Strategic Learning Agents in Equilibrium-based Markets for Resource Allocation

Eduardo Rodrigues Gomes

Submitted in fulfillment of the requirements of the degree of Doctor of Philosophy

Faculty of Information & Communication Technologies
Swinburne University of Technology

December 2009
Abstract

The Commodity Market (CM) economic model offers a promising approach for resource allocation in large-scale distributed systems. A CM provides a marketplace where sellers and buyers trade resources based on a common price known by all the participants. A traditional approach for resource allocation with CM is to apply concepts from the general equilibrium theory of microeconomics, assuming price-taking participants that will not attempt to strategically influence the mechanism to improve their profits or welfare. Such a condition, apart from being criticized as not realistic, is hardly satisfied in large-scale distributed systems where there is little control over the behaviour of the agents, making it impossible to guarantee that they will behave in an ordered manner. To understand the impacts of these attempts and to develop mechanisms that are robust in the presence of strategic participants are important aspects of the problem. Additionally, most mechanisms focus on the achievement of a Pareto-Optimal (PO) allocation, usually disregarding how fair and how desirable the solution is for both the system and the participants. Different PO outcomes can generate different gains to the involved parties. Therefore, being able to find a mutually desirable PO allocation is also an important aspect of the problem.

This thesis addresses the above issues and proposes a framework to optimise the individual and social allocation efficiency in equilibrium-based commodity market mechanisms composed of strategic learning agents. It addresses the problem from the premise that participants can behave like the entities composing real economies and will engage in strategic behaviour in order to satisfy their preferences.

We propose the Iterative Price Adjustment with Reinforcement Learning (IPA with RL), a CM-based mechanism in which agents use utility functions to describe preferences over different resource attributes and develop strategic behaviour by learning demand functions adapted to the market through Reinforcement Learning. We investigate and compare the individual and social performance of the mechanism in the presence of two types of strategic learning agents: selfish, whose learning goal is to improve their individual utility; and altruistic, whose learning goal is to improve the social utility. The results show that the market composed exclusively of selfish learning agents can achieve social performance similar to the performance obtained by the market composed exclusively of altruistic learning agents, both achieving near-optimal
social welfare measured by the Nash Product function. The results also show that the selfish agents are able to approximate the solution to the fairest PO allocation in situations where the altruistic agents fail. We further investigate these outcomes and present their theoretical analysis first from the perspective of game-theory, highlighting the properties of the results in terms of Nash Equilibrium and Pareto-Optimality, and then from the perspective of the dynamic process generated by the agents’ learning algorithm.

The thesis advances the knowledge base in a number of areas of Market-based Resource Allocation, Multiagent Learning and Agent-based Computational Economics. First, it formalizes a new conceptual framework involving multiagent reinforcement learning in commodity-market resource allocation mechanisms. Second, it introduces strategic learning agents in market-based resource allocation mechanisms founded on general equilibrium theory. So far, these mechanisms have assumed the existence of rational price-taking participants. The approach proposed in the thesis, instead, is more realistic as it explicitly addresses the existence of strategic participants that can try to exploit the mechanism. Third, it simultaneously optimises both the individual and social efficiency of the allocation. Existing approaches typically focus on the achievement of a PO allocation only, usually disregarding the individual utility and social welfare resulting from it. Fourth, it develops a theoretical model for the dynamics of Multiagent Q-learning with ε-greedy exploration. Despite the popularity of this algorithm, such a model has not been developed before. Finally, it develops a new technique to support the application and scalability of reinforcement learning in commodity-market resource allocation.
Acknowledgements

First of all, I would like to express my profound gratitude to my wife: Lari, thank you very much for making part of the journey, I would not have completed it without you.

I also would like to express my gratitude to my family. To my brother and sisters Evandro, Marcia and Silvia, thank you for your great affection and friendship. To my nephews José, João and Marco, thank you for the many joys and saudades\(^1\). To my parents, José and Enice, thank you for the immense care and dedication committed to your family and for your permanent incentive and support in each one of our projects (even if the project is to move half around the world). To the other members of my big family, Paulo, Neco, Adriana, Daniel, Bibi, uncle João and family, Chico, Dênis, Braulio, Gilso and Izete, thank you very much.

I am also thankful to my friends, the ones I have left in Brazil, whose happy memories have given me so many nostalgic times, and the ones I have made in Australia, whose company made the whole experience so much easier and enjoyable: thank you.

In particular, I am profoundly thankful to my supervisor Professor Ryszard Kowalczyk, who gave me orientation, encouragement and support in every single moment of difficulty, and to the members of the Intelligent Agents and Multiagent Systems (IA-MAS) research group at Swinburne University. I am also very thankful to the staff at the Faculty of Information and Communication Technologies, in particular Gillian Foster, Li Sim Loh and Mandish Webb.

Finally, to God: thank you for everything.

---

\(^{1}\)Portuguese word for the feeling of “missing someone”. The word does not exist per se in English.
Declaration

This is to certify that this thesis contains no material which has been accepted for the award of any other degree or diploma and that to the best of my knowledge this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

Eduardo Rodrigues Gomes
List of Publications


Contents

Abstract

Acknowledgements iii

Declaration iv

List of Publications v

1 Introduction 1
   1.1 Aim, Hypothesis and Research Questions 4
   1.2 Approach 5
   1.3 Contributions 7
   1.4 Thesis Overview 9

2 Market-based Resource Allocation 12
   2.1 Introduction 12
   2.2 Economic Models 13
      2.2.1 Auction 13
      2.2.2 Commodity Market 15
      2.2.3 Bargaining 18
   2.3 Market-based Systems 19
      2.3.1 Auction-based Systems 19
      2.3.2 Commodity-Market-based Systems 21
      2.3.3 Bargaining-based Systems 22
   2.4 Opportunities for Learning 23
      2.4.1 Learning in Auctions 23
      2.4.2 Learning in Commodity Markets 25
      2.4.3 Learning in Bargaining 26
   2.5 Summary 27

3 Reinforcement Learning 28
   3.1 Introduction 28
   3.2 The Learning Problem 29
      3.2.1 Elements of the Problem 30
      3.2.2 Delayed Rewards 32
      3.2.3 Exploration versus Exploitation 32
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>Formal Model</td>
<td>32</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Markov Decision Process</td>
<td>33</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Value Functions</td>
<td>34</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Bellman Equations</td>
<td>35</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Optimal Policies and Value Functions</td>
<td>36</td>
</tr>
<tr>
<td>3.4</td>
<td>Solution Methods</td>
<td>38</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Dynamic Programming</td>
<td>38</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Monte Carlo Methods</td>
<td>39</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Temporal-Difference Learning</td>
<td>39</td>
</tr>
<tr>
<td>3.5</td>
<td>RL Algorithms</td>
<td>40</td>
</tr>
<tr>
<td>3.5.1</td>
<td>The TD(0) Algorithm</td>
<td>40</td>
</tr>
<tr>
<td>3.5.2</td>
<td>The Sarsa Algorithm</td>
<td>41</td>
</tr>
<tr>
<td>3.5.3</td>
<td>The Q-learning Algorithm</td>
<td>42</td>
</tr>
<tr>
<td>3.6</td>
<td>Action Selection Mechanisms</td>
<td>43</td>
</tr>
<tr>
<td>3.6.1</td>
<td>(\epsilon)-greedy Exploration</td>
<td>44</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Boltzmann Exploration</td>
<td>44</td>
</tr>
<tr>
<td>3.7</td>
<td>Practical Reinforcement Learning</td>
<td>45</td>
</tr>
<tr>
<td>3.8</td>
<td>Multiagent Reinforcement Learning</td>
<td>46</td>
</tr>
<tr>
<td>3.8.1</td>
<td>Challenges</td>
<td>47</td>
</tr>
<tr>
<td>3.8.2</td>
<td>Formal Model</td>
<td>48</td>
</tr>
<tr>
<td>3.8.3</td>
<td>Learning Algorithms</td>
<td>49</td>
</tr>
<tr>
<td>3.9</td>
<td>Summary</td>
<td>51</td>
</tr>
<tr>
<td>4</td>
<td>Commodity Market with Reinforcement Learning</td>
<td>53</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>53</td>
</tr>
<tr>
<td>4.2</td>
<td>Commodity Market Scenario</td>
<td>54</td>
</tr>
<tr>
<td>4.2.1</td>
<td>The Market</td>
<td>55</td>
</tr>
<tr>
<td>4.2.2</td>
<td>The Participants</td>
<td>55</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Market-Participants Interaction</td>
<td>55</td>
</tr>
<tr>
<td>4.3</td>
<td>Equilibrium-based Markets</td>
<td>56</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Basic Principles</td>
<td>56</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Competitive Participants</td>
<td>57</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Economic Equilibrium</td>
<td>58</td>
</tr>
<tr>
<td>4.3.4</td>
<td>The Tâtonnement Process</td>
<td>59</td>
</tr>
<tr>
<td>4.3.5</td>
<td>Abstract Architecture</td>
<td>60</td>
</tr>
<tr>
<td>4.4</td>
<td>Market with Strategic Learning Participants</td>
<td>60</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Strategic Learning Participants</td>
<td>61</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Integration Architecture</td>
<td>62</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Dealing with Continuous Spaces</td>
<td>64</td>
</tr>
<tr>
<td>4.5</td>
<td>Summary</td>
<td>65</td>
</tr>
<tr>
<td>5</td>
<td>The Iterative Price Adjustment with Reinforcement Learning</td>
<td>66</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>66</td>
</tr>
<tr>
<td>5.2</td>
<td>The IPA with RL</td>
<td>67</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Iterative Price Adjustment</td>
<td>68</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Reinforcement Learning</td>
<td>69</td>
</tr>
</tbody>
</table>

viii
### Contents

5.2.3 The IPA as a RL problem .............................................. 71
5.3 General Experimental Setup ........................................... 73
5.4 Learning in the Presence of a Static Agent ......................... 74
  5.4.1 Experimental Setup .............................................. 75
  5.4.2 Learning Results .................................................. 75
  5.4.3 Market Results ..................................................... 77
5.5 Learning in the Presence of a Learning Agent ....................... 78
  5.5.1 Experimental Setup .............................................. 78
  5.5.2 Learning Results .................................................. 79
  5.5.3 Market Results ..................................................... 80
5.6 Related Works .......................................................... 83
5.7 Conclusions ............................................................ 85

6 Selfish and Altruistic Learning Participants in the IPA with RL 87
  6.1 Introduction ............................................................ 87
  6.2 The Individual and Social Reward Functions ......................... 88
    6.2.1 Experimental Setup .............................................. 89
    6.2.2 Learning Results .................................................. 90
    6.2.3 Market Results ..................................................... 91
    6.2.4 Summary .......................................................... 95
  6.3 Performance over the Learning Episodes ............................. 95
    6.3.1 Experimental Setup .............................................. 96
    6.3.2 Market Results ..................................................... 96
    6.3.3 Summary .......................................................... 98
  6.4 Multiple Learning Agents in the Market ............................ 99
    6.4.1 Experimental Setup .............................................. 100
    6.4.2 Market Results ..................................................... 100
    6.4.3 Summary .......................................................... 101
  6.5 Agents with Non-symmetric Preferences ............................. 102
    6.5.1 Experimental Setup .............................................. 102
    6.5.2 Market Results ..................................................... 103
    6.5.3 Summary .......................................................... 108
  6.6 Conclusions ............................................................ 109

7 Theoretical Analysis of the IPA with RL 111
  7.1 Introduction ............................................................ 111
  7.2 Static Analysis ........................................................ 112
    7.2.1 The Market-based Resource Allocation Games ................. 112
    7.2.2 Learning Experiments ............................................ 115
    7.2.3 Summary .......................................................... 116
  7.3 Dynamic Analysis ...................................................... 117
    7.3.1 A Model of Multiagent Q-learning with \( \epsilon \)-greedy Exploration 117
    7.3.2 Analysis of the Market-based Resource Allocation Games .... 133
    7.3.3 Summary .......................................................... 146
  7.4 Conclusion ............................................................. 147
8 Conclusion 148
  8.1 Answers to the Research Questions 151
  8.2 Summary of Contributions 154
  8.3 Future Works 154
    8.3.1 Equilibrium-based Markets Aspects 155
    8.3.2 Multiagent Reinforcement Learning Aspects 156
    8.3.3 Grid and Cloud Computing Contexts 157

Bibliography 160
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>The main contributions of the thesis in relation to its organization and research areas</td>
</tr>
<tr>
<td>3.1</td>
<td>Agent-environment interaction in Reinforcement Learning</td>
</tr>
<tr>
<td>4.1</td>
<td>A Commodity Market</td>
</tr>
<tr>
<td>4.2</td>
<td>Examples of Demand and Supply Curves</td>
</tr>
<tr>
<td>4.3</td>
<td>Economic Equilibrium</td>
</tr>
<tr>
<td>4.4</td>
<td>General idea of the tâtonnement process (and of equilibrium-based markets)</td>
</tr>
<tr>
<td>4.5</td>
<td>Abstract architecture for equilibrium-based mechanisms with competitive participants</td>
</tr>
<tr>
<td>4.6</td>
<td>Abstract architecture for equilibrium-based mechanisms with strategic learning participants</td>
</tr>
<tr>
<td>4.7</td>
<td>General idea of the tâtonnement process (and of equilibrium-based markets) with strategic learning participants</td>
</tr>
<tr>
<td>4.8</td>
<td>A technique to support the application and scalability of Reinforcement Learning in problems with continuous spaces</td>
</tr>
<tr>
<td>5.1</td>
<td>The IPA mechanism with RL</td>
</tr>
<tr>
<td>5.2</td>
<td>Agents’ utility functions</td>
</tr>
<tr>
<td>5.3</td>
<td>Static Agents’ demand functions</td>
</tr>
<tr>
<td>5.4</td>
<td>Evolution of the demand function of a learning agent</td>
</tr>
<tr>
<td>5.5</td>
<td>Examples of learning agents’ demand functions with trends</td>
</tr>
<tr>
<td>5.6</td>
<td>Utilities over experiments with S1</td>
</tr>
<tr>
<td>5.7</td>
<td>Utilities over experiments with S2</td>
</tr>
<tr>
<td>5.8</td>
<td>Utilities over experiments with S3</td>
</tr>
<tr>
<td>5.9</td>
<td>Evolution of the demand function of one of the learning agents</td>
</tr>
<tr>
<td>5.10</td>
<td>Learning agents’ demand functions with trends</td>
</tr>
<tr>
<td>5.11</td>
<td>Individual utilities in learner vs. learner and static vs. static experiments</td>
</tr>
<tr>
<td>5.12</td>
<td>Individual utilities in learner vs. static experiments</td>
</tr>
<tr>
<td>5.13</td>
<td>a) Average USW; b) Average ESW; c) Average NP</td>
</tr>
<tr>
<td>5.14</td>
<td>Average individual utility of learning agents over previous and current experiments</td>
</tr>
<tr>
<td>6.1</td>
<td>Agents’ utility functions</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>6.2</td>
<td>Examples of demand functions obtained by the learning agents, with f’t curves.</td>
</tr>
<tr>
<td>6.3</td>
<td>Average NP of the market using the individual and the social reward functions.</td>
</tr>
<tr>
<td>6.4</td>
<td>Average utility received by the agents using the individual and the social reward functions.</td>
</tr>
<tr>
<td>6.5</td>
<td>Comparison of Average NP obtained by the market using the individual and the social reward functions over the learning episodes.</td>
</tr>
<tr>
<td>6.6</td>
<td>Comparison of the average utility obtained by learning and static agents using the individual and the social reward functions over the learning process.</td>
</tr>
<tr>
<td>6.7</td>
<td>Average NP obtained by the market using the individual and social reward functions over the learning episodes.</td>
</tr>
<tr>
<td>6.8</td>
<td>Average utility obtained by the agents using the individual and social reward functions over the learning episodes.</td>
</tr>
<tr>
<td>6.9</td>
<td>Utility functions for price.</td>
</tr>
<tr>
<td>6.10</td>
<td>Utility functions for amount of resource.</td>
</tr>
<tr>
<td>6.11</td>
<td>Median Nash Product (NP) of the market for the Individual and Social Reward Functions (RF) over the Learning Episodes (LE).</td>
</tr>
<tr>
<td>6.12</td>
<td>Median Utility (U) of the agents for the Individual and Social Reward Functions (RF) over the Learning Episodes (LE).</td>
</tr>
<tr>
<td>6.13</td>
<td>Median demand (d) requests of the agents over the Learning Episodes (LE) for experiments of type E.</td>
</tr>
<tr>
<td>6.14</td>
<td>Median Utility (U) for amount of resource (UM) and price (UP) of the agents in experiments of type E.</td>
</tr>
<tr>
<td>6.15</td>
<td>Relationship between individual utility and social welfare.</td>
</tr>
<tr>
<td>7.1</td>
<td>Actions chosen by the players over the learning episodes in the individual reward game.</td>
</tr>
<tr>
<td>7.2</td>
<td>Actions chosen by the players over the learning episodes in the social reward game.</td>
</tr>
<tr>
<td>7.3</td>
<td>Joint-actions chosen by the players over the learning episodes.</td>
</tr>
<tr>
<td>7.4</td>
<td>Slope feld associated with Equation 7.5 ((\alpha = 0.2) and (r_{ai} = 5)) and examples of specific solutions obtained when (Q_{ai}(0) \in {0, 2, 8, 10}).</td>
</tr>
<tr>
<td>7.5</td>
<td>Slope felds associated with Equation 7.10 ((\alpha = 0.2) and (r_{ai} = 5)) for (x_i = 0.1) (Left) and (x_i = 0.9) (Right), and examples of specific solutions obtained when (Q_{ai}(0) \in {0, 2, 8, 10}).</td>
</tr>
<tr>
<td>7.6</td>
<td>Graphs for the Prisoner’s Dilemma.</td>
</tr>
<tr>
<td>7.7</td>
<td>Example of theoretical Q-values obtained by the model in the Prisoners Dilemma.</td>
</tr>
<tr>
<td>7.8</td>
<td>Example of an individual run of Q-learning in the Prisoners Dilemma.</td>
</tr>
<tr>
<td>7.9</td>
<td>Graphs for the Battle of the Sexes.</td>
</tr>
<tr>
<td>7.10</td>
<td>Example of an individual run of Q-learning in the Battle of the Sexes.</td>
</tr>
<tr>
<td>7.11</td>
<td>Graphs for the 3rd game.</td>
</tr>
<tr>
<td>7.12</td>
<td>Example of an individual run of Q-learning in the 3rd Game.</td>
</tr>
</tbody>
</table>
7.13 Graphs for the Social Reward Game when the initial $Q$-values are $Q_a = Q_b = [0, 0, 0, 0, 0]$ .............................................. 135
7.14 Graphs for the Social Reward Game when the initial strategy is $<1, 5>$ 136
7.15 Graphs for the Social Reward Game when the initial strategy is $<2, 4>$ 137
7.16 Graphs for the Social Reward Game when the initial strategy is $<3, 3>$ 138
7.17 Example of the typical behaviour found in the learning experiments with the Social Reward Game when the initial $Q$-values are $Q_a = Q_b = [0, 0, 0, 0, 0]$ .................................................. 139
7.18 Example of the typical behaviour found in the learning experiments with the Social Reward Game when the initial $Q$-values are $Q_a = Q_b = [0, 0, 0, 0, 0]$ .................................................. 139
7.19 Graphs for the Individual Reward Game when the initial $Q$-values are $Q_a = Q_b = [0, 0, 0, 0, 0]$ .................................................. 140
7.20 Graphs for the Individual Reward Game when the initial strategy is $<5, 5>$ ................................................................. 142
7.21 Graphs for the Individual Reward Game when the initial strategy is $<1, 5>$ ................................................................. 142
7.22 Example of the typical behaviour found in the learning experiments with the Individual Reward Game when the initial $Q$-values are $Q_a = Q_b = [0, 0, 0, 0, 0]$ .................................................. 144
7.23 Example of the typical behaviour found in the learning experiments with the Individual Reward Game when the initial $Q$-values are $Q_a = Q_b = [0, 0, 0, 0, 0]$ .................................................. 144
7.24 Graphs for the Individual Reward Game when the initial $Q$-values strategy is $<2, 4>$ ................................................................. 145
7.25 Graphs for the Individual Reward Game when the initial strategy is $<3, 3>$ ................................................................. 146
8.1 Integrated high-level architecture for the application of the IPA with RL in Cloud computing contexts (adapted from Buyya et al. [22]) 158
List of Tables

