs presented in Chapters 3, 4 and 5 have been defined in an ad-hoc manner. Additional experiments may be required to provide more guidelines on the modelling of the trading MDPs and the tuning of learning parameters ($\alpha$, $\gamma$ or $\lambda$).

In future work, the focus should be on enabling the trading agent to fully adapt and evolve without human intervention by automatically designing the new MDPs needed, specifying the mapping rules ($h_i^j$, the tuple $\langle f_i^j, g_i^j, z_i^j \rangle$ as defined in Chapter 7 ) required in the new environment, while being able to avoid negative transfer. Moreover, the multi-agent learning techniques can be applied to improve the learning of a single agent.

The design of a flat MDP is generally a human task, which consists of identifying the relevant environment variables which will make up the state components as well as identifying the actions required and the reward functions. Although, it is not trivial

to imagine the automation of such design process, many studies have been carried out in order to enable automated discovery of SMDP options. These studies can be extended to enable the discovery of lower level MDP structures (Jonsson and Barto, 2001; Mcgovern, 2002; Hengst, 2002; Mehta et al., 2008b; Busoniu et al., 2008; Luo et al., 2008; Konidaris and Barreto, 2009).

Learning of mapping rules $h_i^j$ is difficult, but this could also be inspired by the same discovery techniques that are suggested for enabling automated design of the MDP. The challenge here is that the real-life market environments are more complex than the environment considered in the presented studies. One of the questions that was not fully answered is how to specify the mapping rules for $h_i^j$ so that a negative transfer can be avoided. Some previous works provide an initial answer to this question, as the function $h_i^j$ is a transformation function, which transforms $M_i^j$ to the abstract $M_c^j$ (see Chapter 7). Given this, existing studies have addressed the formal mapping between MDPs. Ng et al. (1999) investigated the policy invariance under transformation of the reward function. The SMDP homomorphism approach of Ravindran (2003, 2004) is used by Soni and Singh (2006); Rajendran and Huber (2009) to enable transfer of skills. I believe that the ideas of these two approaches can be combined in order to specify a transformation that would always eliminate the negative transfer between $M_i^j$ and $M_c^j$. Furthermore, multi-agent learning can enable agents to have a better understating of the environment so that less successful agents can learn how to trade from more effective agents (Zhang, 2011).

Overall this research is an innovative and promising work that has proposed a design approach to develop adaptive and portable agent reasoning that can be trained in one market and still performs well when deployed in a new and different market.

# List of References


Aberdeen, D. and Buffet, O. (2007). Concurrent probabilistic temporal planning with policy-gradients. In *International Conference on Automated Planning and Scheduling*, pages 10–17.

Agre, P. E. and Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Association for the Advancement of Artificial Intelligence*, volume 87, pages 286–272.

Ailliot, P. and Monbet, V. (2012). Markov-switching autoregressive models for wind time series. *Environmental Modelling & Software*, 30:92–101.

Amato, A., Di Martino, B., Scialdone, M., and Venticinque, S. (2015). Multi-agent negotiation of decentralized energy production in smart micro-grid. In *Intelligent Distributed Computing VIII*, pages 155–160. Springer.

Andre, D. and Russell, S. J. (2001). Programmable reinforcement learning agents. *Advances in Neural Information Processing Systems: Proceedings of the 2000 Conference*, pages 1019–1025.

Arunachalam, R. and Sadeh, N. M. (2005). The supply chain trading agent competition. *Electronic Commerce Research and Applications*, 4(1):66–84.

Asadi, M. and Huber, M. (2007). Effective control knowledge transfer through learning skill and representation hierarchies. In *International Joint Conferences on Artificial Intelligence*, volume 7, pages 2054–2059.

Attias, H. (1999). Inferring parameters and structure of latent variable models by variational bayes. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 21–30. Morgan Kaufmann Publishers Inc.

Bach, T., Yao, J., Wang, J., and Yang, S. (2012). Research and application of the Q-learning for wholesale power markets. In *2nd International Conference on Consumer Electronics, Communications and Networks*, pages 1192–1197. IEEE.

Banerjee, B. and Stone, P. (2007). General game learning using knowledge transfer. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 672–677.

Baral, C. and Gelfond, M. (1997). Reasoning about effects of concurrent actions. *The Journal of Logic Programming*, 31(1):85–117.

Barrett, L. and Narayanan, S. (2008). Learning all optimal policies with multiple criteria. In *Proceedings of the 25th international conference on Machine learning*, pages 41–47. ACM.

Barto, A. and Mahadevan, S. (2003a). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379.

Barto, A. G. and Duff, M. (1994). Monte carlo matrix inversion and reinforcement learning. *Advances in Neural Information Processing Systems*, pages 687–687.

Barto, A. G., Konidaris, G., and Vigorito, C. (2013). Behavioral hierarchy: Exploration and representation. In *Computational and Robotic Models of the Hierarchical Organization of Behavior*, pages 13–46. Springer.

Barto, A. G. and Mahadevan, S. (2003b). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379.

Beal, M. J. (2003). *Variational algorithms for approximate Bayesian inference*. PhD thesis, University of London.

Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, 6:679–684.

Bengio, Y. and Frasconi, P. (1995). An input output HMM architecture. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in neural information processing systems*, volume 7, pages 427–434. MIT Press, Cambridge, MA.

Bengio, Y. and Frasconi, P. (1996). Input-output HMMs for sequence processing. *Neural Networks, IEEE Transactions on*, 7(5):1231–1249.

Benisch, M., Sardinha, A., Andrews, J., Ravichandran, R., and Sadeh, N. (2009). Cmieux: adaptive strategies for competitive supply chain trading. *Electronic Commerce Research and Applications*, 8(2):78–90.

Bernstein, D. S. (1999). Reusing old policies to accelerate learning on new MDPs. Technical report, Tech. rep., University of Massachusetts, Amherst, MA, USA.

Bertoluzzo, F. and Corazza, M. (2012). Testing different reinforcement learning configurations for financial trading: Introduction and applications. *Procedia Economics and Finance*, 3:68–77.

Bertsekas, D. C. and Tsitsiklis, J. N. (1989). *Parallel and Distributed Computation: NumericalMethods*. Prentice-Hall, Englewood Cliffs, New Jersey.

Bertsekas, D. P., Bertsekas, D. P., Bertsekas, D. P., and Bertsekas, D. P. (1995). *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA.

Bertsekas, D. P. and Tsitsiklis, J. N. (1995). Neuro-dynamic programming: an overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1, pages 560–564. IEEE.

Binmore, K. (1990). *Essays on the foundations of game theory*, volume 23. Basil Blackwell Oxford.

Binmore, K. (2007). *Does game theory work? The bargaining challenge*. MIT Press.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer Science and Business Media, LLC.

Bompard, E., Lu, W., Napoli, R., and Jiang, X. (2010). A supply function model for representing the strategic bidding of the producers in constrained electricity markets. *International Journal of Electrical Power & Energy Systems*, 32(6):678–687.

Boonchuay, C. and Ongsakul, W. (2011). Optimal risky bidding strategy for a generating company by self-organising hierarchical particle swarm optimisation. *Energy Conversion and Management*, 52(2):1047–1053.

Boutilier, C. and Brafman, R. I. (2001). Partial-order planning with concurrent interacting actions. *Journal Of Artificial Intelligence Research*, 14:105–136.

Bridle, J. S. (1990). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2*, pages 211–217. Morgan-Kaufmann.

Brooks, R. A. (1986). Achieving artificial intelligence through building robots. Technical report, Technical Report A.I. Memo 899, Massachusetts Institute of Technology Artig=ficial Intelligence Laboartory, Cambridge, MA.