6.1 Experiments’ configuration. ........................................ 103
7.1 Individual Reward Game. ............................................. 114
7.2 Social Reward Game. ................................................ 114
7.3 Individual Reward Game. ............................................. 133
7.4 Social Reward Game. ................................................ 133
Chapter 1

Introduction

The rapid development experienced in the area of computer networks over the past years and the subsequent emergence of large-scale distributed systems, such as the Grid [40], peer-to-peer (P2P) networks [78], service-oriented architectures (SOAs) [36] and, more recently, Cloud Computing (CC) [23], have been generating a major paradigm shift in the way that research, business and personal software applications are designed, delivered and consumed. Some of the characteristics of this emergent paradigm are computational and geographical distribution, dynamic architecture, lack of coherent global knowledge, and dispersed ownership and control of computing and communication resources [40]. The efficient and flexible allocation of resources is a critical requirement for provision of software applications and services in such distributed environments. The traditional resource allocation approaches, typically based on centralised scheduling and decision-making [98], are not sufficient as they usually require global knowledge and total control over the resources, which is often hard to guarantee in large-scale distributed systems. Therefore it has been recognised the need for a paradigm shift also in the resource allocation approaches. As a result, in recent years, approaches based on economic principles have emerged as promising attempts to address the new requirements and a number of market-based mechanisms, such as auctions and commodity markets, have been proposed for resource allocation in large-scale distributed systems [8, 19, 24, 25, 28, 46, 59, 67, 68, 87, 105, 125, 126, 130, 130].

There are several advantages in framing the distributed resource allocation problem into economic terms [130]. In the Grid context, for example, if users are willing to share or trade access to their idle resources, they need to somehow account for the ratio between the benefit and costs of doing it. A similar need is clearly present in
P2P, SOAs and CCs where sharing and using the resources require incentives and mutual benefit for the providers and consumers. Market-based mechanisms are suitable for these types of problem because they intrinsically incorporate the notion of relative worth, usually abstracting it into a price or other type of exchange unit. Additionally, most economic models are well-understood as they have been subject of study in the Economics disciplines along the years, making it possible to utilise that body of knowledge also to analyse and design market-based resource allocation mechanisms.

The vast majority of market-based resource allocation mechanisms proposed over the years are based on the auction economic model and its many variants, e.g. [24, 25, 41, 44, 46, 59, 85, 87, 90]. It has been shown, however, that auctions can exhibit volatile behaviour [115, 118]. Another promising approach is the Commodity Market (CM) economic model [21, 28, 97, 105, 125, 126]. The CM offers a marketplace where sellers and buyers trade resources based on a common price known by all the participants. The price is defined by a trusted third-party, called market, using a pricing mechanism usually driven by demand-and-supply. Commodity markets can be easily confused with some variants of the auction model, in particular with double auctions. In commodity markets the market define the price and asks for the participants’ demand-and-supply information. In double auctions, on the other hand, the participants define the price in which they want to trade and inform the auctioneer, which then matches demand requests with relevant supply offers. Wolski et al. [126] have experimentally compared commodity markets and auctions, concluding that the commodity market model is in general more superior to the auction model. This thesis focuses on the commodity market economic model.

A traditional approach for resource allocation with CM is to apply concepts from the general equilibrium theory of microeconomics. The work, based on the ideas of the nineteenth-century French economist Léon Walras [119], proposes that markets are able to coordinate the allocation of resources by finding an equilibrium price where the total demand matches the total supply. When this price is found, the market enters into a state of economic equilibrium that, by the First Fundamental Theorem of Welfare Economics [35, 103], is a Pareto-Optimal (PO) solution for the resource allocation problem. A solution is Pareto-Optimal if there is no other solution that can improve one agent’s outcome without deteriorating the others’.

One of the conditions for the Walrasian economic equilibrium is the existence of a Perfectly Competitive Market where no participant (buyer or seller) has significant impact on price. In this case, the participants are assumed to be rational price-taking
entities, acting towards the maximization of their gains but taking the price as a
given aspect of the decision making problem, an aspect outside their control. In other
words, participants are assumed to be passive entities that will not actively attempt to
influence the mechanism in order to obtain higher gains. One participant’s strategy
is thus reduced to solving simple optimization problems, neglecting the dependence
between its actions and the actions of the other participants.

The price-taking assumption, although convenient to ensure PO allocations, is often
criticized as not realistic [112] and hard to be satisfied in large-scale distributed sys-
tems. In such systems there is little control over the behaviour of the participants,
making it practically impossible to guarantee that they will behave in an ordered man-
er. Buyers and sellers will typically have different objectives, preferences and de-
mand/supply patterns which they will try to satisfy. Therefore, one cannot assume
they will not attempt to exploit the system by engaging in strategic behaviour to ob-
tain higher gains, possibly decreasing the overall quality of the allocation. Hence, to
understand the impacts of these attempts and to develop mechanisms that are robust in
the presence of strategic participants, being able to deliver optimal allocation also in
this situation, are important aspects to enable the next generation of large-scale distri-
buted systems. In this thesis we investigate these aspects by introducing participants
with learning capabilities into the market. For this, we use Multiagent Reinforcement
Learning (MARL) [17, 99].

Reinforcement Learning (RL) is a sub-area of Machine Learning that studies the pro-
blem in which an agent, e.g. a buyer or seller, learns optimal strategies from interaction
with the environment. The agent explores the environment by trying different actions
in different situations, and receives scalar reinforcement signals indicating how good
the tried actions are. Its task is then to choose actions in order to maximize some
long-run measure of the reinforcement signals. When more than one agent is applying
reinforcement learning, then we have a MARL problem, the extension of RL to mul-
tiagent scenarios.

Another problem with most current commodity market mechanisms is that they focus
on the achievement of a PO allocation, but usually disregard how fair and how desi-
rable the allocation is for both the system and the participants. Different PO outcomes
can generate different utility gains to the parties involved in the resource allocation
process. Therefore, being able to find a fair PO outcome, in which all the participants
are equally or near-equally satisfied is also an important aspect of the problem. In this
thesis we aim at optimising both the individual utility of the participants and the social
welfare of the market. For this, we introduce utility-aware agents into the market, that is, agents that explicitly model preferences through utility functions, allowing for the interpersonal evaluation of the outcomes, and measure the market’s performance using Social Welfare (SW) functions. In particular, we apply the Nash Product SW function, which is well-suited to the resource allocation problem since it emphasizes both the improvement and the balance between the utility of the participants.

This thesis addresses the above issues and proposes a framework to optimise the individual and social allocation efficiency in equilibrium-based markets for resource allocation composed of strategic learning agents. It addresses the market-based resource allocation problem from the premise that participants can behave like the entities composing real economies and will engage in strategic behaviour in order to satisfy their preferences. This premise enables the development of a more realistic and robust scenario for the resource allocation in distributed systems and the improvement of the applicability of mechanisms for commodity-market resource allocation in these systems. The next section formalizes the aim, hypothesis and research questions of the thesis. Section 1.2 presents a brief overview of the approach taken and areas involved in this research. Section 1.3 presents the main contributions and Section 1.4 presents the organization of the thesis.

1.1 Aim, Hypothesis and Research Questions

The aim of this thesis is to investigate and develop a framework for the optimization of the individual and social allocation efficiency in equilibrium-based commodity market mechanisms composed of strategic learning agents.

The hypothesis driving this research is that:

**The individual and social allocation efficiency of equilibrium-based commodity market resource allocation mechanisms with strategic learning agents can be optimised.**

In this context, the specific research questions are:

- How to integrate strategic learning participants into equilibrium-based commodity market mechanisms?
1.2. Approach

- What are the impacts of the presence of strategic learning participants on the individual and social performances of equilibrium-based commodity market mechanisms?

- What mechanism can optimize the individual and social efficiency in equilibrium-based commodity market mechanisms with strategic learning participants?

- How to model the dynamics of the agents and to predict the expected behaviour of equilibrium-based commodity market mechanisms with strategic learning participants?

- How to support the application and scalability of reinforcement learning in the context of market-based resource allocation?

1.2 Approach

This thesis considers the existence of strategic participants in equilibrium-based markets by introducing agents with learning capabilities. It addresses the general scenario in which a limited amount of resources has to be allocated to a set of agents in a commodity market resource allocation mechanism using an equilibrium-based approach. The agents use utility functions to describe preferences over different resource attributes and develop strategic behaviour via Machine Learning techniques, more specifically Reinforcement Learning. In this section we give a brief introduction to the main areas involved in this research.

The general area addressed in this thesis is that of **Market-based Resource Allocation** (MBRA). MBRA is concerned with the application of economic principles and models to abstract the problem of resource allocation. The advantages of framing the problem into economic terms are many [130], some of them have been described in the introductory section.

Directly related to MBRA is the area of **Multiagent Systems** (MAS) [127]. A MAS is composed by a collection of interactive entities called agents. An agent is a software entity that can sense its environment and act autonomously in order to satisfy its objectives. In this context, the participants of some economic model can be seen as agents and the whole economy a MAS.
Chapter 1. Introduction

The area of Multiagent Resource Allocation (MARA) [27] is also relevant for this research. It studies the problem of resource allocation in multiagent contexts. In the typical MARA problem, the agents have preferences over the resources and actively participate in the allocation process. Since it addresses a similar problem, the adoption of MARA concepts is quite natural.

Also very relevant is the area of Agent-based Computational Economics (ACE) [112]. This area first appeared in the economics field and studies the simulation of economies through MAS principles. The addressed research problem is to explain how the interactions between individuals can give rise to certain economic or social phenomena that are so far unexplained by traditional economics. The area combines concepts from economics, social sciences and artificial intelligence.

Multiagent Learning (MAL) is the branch of Machine Learning concerned with the development of methods to enable agents to learn while involved in a society. Current work in this area can be broadly divided into concurrent learning and team learning [79]. Team-learning typically uses a single learner to learn on behalf of all the team members. Its main advantage is that the application of standard machine learning techniques is made relatively easy. The disadvantage, however, is the centrality required. Concurrent-learning approaches use multiple simultaneous learners, usually one per agent. While its distribution is seen as one of its good characteristics, particularly for this research, it is also one of the big difficulties in its implementation. In this thesis we apply the concurrent learning approach. Each participant will be equipped with a learning component.

Reinforcement Learning (RL) studies the problem faced by an active decision-making entity that has to learn to achieve a goal from interaction with its environment. The entity is not told which decisions to take nor is provided with examples of how to solve the problem, as in most forms of machine learning. Instead, it receives feedbacks on the quality of its performance through scalar reward signals. The entity’s learning task is to map situations it encounters during the interaction with the environment to the decisions to be taken when in those situations so as to maximize some long-term measure of these reward signals. The main rationale for the application of Reinforcement Learning in this thesis is that it enables the problem to be solved on-line and without the necessity of having a model of the environment. In our specific case, these characteristics permit the participants to develop their strategic behaviours while interacting with the market.
1.3 Contributions

The main contributions of this thesis are:

- **A conceptual framework for commodity-market resource allocation with multiagent reinforcement learning.** We identify the relationships between the two areas and develop an integrated conceptual framework for commodity-market resource allocation with reinforcement learning. The framework enables the integration of strategic learning participants into equilibrium-based mechanisms for market-based resource allocation.

- **The IPA with RL, a utility-aware market-based resource allocation mechanism supported on the multiagent learning approach.** The mechanism results from the combination between the Iterative Price Adjustment (IPA) pricing mechanism and Reinforcement Learning. Unlike the usual approaches of defining demand functions, agents of this new mechanism define utility functions to describe preferences over resource attributes and learn demand functions adapted to the market by Reinforcement Learning. The mechanism presents the interesting property in which agents that apply a reward function based solely on their individual utility can achieve near-optimal social welfare, measured by the Nash Product function. Therefore, this mechanism can be used in domains where social utility should be maximized but agents are not guaranteed to be cooperative or are unwilling to reveal private information, such as their preferences.

- **The analysis and comparison of individual and social-based rewards for market-based resource allocation.** During the experimental investigation in the IPA with RL we evaluate the application of two different reward functions: one based on the individual utility of the agents; and the other based on the social welfare of the allocation, generating selfish and altruistic learning agents respectively. Our experiments show that the market composed exclusively of selfish agents can achieve social performance similar to the performance obtained by the market composed exclusively of altruistic agents, both reaching near-optimal SW, measured by the Nash Product social welfare function. This outcome is significant not only for the market-based resource allocation domain but also for a series of other domains where individual and social utility should be maximized but agents are not guaranteed to act cooperatively in order to achieve it or they do not want to reveal their private preferences.
• The theoretical model for the dynamics of Multiagent $Q$-learning with $\epsilon$-greedy exploration. Despite the popularity of this learning algorithm, the problem of analysing and modelling the expected behaviour of agents using it has not been addressed before. To develop the model, we first analyse a continuous-time version of the $Q$-learning update rule and study how the $\epsilon$-greedy mechanism and the presence of other agents affect it. We then use this analysis to model the problem as a system of difference equations which is used to theoretically calculate the expected evolution of the $Q$-values and, consequently, the expected behaviour of the agents. We demonstrate the feasibility of the model in typical games selected from the literature and then apply it to understand and justify the results found in our experimental investigation in the IPA with RL.

• A new technique to support the application and scalability of Reinforcement Learning in problems with continuous spaces. Traditional reinforcement learning algorithms are based on tabular representations of the action and state spaces and, therefore, cannot be directly applied in problems with continuous spaces. Continuous spaces also have a direct effect on the scalability aspects of the algorithms, which usually rely on exhaustive trial-and-error searches. The new technique consists in the discretization of the state and the action spaces during the learning process and the application of curve-fitting or interpolation procedures in the learnt discrete policy to obtain the agent’s continuous policy. The approach enables the application of reinforcement learning in problems with continuous spaces and leads to the reduction of the required amount of learning, as evidence by the experimental investigation in the IPA with RL.

This thesis advances the knowledge base in a number of areas of Market-based Resource Allocation, Multiagent Learning and Agent-based Computational Economics. First, it formalizes a new conceptual framework involving multiagent reinforcement learning in commodity market resource allocation mechanisms. Second, it introduces strategic learning agents in market-based resource allocation mechanisms founded on general equilibrium theory. So far, these mechanisms have assumed the existence of rational price-taking participants. The approach proposed in the thesis, instead, is more realistic as it explicitly addresses the existence of strategic participants that can try to exploit the mechanism. Third, it simultaneously optimises both the individual and social efficiency of the allocation. Existing approaches typically focus on the achievement of a PO allocation only, usually disregarding the individual utility and social welfare resulting from it. Fourth, it develops a theoretical model for the dynamics of
Multiagent $Q$-learning with $\epsilon$-greedy exploration. Despite the popularity of this algorithm, such a model has not been developed before. Finally, it develops a new technique to support the application and scalability of reinforcement learning in problems with continuous spaces.

It is important to clarify that, while resource allocation for large-scale distributed systems is used as an illustrative example, the study carried out in this thesis focuses on the development of the mechanisms and algorithms and their respective experimental and theoretical evaluation and analysis. The application of the proposed approach in Grid and Cloud computing contexts is subject of future work. More details are presented in Section 8.3.3.

### 1.4 Thesis Overview

The thesis is organized as follows. In the next two chapters we present an overview of the main areas related to this research. **Chapter 2** addresses the field of Market-based Resource Allocation. It describes the concepts of the different economic models for resource allocation and presents relevant works that apply those concepts. It also discusses specific points where the application of machine learning and/or strategic learning participants could be used. **Chapter 3** addresses the Reinforcement Learning area. It reviews its theoretical foundations, solution methods and aspects affecting its application in practical problems. It also reviews the case of Multiagent Reinforcement Learning, presenting the rationale for the choice of learning algorithm applied in the thesis.

**Chapter 4** introduces the conceptual framework for the realisation of commodity-market resource allocation with reinforcement learning. The framework enables the integration of strategic learning participants in equilibrium-based markets and forms the basis for the experimental and theoretical investigation performed in the thesis. The chapter also introduces the new technique to support the application and scalability of reinforcement learning in problems with continuous spaces.

**Chapter 5** investigates the application of strategic learning agents in a specific type of equilibrium-based mechanism called Iterative Price Adjustment (IPA). The chapter introduces the IPA with RL, a market-based mechanism that enhances the original IPA by considering the scenario where agents use utility functions to describe preferences.
over different resource attributes and develop strategic behaviour by learning demand functions adapted to the market through RL. The experimental investigation in the chapter considers strategic learning agents that exhibit *selfish* behaviour.

**Chapter 6** presents further experiments in the IPA with RL. It investigates the impacts of *selfish* and altruistic learning agents in the mechanism. The learning process of the altruistic agents is based on the social welfare of the allocation, using a social-based reward function. The *selfish* agents, on the other hand, learn based on their individual utilities, using an individual-based reward function. The impacts of both types of strategic learning agent on the individual and social performances of the market are experimentally investigated and the results compared.

**Chapter 7** presents the theoretical analysis of the results found in the IPA with RL. The goal is to gain an understanding on the reasons for the behaviour found in the mechanism. The study is performed in two parts. The first part focuses on the game-theoretical analysis of the scenario. For this, we develop two games which simulate the application of the individual and the social reward functions in the IPA with RL and highlight the properties of the solutions found by the agents in these games in terms of Nash Equilibrium and Pareto-Optimality. The second part focuses on the analysis of the agents’ behaviour from the perspective of the dynamic process generated by the agents’ learning algorithm. We develop a theoretical model of the Multiagent Q-learning with $\epsilon$-greedy exploration algorithm and, then, supported by this model, we investigate the dynamic behaviour of the agents in the formulated games.

**Chapter 8** concludes the thesis, summarizing it, answering to the research questions and pointing out some future works.

The main contributions of the thesis in relation to its organization and research areas is shown in Figure 1.1.
1.4. Thesis Overview

Figure 1.1: The main contributions of the thesis in relation to its organization and research areas
Chapter 2

Market-based Resource Allocation

This chapter provides the fundamentals of the Market-based Resource Allocation area. It reviews the main economic models applied in resource allocation and presents examples of existing mechanisms applying those models. It also presents a discussion on aspects and opportunities for the application of machine learning techniques in market-based resource allocation and gives examples of existing mechanisms applying it.

2.1 Introduction

There has been an increasingly research interest on the application of market-based mechanisms for resource allocation over the past years. Most of this interest results from the emergence of large-scale distributed systems, such as the Grid [40], peer-to-peer (P2P) networks [78], service-oriented architectures (SOAs) [36] and, more recently, Cloud Computing (CC) [23], and their needs for efficient and flexible resource allocation.

Typical features of those systems include architectures in which users and resource are distributed not only computationally but also geographically and can enter and leave the system at any time, dynamically changing the system’s structure [40]. In such a scenario, the traditional resource allocation approaches, typically based on the centralised scheduling and decision making, are not sufficient as they require global knowledge and total control over the resources, which is usually not possible to guarantee [98].
There are several advantages in considering the resource allocation problem in economic terms [98, 130]. In the Grid context, for example, if users are willing to share or trade access to their idle resources, they need to somehow account for the ratio between the benefit and costs of doing it. A similar need is clearly present in P2P, SOA and CC where sharing and using the resources require incentives and mutual benefit for the providers and consumers. Market-based mechanisms are suitable for these types of problem because they intrinsically incorporate the notion of relative worth, usually abstracting it into a price or other type of exchange unit. Additionally, most economic models are well-understood as they have been subject of study in the Economics disciplines along the years, making it possible to utilise this body of knowledge to analyse and design market-based mechanisms.

The chapter is organized as follows. The next section introduces the relevant concepts of the main economic models for resource allocation. Section 2.3 give examples of mechanisms applying the models. Section 2.4 discusses some opportunities for the application of learning in the existing models together with examples of existing mechanisms applying it. Section 2.5 summarizes the chapter. In the discussion, the terms buyer, consumer and client are used interchangeably to identify the entity that participates in the allocation process in order to use resources; and seller, producer and server to identify the entity that participates in the process in order to provide resources.

### 2.2 Economic Models

A comprehensive survey on the different economic models applied in resource allocation is presented by Yeo and Buyya [130]. The authors classify the existing approaches into commodity market, posted price, bargaining, tendering / contract-net, auctions, bid-based proportional resource sharing, community / coalition / bartering / share holders, and monopoly / oligopoly systems. We will focus our description on the three main approaches: auction, commodity market and bargaining.

#### 2.2.1 Auction

Auctions are the most common of the market-based mechanisms. They have been used to allocate resources from very remote times and are used with success for a series of
different real-world applications, from ordinary objects, as in eBay (http://www.ebay.com.au), to telecommunication rights.

### 2.2.1.1 Concepts

Classic auction models differ mainly in three aspects: pricing procedure, bidding/asking procedure and nature. The pricing procedure can be ascending or descending. In price ascending auctions, the price starts at a low level and rises until no bid is made. Price descending auctions start with a high price, which is decreased until someone accepts it. With regards to the bidding/asking procedure, auctions can be open or closed. In open auctions, bids and asks are made public during the bidding process. In closed, or sealed, auctions, they are not. With regards to the nature, auctions can be: for supply, when the sellers offer a good wanted by a buyer; for demand, when the buyers bid for a good being sold; or double, when a number of buyers bid to buy goods being sold by a number of sellers.

There are several auction models, and variants as well. For example, the auction can have a reserve price, a least/maximum acceptable price for the good. Another example, the price paid by the winner may not be the highest price, but the second-highest, which is the case of Vickrey auctions.

Some classic auction models are described bellow:

- The **English auction** is open and price ascending. The auctioneer begins with a low price and participants bid openly against each other. The auction ends when no participant is willing to bid further. Depending on what the other participants bid, it is possible for a bidder to win the auction by bidding a value less than the bidder’s true valuation for the good. It is also well-known the possibility of collusion among participants to exploit the mechanism.

- The **Dutch auction** is open and price descending. The auctioneer begins with a high asking price and decreases it until some participant is willing to pay such amount or the reserve price is reached. The winning participant pays the last announced price.

- In **first-price sealed-bid auction**, bids are submitted simultaneously and no bidder knows the bid of any other participant. The highest bidder pays the price he/she submitted.
2.2. Economic Models

- The second-price sealed-bid auction, also referred to as Vickrey auction, is similar to the first-price sealed-bid auction. In this model, however, instead of paying the value of its own bid, the highest bidder pays value of the second-highest bid. As a consequence, the best strategy for a bidder is to bid the true valuation for the good, which is a very good property. However, the model has been rarely applied. An overview of its drawbacks and explanations for its rare use is presented by Sandholm [89].

In addition to the classic models, we have:

- **Double auctions**, in which sellers and buyers submit the prices or price ranges in which they are willing to trade. The auctioneer task is to match the proposals. This procedure can be made at determined intervals, when it is called periodic double auction, or as soon as a new proposal arrives, when it is called continuous double auction.

- **Combinatorial auctions**, which allow for bids on combinations of goods. A bid language is used by the participants. The expressiveness of the language determines the types of combinations the participants can bid for. Resolving the optimal allocation in combinatorial auctions is usually NP-Complete [124]. For this reason, several works address the problem using approximation approaches [7, 41, 90].

- The contract-net protocol, in which a consumer advertises its requirements and asks for bidding proposals from producers. Having received the information, the consumer chooses the best proposal. Although, this protocol is not always regarded as an auction model, it can be described as a bid-sealed auction in the reverse direction.

- And, the bid-proportional sharing auctions, in which bidders receive shares of resources proportional to their bids. The auctioneer is responsible for aggregating the bids and determining how much each participant will get.

2.2.2 Commodity Market

The commodity market economic model offers a place for trading of resources. In commodity markets, a third party, generally called the market or auctioneer, sets the
prices for the resources and asks both buyers and sellers for willingness to buy and sell at that price. Those wishing to participate agree to make business and an exchange of currency for resources takes place. In some mechanisms, the market observes the unsatisfied supply and demand and uses that information to adjust the price.

According to Wolski et al. [126], the main difference between commodity markets and auctions is that in the first the buyer does not purchase a “specific item, but rather takes one of many equivalents. However, there are variations of the model in which buyers have the option for choosing a specific item. In special, commodity market can be easily confused with double auctions (see previous section). In commodity markets the market usually define the price and asks for the participants’ supply and demand information. In double auctions, the participants define the price in which they want to trade and inform the auctioneer.

2.2.2.1 Concepts

The core aspect of a commodity market is the pricing mechanism. It is generally responsible for adjusting the price in order to maximize some quality measure of the allocation. For example, the pricing mechanism may target to find a price able to maximize resource usage, or the aggregate utility of the participants. The pricing policy can be derived by various methods and can be flat or variable depending on the resource supply and demand. In this section we discuss some pricing approaches.

Several of the pricing mechanisms found in the literature are based on the tâtonnement processes, which are of particular importance for this thesis. Tâtonnement was suggested by Walras [119] as a way to describe how real-world markets come to equilibrium. The rough idea is to increase or decrease the of each resource type according to its excess demand until demand-and-supply is equilibrated.

An example of mechanism based on tâtonnement is the Iterative Price Adjustment (IPA) [37]. Derived from the Lagrangian Relaxation method, the mechanism decomposes the resource allocation optimization problem into smaller and easier sub-problems. The process is a cycle that begins with a facilitator (the market) announcing the initial prices for the resources. Based on this price, the participants calculate the amount of resources that maximize their private utilities (the sub-problems) and send these values to the facilitator. The facilitator adjusts the prices according to the total demand of the agents and announces the new prices. If the demand exceeds the supply,
the price is increased, otherwise it is decreased. The process continues until an equilibrium price is reached, when the resources are sold and the market is “cleared”. In the equilibrium, the total demand equals the supply or the price of the excessive supply is equal to zero.

The IPA may fail to converge when agents have local constraints. Jennergren [56] proposed the price schedule method to overcome such problems. In this method, the price used by the participants to calculate the demand is related to the amount demanded, changing from a linear strategy (as in the IPA) to a non-linear strategy. It has been proven to converge if the participants have concave objectives and compact convex feasible sets.

Both the IPA and the price schedule mechanisms do not consider the presence of sellers in the market. The participants are all buyers. And also, the market does not target at making revenue. The resources are sold at a price that maximizes the resource usage.

Chunlin and Layuan [29] developed a pricing mechanism in which the goal is to maximize the aggregated utility of the participants. This pricing mechanism uses the same principles of the IPA: the problem is decomposed using Lagrangian Relaxation and the price is adjusted in an iterative fashion following the law of demand-and-supply. Although the authors claim the mechanism optimizes aggregated utility of the participants, the preferences of the resources owners are assumed to be the maximization of the resource usage. Therefore, there is no significant difference between this pricing approach and the IPA’s pricing approach. The underlying resource allocation mechanism, however, is significantly different, as shall be seen in Section 2.3.2.

The pricing mechanism implemented in Wolski et al. [126] considers sellers and buyers in the market. It uses a price adjustment procedure to approach an equilibrium price, where supply equalizes demand. Although it is the same goal of Everett [37], Jennergren [56] and Chunlin and Layuan [29], the sellers’ preferences are really taken into account. Sellers express their preferences by supply curves, which are defined based on historical information of resource price. Therefore, the market works in the sense of also making revenue for the sellers. The mechanism implements the theoretical work of Smale [102] through an algorithm to approximate it. The approximation algorithm, called G-commerce, was developed given the non-viability of the Smale’s method for the “real world” as it requires polling consumers and producers for demand and supply information for all possible prices. The approximation algorithm does not rely on this assumption.

The pricing mechanism applied in Libra [97] is based on a cost formula. Users submit the jobs with estimated runtime, deadline and budget they are willing to expend to have the job completed within the deadline. The estimated runtime and deadline are used by a central server in the cost formula to calculate the price. The formula is defined in a way that the price will be higher for large jobs with short deadlines and lower for small jobs with long deadlines. It uses two constants which are defined a priori. The paper mentions the possibility of driving these constants by demand and supply, but does not describe the algorithm for it.

2.2.3 Bargaining

In the bargaining economic model, the allocation emerges from local negotiations among the participants. It enables both users and producers to negotiate for a mutually agreeable price. According to Yeo and Buyya [130], a consumer will typically start the negotiation with lower prices, to minimize the costs, and producers with higher prices, to maximize profit.

There is a large amount of work on automated or agent-based negotiation that can be easily extended to realise bargaining-based mechanisms. For an overview of the research area on negotiation, please see Jennings et al. [57], Lai et al. [66] and Braun et al. [15].

2.2.3.1 Concepts

According to Jennings et al. [57], an automated negotiation model is composed of Negotiation Protocols, Negotiation Objects and the Agent's Decision Making Models.

A Negotiation Protocol define the rules that govern the interaction. This includes the permissible types of participants, the negotiation states, the events that cause negotiation states to change and the valid actions of the participants.

A Negotiation Object comprehends the range of issues over which agreement should be reached. The object may contain a single issue (e.g. price) or several issues (e.g. QoS requirements). The negotiation object can be fixed, so the participants can either accept or reject it, or it can be variable, so the participants can alter the values of the issues (e.g. to make counter proposals) or the structure of the object (e.g. to add or
remove issues). The types of operations allowed on the agreement depends on the negotiation protocol.

Finally, the *Agent’s Decision Making Models* covers the negotiation strategies that define the behaviour of the participants, which should comply with the negotiation protocol. The sophistication of the model depends on the protocol in place, the nature of the negotiation object, and the range of operations that can be performed on it.

## 2.3 Market-based Systems

This section presents some examples of systems based on the economic models described in the previous section. It is not the goal to exhaustively present all the existing mechanisms, but to give examples of how the concepts are applied in existing approaches.

### 2.3.1 Auction-based Systems

Bellagio [6] employs a centralized combinatorial period auction, allowing for the allocation of combinations of heterogeneous goods. Payments are determined using the threshold rule [82], which is similar to the payment rule used in second-price auctions, enforcing the users to bid their true valuations for the goods. Auctions are cleared hourly. Users have a virtual bank account with bounded balance. The resources are owned by the system and the revenue divided among the users.

GridMarket [25] uses a continuous double auction. Bids from sellers and buyers are adjusted according to simple rules. Sellers and buyers set starting prices and the elasticity they would accept in such a price. The system then generates automatic bids based on such parameters and the time elapsed in the auction. Resources are standardized in the sense that sellers and buyers bid in classes of resources and not in the resources themselves.

Grosu and Das [46] investigated the use of first-price, second-price and double auctions. Considering a mix of risk-averse and risk-neutral users, they found first-price auctions better from a resource’s perspective and Vickrey auctions better from a user’s perspective, while double auctions favour both users and resources. In Das and Grosu

[34], this study is extended to include combinatorial auctions. However, the results of this later work do not include comparisons with the previous models studied.

Chen et al. [24] investigated co-allocation in auctions. For this end, they proposed the co-bids approach and two heuristics for the winner selection. The co-bids first approach allocates co-bids first because they have more restrictions than other bids. The no-preferences approach does not differentiate among the different types of bids.

Tycoon [67] employs the Auction Share scheduling algorithm. It allocates resources estimating proportional shares with consideration for latency-sensitive and risk-averse applications.

Spawn [118] uses a sealed-bid, second-price auction. Each seller runs its own auction. Bids are accepted for the next time slice available until close to the end of the current time slice.

JaWS [68] is a framework for the deployment of grid applications. The resource allocation is made based on a continuous double auction. The auction is run by a central entity called market server. A credit system is cited but not described. The authors also point out the need for multiple market servers and for the investigation of strategies for applications (client) and hosts (provider) in order to improve the bidding procedure.

POPCORN [87] provides three auction models for trade in CPU time: a repeated Vickrey auction; a double auction; and a variant of double auctions called Clearinghouse double auction. In the double auction, sellers and buyers define a top and floor prices and a rate of change. The generation of new bids is based on these parameters. The Clearinghouse double auctions differs from the common double auctions in the aspect that more than a buyer-seller pair is matched in each round. At given intervals, the bids and asks from buyers and sellers are used to calculate demand and supply curves. The curves are intersected to find the equilibrium price in which the resources will be traded.

In WALRAS [26, 125], each good is associated with an auction. Consumers and producers participate in the auction by submitting their demand and supply curves. The auction derives a market-clearing price based on those curves. The mechanism is actually an implementation of the tâtonnement procedure and, for this reason, could be classified as a commodity-market-based approach. However, as the submissions of demand and supply curves can be seen as bids to consume and provide resources, the classification as an auction-based approach is more appropriate.
2.3. Market-based Systems

The contract-net protocol is used in several systems. In Faucets [59], for example, it is used in a centralized way. There is a centralized server that maintains a list of resources and applications that users can execute. Users specify QoS contracts. The server determines the resources that satisfy the QoS requirements and then follows the contract-net protocol with the Grid resources to conclude the allocation. Grid resources compete to maximize their own profit and resource utilization.