Brooks, R. A. (1995). Intelligence without reason. *The artificial life route to artificial intelligence: Building embodied, situated agents*, pages 25–81.

Buffett, S. and Scott, N. (2004). An algorithm for procurement in supply chain management. *In AAMAS 2004 Workshop on Trading Agent Design and Analysis*.

Busoniu, L., Babuska, R., and De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156–172.

Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D. (2010). *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC press.

Cancelo, J., Espasa, A., and Grafe, R. (2008). Forecasting the electricity load from one day to one week ahead for the spanish system operator. *International Journal of Forecasting*, 24(4):588–602.

Carrión, M., Philpott, A. B., Conejo, A. J., and Arroyo, J. M. (2007). A stochastic programming approach to electric energy procurement for large consumers. *IEEE Trans. Power Syst*, 22(2):744–754.

Cheema, A. (2008). Surcharges and seller reputation. *Journal of Consumer Research*, 35(1):167–177.

Chen, S.-H., Chang, C.-L., and Du, Y.-R. (2012). Agent-based economic models and econometrics. *The Knowledge Engineering Review*, 27(02):187–219.

Cliff, D. (2005). Zip60: Further explorations in the evolutionary design of online auction market mechanisms. *Hewlett-Packard Laboratories Technical Report HPL-2005-85*.

Collins, J., Arunachalam, R., Sadehb, N., Eriksson, J., Finne, N., and Janson, S. (2006). *The Supply Chain Management Game for the 2007 Trading Agent Competition*. ERIM Report Series Reference No. CMU-ISRI-07-100.

Conejo, A., Fernandez-Gonzalez, J., and Alguacil, N. (2005a). Energy procurement for large consumers in electricity markets. In *Generation, Transmission and Distribution, IEE Proceedings-*, volume 152, pages 357–364. IET.

Conejo, A., Plazas, M., Espinola, R., and Molina, A. (2005b). Day-ahead electricity price forecasting using the wavelet transform and ARIMA models. *IEEE Transactions on Power Systems*, 20(2):1035–1042.

Contreras, J., Espinola, R., Nogales, F., and Conejo, A. (2003). ARIMA models to predict next-day electricity prices. *IEEE Transactions on Power Systems*, 18(3):1014–1020.

Corrêa, M. F., Vellasco, M., Figueiredo, K., and Vellasco, P. (2009). Trading strategy in foreign exchange market using reinforcement learning hierarchical neuro-fuzzy systems. In *Advances in Neuro-Information Processing*, pages 461–468. Springer.

Croonenborghs, T., Driessens, K., and Bruynooghe, M. (2008). Learning relational options for inductive transfer in relational reinforcement learning. In *Inductive Logic Programming*, pages 88–97. Springer.

Cuayahuitl, H. (2009). *Hierarchical Reinforcement Learning for Spoken Dialogue Systems*. PhD thesis, University of Edinburgh.

Dai, W., Xue, G.-R., Yang, Q., and Yu, Y. (2007). Transferring naive bayes classifiers for text classification. In *Proceedings 22nd Association for the Advancement of Artificial Intelligence*, volume 22, page 540. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

De Giacomo, G., Lesperance, Y., and Levesque, H. J. (2000). Congolog a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1):109–169.

De Giacomo, G., Lesperance, Y., Levesque, H. J., and Sardina, S. (2009). Indigolog: A high-level programming language for embedded reasoning agents. In *Multi-Agent Programming: Languages, Platforms and Applications*, pages 31–72. Springer.

De Ladurantaye, D., Gendreau, M., and Potvin, J.-Y. (2007). Strategic bidding for price-taker hydroelectricity producers. *Power Systems, IEEE Transactions on*, 22(4):2187–2203.

Dean, T. and Lin, S.-H. (1995). Decomposition techniques for planning in stochastic domains. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1121–1127. Morgan Kaufmann Publishers Inc.

Degris, T., Pilarski, P. M., and Sutton, R. S. (2012). Model-free reinforcement learning with continuous action in practice. In *American Control Conference (ACC), 2012*, pages 2177–2182. IEEE.

Deisenroth, M. P., Neumann, G., Peters, J., et al. (2013). A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142.

Dietterich, T. G. (2000a). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303.

Dietterich, T. G. (2000b). An overview of MAXQ hierarchical reinforcement learning. In *Abstraction, Reformulation, and Approximation*, pages 26–44. Springer.

Dignum, V. and Dignum, F. (2010). Designing agent systems: state of the practice. *International Journal of Agent-Oriented Software Engineering*, 4(3):224–243.

Doshi-Velez, F. (2009). The infinite partially observable markov decision process. In *Advances in neural information processing systems*, pages 477–485.

Dutta, P. K. (1999). *Strategies and games: theory and practice*. MIT press.

Ephraim, Y. and Roberts, W. J. (2005). Revisiting autoregressive hidden markov modeling of speech signals. *IEEE Signal processing letters*, 12(2):166–169.

Esfahanipour, A. and Mousavi, S. (2011). A genetic programming model to generate risk-adjusted technical trading rules in stock markets. *Expert Systems with Applications*, 38(7):8438–8445.

Fasli, M. (2001). Conceptualizing BDI agents for financial markets. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 2, pages 829–834. IEEE.

Fasli, M. (2003). Interrelations between the BDI primitives: Towards heterogeneous agents. *Cognitive Systems Research*, 4(1):1–22.

Ferber, J. (1999). *Multi-agent systems: an introduction to distributed artificial intelligence*. Addison-Wesley Reading.

Ferguson, I. A. (1992). *Touring Machines: An architecture for dynamic, rational, mobile agents*. PhD thesis, University of Cambridge Cambridge.

Ferrante, E., Lazaric, A., and Restelli, M. (2008). Transfer of task representation in reinforcement learning using policy-based proto-value functions. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 3, pages 1329–1332. International Foundation for Autonomous Agents and Multiagent Systems.

Fikes, R. E., Hart, P. E., and Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial intelligence*, 3:251–288.

Fleten, S.-E. and Pettersen, E. (2005). Constructing bidding curves for a price-taking retailer in the norwegian electricity market. *Power Systems, IEEE Transactions on*, 20(2):701–708.

Frommberger, L. and Wolter, D. (2010). Structural knowledge transfer by spatial abstraction for reinforcement learning agents. *Adaptive Behavior*, 18(6):507–525.

Fudenberg, D. and Tirole, J. (1991). Game theory. *MIT Press Books*, 1.

Gábor, Z., Kalmár, Z., and Szepesvári, C. (1998). Multi-criteria reinforcement learning. In *ICML*, volume 98, pages 197–205.

Gajjar, G., Khaparde, S., Nagaraju, P., and Soman, S. (2003). Application of actor-critic learning algorithm for optimal bidding problem of a Genco. *Power Systems, IEEE Transactions on*, 18(1):11–18.

Gao, F., Guan, X., Cao, X., and Papalexopoulos, A. (2000). Forecasting power market clearing price and quantity using a neural network method. In *Power Engineering Society Summer Meeting*, volume 4, pages 2183–2188. IEEE.

Gao, F. and Sheble, G. (2010). Electricity market equilibrium model with resource constraint and transmission congestion. *Electric Power Systems Research*, 80(1):9–18.

Garcia, R., Contreras, J., Van Akkeren, M., and Garcia, J. (2005). A GARCH forecasting model to predict day-ahead electricity prices. *IEEE Transactions on Power Systems*, 20(2):867–874.

Geanakoplos, J., Axtell, R., Farmer, D. J., Howitt, P., Conlee, B., Goldstein, J., Hendrey, M., Palmer, N. M., and Yang, C.-Y. (2012). Getting at systemic risk via an agent-based model of the housing market. *The American Economic Review*, 102(3):53–58.