2.3.2 Commodity-Market-based Systems

In Wolski et al. [126], the authors investigate the use of commodity markets and auctions for Grid resource allocation. The model is divided between consumers and producers of resources. The commodity market is implemented using a tâtonnement-based mechanism. The auction is implemented using a second-price-like mechanism. The simulations dealt with CPU and disk storage resources. Resource types are complementary for the consumer model, e.g. if the consumer does not hold enough CPU it does not express demand for disk storage, and non-complementary for the producer model. Producers define the amount of resources they will supply to the market based on historical price information. Consumers define their demand request based on the current price of resources and budget. The simulation results have shown the superiority of commodity markets over auctions. As an important comment, the authors cite that auctions may be locally advantageous but exhibit volatile emergent behaviour system-wide as presented in Waldspurger et al. [118].

Subramoniam et al. [105] developed a commodity market mechanism based on tâtonnement pricing. The market is organized on a hierarchical structure, with local and global markets. They considered individual and bundled resource allocation and also interdependency of resources.

Chunlin and Layuan [29] developed a commodity-market-like mechanism. The market is decomposed in grid resource and grid task agents. Grid resource agents are responsible for adjusting their own resource prices based on the excess demand. Grid task agents receive an endowment to complete their tasks. They maintain a list of resource prices and prepare demand requests that maximize their utility based on this price list. The utility functions that are maximized by the task agents use concrete parameters such as the time to complete a job and price. Each task agent receives a resource share proportional the amount it wants to pay.

Libra [97] uses the commodity market model for resource allocation in clusters. The system is composed by clients, a server and a set of nodes. Clients submit their jobs to the server, a central entity responsible for charging clients. Jobs are submitted with estimated runtime, deadline and the budget the client is willing to pay for it. The server uses the estimated runtime and deadline to calculate the cost of the job. The cost is compared with the budget to check whether the job can be accepted or not. Once a job is accepted, the server asks the nodes if any of them is able to finish the job in the deadline. Among the nodes that are able, the node with the lower load is chosen to run the job. The node then is responsible for the local scheduling of the job, assuring its completion by the deadline.

Nimrod/G [4] is a resource management and scheduling system for Grids. It supports deadline and cost-based scheduling. Although the native cost mechanism is static, GRACE is a middleware infrastructure that can be used by Nimrod/G. GRACE allows for the dynamic online negotiation for access to Grid resources at lower cost. It also allows deadline versus cost trade-off during scheduling. The architecture is composed of Trade Managers (TM) and Trade Servers (TS). A TM interacts with TS to negotiate for access to the resources. TM’s goal is to minimize the cost for resource users while TS’s goal is to maximize the profit for resource owners. TS uses pricing algorithms as define by resource owners, which may be driven by demand and supply.

2.3.3 Bargaining-based Systems

Only a small fraction of mechanisms has fully explored the bargaining economic model for resource allocation. In this section we will discuss the CATNETS initiative [8, 38, 93]. Other mechanisms can be found in Sim [100] and Braun et al. [15].

CATNETS is based on the Catallaxy approach, which is founded on the ideas of the Austrian economist Friedrich August von Hayek [48]. He understood markets as decentralized coordination mechanisms, as opposed to a centralized economy control. The concept assumes self-interested participants that try to maximize their own utility acting under incomplete information and bounded rationality. CATNETS divides the environment in two different markets: one for services and other for resources. The scenario is composed of a set of basic services (Service-Copies), a set of complex services (Clients) demanding basic services, and a set of resource services (Resources) capable of providing computational resources for basic services. Complex services act
as buyers in the service market. Basic services act as sellers in the service market and as buyers in the resource market. And, resource services act as sellers in the resource market.

The CATNETS allocation system is completely decentralized and distributed. The allocation process involves a two-step negotiation. Clients have to negotiate with service-copies for service provision and service-copies have to negotiate with resource agents for resource provision. Therefore, each agent (clients, service-copies and resources) is equipped with a negotiation mechanism. The mechanism is driven by a set of parameters that compose the negotiation strategy of the agent. The parameters are \( \text{priceNext} \), \( \text{weightMemory} \), \( \text{satisfaction} \), \( \text{acquisitiveness} \) and \( \text{priceStep} \). The parameter \( \text{priceNext} \) is used to modify the starting value of future negotiations. The parameter \( \text{weightMemory} \) defines the ratio of current and historic price information. The higher it is, the faster the agent will adapt to the current market information. The parameter \( \text{satisfaction} \) defines the probability of continuing in the negotiation. The parameter \( \text{acquisitiveness} \) defines the probability of making an unilateral concession in the following negotiation. And finally, the parameter \( \text{priceStep} \) is used to generate the concessions during the negotiation process. The parameter is defined as the percentage of the difference between the stated initial values of the parties.

2.4 Opportunities for Learning

In the previous section we have presented some examples of mechanisms based on the market concept. In this section we provide examples of how those mechanisms could be improved by the application of machine learning techniques and describe some mechanisms applying it.

2.4.1 Learning in Auctions

Most of the auction-based systems described in the previous section can be improved with the use of learning algorithms. Although there is no much room for enhancement at the auctioneer side, as they usually work with very strict rules, there are clear opportunities for the application of learning at the participants’ side: buyers and sellers.

The most visible opportunity is to use learning to guide the bidding and asking proce-
dures of the participants. None of the systems described above use it. In those systems, bids and asks are made based on either users’ expertise or simple arithmetic rules. For example, in GridMarket [25], bids and asks are define statically. Users set some parameters that are used to generate bids and asks according to the time elapsed in the auction. A similar approach is used in Popcorn [87].

Considering an environment composed of a centralized single auction, it is likely that users will present similar pattern of behaviours over different auction rounds. Therefore, an individual agent can apply learning and explore such patterns to obtain further gains from the allocation. The English auction model is the one that makes this task easier. In such a model, the best strategy is to bid less than the true valuation for the good. It is possible to win the auction by following this strategy, but it all depends on what the others bid. Therefore, strategic knowledge about the others can be rewarding. Participants in double, combinatorial and bid-proportional sharing auctions and contract-net protocol can also benefit from observing regularities. Vickrey auctions, however, offer a more difficult task. In this model, the best strategy is to bid the true valuation for the good. It guarantees that the price paid will be equal or lower than the valuation in case of winning. We could not see how learning could be used to improve such a mechanism in the single auction case.

An environment composed of multiple auctions opens more room for improvements because it is possible to explore not only the patterns in the auction level but also in the environment or society level. This is what Preist et al. [85] do. They developed an agent that constructs a belief function to model probabilities of closing prices in different auction houses. The function is used to coordinate the agent’s bids in the houses. The strategy improved the gains of the agent. Although the study was conducted for English auctions, we believe it may also be useful for Vickrey auctions. The agent may benefit from bidding its true valuation in the auction it believes will finish with the lowest prices. The probability of getting a better outcome using this strategy is not worse than using no learning at all.

We discussed above some opportunities for learning in the view of a single agent and how it can bring gains from the individual perspective, but what are the effects at the society level? Are we able to improve the allocation as a whole if all agents are learners? This is the direction of this thesis. For example, the study of Preist et al. [85] found that the market efficiency is enhanced with the number of agents using the belief function. However, the gains of the agents are worsened.
It is also important to comment that most of the theoretical formulations of auctions are lost when we extend the model to a multiple case, which is needed to reflect the necessity for decentralized allocation requested by large distributed systems. Therefore, if one model exhibits all the desired properties in the single auction environment, it does not necessarily imply that those properties will be hold in the multiple-auction environment. Learning can be a way to overcome this situation.

2.4.2 Learning in Commodity Markets

As in the auction case, most of the commodity-market-based systems described in the previous section can be improved with the use of learning algorithms. There are room for improvement in the market’s side and also in the participants’ side.

Starting by the market’s side, its main task is to define the price of the resources. Systems in which the price is derived from static formulas or is fixed can definitely be improved by learning. For example, some of the works described above say the price may be driven by demand and supply. LIBRA is one of those. The cost applied in Libra, however, is given by a simple formula which uses two constants define a priori. There is a clear chance for learning those constants on-the-fly. It is very likely that supply and demand will follow patterns, opening the opportunity to explore which configuration for the constants are more efficient. In addition, if the pattern changes over the time, a configuration define beforehand may produce poor results. Therefore, any adaptation of the configuration to the actual state of the market will be an improvement. On the other hand, it is difficult to evaluate the possibility of learning in equilibrium-based markets such as the ones used in Wolski et al. [126], Chunlin and Layuan [29], Wellman [125] and Subramoniam et al. [105]. This is due to the fact that the price is actually bounded to the rules of the mechanism.

The opportunities for learning are clearer at the participants’ side. Their task is to decide when and how much to sell and buy. In G-commerce [126], for example, producers define their supply offers based on the current price of the resources in comparison to the average price over the past and consumers define their demand requests based on the current price, the size of the tasks they need to run and the budget available. It would be interesting to investigate what happens if those strategies are changed: what if they learn it? A hypothetical situation would be describing the preferences of sellers and buyers by utility functions, and let a learning algorithm decide the buying and sel-

ling strategies based on it. This is the direction we have taken in this thesis to study the presence of strategic agents in the Iterative Price Adjustment mechanism.

2.4.3 Learning in Bargaining

The bargaining economic model offers, perhaps, the largest opportunities for application of machine learning. Since the allocation emerges from local negotiations between participants, the negotiation strategies employed by them have high impacts not only on the individual gain of a particular participant but also on the final social gains obtained from the set of negotiations.

The CATNETS approach described in Section 2.3.3 is an example of mechanism in which learning is used to improve the participants’ strategies. While the experiments presented in Ardaiz et al. [8] have used a pre-defined set of values for the strategy parameters, in Schnizler et al. [93] an evolutionary approach has been applied to allow for the autonomous adaptation of them. For this, the authors used the Smith Taylor Decentralized Evolutionary Algorithm (STDEA). This algorithm relies on decentralized communication and fitness evaluation, using locally available data. Every agent sends its average income (fitness and its genes (genotype) to all agents of the population after it has carried out a certain number of negotiations with this genotype. A receiver agent decides using a blindness probability, whether the information is evaluated, avoiding premature unification of the genotype. Sender and recipient remain anonymous. If a certain maturity threshold of received information is exceeded, the agent replaces his old genotype with the evolved version. Evaluation, selection, recombination and mutation phases occurs as in normal genetic algorithms.

CATNETS is particularly appealing because of its high decentralization. One of its drawbacks, however, is the advertisement of private information by the agents. In order to evolve the society, agents have to share their strategy with other agents, which is not natural in truly competitive markets. To explore ideas to eliminate it would be very interesting.
2.5 Summary

In this chapter we have provided the fundamentals of Market-based Resource Allocation. We have described the main economic models used in resource allocation systems and given examples of mechanisms applying them. We have also discussed opportunities for the application of learning in the models together with examples of existing mechanisms applying it.

In summary, the three economic models described in the chapter (i.e. auction, commodity market and bargaining) present features that can be explored by the application of machine learning techniques. As a consequence, most market-based resource allocation mechanisms can be improved with the use of learning. Yet, research in the area is still very incipient. In this thesis we contribute to this effort by investigating the application of learning in commodity market mechanisms based on economic equilibrium concepts.
Chapter 3

Reinforcement Learning

This chapter provides the basics of Reinforcement Learning. It first presents an informal description of the problem, followed by a more formal definition using the framework of Markov Decision Processes. It then reviews the fundamental classes of solution methods and addresses aspects affecting the use of these methods in practical problems. Finally, it extends the discussions to the case of Multiagent Reinforcement Learning, presenting the rationale for the choice of learning algorithm applied in the thesis.

3.1 Introduction

There are basically three major paradigms for machine learning: supervised learning, unsupervised learning and reinforcement learning. They differ from each other mainly in respect to the type of the information provided to the learner. In the supervised case, it receives a training set containing examples of solved problems, mapping correct outputs to inputs. The learning task is to obtain a function that abstracts these examples and generalizes them to unseen data. In the unsupervised approach, on the contrary, no examples are provided. The learner receives a set of unclassified data and has to identify features and relationships between the members of it. The goal is usually to classify the data in a number of different classes.

Reinforcement Learning is located between those two approaches. Instead of receiving information on the correct choices, as in the supervised case, a RL learner interacts
3.2. The Learning Problem

with the environment and receives feedbacks in the form of quantitative assessments of its performance. The learning task is to map situations the learner encounters in the environment to the actions to be taken when in those situations so as to maximize a long-term measure of these feedbacks. Reinforcement Learning is learning from interaction to achieve a goal.

Neither the supervised nor the unsupervised approaches are adequate to solve the problem of developing strategic behaviour proposed in this thesis. We cannot apply the supervised approach because it is impossible to provide examples beforehand to the participants without complete knowledge of the market’s characteristics, including the preferences of the other participants. For this same reason, the participants must learn from interaction in the market mechanism and based on their own experience, without the assistance of an external supervisor. Moreover, the individual objective for the participants should be to develop an strategy to act in the market, which excludes the unsupervised approach and lead us to Reinforcement Learning.

The discussions herein are mainly based on the book of Sutton and Barto [108]. Please see that book for a more detailed discussion of most topics covered here. Another interesting source is the survey by Kaelbling et al. [58]. For the case with multiple learners, the surveys by Buşoniu et al. [17] and Shoham et al. [99] are excellent resources. For more broad aspects of multiagent learning, the surveys by Stone and Veloso [104] and Panait and Luke [79] are recommended.

3.2 The Learning Problem

Reinforcement Learning studies the problem faced by an active decision-making entity that has to learn to achieve a goal from interaction with its environment. The entity is not told which decisions to take nor is provided with examples of how to solve the problem, as in most forms of machine learning. Instead, it receives feedbacks on the quality of its performance through scalar reward signals. The entity’s learning task is to map situations it encounters during the interaction with the environment to the decisions to be taken when in those situations so as to maximize some long-term measure of these reward signals.

In this section we informally present the elements and the relevant aspects of the Reinforcement Learning problem. A more formal description is presented in Section 3.3.
3.2.1 Elements of the Problem

In Reinforcement Learning, the decision-making entity is usually called an agent and the entity it interacts with is the environment. The agent interacts with the environment by performing actions and the environment interacts with the agents by reacting to those actions, presenting new situations to the agent and giving rise to the reward values. Figure 3.1 illustrates this concept.

![Agent-environment interaction in Reinforcement Learning.](image)

The interaction proceeds as follows. At each time step $t$, the agent receives a signal from the environment indicating the current state $s_t$ and selects an action $a_t$. Once this action is performed, it changes the environment state to $s_{t+1}$, generating a reward signal $r_{t+1}$ that is used by the agent to evaluate the quality of the decision.

3.2.1.1 Policies

The behaviour of the agent during the interaction with the environment is defined by its policy, usually denoted $\pi$. A policy is a mapping from perceived states to the actions to be taken when in those states. The objective of a Reinforcement Learning agent is to find a policy that maximizes a long-term measure of the rewards.

Policies can be deterministic or stochastic. In the deterministic case, the agent always selects the same actions at the same states. In the stochastic case, on the other hand, the agent selects the actions based on a probability distribution. In this case, $\pi_t(s, a)$ denotes the probability of selecting action $a$ at time $t$ given the state $s$.

If the policy does not depend on time, i.e., if the action is always selected from the same probability distribution at the same states, independent of time, then the policy is called stationary.
3.2. The Learning Problem

3.2.1.2 Rewards

The rewards are given by the reward function, a mapping from environment states to numerical reward signals indicating the immediate desirability of those states. To some extent, the reward function formalizes the idea of goal in the Reinforcement Learning problem. It should be defined in such a way that by maximizing a particular function of the rewards the learner will achieve the goal.

As policies, reward functions can be deterministic or stochastic. In the formal description presented in Section 3.3, we consider only deterministic reward functions, meaning that the immediate reward received for achieving a particular state is always the same.

3.2.1.3 Returns

The objective of the agent is usually to accumulate the maximum amount of reward. In this case, the long-term measure that the agent tries to maximize, called expected return, is the sum of the rewards it expects to accumulate in the future. If the learning task has a finite length known by the agent a priori, then the total cumulative reward can be used to evaluate it:

\[ R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T = \sum_{k=0}^{T} r_{t+k+1} \]  

(3.1)

where \( T \) is a finite time step. In the general case, however, the length is not known or is infinite. In these situations, the use of the discounted cumulative reward is more sound:

\[ R_t = \gamma^0 r_{t+1} + \gamma^1 r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \]  

(3.2)

where \( \gamma \in [0, 1] \) is the discount rate preventing the sum to become infinite. The discount rate also represents a measure of the relative importance of future rewards. A reward received \( k \) steps in the future will be worth \( \gamma^{k-1} \) times of what it would be worth if it were received immediately. In this context, the discount rate is used to define how far in the future the agent “sees”. If \( \gamma \) is set to 0, the agent sees only the
next time step and will try to maximize only immediate rewards. On the other hand, as \(\gamma\) approaches 1, the agents sees further in the future and will give strong importance to future rewards.

### 3.2.2 Delayed Rewards

An important characteristic of Reinforcement Learning is the existence of delayed rewards: rewards that are received as consequence of actions taken in the past. One aspect evidenced by the issue is that the agent must be able to learn which of its actions are desirable based on rewards that can take place arbitrarily far in the future. Another aspect is that the agent may have to sacrifice short-term rewards in favour of larger rewards that may be received in the future.

RL solves the problem of delayed rewards through the estimation of value functions (see Section 3.3.2). Roughly speaking, a value function indicates the long-term desirability of states or state-action pairs based on long-term measures of the rewards received from that state.

### 3.2.3 Exploration versus Exploitation

Another distinguishing feature of Reinforcement Learning is the trade-off between exploration and exploitation. In order to learn optimal policies, the agent needs to explicitly explore the environment. At the same time, it also needs to reinforcement those strategies it already knows to be good. To find the correct balance between these two aspects is a ongoing research topic in the area. Some mechanisms for it will be reviewed in Section 3.6.

### 3.3 Formal Model

In this section we provide a more formal description of the Reinforcement Learning problem using the formalism of Markov Decision Processes (MDPs) [50, 86].
3.3. Formal Model

3.3.1 Markov Decision Process

Formally, a Reinforcement Learning Problem can be represented as a Markov Decision Process.

**Definition 3.1** A finite Markov Decision Process is a tuple \((S, A, T, R)\), where

- \(S\) is a finite set of states
- \(A\) is a finite set of actions
- \(T\) is a state transition function \(S \times A \times S \rightarrow [0, 1]\)
- \(R\) is a reward function \(S \times A \rightarrow \mathbb{R}\)

In Definition 3.1, \(S\) define the set of states that the environment can assume; \(A\) define the set of actions available to the agent; \(T\) define how the environment changes its state in respect to the actions performed by the agent, with \(T(s'|s, a)\) denoting the probability of making a transition from state \(s\) to \(s'\) when action \(a\) is selected; and \(R\) define the rewards the agent will receive as a result of the state transitions.

It is assumed that the dynamics of the system obeys to the Markov property, i.e., that the reward signal and the next state of the environment, given by \(R\) and \(T\), depend only on the current state and the current action selected by the agent. In other words, the dynamics of a MDP is independent of the history of the process.

**Definition 3.2** Given all possible values of past events \(s_t, a_t, r_t, \ldots, r_1, s_0, a_0\), a Reinforcement Learning task contains the Markov property and is a Markov Decision Process if and only if

\[
Pr\{s_{t+1} = s', r_{t+1} = r|s_t, a_t\} = Pr\{s_{t+1} = s', r_{t+1} = r|s_t, a_t, r_t, \ldots, r_1, s_0, a_0\}
\]

For the Markov property to hold, the description of the environment states must fully capture all the information that can influence the future evolution of the process. That is, the description of the current situation makes it unnecessary to know about the system’s past. It is important to emphasize that the Markov property is not a feature
of the real process, but of the model used to describe it. Any state signal can be made detailed enough to ensure that the description of the current state contains all the aspects that are relevant for the prediction of future state-transitions and rewards.

### 3.3.2 Value Functions

As in the RL problem, the goal in a MDP is usually to maximize the accumulated reward. For this reason, decisions are not taken with basis on immediate rewards but on long-term measures represented by value functions. Two types of value functions are commonly used: the state-value function and the action-value function. The state-value function associates states of the environment with their expected returns when a given policy \( \pi \) is followed. Informally, the value of a state indicates how good it is for the agent to be in a particular state if policy \( \pi \) is followed, i.e., how much reward the agent can expect to accumulate from that state by following the policy \( \pi \). The action-value function uses a similar concept but works with state-action pairs. In this case, the value of a state-action pair indicates how much reward the agent can expect to accumulate when a particular action is chosen at that state and a policy \( \pi \) is followed afterwards.

The state-value function is defined as follows.

**Definition 3.3** The value of a state \( s \), given a policy \( \pi \), denoted \( V^\pi(s) \), is the expected return when starting from state \( s \) and following the policy \( \pi \) thereafter:

\[
V^\pi(s) = E_\pi \{ R_t | s_t = s \} \tag{3.3}
\]

where \( V^\pi \) is the state-value function for policy \( \pi \).

If the agent’s goal is to accumulate the maximum amount of reward and the discounted cumulative reward (Equation 3.2) is applied, then the value of state \( s \) can be rewritten as:

\[
V^\pi(s) = E_\pi \{ R_t | s_t = s \} = E_\pi \{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \} \tag{3.4}
\]

At this stage, it is necessary to highlight the relationship between rewards and state
values: while the rewards define what is good for the agent in the immediate sense, it is the state values that define what is good in the long term. Therefore, the agent should take decisions based on the values of the states and not on the immediate reward returned by them. The goal should be to bring about those states with highest value because they are the ones that return the highest amount of reward over the lifetime of the agent.

Similar to the state-value function, the action-value function can be defined as follows.

**Definition 3.4** The value of action $a$ in state $s$, given a policy $\pi$, denoted $Q^\pi(s, a)$, is the expected return when starting in state $s$, taking action $a$ and following the policy $\pi$ thereafter:

$$Q^\pi(s, a) = E_\pi\{R_t | s_t = s, a_t = a\}$$  (3.5)

where $Q^\pi$ is the action-value function for policy $\pi$.

As in $V^\pi$, if the agent’s goal is to maximize the discounted cumulative reward (Equation 3.2), then the value of action $a$ in state $s$ is expressed as:

$$Q^\pi(s, a) = E_\pi\{R_t | s_t = s, a_t = a\} = E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$  (3.6)

### 3.3.3 Bellman Equations

One important property of the value functions used in MDPs is that they obey to recursive relationships between the values of successive states. For the state-value function, given any policy $\pi$, the value of a starting state $s$ can be written in terms of the expected values of the next reward and state $s'$ as follows.
Chapter 3. Reinforcement Learning

\[ V^\pi(s) = E_\pi \{ R_t|s_t = s \} \]
\[ = E_\pi \{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s \} \]
\[ = E_\pi \{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2}|s_t = s \} \]
\[ = \sum_a \pi(s,a) \sum_{s'} P_{r}(s'|s,a)[r(s,a) + \gamma E_\pi \{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2}|s_{t+1} = s' \}] \]
\[ = \sum_a \pi(s,a) \sum_{s'} P_{r}(s'|s,a)[r(s,a) + \gamma V^\pi(s')] \quad (3.7) \]

Equation 3.7 is known as the Bellman Equation for \( V^\pi \). It states that the value of any starting state \( s \) must equal the (discounted) value of the next expected state \( s' \) plus the expected next reward. This recursive relationship forms the basis of several methods to compute, estimate, approximate and learn \( V^\pi \), as shall be seen in the next sections.

### 3.3.4 Optimal Policies and Value Functions

As presented above, a policy’s value functions assign to each state, or state-action pair, the return expected from that state, or state-action pair, when the policy is followed. Therefore, it enables the evaluation of different policies in terms of their expected returns, defining a partial ordering relation over the set of policies.

**Definition 3.5** A policy \( \pi \) is better than or equal to a policy \( \pi' \) if and only if its expected return is greater than or equal to that of \( \pi' \) for all states

\[ \pi \geq \pi' \text{ if and only if } V^\pi(s) \geq V^{\pi'}(s), \forall s \in S \]

A policy that is better than or equal to all the others is an optimal policy, denoted \( \pi^* \). An optimal policy indicates the best action to take in each state in the sense that following that policy will lead to the highest possible expected return. There may exist more than one optimal policy but they all share the same state-value function.
3.3. Formal Model

Definition 3.6 The optimal state-value function, denoted $V^*$, is defined as

$$V^*(s) = \max_{\pi} V^\pi(s), \forall s \in S$$

Similarly, optimal policies also share the same optimal action-value function.

Definition 3.7 The optimal action-value function, denoted $Q^*$, is defined as

$$Q^*(s, a) = \max_{\pi} Q^\pi(s, a), \forall s \in S, \forall a \in A$$

The optimal action-value function can also be expressed in terms of the optimal state values:

$$Q^*(s, a) = E\{r_{t+1} + \gamma V^*(s_{t+1})|s_t = s, a_t = a\} \quad (3.8)$$

Because $V^*$ is the value function of a policy, it also obeys to the recursive relationship presented in the previous section. In this case, however, the Bellman Equation can be written without references to any specific policy and is called the Bellman Optimality Equation. This equation declares that the value of a state under an optimal policy must equal the expected return for the best action from that state:

$$V^*(s) = \max_{a \in A} Q^\pi(s, a)$$

$$= \max_a E_{\pi}\{R_t|s_t = s, a_t = a\}$$

$$= \max_a E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\}$$

$$= \max_a E_{\pi}\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2}|s_t = s, a_t = a\}$$

$$= \max_a E\{r_{t+1} + \gamma V^*(s_{t+1})|s_t = s, a_t = a\}$$

$$= \max_a \sum_{s'} Pr(s'|s, a) [r(s, a) + \gamma V^*(s')] \quad (3.9)$$

Likewise, the Bellman optimality equation for $Q^*$:
Chapter 3. Reinforcement Learning

\[ Q^*(s, a) = E\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a \} \]

Suppose that we have the value functions for all the possible policies. Then the optimal policy \( \pi^* \) can be extracted simply by, at each state, selecting the action with the optimal value function (\( V^* \) or \( Q^* \)). If only the state values are known, then the agent has to perform a one-step-ahead search to evaluate which actions lead to the state with highest expected return. Knowing the action values makes the process easier. In this case there is no need for the additional search and the agent can select the action with the highest value directly. Effectively, the action-value function caches the results of all the one-step-ahead searches.

### 3.4 Solution Methods

In this section we discuss the fundamental classes of methods for finding or approximating optimal policies in Markov Decision Processes and Reinforcement Learning problems. They are: Dynamic Programming, Monte Carlo and Temporal-Difference Learning.

#### 3.4.1 Dynamic Programming

Dynamic Programming is the name given to a class of well-developed mathematical methods that can be used to compute optimal policies in Markov Decision Processes. These methods, however, require a complete and accurate model of the environment, which limits their utility in Reinforcement Learning problems. Nevertheless, they are still important as they have served as inspiration for most Reinforcement Learning algorithms and provide the theoretical foundation for understanding them.

Dynamic Programming algorithms are heavily based on the recursive principles of the Bellman Equations. In fact, they turn the Bellman Equations into update rules for improving approximations of the value functions. As such, the algorithms are essentially iterative processes in which each state value is built upon the values of the successor states, a procedure known as bootstraping.
3.4. Solution Methods

3.4.2 Monte Carlo Methods

Unlike Dynamic Programming, Monte Carlo methods do not require a perfect model of the environment. The value functions are estimated from experience obtained through sample sequences of states, actions and rewards. Therefore, Monte Carlo is more suitable than Dynamic Programming to solve Reinforcement Learning problems.

Monte Carlo methods calculate $V^\pi$ by running several simulated trajectories of the policy $\pi$ and averaging the cumulative returns obtained. Each simulated trajectory is called an episode. The idea is that, as the number of episodes goes to infinity, the average of the returns experienced will converge to a good approximation of the actual expected return. Such an approach is fundamental to all Monte Carlo methods.

The most common examples of Monte Carlo methods are the First-visit and the Every-visit methods. Suppose that we want to estimate $V^\pi(s)$ given a series of episodes obtained by following $\pi$ and passing through $s$. Each occurrence of $s$ in an episode is called a visit to $s$. The first occurrence of $s$ is the first-visit. While the first-visit method estimates $V^\pi(s)$ using the average return following only first-visit to $s$, the every-visit method estimates using all the visits. The methods have slightly different theoretical properties but they both converge to $V^\pi(s)$ as the number of visits (or first-visits to $s$) goes to infinity.

3.4.3 Temporal-Difference Learning

Temporal Difference combines ideas from both Dynamic Programming and Monte Carlo. It is similar to Dynamic Programming because the estimation of value functions are based on the values of the successor states, using the recursive principles of the Bellman Equations. And similar to Monte Carlo because the estimations are obtained by experience from sample sequences of states, actions and rewards, without requiring the model of the environment. Thus, Temporal Difference combines the bootstrapping from Dynamic Programming with the sampling of Monte Carlo.

Algorithms based on Temporal Difference form the core of Reinforcement Learning. One reason for this is that they do not require a model of the environment, which is a clear advantage over Dynamic Programming. Another reason is that they can update the value functions after one single time step rather than having to wait until the end.
of the episode, which is an advantage over Monte Carlo methods and quite useful in continuing or very long learning tasks.

### 3.5 RL Algorithms

In this section we will briefly introduce the two fundamental algorithms for obtaining optimal policies using Temporal Difference: Sarsa and Q-learning. Before that, however, it is interesting to describe the TD(0) algorithm, which illustrates how Temporal Difference works for estimating value functions.

#### 3.5.1 The TD(0) Algorithm

The general idea of TD(0) is as follows. On each time step, the agent selects an action and receives a reward and the next state of the environment. Knowing the next state, the agent can access its current estimated value and, together with the received reward, use it to update the value of the current state. Therefore, the reward and the current value of the next state form a sample of the expected return for the current state. The idea is illustrated in the following formula:

\[
V(s_t) \leftarrow V(s_t) + \alpha [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]
\] (3.10)

In the formula, \(\alpha\) is called the **learning rate** and serves as a step-size parameter. The formula’s idea is to move the value of the state slightly towards its expected return using as immediate target the value of \(r_{t+1} + \gamma V(s_{t+1})\), that is, the value of the immediate reward plus the current estimation of the immediate next state’s value. The complete algorithm for estimating the \(V^\pi\) using TD(0) is presented in Algorithm 3.1.

TD(0) converges to \(V^\pi\) for any fixed policy \(\pi\), in the mean if the learning rate is sufficiently small and kept constant, and with probability 1 if it is decayed according to usual stochastic approximation conditions.

It is not very useful to know the state values if the transitions between the states are not known, that is, if the agent does not have access to the model to identify what action to take to go to the most beneficial next state. One alternative for this problem is to
3.5. RL Algorithms

Data: \( \pi \), the policy to be evaluated

Result: \( V \approx V^\pi \)

1. For all \( s \in S \) do
   2. \( V(s) \leftarrow 0 \)
   3. end

4. For each episode do
   5. Observe state \( s \)
   6. while \( s \) is not terminal do
      7. \( a \leftarrow \pi(s) \), (the action given by \( \pi \) for the state \( s \))
      8. Take action \( a \); observe reward \( r \) and next state \( s' \)
      9. \( V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)] \)
     10. \( s \leftarrow s' \)
   11. end
   12. end

Algorithm 3.1: TD(0) for estimating \( V^\pi \)

estimate the action values. In this case, it is also important to take into account the exploration aspect so all the actions in a particular state are visited enough times. In the next subsections we will present two algorithms that implement these ideas: Sarsa and \( Q \)-learning.

3.5.2 The Sarsa Algorithm

Sarsa [108] approximates \( Q^\pi \) by considering transitions from state-action to state-action pairs. Its update rule is quite similar to that of TD(0), with the difference that, instead of using state values, the rule employs action values:

\[
Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]
\]  

(3.11)

where \( \alpha \in [0, 1] \) is the learning rate. The value of the current action-state pair is updated using the value of the next state-action pair. This update uses every element of the quintuple of events \((s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})\) giving rise to the name Sarsa. The complete algorithm is shown in Algorithm 3.2.

Sarsa is a type of on-policy algorithm, meaning that the policy that is being evaluated
and improved is the same policy that is being used to select the actions. For this reason, its convergence properties depend on the mechanism used to implement this policy, that is, the mechanism used to select the actions (lines 6 and 9 of the algorithm). Such a mechanism must make sure that the policy converges to the greedy policy in the limit. For this, one can use the $\varepsilon$-greedy mechanism (see Section 3.6) and decay $\varepsilon$ as $\varepsilon = 1/t$. As long as the policy converges to the greedy one and all state-action pairs are visited an infinite number of times, then Sarsa will converge to the optimal action-value function and policy with probability 1.

### 3.5.3 The $Q$-learning Algorithm

On contrary to Sarsa, $Q$-learning [122] is an off-policy method, meaning that the policy used to generate the agent’s behaviour is different from that that is being evaluated or improved. Therefore, $Q$-learning is able to learn the optimal $Q$-values independent of the exploration policy that is being followed. For this, it updates the current estimation with the value of the best action for the next state:
3.6 Action Selection Mechanisms

An important component of Reinforcement Learning algorithms is the action selection mechanism used to harmonize the trade-off between exploration and exploitation.

There are many possible mechanisms to address the problem and no clear results indicating the best alternative. Instead, the results indicate that the performance of different

\[ Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \]  

(3.12)

where \( \alpha \) is the learning rate. Because the rule uses the value of the best action for the next state, the algorithm is able to approximate \( Q^* \) directly, without requiring the convergence of the control policy to the greedy one, as in Sarsa.

The complete Q-learning algorithm is presented in Algorithm 3.3. It is proven to converge with probability 1 to the optimal Q-values if the environment is stationary and the learning rate, \( \alpha \), is decayed appropriately [55, 108, 113]. In a stationary environment, the probabilities of making state transitions \( T \) and receiving specific reinforcement signals \( R \) do not change over time.

<table>
<thead>
<tr>
<th>Result: ( \pi \approx \pi^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>For all ( s \in S, a \in A ) do</td>
</tr>
<tr>
<td>( Q(s, a) \leftarrow 0 )</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>For each episode do</td>
</tr>
<tr>
<td>Observe state ( s )</td>
</tr>
<tr>
<td>while ( s ) is not terminal do</td>
</tr>
<tr>
<td>Choose ( a ) from ( s ) using policy derived from ( Q ) (e.g. ( \epsilon )-greedy)</td>
</tr>
<tr>
<td>Take action ( a ); observe reward ( r ) and next state ( s' )</td>
</tr>
<tr>
<td>( Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)] )</td>
</tr>
<tr>
<td>( \pi(s) \leftarrow \arg \max_a Q(s, a) )</td>
</tr>
<tr>
<td>( s \leftarrow s' )</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

Algorithm 3.3: Q-learning Algorithm
mechanisms is highly dependent of the problem. In this section we will present the two most famous examples: the $\epsilon$-greedy and the Boltzmann action selection mechanisms.

### 3.6.1 $\epsilon$-greedy Exploration

The $\epsilon$-greedy mechanism selects actions based on the current configuration of the action-value function. At each state, the best action is selected with probability $1 - \epsilon$ and a random action with probability $\epsilon$. As such, the mechanism define a semi-uniform probability distribution over the set of actions available to the agent at each state.

According to this mechanism, the probability of playing action $a$ in state $s$ is:

$$Pr(a|s) = \begin{cases} (1 - \epsilon) + (\epsilon/n), & \text{if } Q(s, a) \text{ is currently the highest} \\ \epsilon/n, & \text{otherwise} \end{cases} \quad (3.13)$$

where $n$ is the number of actions in the set.