Gelfond, M. and Lifschitz, V. (1992). Representing actions in extended logic programming. In *JICSLP*, volume 92, page 560.

Girardi, R. and Leite, A. (2013). A survey on software agent architectures. *IEEE Intelligent Informatics Bulletin*, 14(1):8–20.

Giunchiglia, F. and Walsh, T. (1992). A theory of abstraction. *Artificial Intelligence*, 57(2-3):323–390.

Givan, R., Dean, T., and Greig, M. (2003). *Equivalence notions and model minimization in Markov decision processes*, volume 147.

Gmytrasiewicz, P. J. and Durfee, E. H. (2000). Rational coordination in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, 3(4):319–350.

Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica: Journal of the Econometric Society*, pages 357–384.

Han, W., Liu, L., and Zheng, H. (2008). Dynamic pricing by multiagent reinforcement learning. In *International Symposium on Electronic Commerce and Security*, pages 226–229. IEEE.

Harvey, A. and Koopman, S. (1993). Forecasting hourly electricity demand using time-varying splines. *Journal of the American Statistical Association*, 88(424):1228–1236.

Hassan, G. N. A. (2010). *Multiobjective genetic programming for financial portfolio management in dynamic environments*. PhD thesis, University College London.

He, M., Leung, H.-f., and Jennings, N. R. (2003). A fuzzy-logic based bidding strategy for autonomous agents in continuous double auctions. *Knowledge and Data Engineering, IEEE Transactions on*, 15(6):1345–1363.

He, M., Rogers, A., Luo, X., and Jennings, N. (2006). Designing a successful trading agent for supply chain management. In *Proceedings of the fifth international Joint Conference on Autonomous agents and multiagent systems*, pages 1159–1166. ACM.

Heij, C., De Boer, P., Franses, P. H., Kloek, T., Van Dijk, H. K., et al. (2004). *Econometric methods with applications in business and economics*. Oxford University Press.

Hengst, B. (2002). Discovering hierarchy in reinforcement learning with HEXQ. In *International Conference on Machine Learning*, volume 2, pages 243–250.

Herranz, R., Munoz San Roque, A., Villar, J., and Campos, F. A. (2012). Optimal demand-side bidding strategies in electricity spot markets. *Power Systems, IEEE Transactions on*, 27(3):1204–1213.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Hommes, C. H. (2006). Heterogeneous agent models in economics and finance. *Handbook of computational economics*, 2:1109–1186.

Howard, R. A. (1971). *Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes*. John Wiley and Sons, Incorporated.

Huber, M. and Grupen, R. A. (1997). A feedback control structure for on-line learning tasks. *Robotics and autonomous systems*, 22(3):303–315.

Hunter, I. W., Hollerbach, J. M., and Ballantyne, J. (1991). A comparative analysis of actuator technologies for robotics. *Robotics Review*, 2.

Iba, G. A. (1989). A heuristic approach to the discovery of macro-operators. *Machine Learning*, 3(4):285–317.

Irodova, M. and Sloan, R. H. (2005). Reinforcement learning and function approximation. In *Florida Artificial Intelligence Research Society Conference*, pages 455–460.

Jewell, W. S. (1963). Markov renewal programming i: Formulation, finite return models. *Operations Research*, 11(6):938–948.

Jong, N. K. and Stone, P. (2007). Model-based function approximation in reinforcement learning. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, page 95. ACM.

Jonsson, A. and Barto, A. G. (2001). Automated state abstraction for options using the u-tree algorithm. *Advances in neural information processing systems*, pages 1054–1060.

Jónsson, T., Pinson, P., and Madsen, H. (2010). On the market impact of wind energy forecasts. *Energy Economics*, 32(2):313–320.

Juba, B. (2013). Implicit learning of common sense for reasoning. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 939–946. AAAI Press.

Kaelbling, L., Littman, M., and Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134.

Kaelbling, L. P. and Rosenschein, S. J. (1990). Action and planning in embedded agents. *Robotics and autonomous systems*, 6(1):35–48.

Karlsson, J. (1997). *Learning to solve multiple goals*. PhD thesis, University of Rochester.

Kazakov, D. and Kudenko, D. (2001). Machine learning and inductive logic programming for multi-agent systems. In Luck, M., Marik, V., Stepankova, O., and Trappl, R., editors, *Multi-Agent Systems and Applications*, volume 2086 of *Lecture Notes in Computer Science*, pages 246–270. Springer Berlin Heidelberg.

Ketter, W., Collins, J., Gini, M., Gupta, A., and Schrater, P. (2006). Identifying and forecasting economic regimes in TAC SCM. In *Agent-Mediated Electronic Commerce. Designing Trading Agents and Mechanisms*, pages 113–125. Springer.

Ketter, W., Collins, J., Reddy, P., Flath, C., and Weerdt, M. (2015). *The 2015 Power Trading Agent Competition*. ERIM Report Series Reference No. ERS-2015-001-LIS.

Ketter, W., Peters, M., and Collins, J. (2013). Autonomous agents in future energy markets: The 2012 power trading agent competition. In *Association for the Advancement of Artificial Intelligence Conference, Bellevue*.

Khatib, O. and Burdick, J. (1986). Motion and force control of robot manipulators. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1381–1386. IEEE.

Kiekintveld, C., Miller, J., Jordan, P. R., Callender, L. F., and Wellman, M. P. (2009). Forecasting market prices in a supply chain game. *Electronic Commerce Research and Applications*, 8(2):63–77.

Kiekintveld, C., Wellman, M. P., Singh, S. P., Estelle, J., Vorobeychik, Y., Soni, V., and Rudary, M. R. (2004). Distributed feedback control for decision making on supply chains. In *International Conference on Automated Planning and Scheduling*, pages 384–392.

Kiekintveld, C. D. (2008). *Empirical Game-Theoretic Methods for Strategy Design and Analysis in Complex Games*. PhD thesis, University of Michigan.

Kirschen, D. S. (2003). Demand-side view of electricity markets. *Power Systems, IEEE Transactions on*, 18(2):520–527.

Knoblock, C. A. (1994). *Generating parallel execution plans with a partial order planner*.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*.

Kober, J. and Peters, J. (2012). Reinforcement learning in robotics: A survey. In *Reinforcement Learning*, pages 579–610. Springer.

Kober, J. and Peters, J. R. (2009). Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, pages 849–856.

Konidaris, G. and Barreto, A. S. (2009). Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems*, pages 1015–1023.

Konidaris, G. and Barto, A. G. (2007). Building portable options: Skill transfer in reinforcement learning. In *International Joint Conferences on Artificial Intelligence*, volume 7, pages 895–900.

Konidaris, G., Scheidwasser, I., and Barto, A. (2012). Transfer in reinforcement learning via shared features. *Journal of Machine Learning Research*, 13:1333–1371.

Korf, R. E. (1985). Macro-operators: A weak method for learning. *Artificial intelligence*, 26(1):35–77.

Kormushev, P., Calinon, S., and Caldwell, D. G. (2010). Robot motor skill coordination with em-based reinforcement learning. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3232–3237. IEEE.

Kristiansen, T. (2014). A time series spot price forecast model for the Nord Pool market. *International Journal of Electrical Power & Energy Systems*, 61:20–26.

Kuate, R. T., Chli, M., and Wang, H. H. (2014). Optimising market share and profit margin: SMDP-based tariff pricing under the smart grid paradigm. In *Innovative Smart Grid Technologies Europe, 2014 5th IEEE/PES*. IEEE.

Kuate, R. T., Chli, M., and Wang, H. H. (2015). An efficient knowledge transfer solution to a novel SMDP formalization of a broker decision problem. In *International Conference on Autonomous Agents and Multiagent Systems*.