### 3.6.2 Boltzmann Exploration

The Boltzmann mechanism is also based on the current configuration of the action-value function. This mechanism, however, takes into account the relative values of the state-action values. The probability that an action is selected depends on how high it is compared to the other state-action values. The higher the value in comparison to the others, the more likely that this action will be taken.

According to this mechanism, the probability of playing action $a$ in state $s$ is:

$$Pr(a|s) = \frac{e^{\tau Q(s,a)}}{\sum_{b=1}^{n} e^{\tau Q(s,b)}} \quad (3.14)$$

where $n$ is the number of actions in the set and $\tau$ is a parameter called the temperature that controls the degree of exploitation: the highest the value, the highest the probability of exploitation.


3.7 Practical Reinforcement Learning

All the methods presented in the previous section rely on tabular representations of the state and action spaces, making their application not viable in large problems. In addition, they are based on discrete sets, while most real applications have continuous spaces. In such cases, there is effectively an infinite number of states and actions and the probability of returning to exactly the same state-action ever again is negligible.

For the reasons above, practical applications of RL algorithms are typically coupled with function approximation mechanisms. The general idea is to represent a value function as a parameterized function of some parameter vector $\tilde{\theta}$ and use supervised learning to update it. The value function then becomes totally dependent on $\tilde{\theta}$ and will vary from time step to time step only as $\tilde{\theta}$ varies. The number of parameters of $\tilde{\theta}$ is typically much lower than the number of states and changing one of its parameters will affect many states.

The most common approach is based on linear function approximation. In this case, the value function is approximated as a weighted sum of a set of features $\phi_1, \cdots, \phi_n$ constructed from the state space:

$$ V_t(s) = \tilde{\theta}_t^T \phi_s = \sum_{i=1}^{n} \theta_t(i) \phi_s(i) $$

Gradient-descent methods can be applied to adjust the parameter vector. The goal is usually to minimize the mean-squared error (MSE) between the approximate value function $V_t$ and the true value function $V^\pi$ over the training examples. The update rule for minimizing MSE is:

$$ \tilde{\theta}_{t+1} = \tilde{\theta}_t - \frac{1}{2} \alpha \nabla_{\tilde{\theta}} [V^\pi(s_t) - V_t(s_t)]^2 $$

$$ = \tilde{\theta}_t + \alpha [V^\pi(s_t) - V_t(s_t)] \nabla_{\tilde{\theta}} V_t(s_t) $$

where $\alpha$ is a positive step-size parameter and $\nabla_{\tilde{\theta}} V_t(s_t)$ is the gradient of $V_t(s_t)$ with respect to $\tilde{\theta}_t$.

The features, also referred to as basis functions, can be constructed in many ways.
Sutton and Barto [108] describe the use of Coarse Coding, Tile Coding (CMACs [5]), Radial Basis Functions and Kanerva Coding. Many other approaches have been proposed, for example Normalized Gaussian Networks [77], Fourier Basis [62], Polynomial Basis [65] and Proto-value functions [75, 76]. Constructing the appropriate feature set, however, is critical for most problems and requires significant design effort and problem insight.

Another common approach for approximation in RL is the use of non-linear function approximators such as Neural Networks (NN) [110, 120]. A NN consists of a graph of nodes, called neurons, connected by weighted links. The neurons receive input values and produce output values. The mapping from input to output depends on the weight of the links, which can be updated using, for example, the backpropagation algorithm [88]. A very successful application of RL with NNs is TD-Gammon, a backgammon player by Tesauro [110]. TD-Gammon managed to reach the level of human masters in a game where the number of possible positions is estimated at $10^{20}$. A value function over such a number of states cannot be stored in a lookup-table.

Although being able to handle large state and action spaces, NNs are usually rejected in favour of linear approximators schemes. The reason is that NNs are much more tricky to handle and design than linear approximation methods. NNs are also subject to the problem of local-optima convergence. Additionally, the general convergence properties of RL with non-linear function approximation is mostly unclear.

In general, the application of RL with function approximation requires significant design effort and problem insight. The choices of the parametric approximation architectures and parameter adjustment methods are crucial factors for a successful approximate algorithm [65]. Poor design choices can result in estimates that diverge from the optimal value function and agents that perform poorly [53, 54].

### 3.8 Multiagent Reinforcement Learning

So far we have only considered the case where a single learning entity inhabits the environment. However, in the market-based resource allocation scenario addressed in this thesis, the environment may be composed by multiple learning entities. Therefore, it is necessary to address this case.
In this section we review the main ideas of Multiagent Reinforcement Learning (MARL). We comment on the main challenges faced by MARL, present a formal model for the problem and give a general overview on the algorithms proposed to solve it.

### 3.8.1 Challenges

One of the main challenges in MARL is the non-stationarity generated by multiple learners [17, 79]. Each time one agent learns a new strategy, it changes the environment as perceived by the other agents, possibly invalidating the very strategies learnt by them and making a series of new adaptations necessary. From a single-agent perspective, the phenomenon generates a dynamic environment and violates the Markov property.

Another challenge is the curse of dimensionality, which is related to the complexity of the state-action space that the agents have to explore. The complexity is exponential in both the number of states and actions and the number of agents. It is exponential in the number of states and actions because the algorithms typically estimate values for every possible state-action pair. And it is exponential in the number of agents because each agent adds its own states and actions to the problem.

Another important challenge regards to the difficult of specifying a good goal for MARL algorithms. Because the optimality of the agents’ strategies are interrelated, specifying a good goal is typically a difficult task and an ongoing research topic. Several criteria have been proposed [17]. They include aspects of stability and adaptation.

Stability is related to the convergence of the learning dynamics to some stationary set of strategies. The convergence of the strategies to a Nash Equilibrium is frequently used to assess this criterion. However, other concepts, such as the Correlated Equilibrium [45] and the so-called “AI Agenda”, have also been proposed, in particular because of concerns related to the usefulness of the Nash Equilibrium in MARL [45, 99, 123].

Adaptation is related to the maintenance or improvement of performance as the other agents change their strategies. A frequently used concept for adaptation is rationality\(^1\), define by Bowling and Veloso [14] as the requirement of convergence to a best response when the other agents are stationary. Also applied is the concept of no-regret, which requires the algorithm to perform as well as any stationary strategy [13].

---

\(^1\)Please note that the term has a different meaning in economics.
Chapter 3. Reinforcement Learning

Relationships between stability, adaptation and other properties of learning algorithms are also discussed in the literature. Buşoniu et al. [17] provide a more detailed discussion on the subject.

3.8.2 Formal Model

Formally, a Multiagent Reinforcement Learning Problem can be represented as a Stochastic Game (SG) [96]:

**Definition 3.8** A Stochastic Game is a tuple \((n, S, A_1, \ldots, A_n, T, R_1, \ldots, R_n)\), where

- \(n\) is the number of agents
- \(S\) is the set of states
- \(A_i\) is the set of actions available to agent \(i\), with \(A\) being the joint action space \(A_1 \times \cdots \times A_n\)
- \(T\) is the transition function \(S \times A \times S \rightarrow [0, 1]\)
- \(R_i\) is the reward function of agent \(i\); \(S \times A \rightarrow \mathbb{R}\)

SGs can be considered as an extension of the MDP framework to multiple agents, where the next state and one agent’s reward depend on the joint actions of all agents. Note that the Definition 3.8 is very similar to Definition 3.1 of a MDP, except that both \(T\) and \(R\) are defined over the joint-action space. SGs are also an extension of matrix games to multiple states. Each state is viewed as a matrix game and the payoffs (rewards) for each joint-action \(a\) is determined by \(R_i(s, a)\). The transitions between the states (matrix games) is determined by \(T(s, a)\).

As discussed in the previous section, one important aspect of the multiagent problem, and of SGs, is that one agent’s policy can only be evaluated in the context of the policies of all the other agents. Therefore, it is often useful to evaluate the policy’s optimality in terms of a best response to the other agents’ policies:
\textbf{Definition 3.9} The \textit{best-response function} for agent $i$, $BR_i(\pi_{-i})$, is the set of all policies that are optimal given the other agents’ joint policy $\pi_{-i}$. A policy $\pi_i$ is optimal given $\pi_{-i}$ if and only if,

$$\forall \pi'_i \in \Pi_i \quad V_i(\pi_i, \pi_{-i}) \geq V_i(\pi'_i, \pi_{-i})$$

where $\Pi_i$ is the set of policies for agent $i$ and $V_i$ is the expected reward for playing $\pi_i$ given $\pi_{-i}$.

If all the agents are playing best-response policies to each other, then there is a \textit{Nash Equilibrium} (NE):

\textbf{Definition 3.10} A \textit{Nash Equilibrium} is a joint policy, $\pi_i = 1, \ldots, n$, with

$$\forall i = 1, \ldots, n \quad \pi_i \in BR_i(\pi_{-i})$$

An important result shown by Fink [39] guarantees that every $n$-agent general-sum discounted SG possesses at least one NE in stationary policies. Recall that a stationary policy is one that depends only on the current state (see Section 3.2.1.1).

\subsection*{3.8.3 Learning Algorithms}

The several MARL algorithms proposed in the literature can be classified according to the type of their targeted task [17]. This classification gives rise to fully cooperative, fully competitive and mixed algorithms. Fully cooperative algorithms are tailored to problems in which the agents share the same global reward function and the goal is to maximize the common discounted return. \textit{Joint Action Learners} [30], \textit{Frequency Maximum Q Value} [60], Team-Q [74], Distributed-Q [69] and \textit{Optimal Adaptive Learning} [121] are examples of algorithms in this class. Fully competitive algorithms, on the other hand, are tailored to solve zero-sum games, where the gains of one agent corresponds to the losses of the other. An example of such an algorithm is the Minimax-Q [72].

The market-based resource allocation scenario addressed in this thesis is best described as a general-sum game. In these games, the relationships between the agents’ payoffs
are arbitrary and what one agent gains is not necessarily what the other loses. Therefore, neither the fully cooperative nor the fully competitive algorithms are suitable to solve our problem, which lead us to algorithms in the mixed class.

Unfortunately, most algorithms in the mixed class are also not suitable. For instance, a significant number of them have been developed for and are only applicable in single-state scenarios, where the learning problem can be represented as a matrix game. Examples of algorithms in this case are Fictitious Play [16], MetaStrategy [84], Friend-or-Foe [73], ReDValLeR [10], IGA [101], WoLF-IGA [14], GIGA [131], GIGA-WoLF [13], AWESOME [32] and Hyper-Q [111]. Other mixed algorithms assume that the agents have some knowledge over the game structure. For instance, the agents must know their payoff matrix, the actions of the opponents or the payoffs of the opponents. These algorithms are also not suitable to our problem since no such information is realistically available in the market scenario. Examples of algorithms in this case are Nash Q-learning [51], Correlated Q-learning [45], EXORL [106], Nonstationary Converging Policies [123], Asymmetric-Q [63] and the extension of Fictitious Play to SGs [117].

To the best of our knowledge, the list of algorithms that are general enough to be applied in our scenario is composed of some straightforward extensions of single-agent RL algorithms (e.g. the Multiagent Q-learning), the PHC-WoLF [14] and some of its variations (e.g. the PDWoLF [9]).

### 3.8.3.1 Multiagent Q-learning

The Multiagent Q-learning algorithm is the natural extension of Q-learning to multiagent scenarios. Each agent is equipped with a standard Q-learning algorithm and learns independently, without considering the presence of the others in the environment. The rewards and the state transitions, however, depend on the joint actions of all the agents.

As seen in Section 3.5.3, Q-learning is proven to converge with probability 1 in the single-agent case [55, 108, 113]. When extended to multiagent, however, its supporting theoretical framework and convergence guarantees are lost. The main reason for this is the non-stationary environment generated by multiple learners, which violates the Markov property. Nevertheless, the efficacy and simplicity of this algorithm have lead to its successful application in several multiagent scenarios, for example Claus and
3.8.3.2 WoLF-PHC

The WoLF-PHC algorithm [14] extends $Q$-learning by combining Policy Hill Climbing (PHC) with the Win or Learn Fast (WoLF) heuristic. The algorithm has the desirable properties of rationality and convergence with the capability of learning without needing to know the agent’s entire payoff matrix or the actions or payoffs of the opponents.

PHC is an extension of $Q$-learning to mixed policies. The algorithm maintains a table of $Q$-values, just like standard $Q$-learning, but also includes a mixed policy. This mixed policy is updated using a hill-climbing process based on the $Q$-values. On each iteration, the probability of the highest valued action is increased by a learning rate $\delta \in [0, 1]$.

The WoLF-PHC algorithm, on its turn, modifies the PHC by making the learning rate $\delta$ variable using the WoLF heuristic. The general idea of WoLF is to learn quickly when the algorithm is doing poorly than expected (i.e., when it is “loosing”) and learn slowly when it is doing better than expected (“winning”). To assess whether it is loosing or winning, the algorithm compares the expected payoff of the current policy with the payoff of an average policy.

One variant of the WoLF-PHC algorithm is PDWoLF [9]. Instead of comparing an average policy with the current one, this algorithm assess the win/lose criterion based on second-order difference of policy elements.

3.9 Summary

This chapter has provided the basics of Reinforcement Learning. We have presented an informal description of the problem, followed by a more formal definition using the framework of Markov Decision Processes (MDPs). We have also reviewed the fundamental classes of solution methods and addressed aspects affecting the use of those methods in practical problems. Finally, we have extended the discussion to review the relevant aspects of Multiagent Reinforcement Learning.
Chapter 3. Reinforcement Learning

In summary, the main rationale for the application of Reinforcement Learning in this thesis is that it enables the problem to be solved on-line and without the necessity of having a model of the environment. In our specific case, these characteristics permit the participants to develop their strategic behaviours while interacting with the market.

Additionally, from the analysis of the RL algorithms that have been specifically developed for multiagent problems, such as the one addressed in this thesis, we have seen that just a few of them are general enough to be applied. In fact, the algorithms choice are reduced to straightforward extensions of single-agent RL algorithms, such as the Multiagent $Q$-learning, the WoLF-PHC algorithm and some of its variants. While WoLF-PHC has some desirable properties, it has not been as widely studied and successfully applied as Multiagent $Q$-learning. For this reason, in this thesis we apply the Multiagent $Q$-learning algorithm.
Chapter 4

Commodity Market with Reinforcement Learning

This chapter introduces a conceptual framework for the realisation of commodity-market resource allocation with reinforcement learning. The framework enables the integration of strategic learning participants in equilibrium-based markets and forms the basis for the experimental and theoretical investigation performed in the next chapters. The chapter also introduces a new technique to support the application and scalability of reinforcement learning in problems with continuous spaces.

4.1 Introduction

The overall goal of this thesis is to develop a framework for the optimization of the individual and social allocation efficiency of equilibrium-based markets when their participants exhibit strategic behaviour. For this, the thesis proposes and studies the scenario in which the strategic participants are intelligent learning agents that use utility functions to describe preferences over different resource attributes and develop strategic behaviour via Reinforcement Learning.

In this chapter we introduce the conceptual framework that enables the integration of the strategic learning participants in the market. We start by describing the general scenario addressed in commodity markets. We then refine that scenario to the case
of equilibrium-based markets, reviewing some concepts of *general equilibrium theory* and presenting an abstract architecture for the standard mechanism. Finally, we modify the standard mechanism’s architecture to permit the learning of strategic behaviour by the participants. Additionally, we describe a new technique to support the scalability and application of reinforcement learning in the addressed scenario.

### 4.2 Commodity Market Scenario

A commodity market for computational resource allocation is characterized by the scenario in which a set of sellers and buyers want to trade resources and trust in a central entity called the market, or auctioneer, to coordinate the process. The market is responsible for setting the prices of the resources, so as to maximize some quality measure of the trading, and the participants, buyers and sellers, for specifying the amounts of resources they would like to trade at the specific prices. The idea is represented in Figure 4.1.

![Figure 4.1: A Commodity Market](image)

In this thesis we consider that the resources are of basic types such as CPU time, memory space or disk storage. However, the model is also applicable for the allocation of more complex concepts, such as services. Additionally, what is traded in our case is not the actual resource but access to it, in the form of a *rental agreement* for example.
4.2. Commodity Market Scenario

4.2.1 The Market

The market is the central entity in the model. Responsible for the coordination of the trading process, it sets the prices of the resources and aggregates the demand and supply information from buyers and sellers so as to identify the final allocation.

The prices are set according to a pricing mechanism, which can be derived by various methods and with several purposes (see Chapter 2). For example, the mechanism may target to find a price able to maximize resource usage, or the aggregate utility of the participants. This thesis focuses on equilibrium-based markets, where the price is driven by demand and supply and the objective is to find an equilibrium between these two quantities.

4.2.2 The Participants

The participants are the entities that want to trade resources. A buyer will usually represent a resource user that is interested in renting a particular type of resource to execute some computational task. A seller, on the other hand, will typically represent a resource provider that has some resources available for renting.

The main task of the participants in the mechanism is to decide how much to trade given the current prices specified by the market. There are several models to realise this decision process. In equilibrium-based markets, the usual approach is to specify demand and supply functions, mapping different price levels to the quantities of resources that the entities are prepared to trade at each level. Examples of demand and supply curves are shown in Figure 4.2.

4.2.3 Market-Participants Interaction

The interaction between market and participants can take different forms. Typically, the market will set the prices for the resources and inform the participants, asking for their desire to trade at the specified levels. Based on this information, the participants will decide how much they would like to buy or sell and send it back to the market. The market will then aggregate the demand and supply information from the participants and identify the allocation. In some cases, a new “negotiation” cycle may be required.
before the allocation takes place. In both situations, however, the pricing mechanism can be activated to update the prices.

### 4.3 Equilibrium-based Markets

The scenario presented in the previous section can describe a wide range of commodity market mechanisms for resource allocation. In this section we refine that scenario to the case of equilibrium-based markets and show an abstract architecture for it. This architecture will be extended in the next section to enable the introduction of strategic learning participants in the market.

#### 4.3.1 Basic Principles

Equilibrium-based markets are founded on the microeconomics’ concepts of general equilibrium theory [35, 103]. The goal of the mechanism is to find an state at which the total quantity of resources demanded by the buyers is equal to the total quantity supplied by the sellers. Such a state is called an economic equilibrium.

The concept of a perfectly competitive market is central to general equilibrium theory. It abstracts the notion of an ideal economy in which no single participant has significant impact on price. One of conditions usually cited as necessary for perfect competition is the existence of a large (possibly infinite) number of buyers and sellers, such that each one is responsible only for a very small part of the market. In addition, buyers and sellers must have perfect information and all the resources must be homogeneous, so that buyers are indifferent between alternative sellers. Additionally, buyers and sellers must act independently. The buyers must act towards utility maximization and the sellers towards profit maximization. Finally, there should exist no entry or exit barriers, making it relatively easy for buyers and sellers to enter and exit the market.

If an economic equilibrium is found through perfect competition, then, by the First Fundamental Theorem of Welfare Economics, that equilibrium is a Pareto-Optimal solution for the resource allocation problem [35, 103]. It is important to stress, however, that the concept of Pareto-Optimality does not necessarily relate to the fairness or desirability of the outcome. There can be many possible Pareto-Optimal outcomes and not all of them may be equally desirable by the participants. We will get back to this...
4.3. Equilibrium-based Markets

issue latter, when we introduce the strategic learning participants.

4.3.2 Competitive Participants

In a perfectly competitive market, the best strategy for any participant, individually, is to act competitively. A competitive participant is a price-taking rational entity. The rational aspect is related to the idea that the participant will act towards the maximization of its utility or profit. The price-taking aspect is related to the concept that it will consider the price as exogenous, that is, as a given parameter of the decision-making problem.

The strategies of competitive participants are usually represented by demand and supply functions. A demand function represents the purchasing intentions of a buyer in relation to a type of resource, expressed as a function of the resource’s price. Similarly, a supply function represents the selling intentions of a seller, also expressed as a function of the resource’s price.

A buyer is faced with the problem of utility maximization. The buyer is associated with a set of preferences and an income. The preferences reflect the relative value the buyer puts into resource types that can be purchased and the income represent the budget that the buyer has available to fulfill those preferences. It is assumed that the buyer is completely aware of the relative worth of each type and quantity of resource and that he/she is able to represent it in terms of utility functions. Because the prices are not affected by the participant’s decisions, it can calculate, for every possible price, the amount of resources that maximizes its utility. The result of this maximization problem is the buyer’s demand function. In general, the quantity demanded by a buyer will move in the opposite direction of the price, characterizing the downward slope shown in Figure 4.2.

A seller is faced with the problem of profit maximization. The seller can be seen as having a revenue function and a cost function. The revenue function describes the amount of money the seller will gain from selling and the cost function represents cost it incur to, for example, produce the resources. The task of the seller is to determine, for every possible price, the optimal amount of resource to sell given these functions. The solution of this maximization problem is the seller’s supply function. In general, the quantity supplied by a seller will move in the same direction of the price, characterizing the upward slope shown in Figure 4.2.
4.3.3 Economic Equilibrium

The equilibrium, or market-clearing, price is found at the intersection of the market’s demand and supply curves, as shown in Figure 4.3. The market’s demand curve represents the amount of resources that the buyers are willing to buy for each price. It is calculated from the sum of the individual demands of the buyers at each price. Similarly, the market’s supply curve represents the amount of resources that the sellers are willing to sell for each price and is calculated from the sum of the individual supplies at each price.

It can be seen in Figure 4.3 that there will be a surplus of supply if the price is above the equilibrium. For prices below the equilibrium, on the other hand, there will be a shortage of supply. It can also be seen that different supply and demand curves may generate different points of economic equilibrium.

Each resource type is typically associated with one market. When the market for one resource type is in equilibrium, there is a partial economic equilibrium. When all the markets are in equilibrium, there is a general economic equilibrium.
4.3.4 The Tâtonnement Process

Tâtonnement, or groping, is the name given to the dynamic process proposed by Walras [119] to explain how general equilibrium might be reached. The rough idea of the process is to increase or decrease the price of each individual resource type according to its excess demand until demand-and-supply is equilibrated. Prices are increased for resources with more demand than supply and decreased for resources with more supply than demand. No transactions take place at disequilibrium prices.

The general idea of tâtonnement, and of equilibrium-based markets, is depicted in Figure 4.4. The process consists of cyclic iterations between the market and the participants. The market sets the initial prices of the resources and asks the participants for their desire to trade at the specific levels. Based on the prices, the participants decide how much they want to buy or sell and send it to the market. The market then aggregates the information received from the participants and checks whether the equilibrium has been reached, that is, if the total demand is equal to the total supply. If there is an equilibrium, the process stops and the resources are sold. Otherwise, the market adjusts the prices and asks for the new demand and supply information from the participants. The process continues until the equilibrium is reached.

![Diagram of the tâtonnement process](image-url)

Figure 4.4: General idea of the tâtonnement process (and of equilibrium-based markets)
4.3.5 Abstract Architecture

Based on the previous sections, we can now define the abstract architecture of an equilibrium-based market with competitive participants. The architecture is shown in Figure 4.5.

A competitive participant can be seen as composed of two components: a preferences component and a strategy component. The preferences component represents the participant’s preferences, which include its income and set of utility functions if the participant is a buyer or its cost and revenue functions if it is a seller. The preferences are used to define the participant’s demand or supply function, represented in the architecture by the strategy component. The functions are calculated before the participant enters the market and are kept static during the allocation process, not changing unless the preferences of the participant change. As such, the decision making of the participant during the allocation process is based exclusively on its demand or supply function. The preferences are used only for the initial definition of these functions.

![Figure 4.5: Abstract architecture for equilibrium-based mechanisms with competitive participants](image)

4.4 Market with Strategic Learning Participants

The First Fundamental Theorem of Welfare Economics states that any economic equilibrium found through perfect competition is a Pareto-Optimal solution for the resource
4.4. Market with Strategic Learning Participants

The statement raises two concerns for this thesis. The first concern is related to the Pareto-Optimality of the outcome. It is important to emphasize that a Pareto-Optimal solution is not necessarily a fair or a desirable solution. A solution is Pareto-Optimal if there is no other solution that can improve one agent’s utility without deteriorating the others’. The concept merely indicates that no one can be made better off without someone else being made worse off. In fact, different Pareto-Optimal outcomes can generate different utility gains to the parties involved. Being able to find a fair Pareto-Optimal allocation, in which all the participants are equally or near-equally satisfied is an important aspect of the resource allocation problem.

The second concern is related to the requirement of perfect competition. In particular, the concept assumes that the participants are price-taking entities and have perfect knowledge. Apart from being criticized by many researchers as not realistic, such assumptions are hard to be satisfied in the distributed systems targeted in this thesis. In these systems, there is little control over the behaviour of the participants, making it practically impossible to guarantee that they will not attempt to exploit the system by engaging in strategic behaviour. Hence, to understand the impacts of these attempts and to develop mechanisms for optimal allocation in the presence of strategic participants are also important aspects of the problem.

This thesis sets to investigate the above aspects by introducing participants with learning capabilities in the market. We now describe an abstract architecture to enable it.

4.4.1 Strategic Learning Participants

We have seen in the previous section that the strategy of a competitive participant, composed of its demand or supply function, is not updated during the allocation process. It is defined at the beginning of the “participant’s life”, based on both the participant’s preferences and the assumption of price-taking behaviour, and kept unchanged, unless the preferences change. Such an approach does not enable the development of any type of strategic behaviour. In practice, a demand or supply function calculated beforehand based on the price-taking assumption implies that all the points lying in the characteristic curve are equally preferred by the participant during the allocation process. Whenever a new price is defined by the market, the participant has no alternative...
other than replying with the demand request or supply offer dictated by its demand or supply function.

Therefore, that approach needs to be modified in order to enable the development of strategic behaviour. We propose the scenario in which the participants learn their demand and supply functions from interaction with market. The general idea is that, given a particular market configuration there is at least one demand/supply function that maximizes the utility or profit derived from the participant’s preferences. Learning such a demand/supply function leads to the development of the participant’s strategic behaviour.

More specifically, in the proposed scenario, the strategic participants are intelligent learning agents that use utility functions to describe preferences over different resource attributes and develop strategic behaviour from interaction with the market via Reinforcement Learning.

### 4.4.2 Integration Architecture

The abstract architecture for a market composed of strategic learning agents is shown in Figure 4.6. It modifies the architecture presented in the previous section by integrating a reinforcement learning component within each participant. The inputs to this component are the price information received from the market and the preferences received from the preferences component. The price information is used as state signal and the preferences define the agent’s reward function. The output of the component is the participant’s strategic demand or supply function.

The parallel of Figure 4.4 for the market with strategic learning participants is depicted in Figure 4.7. For the market, the process is the same as defined before. The market sets the initial prices of the resources and sends this information to the participants. Once the participants reply with their demand and supply information, the market checks for the equilibrium, allocating the resources if the equilibrium is found or adjusting the prices and beginning a new cycle otherwise. For the participants, however, the process is quite different. The price information received from the market is deviated to the learning component, which is responsible for the learning of the participant’s strategic behaviour (demand or supply function) and the selection of the amount of resources the participant will buy or sell. The selection depends on the RL algorithm used by the agent and will follow the algorithm’s approach for exploration-exploitation (see...
4.4. Market with Strategic Learning Participants

Commodity Market with Reinforcement Learning

Market
Pricing Mechanism

Commodity Market

Commodity Market with Reinforcement Learning

Strategic Learning Buyer/Seller

Commodity Market

Strategic Learning Buyer/Seller

Figure 4.6: Abstract architecture for equilibrium-based mechanisms with strategic learning participants

Chapter 3).

Figure 4.7: General idea of the tâtonnement process (and of equilibrium-based markets) with strategic learning participants
4.4.3 Dealing with Continuous Spaces

One particular aspect that needs to be addressed in the framework is related to the continuous spaces present in our problem. In the proposed approach, prices are used as the states of the environment and amounts of resource (to request or offer) are used as the actions of the agents. As such, our problem has both the action and the state spaces as continuous components.

However, as it has been seen in Chapter 3, traditional reinforcement learning algorithms are based on tabular representations of the action and state spaces and, therefore, cannot be directly applied in problems with continuous spaces. Continuous spaces also have a direct effect on the scalability aspects of the algorithms, which usually rely on exhaustive trial-and-error searches. For these reasons, reinforcement learning algorithms are usually coupled with some additional technique, mainly function approximation.

In this thesis we propose a novel approach to support the application and scalability of reinforcement learning in problems with continuous spaces. It consists in the discretization of the state and the action spaces for the learning and the application of curve-fitting or interpolation procedures in the learnt discrete policy to obtain the agent’s continuous policy. The general idea is represented in Figure 4.8.

The technique is applied as follows. In our specific scenario, what we want the agents to learn is a demand or supply function, mapping price levels to amount of resources. For this, we use the current price of the resources as the environment states $s \in S$ and the possible amounts of resource as the agent’s actions $a \in A$. In this case, a policy $\pi(s, a)$ represents the agent’s demand or supply function and the $Q$-values $Q(s, a)$ estimate how good it is for the agent to request or offer an amount of resource $a$ at a price $s$.

The first step to apply the technique is to discretize the state and action spaces, using a rounding procedure for example. The result is a grid-like structure, where each component represents a specific state-action pair. The next step is to apply standard reinforcement learning algorithms to learn the values of the state-action pairs, that is, to learn the $Q$-values $Q(s, a)$. After some learning period, the current policy $\pi$ can be extracted by extracting the highest-valued action at each state, so, $\pi(s, a) = \arg \max_a Q(s, a), \forall s$. A curve-fitting procedure can then be applied in this discrete policy to obtain its continuous version, which represents the agent’s demand or supply
4.5 Summary

In this chapter we have introduced a conceptual framework for the realisation of commodity-market resource allocation with reinforcement learning. The framework enables the introduction of strategic learning participants in equilibrium-based markets and forms the basis for the experimental and theoretical investigation performed in the next chapters. As part of the framework, we have also introduced a new technique to support the application and scalability of reinforcement learning in problems with continuous spaces.

Figure 4.8: A technique to support the application and scalability of Reinforcement Learning in problems with continuous spaces

function.

The approach enables the application of reinforcement learning in problems with continuous spaces and leads to the reduction of the required amount of learning, since not all the actions and states need to be exhaustively explored in order to obtain a reasonable policy.

4.5 Summary

In this chapter we have introduced a conceptual framework for the realisation of commodity-market resource allocation with reinforcement learning. The framework enables the introduction of strategic learning participants in equilibrium-based markets and forms the basis for the experimental and theoretical investigation performed in the next chapters. As part of the framework, we have also introduced a new technique to support the application and scalability of reinforcement learning in problems with continuous spaces.
Chapter 5

The Iterative Price Adjustment with Reinforcement Learning

This chapter investigates the application of strategic learning agents in a specific type of equilibrium-based mechanism called Iterative Price Adjustment (IPA). The chapter introduces the IPA with RL, a market-based mechanism that enhances the original IPA by considering the scenario where agents use utility functions to describe preferences over different resource attributes and develop strategic behaviour by learning demand functions adapted to the market through Reinforcement Learning.

5.1 Introduction

In this chapter we investigate the introduction of strategic learning agents in the Iterative Price Adjustment (IPA) [37, 128]. The IPA is an equilibrium-based mechanism in which the price is calculated through an iterative process. The mechanism cyclically asks the agents for the amount of resources they would be willing to buy and uses this information to update the price. The price is increased if the total demand requested by the agents is higher than the supply and decreased otherwise. This iterative process continues until the equilibrium between demand and supply is found, when the market is cleared and the resources are sold.
5.2 The IPA with RL

Under standard assumptions, the strategies of resource requesting agents in the IPA are defined using the price-taking approach. As seen in the previous chapter, such an approach does not enable the agents to develop any type of strategic behaviour. More specifically, demand functions specified beforehand based on the price-taking assumption have all the points lying in the characteristic curve as equally preferred by the agents during the allocation process. Therefore, in practice, those demand functions do not account for the preferences that the agents may have over attributes of the allocation, for example, the price and the resource levels themselves, making it difficult to influence and optimize the resource allocation in terms of the utility received by the agents.

In this chapter we propose and investigate the IPA with RL, a market-based mechanism that enhances the original IPA by introducing agents with learning capabilities into the market.

The chapter is organized as follows. The next section presents the IPA with RL. It reviews the IPA mechanism, the RL algorithm and the modelling of the IPA as a RL problem. Section 5.3 presents the general setup for the experimental investigation on the mechanism, presented in sections 5.3 and 5.4. Some works that are directly related with the approach are discussed in Section 5.6. The conclusions and a summary of the chapter are presented in Section 5.7.

5.2 The IPA with RL

The IPA with RL is a commodity market mechanism resulting from the combination between the Iterative Price Adjustment and Reinforcement Learning. It considers the scenario in which a limited amount of resources has to be allocated to a set of agents in a commodity market resource allocation system using the IPA mechanism. The agents use utility functions to describe preferences in the allocation and learn demand functions adapted to the market using reinforcement learning (as illustrated in Figure 5.1).
Chapter 5. The Iterative Price Adjustment with Reinforcement Learning

5.2.1 Iterative Price Adjustment

The IPA decomposes the resource allocation optimization problem into smaller and easier sub-problems. Its behaviour is based on the tâtonnement process, mimicking the law of demand and supply. The price is increased if the demand exceeds the supply and decreased otherwise. The cycle starts with a facilitator (the market) announcing the initial prices for the resources. Based on the initial price, the agents decide on the amount of resources that maximize their private utilities (the sub-problems) and send these values to the facilitator. The facilitator adjusts the prices according to the total demand received and announces the new prices. The process continues until an equilibrium price is reached, when the resources are sold and the market is cleared. In the equilibrium, the total demand equals the supply or the price of the excessive supply is zero. Under some circumstances, the equilibrium price may not exist [56], but that problem is out of the scope of this thesis.

The standard IPA method is formalized as follows. Let $C = \{C_1, \ldots, C_i, \ldots, C_m\}$ be the total supply of resources available, where $C_i$ is the total supply of resource $i$; Let $P(t) = \{p_1(t), \ldots, p_i(t), \ldots, p_m(t)\}$ be the price vector for the resources $C$ at time $t$, where $p_i$ is the price for the resource $i$; there are $n$ self-interested agents and each agent has a utility function $u_j(D_j)$, which is the utility over $\{d_{1,j}, d_{2,j}, \ldots, d_{m,j}\}$, where $d_{i,j}$ is the amount of resource $i$ that the agent $j$ requires. Let $D_j(t) = \{d_{1,j}(t), d_{2,j}(t), \ldots, d_{m,j}(t)\}$ be the total amount of resources agent $j$ requires at time $t$. At time $t$, the objective of the agent $j$ is to find $D_j(t) = \arg_{D_j(t)} \max(u_j(D_j(t)))$. At time $t + 1$, the price can change, and the new price of resource $i$ is given by:
5.2. The IPA with RL

\[ p_i(t + 1) = \max\{0, p_i(t) + \alpha(\sum_{j=1}^{n} d_{i,j}(t) - C_i)}\}, \quad 1 \leq i \leq m \quad (5.1) \]

where \( \alpha \) is a small positive constant.