Kuate, R. T., He, M., Chli, M., and Wang, H. H. (2013). An intelligent broker agent for energy trading: an MDP approach. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 234–240. AAAI Press.

Lambrecht, A., Seim, K., Vilcassim, N., Cheema, A., Chen, Y., Crawford, G. S., Hosanagar, K., Iyengar, R., Koenigsberg, O., Lee, R., et al. (2012). Price discrimination in service industries. *Marketing Letters*, 23(2):423–438.

Lazaric, A. (2008). *Knowledge Transfer in Reinforcement Learning*. PhD thesis, Politecnico di Milano.

Lazaric, A. (2012). Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer.

Lazaric, A., Ghavamzadeh, M., et al. (2010). Bayesian multi-task reinforcement learning. In *27th International Conference on Machine Learning*, pages 599–606.

Lazaric, A., Restelli, M., and Bonarini, A. (2008). Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 544–551. ACM.

LeBaron, B. (2001). A builder guide to agent-based financial markets. *Quantitative Finance*, 1(2):254–261.

LeBaron, B. (2002). Building the santa fe artificial stock market. *Physica A: Statistical Mechanics and its Applications*.

LeBaron, B. (2006). Agent-based computational finance. *Handbook of computational economics*, 2:1187–1233.

Li, L. and Littman, M. (2005). Lazy approximation for solving continuous finite-horizon MDPs. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1175. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press.

Li, L., Walsh, T. J., and Littman, M. L. (2006). Towards a unified theory of state abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM 2006)*.

Liefers, B., Hoogland, J., and La Poutré, H. (2014). A successful broker agent for power tac. In *Proceedings of the 2014 Workshop on Agent-Mediated Electronic Commerce Trading Agent Design and Analysis (AMEC/TADA 2014)*.

Lin, L.-J. and Mitchell, T. M. (1993). Reinforcement learning with hidden states. In *Proceedings of the second International Conference on From Animals to Animats*, volume 2, pages 271–280.

Lin, S.-H. (1997). *Exploiting Structure for Planning and Control*. PhD thesis, Brown University.

Little, I. and Thiebaux, S. (2006). Concurrent probabilistic planning in the graphplan framework. In *International Conference on Automated Planning and Scheduling*, pages 263–273.

Littman, M. (1996). *Algorithms for sequential decision making*. PhD thesis, Brown University.

Littman, M. L. (1994). Memoryless policies: Theoretical limitations and practical results. In *From Animals to Animats 3: Proceedings of the third International Conference on Simulation of Adaptive Behavior*, volume 3, page 238. MIT Press.

Liu, C., Xu, X., and Hu, D. (2015). Multiobjective reinforcement learning: A comprehensive overview. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, 45(3):385–398.

Liu, H. and Shi, J. (2013). Applying ARMA-GARCH approaches to forecasting short-term electricity prices. *Energy Economics*, 37:152–166.

Liu, M., Liao, X., and Carin, L. (2011). The infinite regionalized policy representation. In *Proceedings of the 28th International Conference on Machine Learning*, pages 769–776.

Lizotte, D. J., Bowling, M., and Murphy, S. A. (2012). Linear fitted-q iteration with multiple reward functions. *The Journal of Machine Learning Research*, 13(1):3253–3295.

Loch, J. and Singh, S. P. (1998). Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. In *International Conference on Machine Learning*, pages 323–331.

Lohpetch, D. and Corne, D. (2009). Discovering effective technical trading rules with genetic programming: Towards robustly outperforming buy-and-hold. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 439–444. IEEE.

Lohpetch, D. et al. (2011). *Evolutionary algorithms for financial trading*. PhD thesis, Heriot-Watt University.

Luo, Z., Bell, D. A., and McCollum, B. (2008). Discover relevant environment feature using concurrent reinforcement learning. In *Association for the Advancement of Artificial Intelligence*, pages 1816–1817.

MacKie-Mason, J. K. and Wellman, M. P. (2006). Automated markets and trading agents. *Handbook of computational economics*, 2:1381–1431.

Maes, P. (1989). The dynamics of action selection. In *In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 991–997.

Mahadevan, S. and Maggioni, M. (2007). Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(2169-2231):16.

Mahvi, M. and Ardehali, M. (2011). Optimal bidding strategy in a competitive electricity market based on agent-based approach and numerical sensitivity analysis. *Energy*, 36(11):6367–6374.

Mandal, P., Senjyu, T., Uezato, K., and Funabashi, T. (2005). Several-hours-ahead electricity price and load forecasting using neural networks. In *Power Engineering Society General Meeting*, volume 3, pages 2146–2153. IEEE.

Mannor, S. and Shimkin, N. (2004). A geometric approach to multi-criterion reinforcement learning. *The Journal of Machine Learning Research*, 5:325–360.

Marx, M. L. and Larsen, R. J. (2006). *Introduction to mathematical statistics and its applications*. Pearson/Prentice Hall.

Mausam and Weld, D. S. (2005). Concurrent probabilistic temporal planning. In *In Proceedings International Conference on Automated Planning and Scheduling*.

Mausam and Weld, D. S. (2008). Planning with durative actions in stochastic domains. *Journal of Artificial Intelligence Research*, 31:33–82.

McCallum, A. (1993). Overcoming incomplete perception with util distinction memory. In *International Conference on Machine Learning*, volume 93, pages 190–196.

McCallum, R. A. (1995). Instance-based utile distinctions for reinforcement learning with hidden state. In *International Conference on Machine Learning*, pages 387–395.

McGovern, A., Sutton, R. S., and Fagg, A. H. (1997). Roles of macro-actions in accelerating reinforcement learning. In *Grace Hopper celebration of women in computing*, volume 1317.

Mcgovern, E. A. (2002). *Autonomous discovery of temporal abstractions from interaction with an environment*. PhD thesis, University of Massachusetts Amherst.

Mehta, N., Natarajan, S., Tadepalli, P., and Fern, A. (2008a). Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning*, 73(3):289–312.

Mehta, N., Ray, S., Tadepalli, P., and Dietterich, T. (2008b). Automatic discovery and transfer of MAXQ hierarchies. In *Proceedings of the 25th International Conference on Machine Learning*, pages 648–655. ACM.

Melo, F. S., Meyn, S. P., and Ribeiro, M. I. (2008). An analysis of reinforcement learning with function approximation. In *Proceedings of the 25th International Conference on Machine Learning*, pages 664–671. ACM.

Meuleau, N., Peshkin, L., Kim, K.-E., and Kaelbling, L. P. (1999). Learning finite-state controllers for partially observable environments. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 427–436. Morgan Kaufmann Publishers Inc.

Moody, J. and Saffell, M. (2001). Learning to trade via direct reinforcement. *Neural Networks, IEEE Transactions on*, 12(4):875–889.

Morales, J. M., Conejo, A. J., and Pérez-Ruiz, J. (2010). Short-term trading for a wind power producer. *Power Systems, IEEE Transactions on*, 25(1):554–564.

Moriyama, K., Matsumoto, M., Fukui, K.-i., Kurihara, S., and Numao, M. (2008). Reinforcement learning on a futures market simulator. *Journal of Universal Computer Science*, 14(7):1136–1153.

Muggleton, S. and Lin, D. (2013). Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 1551–1557. AAAI Press.

Müller, J. P. and Fischer, K. (2014). Application impact of multi-agent systems and technologies: A survey. *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks*, pages 27–53.

Natarajan, S. and Tadepalli, P. (2005). Dynamic preferences in multi-criteria reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 601–608. ACM.

Nevmyvaka, Y., Feng, Y., and Kearns, M. (2006). Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 673–680. ACM.

Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, pages 278–287.

Ng, A. Y. and Jordan, M. (2000). Pegasus: A policy search method for large mdps and pomdps. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 406–415. Morgan Kaufmann Publishers Inc.

Nguyen, H. and Nabney, I. (2010). Short-term electricity demand and gas price forecasts using wavelet transforms and adaptive models. *Energy*, 35(9):3674–3685.

Palmer, R., Brian Arthur, W., Holland, J. H., LeBaron, B., and Tayler, P. (1994). Artificial economic life: a simple model of a stockmarket. *Physica D: Nonlinear Phenomena*, 75(1):264–274.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359.

Pardoe, D. (2011). *Adaptive trading agent strategies using market experience*. PhD thesis, University of Texas at Austin.

Pardoe, D. and Stone, P. (2006). Predictive planning for supply chain management. In *International Conference on Automated Planning and Scheduling*, pages 21–30.

Parlange, M. B. and Katz, R. W. (2000). An extended version of the richardson model for simulating daily weather variables. *Journal of Applied Meteorology*, 39(5):610–622.

Parr, R. and Russell, S. (1998). Reinforcement learning with hierarchies of machines. In *Advances in neural information processing systems*, volume 10, pages 1043–1049.

Parr, R. E. (1998). *Hierarchical control and learning for Markov decision processes*. PhD thesis, University of California at Berkeley.

Peshkin, L. (2002). *Reinforcement learning by policy search*. PhD thesis, Brown University.

Peters, J. and Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697.

Peters, M., Ketter, W., Saar-Tsechansky, M., and Collins, J. (2013). A reinforcement learning approach to autonomous decision-making in smart electricity markets. *Machine Learning*, pages 1–35.

Petrik, M., Taylor, G., Parr, R., and Zilberstein, S. (2010). Feature selection using regularization in approximate linear programs for markov decision processes. *International Conference on Machine Learning*.

Philpott, A. B. and Pettersen, E. (2006). Optimizing demand-side bids in day-ahead electricity markets. *Power Systems, IEEE Transactions on*, 21(2):488–498.

Pinto, J. A. (1994). *Temporal reasoning in the situation calculus*. PhD thesis, University of Toronto.

Power TAC Community (2013). Overview for tournament: finals-2012-12. http://wolf-08.fbk.eur.nl:8080/TournamentScheduler/faces/tourna-ment.xhtml?tournamentId=9. [Online; accessed 28-January-2013].

Precup, D. (2000). *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts - Amherst.

Raina, R., Ng, A. Y., and Koller, D. (2006). Constructing informative priors using transfer learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 713–720. ACM.

Rajendran, S. and Huber, M. (2009). Learning to generalize and reuse skills using approximate partial policy homomorphisms. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 2239–2244. IEEE.

Raju, C., Narahari, Y., and Ravikumar, K. (2006). Learning dynamic prices in electronic retail markets with customer segmentation. *Annals of Operations Research*, 143(1):59–75.

Ramanathan, R., Engle, R., Granger, C., Vahid-Araghi, F., and Brace, C. (1997). Short-run forecasts of electricity loads and peaks. *International Journal of Forecasting*, 13(2):161–174.

Rautiainen, T., Lunden, J., Werner, S., and Koivunen, V. (2013). Demand side management through electricity pricing in competitive environments. In *Innovative Smart Grid Technologies Europe, 2013 4th IEEE/PES*, pages 1–5. IEEE.

Ravindran, B. (2003). SMDP homomorphisms: An algebraic approach to abstraction in semi markov decision processes. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*.

Ravindran, B. (2004). *An algebraic approach to abstraction in reinforcement learning*. PhD thesis, University of Massachusetts Amherst.

Ravindran, B. and Barto, A. G. (2003). Relativized options: Choosing the right transformation. In *International Conference on Machine Learning*, pages 608–615.

Raykar, V. C., Krishnapuram, B., Bi, J., Dundar, M., and Rao, R. B. (2008). Bayesian multiple instance learning: automatic feature selection and inductive transfer. In *Proceedings of the 25th International Conference on Machine Learning*, pages 808–815. ACM.

Reddy, P. and Veloso, M. (2011a). Learned behaviors of multiple autonomous agents in smart grid markets. In *Association for the Advancement of Artificial Intelligence*.

Reddy, P. and Veloso, M. (2011b). Strategy learning for autonomous agents in smart grid markets. In *International Joint Conferences on Artificial Intelligence*.

Reddy, P. P. (2013). *Semi-Cooperative Learning in Smart Grid Agents*. PhD thesis, Carnegie Mellon University.

Reiter, R. (1996). Natural actions, concurrency and continuous time in the situation calculus. *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference*, 96:2–13.

Richiardi, M. G. (2012). Agent-based computational economics: a short introduction. *The Knowledge Engineering Review*, 27(02):137–149.

Rogers, A., Ramchurn, S., and Jennings, N. (2012). Delivering the smart grid: Challenges for autonomous agents and multi-agent systems research. In *Association for the Advancement of Artificial Intelligence*.

Rohanimanesh, K. (2006). *Concurrent Decision Making in Markov Decision Processes*. PhD thesis, University of Massachusetts - Amherst.

Rohanimanesh, K. and Mahadevan, S. (2001). Decision-theoretic planning with concurrent temporally extended actions. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 472–479. Morgan Kaufmann Publishers Inc.

Rohanimanesh, K. and Mahadevan, S. (2002). Learning to take concurrent actions. In *Advances in neural information processing systems*, pages 1619–1626.

Rohanimanesh, K. and Mahadevan, S. (2005). Coarticulation: an approach for generating concurrent plans in markov decision processes. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 720–727. ACM.

Roijers, D. M., Vamplew, P., Whiteson, S., and Dazeley, R. (2013). A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*.

Ross, S. M. (1983). *Introduction to stochastic dynamic programming: Probability and mathematical*. Academic Press, Inc.

Rummery, G. and Niranjan, M. (1994). *On-line Q-learning using connectionist systems*. Technical report no.166, University of Cambridge, Department of Engineering.

Rummery, G. and Niranjan, M. (2014). *Renewable 2014 - Global Status Report*. Tech. rep., Renewable Energy Policy Network for the 21st Century.

Russell, S. J. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach (3rd Edition)*. Prentice-Hall.

Samanidou, E., Zschischang, E., Stauffer, D., and Lux, T. (2007). Agent-based models of financial markets. *Reports on Progress in Physics*, 70(3):409.

Sharma, M., Holmes, M. P., Santamaría, J. C., Irani, A., Isbell Jr, C. L., and Ram, A. (2007). Transfer learning in real-time strategy games using hybrid cbr/rl. In *International Joint Conferences on Artificial Intelligence*, volume 7, pages 1041–1046.

Sherstov, A. A. and Stone, P. (2005). Improving action selection in MDP's via knowledge transfer. In *Association for the Advancement of Artificial Intelligence*, volume 5, pages 1024–1029.

Shoham, Y. and Leyton-Brown, K. (2008). *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.

Singh, D. (2011). *Learning Plan Selection for BDI Agent Systems*. PhD thesis, RMIT University.

Singh, S. (1993). *Learning to solve Markovian decision processes*. PhD thesis, University of Massachusetts.

Singh, S., Cohn, D., et al. (1998). How to dynamically merge markov decision processes. *Advances in neural information processing systems*, (10):1057–1063.

Singh, S. P. (1992a). Reinforcement learning with a hierarchy of abstract models. In *Association for the Advancement of Artificial Intelligence*, pages 202–207.

Singh, S. P. (1992b). Scaling reinforcement learning algorithms by learning variable temporal resolution models. In *Proceedings of the Ninth International Machine Learning Conference*, pages 406–415. Morgan Kaufmann.

Singh, S. P., Jaakkola, T., and Jordan, M. I. (1994a). Learning without state-estimation in partially observable markovian decision processes. In *International Conference on Machine Learning*, pages 284–292.

Singh, S. P., Jaakkola, T., and Jordan, M. I. (1994b). Model-free reinforcement learning for non-markovian decision problems. In *Proceedings of the Eleventh International Conference on Machine Learning*, page 284292.

Singh, S. P. and Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1-3):123–158.

Smallwood, R. and Sondik, E. (1973). The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088.

Snel, M. and Whiteson, S. (2014). Learning potential functions and their representations for multi-task reinforcement learning. *Autonomous agents and multi-agent systems*, 28(4):637–681.

Sniezynski, B. (2014). Agent-based adaptation system for service-oriented architectures using supervised learning. *Procedia Computer Science*, 29(0):1057 – 1067. 2014 International Conference on Computational Science.

Sondik, E. J. (1971). *The optimal control of partially observable Markov processes*. PhD thesis, Standford University.

Song, H., Liu, C.-C., Lawarrée, J., and Dahlgren, R. (2000). Optimal electricity supply bidding by Markov Decision Process. *IEEE Trans. Power Systems*, 15(2):618–624.

Soni, V. and Singh, S. (2006). Using homomorphisms to transfer options across continuous reinforcement learning domains. In *Association for the Advancement of Artificial Intelligence*, volume 6, pages 494–499.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning, an Introduction*. MIT press, Cambridge, MA.

Sutton, R., Precup, D., and Singh, S. (1999a). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112:181–211.

Sutton, R. S. (1995). Temporal abstraction in reinforcement learning. In *In Proceedings of the Twelfth Int. Conference on Machine Learning*. Morgan Kaufmann.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, pages 1038–1044.

Sutton, R. S., McAllester, D. A., Singh, S. P., Mansour, Y., et al. (1999b). Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063. Neural Information Processing Systems.

Tamar, A., Mannor, S., and Xu, H. (2014). Scaling up robust MDPs using function approximation. In *Proceedings of the 31st International Conference on Machine Learning*, pages 181–189.

Tan, A.-H., Ong, Y.-S., and Tapanuj, A. (2011). A hybrid agent architecture integrating desire, intention and reinforcement learning. *Expert Systems with Applications*, 38(7):8477 – 8487.

Taylor, M. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, 10:1633–1685.

Taylor, M. E. and Stone, P. (2007). Cross-domain transfer for reinforcement learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 879–886. ACM.

Taylor, M. E., Stone, P., and Liu, Y. (2007). Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(1):2125–2167.

Tellidou, A. and Bakirtzis, A. (2006). Multi-agent reinforcement learning for strategic bidding in power markets. In *3rd International Conference on Intelligent Systems*, pages 408–413. IEEE.

Tesauro, G. and Bredin, J. L. (2002). Strategic sequential bidding in auctions using dynamic programming. In *Proceedings of the first International Joint Conference on Autonomous Agents and Multiagent Systems: part 2*, pages 591–598. ACM.

Tesauro, G. and Kephart, J. O. (2002). Pricing in agent economies using multi-agent q-learning. *Autonomous Agents and Multi-Agent Systems*, 5(3):289–304.

Tesfatsion, L. (2002). Agent-based computational economics: Growing economies from the bottom up. *Artificial life*, 8(1):55–82.

Thrun, S. and Schwartz, A. (1993). Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*.

Thrun, S. and Schwartz, A. (1995). Finding structure in reinforcement learning. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, Cambridge, MA. MIT Press.

Torrey, L., Shavlik, J., Walker, T., and Maclin, R. (2006). Skill acquisition via transfer learning and advice taking. In *Proceedings of the Sixteenth European Conference on Machine Learning*, pages 425–436.

Torrey, L., Walker, T., Shavlik, J., and Maclin, R. (2005). Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Proceedings of the Sixteenth European Conference on Machine Learning*, pages 412–424. Springer.

Toussaint, M. and Storkey, A. (2006). Probabilistic inference for solving discrete and continuous state markov decision processes. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 945–952. ACM.

Urieli, D. and Stone, P. (2014). Tactex'13: a champion adaptive power trading agent. In *Association for the Advancement of Artificial Intelligence Conference*, pages 465–471.

Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., and Dekker, E. (2011). Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84(1-2):51–80.

Van Hasselt, H. (2012). Reinforcement learning in continuous state and action spaces. In *Reinforcement Learning*, pages 207–251. Springer.

Van Moffaert, K., Drugan, M. M., and Nowé, A. (2013). Hypervolume-based multi-objective reinforcement learning. In *Evolutionary Multi-Criterion Optimization*, pages 352–366. Springer.

Vidal, J. M. and Durfee, E. H. (2003). Predicting the expected behavior of agents that learn about agents: the clri framework. *Autonomous Agents and Multi-Agent Systems*, 6(1):77–107.

Vytelingum, P., Cliff, D., and Jennings, N. R. (2008). Strategic bidding in continuous double auctions. *Artificial Intelligence*, 172(14):1700–1729.

Wang, W. (2014). *Multi-objective sequential decision making*. PhD thesis, Université Paris Sud-Paris XI.

Wangenheim, F. V. and Bayón, T. (2004). The effect of word of mouth on services switching: measurement and moderating variables. *European Journal of Marketing*, 38(9/10):1173–1185.

Waters, D. and Waters, C. D. J. (2008). *Quantitative methods for business*. Pearson Education.

Watkins, C. (1989a). *Learning from Delayed Rewards*. PhD thesis, Cambridge University.

Watkins, C. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Watkins, C. J. C. H. (1989b). *Learning from delayed rewards*. PhD thesis, University of Cambridge.

Wellman, M. P. (2011). Trading agents. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(3):1–107.

Wellman, M. P., Estelle, J., Singh, S., Vorobeychik, Y., Kiekintveld, C., and Soni, V. (2005). Strategic interactions in a supply chain game. *Computational Intelligence*, 21(1):1–26.

Whittle, P. (1982). *Optimization over time*. John Wiley & Sons, Inc.

Wilson, A., Fern, A., Ray, S., and Tadepalli, P. (2007). Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 1015–1022. ACM.

Wilson, G. and Banzhaf, W. (2010). Interday foreign exchange trading using linear genetic programming. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 1139–1146. ACM.

Witten, I. H. (1977). An adaptive optimal controller for discrete-time Markov environments. *Information and control*, 34(4):286–295.

Wooldridge, J. M. (2010). *Econometric analysis of cross section and panel data*. MIT press.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. John Wiley & Sons.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(02):115–152.

Xu, X., Zuo, L., and Huang, Z. (2014). Reinforcement learning algorithms with function approximation: Recent advances and applications. *Information Sciences*, 261:1–31.

Yao, S., Song, Y., Zhang, L., and Cheng, X. (2000). Wavelet transform and neural networks for short-term electrical load forecasting. *Energy Conversion and Management*, 41(18):1975–1988.

Younes, H. L. and Simmons, R. G. (2004). Solving generalized semi-markov decision processes using continuous phase-type distributions. In *Association for the Advancement of Artificial Intelligence*, volume 4, page 742.

Yucekaya, A. D., Valenzuela, J., and Dozier, G. (2009). Strategic bidding in electricity markets using particle swarm optimization. *Electric Power Systems Research*, 79(2):335–345.

Zare, K., Moghaddam, M. P., and Sheikh El Eslami, M. K. (2010). Electricity procurement for large consumers based on information gap decision theory. *Energy Policy*, 38(1):234–242.

Zare, K., Moghaddam, M. P., and Sheikh-El-Eslami, M. K. (2011). Risk-based electricity procurement for large consumers. *Power Systems, IEEE Transactions on*, 26(4):1826–1835.

Zhang, C. (2011). *Scaling multi-agent learning in complex environments*. PhD thesis, University of Massachusetts Amherst.

Zhang, L. and Luh, P. (2005). Neural network-based market clearing price prediction and confidence interval estimation with an improved extended Kalman filter method. *IEEE Transactions on Power Systems*, 20(1):59–66.

Zheng, H., Xie, L., and Zhang, L.-Z. (2005). Electricity price forecasting based on GARCH model in deregulated market. In *The 7th International Power Engineering Conference*, pages 1–410. IEEE.

Zinn, M., Roth, B., Khatib, O., and Salisbury, J. K. (2004). A new actuation approach for human friendly robot design. *The International Journal of Robotics Research*, 23(4-5):379–398.

Zugno, M., Morales, J. M., Pinson, P., and Madsen, H. (2013). Pool strategy of a price-maker wind power producer. *Power Systems, IEEE Transactions on*, 28(3):3440–3450.

Zwiers, F. and Von Storch, H. (1990). Regime-dependent autoregressive time series modeling of the southern oscillation. *Journal of climate*, 3(12):1347–1363.

# Appendix A

# AstonTAC Sensors

The focus of this section is on the description of the sensors that AstonTAC used in Power TAC, as innovative approaches have been put forward. The sensors used by AstonTAC in TAC SCM are similar to the sensors described in He et al. (2006).

## A.1 Sensing the Markets

To act effectively in the Power TAC environment, environment-specific sensors are developed to support agent reasoning. To make decisions in the retail and in the wholesale market, the broker uses the prediction of future prices, demand and supply as inputs. In order to purchase enough energy and satisfy the contracted customers every simulated hour, the broker agent needs to forecast the hourly energy demand and production of the contracted customers. Similarly, to buy the energy at very low price, it needs to forecast the energy price in the wholesale market.

When designing forecasting models in Power TAC, environment-specific challenges need to be considered.

- *Historical data may not be available before the game start*s: As described in Section 2.1.3.1, for the Power TAC each game is initialised with a different seed so that the environment settings such as the number of customers, their preferences and their energy consumption vary from game to game. The broker receives the initial environment data information at the beginning of game, which implies that the broker should be able to generate its prediction models at the beginning of each game.

- *Variable retail energy demand and production*: Each customer in the environment exhibits a particular consumption or production pattern, so that it is difficult to generalise their behaviour in a model with the same model parameters. To address this, a specific model is used to predict the behaviour of each customer. The resulting retail demand is an aggregation of the demand of contracted customers.

- *Behaviour of the broker*s: Depending on the strategy of other competitors and on the game setting, the energy prices in wholesale market may be very variable.

- *Influence of the weather*: Real weather data are used to influence the energy demand, supply and the wholesale market price. The weather is also one of the elements that can increase prediction errors of the forecasting models. Without an accurate prediction of the weather, the energy demand may be surprisingly very high or low. Weather prediction information is offered by the Power TAC game server.

- *Real time acting and reacting:* the energy broker has only 5 seconds to take the hourly decisions in retail and wholesale market.

In Power TAC, environment-specific forecasting models need to be adaptive as the environment is variable and historical data may not be available. The prediction models developed in this thesis work well by learning online and adapting to the environment changes in order to improve its accuracy.

## A.2 Forecasting with Non-Homogeneous Hidden Markov Model

### A.2.1 HMMs for Electricity Market Prediction

The idea is to improve the prediction of target variables by predicting their latent states first, then using state-specific models to forecast their values. This section will introduce the Non Homogeneous Hidden Markov Models (NHHMMs) used for forecasting of energy consumption, electricity production and of wholesale prices.

HMMs are probabilistic models that are used to study sequential data. The homogeneous HMM considers a first order Markov chain where the conditional probability for the next observation depends only on the current observation. The joint distribution for a sequence of $N$ states from the set $Z$ of random variables is noted as follows in a homogeneous Markov model (Bishop, 2006):

$$p(z_1, \cdots, z_N) = \prod_{n=2}^{N} p(z_n|z_1, \cdots, z_{n-1}) = p(z_1) \prod_{n=2}^{N} p(z_n|z_{n-1}), \tag{A.1}$$

where $\{z_1, \cdots, z_N\} \epsilon Z$.

The conditional distribution for the state $z_n$ is therefore: $p(z_n|z_1, \cdots, z_{n-1}) = p(z_n|z_{n-1})$. A basic homogeneous HMM framework is defined with four elements.

1. *Hidden variables*: HMM considers that there are some hidden random variables that influence observed variables. $\{z_1, \cdots, z_j\} \epsilon Z$, $j \epsilon \mathbb{N}$ denotes the subset of hidden variables or latent variables. In this thesis, HMM is considered to have a finite number of known hidden variables or states.

2. *Observed variable*s: These are variables that are observed and that should be predicted. $\{x_1, \cdots, x_j\} \epsilon X$, $j \epsilon \mathbb{N}$ denotes the subset of observed random variables.

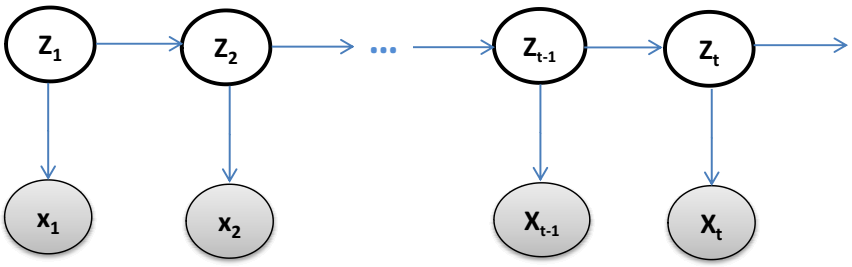



Figure A.1: Graphical Structure of the HMM Model. This is a graphical representation of the HMM model with latent states $(z_1,...,z_t)$ and observed variables $(x_1,...,x_j)$. The random variables are represented here using the nodes. The arrows represent the dependencies. The values of the observed variables are conditioned by the state of the latent variables. There is only one arrow that goes to a latent state. The transition to next latent state depends on the current latent state. The value of an observed variable is only conditioned by the current hidden state. $t \epsilon \mathbb{N}$ represents the time.

The proposed models use discrete observed random variables. Figure A.1 describes the corresponding graphical structural forms of the HMM. It specifies the conditional interdependencies between the random variables. The values of the observed variables are conditioned by the states of the latent variables. In order to compute the probable values of the observations, one needs to predict the state of the latent variables. The joint distribution of this HMM model is therefore (Bishop, 2006):

$$p(x_1, \cdots, x_N, z_1, ..., z_N) \quad = \quad p(z_1)p(x_1|z_1)\prod_{t=2}^{N} p(z_t|z_{t-1})p(x_t|z_t).$$

The transition probabilities $p(z_t|z_{t-1})$ between the latent states and the emission probabilities $p(z_t|z_{t-1})p(x_t|z_t)$ are considered to be time-independent.

3. *Transition matrix*: is the matrix of transition probabilities. The transition probability is the probability of transition from a latent variable $z_t$ at time $t$ to another $z_{t+1}$ at time $t+1$. $T_{ij} = p(z_{t+1} = j|z_t = i)$ denotes the probability of moving from hidden state $z_t = i$ at time $t$ to state $z_{t+1} = j$ at time $t+1$. The transition matrix satisfies $0 \leq T_{ij} \leq 1$ with $\sum_j T_{ij} = 1$.

4. *Emission matrix*: is the matrix of emission probabilities. The emission probability is the probability of having an observable variable knowing the hidden state. $E_{jk} = p(x_t = k|z_t = j)$ denotes the emission probability of the observed variable $x_t$ from the latent state $z_t$. The emission matrix satisfies $0 \leq E_{jk} \leq 1$ with $\sum_k E_{jk} = 1$.

The proposed HMMs are considered to be non-homogeneous, because the transition probability between two states is time dependent. This means that it is possible to have two energy consumption states $i$ and $j$ so that:

$$p(z_t = j|z_{t-1} = i) \neq p(z_{t+\tau} = j|z_{t-1+\tau} = i), \tag{A.2}$$

where $\tau > 0$ is a timespan. Technically, the transition between two consumption states differs depending on the time of the day or of the year. The transitions between energy consumption directly depend on the time of the day, the values of the weather parameters and on the consumer behaviour.

To predict the energy consumption and prices in the Power TAC environment, the autoregressive HMM (ARHMM) is used. The ARHMM is widely used for prediction of time series: speech recognition (Ephraim and Roberts, 2005), econometrics (Hamilton, 1989), meteorology (Zwiers and Von Storch, 1990; Parlange and Katz, 2000) and recently for wind energy forecast (Ailliot and Monbet, 2012). Furthermore, as it was possible to make use of the weather parameters to improve the forecast of energy production, the input-output HMM (IOHMM) is also applied (Bengio and Frasconi, 1995, 1996; Bishop, 2006).

### A.2.2 Implementation with Matlab

At the beginning of each game, the historical data from the set-up game are used to determine the hidden states of the customer energy consumption, production and clearing prices. A Matlab implementation of the Expectation Maximisation algorithm is used to determine the hidden states from the observed variables and the implementation of regression models to build the auto-regressive models of each hidden state. Moreover, to determine the number of states of each target variables, this thesis uses an implementation of the variational Bayesian mixture (Hoffman et al., 2013; Bishop, 2006; Attias, 1999) provided by Beal (2003). Finally a Java API [1] was used to interact with MATLAB.

### A.2.3 Evaluation during the Power TAC 2012

To provide some details about the performance of the prediction techniques in the PowerTAC environment, this section analyses an arbitrary chosen game with two brokers by observing the performance of the prediction one hour ahead over several days.

Figure A.2 shows the performance of the NHHMM for the forecast of the energy consumption in KWh for a customer with Customer ID: 513 during the game 562 between time slots 1131 and 1298. The standard deviation of the prediction is 501.24 with a mean value of 5027.46 of the observations. For the same period of the game, the performance of the IOHMM is analysed for the energy production as illustrated in Figure A.3. In general, the standard deviation of the NHHMM predictions is 10% of

[1] matlabcontrol-4.1.0.jar: https://code.google.com/p/matlabcontrol/

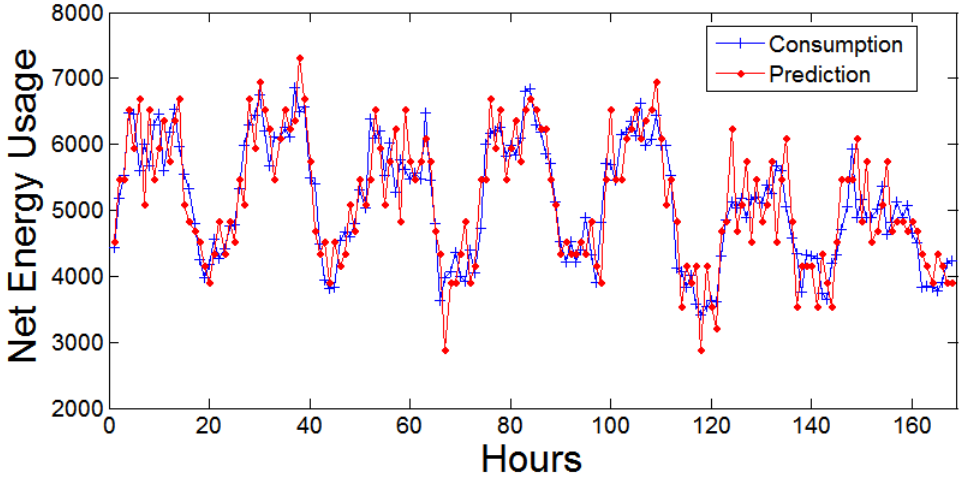



Figure A.2: Prediction of the Energy Consumption in Game 562 for the Customer-ID 513. The ARHMM works well for prediction, in the Power TAC using the dynamically generated models at the beginning of the game.

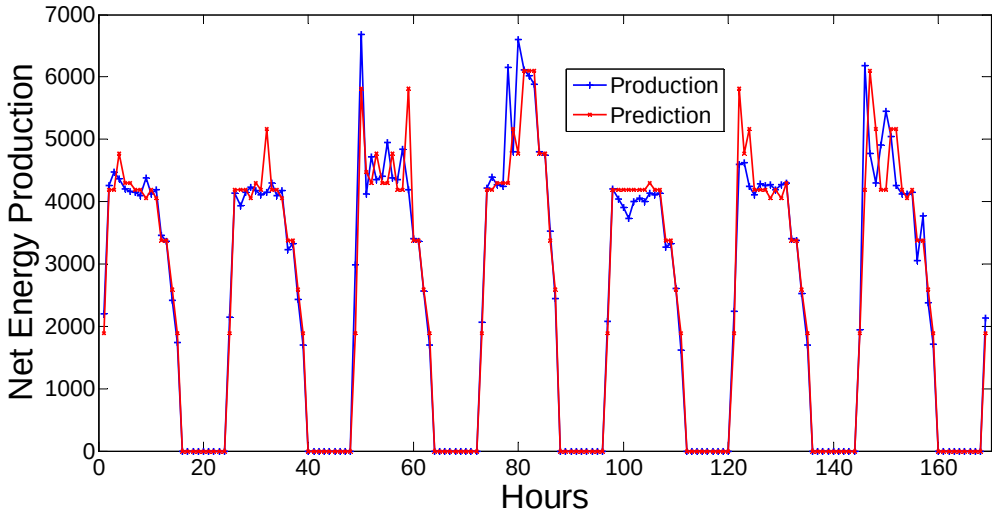



Figure A.3: Prediction of the Energy Production in Game 562 for the Customer-ID 525. Using the set-up data, the prediction of retail production with IOHMM works well enough in the Power TAC environment to enable the retailer's reasoning engine to function effectively.

the mean value of the observations. This poor prediction result is due to the fact that the models were generated online using the limited data provided by the setup game. However, these online models were able to support the agent decision making as presented in the thesis. The next section will brief outline robust prediction techniques that are used in the real world.

## A.3 Forecasting Techniques in Real World

This section reviews briefly the forecasting techniques used in the real world for predicting energy prices, demand or supply. A vast number of techniques have been proposed to deal with energy trading, energy demand forecast and price forecast in real life. The techniques used for energy demand and price forecasting can be classified in two trends: times series models and machine learning. The commonly used time series models include Autoregressive Integrated Moving Average (ARIMA) (Contreras et al., 2003; Conejo et al., 2005b; Cancelo et al., 2008), Generalised Autoregressive Conditional Heteroskedasticity (GARCH) (Garcia et al., 2005; Zheng et al., 2005), structural time series models (Harvey and Koopman, 1993) and multiple regression models (Ramanathan et al., 1997). The machine learning techniques include Artificial Neural Networks (ANN) (Gao et al., 2000; Yao et al., 2000; Zhang and Luh, 2005; Mandal et al., 2005) and Wavelet transform (Yao et al., 2000; Conejo et al., 2005b; Nguyen and Nabney, 2010). AR-HMM and IOHMM provide a more robust and faster way for generating prediction models at run time. The HMMs enable to automatically consider the intra-weekly and intra-daily behaviour of the energy demand and price.