It should be noted that the agent’s utility maximization task in the IPA is actually the maximization of its instantaneous profit. As described by Wu et al. [128], the agent can be considered as having a revenue function and a cost function over the resources and, at each time step, its task is to find the demand request that maximizes the difference between these two functions given the current price. The result is the existence of a demand function, in which all the points lying in the characteristic curve are equally preferred by the agent. As commented before, such an approach limits the power of the agents. As they cannot express preferences over different attributes of the allocation, in particular over the price and the amount of resources, and take these preferences into account during the allocation process, they cannot develop a strategic behaviour to influence the mechanism.

We approach the problem identified above by employing the conceptual framework defined in Chapter 4, addressing the scenario in which agents use utility functions to describe their preferences in the allocation and learn demand functions adapted to the market by RL. The idea is that, given a particular market configuration, there is at least one demand function that maximizes the utility obtained from the aggregation of the agent’s utility functions. Learning such a demand function leads to the development of the agent’s strategic behaviour.

5.2.2 Reinforcement Learning

RL agents learn how to map states of the environment to actions so as to maximize a numerical reward signal. The RL model consists of a discrete set of environment states \( S \), a discrete set of agent actions \( A \) and a set of scalar reinforcement signals \( R \). On each step, the agent receives a signal from the environment indicating its state \( s \in S \) and chooses an action \( a \in A \). Once the action is performed, it changes the state of the environment, generating a reinforcement signal \( r \in R \). The agent uses this reinforcement signal to evaluate the quality of the decision. The task of the agent is then to maximize (or minimize) some long-run measure of the reinforcement signal. This signal has a vital importance in the model. It is the way to communicate to the
agent what to do without having to say how to do.

According to the analysis presented in Chapter 3, the Multiagent Q-learning algorithm is appropriate for the scenario addressed in this thesis. The algorithm is simple, easy to implement and has been applied with success in many scenarios [30, 43, 92, 94, 95, 109, 132]. In the algorithm, each agent maintains a table of $Q(s, a)$-values representing estimations of how good it is for the individual agent to take a particular action $a$ in a particular state $s$. The $Q$-values are in fact estimations of $Q^*(s, a)$-values, which represent the sum of the immediate reward $r$ obtained by performing action $a$ at state $s$ and the total discounted expected future rewards obtained by following the optimal policy thereafter. By updating $Q(s, a)$, the agent eventually makes it converge to $Q^*(s, a)$. The optimal policy $\pi^*$ is then followed by selecting the actions where the $Q^*$-values are maximum. $Q$-values are updated using:

$$Q(s, a) = Q(s, a) + \alpha (r(s, a) + \gamma \max_{a'} Q(s', a') - Q(s, a))$$  \hspace{1cm} (5.2)$$

where $\alpha \in ]0, 1[$ is the learning rate and $\gamma \in ]0, 1[$ is the discount rate. If the environment is stationary, each action is executed in each state an infinite number of times and $\alpha$ is decayed appropriately, the $Q$-values will converge with probability 1 to the optimal ones [108].

An important component of Q-learning is the action selection mechanism. It is used to harmonize the trade-off between exploration and exploitation. We use the $\epsilon$-greedy method, which selects a random action with probability $\epsilon$ and the greedy, the one that is currently the best, with probability $1-\epsilon$:

$$Pr(a|s) = \begin{cases} 
(1 - \epsilon) + (\epsilon/n), & \text{if } Q(s, a) \text{ is currently the highest} \\
\epsilon/n, & \text{otherwise}
\end{cases}$$ \hspace{1cm} (5.3)$$

where $Pr(a|s)$ is the probability of playing action $a$ in state $s$ and $n$ is the number of actions available to the agent.
5.2.3 The IPA as a RL problem

The first task to apply the $Q$-learning algorithm is to define what are the states, the actions and the rewards. As described in Chapter 4, we use the current price of the resources as the environment states and the possible demand requests as the actions of the agents. In this case, $Q$-values represent estimations of how good it is for the agent to request demand $d$ at price $p$, so $Q(p, d)$. The optimal policy $\pi^*$ is then an optimal demand function and is followed by selecting the demand requests with highest $Q^*$-values.

The application of RL in the IPA changes the objective of the agents in the resource allocation. Instead of maximizing their private utility functions in an immediate fashion, they now have to maximize the accumulated reward:

$$\sum_{t=0}^{\infty} \gamma^t r_t$$

where $r$ is the reward given by the reward function in use and $\gamma \in [0, 1]$ is the discounting factor for infinite horizon problems.

The experimental investigation in this chapter considers strategic learning agents that exhibit selfish behaviour. For this, we define the Individual Reward Function (IRF). The IRF is based on the private utility of the agents. Using only local information, the agent receives a positive reward equal to its utility when the market reaches an equilibrium state and zero for all the other states:

$$ r = \begin{cases} U & \text{if equilibrium found,} \\ 0 & \text{otherwise} \end{cases} $$

where $U$ is the utility of the agent. This reward function generates a competitive learning problem, where one agent’s goal is to learn a demand function that maximizes its discounted expected individual utility.

Algorithms 5.1 and 5.2 describe the IPA Market modified with RL. Algorithm 5.1 is the algorithm for the Market Agent and Algorithm 5.2 is the algorithm for the selfish

---

1In the next chapter we evaluate the impacts of the presence of selfish and altruistic learning agents in the market.
learning agents.

\begin{algorithm}
\caption{IPA Market}
\begin{algorithmic}[1]
\State Initialize price $p$ with arbitrary value
\Repeat
\hspace{1em} Announce $p$
\For {each agent $i$}
\hspace{2em} Receive demand request $d_i$;
\EndFor
\hspace{1em} Update price: $p \leftarrow \max\{0, p + \alpha(\sum_{i=1}^{n} d_i - C)\}$
\Until {equilibrium is found}
\State Sell resources at price $p$
\end{algorithmic}
\end{algorithm}

\begin{algorithm}
\caption{Selfis Learning Agent (using the Individual Reward Function)}
\begin{algorithmic}[1]
\State Initialize $Q$-table with arbitrary values
\While {equilibrium is not found}
\hspace{1em} Observe price $p$
\hspace{1em} Select demand request $d$ using $\epsilon$-greedy
\hspace{1em} Request $d$
\hspace{1em} Observe new price $p'$
\hspace{1em} Observe reward $r$:
\hspace{2em} \If {equilibrium is found}
\hspace{3em} $r \leftarrow$ individual utility $U$
\Else
\hspace{3em} $r \leftarrow 0$
\EndIf
\hspace{1em} Update $Q$-table:
\hspace{2em} $Q(p, d) \leftarrow Q(p, d) + \alpha(r + \gamma \max_{p'} Q(p', d') - Q(p, d))$
\hspace{2em} $p \leftarrow p'$
\EndWhile
\end{algorithmic}
\end{algorithm}

Chapter 5. The Iterative Price Adjustment with Reinforcement Learning
5.3 General Experimental Setup

The general configuration for the experiments is as follows. The agents have preferences over price and amount of resources, using a utility function for each attribute. \( U_1(p) \) is the utility function for price and \( U_2(m) \) is the utility function for amount of resource. The total utility of an agent is given by the product of these functions, \( U(p, m) = U_1(p) \times U_2(m) \). The product stresses the existence of dependency between the two properties.

Figure 5.2 shows the actual utility functions used by the agents in the experiments. The functions are normalized between 0 and 1, representing the maximum and the minimum satisfaction the agents can achieve. A possible interpretation for \( U_2(m) \) is that an agent, representing a task, requires a minimum of 2 units of memory to execute. If the agent is allocated an amount lower than 2, it will be completely dissatisfied as it will not be able to run. Its complete satisfaction is achieved when it is allocated an amount equal to or higher than the recommended requirements, in our case approximately 8.39 units of memory. Likewise, interpretation is also possible for \( U_1(p) \).

The experiments include the use of learning and static agents in different configurations. The demand functions of the static agents are defined “by hand” and based on subjective criteria taking into account the utility functions \( U_1(p) \) and \( U_2(m) \). The static agents and the criteria used to define them are:

- **Static Agent 1 (S1):** The demand function of S1 was defined based on the idea that the agent should request the demand that gives the same level of satisfaction of price. For example, if the current price of the resource gives to the agent a utility of 0.5, it will require the demand that gives the same 0.5 of utility.

- **Static Agent 2 (S2):** The idea of S2’s demand function is that the agent should request the minimum demand that gives utility 1 for every price that also gives
utility 1 and the maximum demand that gives utility 0 exactly in the point where price utility is 0. Between these two points, the demand function should be mapped to a line crossing them.

- **Static Agent 3 (S3):** The idea of S3’s demand function is that the agent should request the minimum demand for maximum utility (one) only when the price is minimum (zero), and a demand equal to 0 only when the price utility is 0. Between these two points, the demand should be mapped to a line crossing them. So, in our case, S3 will request 8.39 units of resource for prices lower than 0 and 0 units of resource for prices higher 5.71. For all other prices, it will request the amount dictated by the line equation crossing those points.

The demand functions of the static agents are shown in Figure 5.3.

![Figure 5.3: Static Agents’ demand functions.](image)

During the experiments price and demand requests are bounded in \([0, 10]\). Following the technique introduced in Chapter 4, both prices and demand requests are rounded to 1 decimal place. In the IPA market, the only information available to the agents is the current price of the resources. Therefore, the agents do not know the actions taken and the rewards received by the other agents.

The market is set with 10 units of resource. From the analysis of the utility functions, it can be noted that such an amount does not allow for both agents to have a complete satisfaction in the allocation but it permits the analysis of the market and the learning under a condition of limited supply, which is the most interesting situation.

### 5.4 Learning in the Presence of a Static Agent

We first investigate the case of a market composed of a learning and a static agent. The learning agent’s task in this case is to learn a demand function able to maximize its
displaced individual utility. To analyse this learning task, let $C$ be the total amount of resources in the market, $p_t$ be the price of the resources at time $t$, $d_i(p)$ be the demand function of agent $i$ and $u_i(p, d)$ be the utility function of agent $i$. Assume that the sum of the agents’ demands for maximum utility is greater than the total amount of resources, what implies that the price will not reach zero at any time during the negotiation, so $\sum d_i(p) > C$ when $u_1(p, d) = \ldots = u_n(p, d) = 1$. Also assume that no demand request can be negative. Any set of demands that is able to equilibrate the system at any time of the negotiation process is then given by $\sum d_i(p_t) = C$. Considering our learning scenario, let $d_1(p)$ be a demand function for the learning agent and $d_2(p)$ be the demand function of the static agent. As $d_2(p)$ is fixed, the demands the learning agent can request and that are able to equilibrate the system for any price $p$ are given by $d'_1(p) = C - d_2(p)$. The optimal utility the learning agent can get from the allocation process is therefore given by the point $Q(p, d) \in d'_1(p)$ that maximizes $u_1(p, d)$. An optimal demand function $d^*_1(p)$ for the learning agent will then be any demand function that is able to direct the IPA negotiation process to the point $Q(p, d)$. Therefore, the learning task is to find a demand function $d^*_1(p)$.

### 5.4.1 Experimental Setup

A series of preliminary experiments had been performed to identify a feasible configuration for the values of the parameters used in the learning algorithm. Based on these experiments we set $\alpha = 0.1$, $\gamma = 0.9$ and $\epsilon = 0.4$. The price of the resources is adjusted by the IPA market using a constant parameter $\alpha$ set to 0.05.

We run 10 learning experiments of 1 million episodes for each static agent.

### 5.4.2 Learning Results

The evolution of the demand functions over the episodes presented a similar behaviour. In the first 50,000 episodes, the demand functions were still not so consistent, what can be explained by the fact that the agent had probably not visited all the actions in all the states enough times yet. From that point on, they started to evolve towards a well-defined trend. This behaviour can be observed in Figure 5.4, which shows 4 points during the learning process of one of the agents. The evolution was also not completely stable. Although the shape was always there and was quite stable between the same
values for demand and price, the individual values presented some instability from one episode to another.

The instabilities found here are in the sense that the $Q$-table has not converged to a unique value configuration. The $Q$-values changed from one episode to another, generating slightly different demand functions each time. These changes can be attributed to several factors. In particular, $Q$-learning is only proven to converge if the environment is stationary and the learning parameter is decayed appropriately. The environment addressed in this section is stationary since we have only one learner. Thus, the non-application of decay rules for the learning parameter, as implemented in this work, is the most likely cause. We expect the instabilities to be reduced with the use of decay rules.

The observed trend of the learnt demand functions presents high demands for lower prices, some middle demands in the central area and low demands for high prices. This overall trend was expected. The only unexpected point is the existence of a change in the direction of the functions in their central area. See Figure 5.4 and Figure 5.5 for some examples. This behaviour will be issue of further studies, but is likely to be generated by the fact that, after experiencing a sudden drop in the received utility, the agent tries to direct the negotiation process to a more favourable point.

Even though the agents have not achieved exactly the same demand function, the functions obtained were consistent and presented a very similar overall trend.
5.4.3 Market Results

Following the technique introduced in Chapter 4, the evaluations in the market are made with the trends of the actual demand functions learnt by the agents. The trends are obtained by a process of curve fitting where the best fitted curve from a set of curve models is selected for each learning agent. We use the demand functions obtained at step $450\times 10^3$. The rationale for this is that at that step most of the agents have already developed a demand function with a very consistent shape. Figure 5.5 shows an example of the trend obtained for one learner of each static agent.

![Figure 5.5: Examples of learning agents’ demand functions with trends.](image)

The resulting demand functions are then applied in the ordinary IPA market. Figure 5.6 presents the utilities received by the 10 learning agents over the experiments with S1. The utility received by them in all the experiments is very close to the optimal: $\sim 0.482$ for S1, $\sim 0.502$ for S2, $\sim 0.83$ for S3. It is also seen a consistency in the received utilities over the experiments. The same type of results is observed for the experiments with S2 and S3 shown in Figure 5.7 and Figure 5.8, respectively.

![Figure 5.6: Utilities over experiments with S1.](image)

It is important to comment that these experiments are not about the learning agents beating the static ones, but about the learning agents getting the most utility they can, given the utility functions, the static agents’ demand functions and the amount of resources in the market. Recall that, as the static agent has a fixed demand function, the set of optimal demands that the learning agent can request for every price and that are
able to equilibrate the system is fully define if the amount of resources in the market is known. Therefore, the task of the agent is to fin a point that maximizes its total utility, which is not necessarily higher than the utility of the static agent. The results of the experiments show that the learning agents are able to achieve such a objective quite well and, more important, they show the feasibility of the $Q$-learning algorithm for the addressed scenario.

5.5 Learning in the Presence of a Learning Agent

We now focus on the case where the market is composed of two selfis learning agents. We use the same market configuratio used in the previous scenario, a single IPA Market with one type of resource, but train the agents jointly.

5.5.1 Experimental Setup

We run 4 learning experiments with 1 million episodes each. The learning parameters are set to $\alpha = 0.1$, $\gamma = 0.4$ and $\epsilon = 0.4$, and the market constant set to $\alpha = 0.1$. From
these experiments we obtain 8 agents.

5.5.2 Learning Results

The learning results are very similar to the ones obtained in the previous experiments. The evolution of the demand functions is also similar over the agents, being not completely consistent in the first 50,000 learning episodes and evolving towards a well-defined trend from that point on (see Figure 5.9). However, the instability in the learnt demand functions is slightly higher.

As commented for the previous experiments, these instabilities can be attributed to several factors. In particular, Q-learning is only proven to converge if the environment is stationary and the learning parameter is decayed appropriately. For the cases with a single learner, as in the previous scenario, the environment is stationary, thus the non-application of decay rules for the learning parameter, as implemented in this work, is the most likely cause. For the cases with multiple learners, as in this scenario, it adds up to the fact that the environment becomes dynamic [79]. Each time one agent learns a new strategy, it changes the environment as perceived by the other agents, possibly invalidating the very strategies learnt by them and making a series of new adaptations necessary. As a consequence, each agent has to adapt its strategy to the others’ in a continuous cycle. Usually called co-adaptation, the dynamic environment created by this effect violates the principles behind most single-agent learning algorithms, which includes Q-learning.

An interesting point shown in Figure 5.9 is that the demand functions obtained in this scenario do not presented high demands for lower prices as it might be expected from observing the utility functions and is actually seen in the previous experiments (see Figure 5.5). Another interesting point is the non existence of unexpected changes in the demand function’s direction in the central area, as seen in the previous section (see Figure 5.5).

As in the previous experiments the agents have not achieved exactly the same demand function, but the obtained functions are consistent presenting a very similar overall trend.
Chapter 5. The Iterative Price Adjustment with Reinforcement Learning

5.5.3 Market Results

We use the trends of the learnt demand functions to obtain the results in the market. We randomly selected 4 of our 8 agents to apply the curve-fitting method. As in the previous experiments we use the demand functions obtained at step $450 \times 10^3$. The demand functions with respective trends are shown in Figure 5.10.

![Figure 5.9: Evolution of the demand function of one of the learning agents.](image)

![Figure 5.10: Learning agents’ demand functions with trends.](image)

The resulting demand functions are then applied in the ordinary IPA market. In these particular experiments we run the learning agents against each other and against the static agents. Next, we show the results obtained for these two cases. For comparison
purposes, we also show the results obtained when only static agents are present in the market.

5.5.3.1 Individual Results

We first discuss the results from the individual perspective. Figure 5.11 shows the individual utility of the agents for experiments of type learner vs. learner and static vs. static. In general, the utility of the learning agents are better than the utilities of the static agents. There is also some similarity among the utility received by the learning agents. The situation is different for the static agents, where one of them, S3, has achieved a very poor individual performance.

Experiments of type learner vs. static (Figure 5.12) revealed the four learning agents with similar performances against the same static agents. This result was expected as they learn using the same set of utility functions. These experiments also reveal a good performance of the learning agents. The utilities received by them are not so far from the optimum they can get. The learners received an average utility of approximately 0.45 for S1, 0.48 for S2 and 0.7 for S3, the optimum are approximately 0.482, 0.502, 0.83, respectively.

5.5.3.2 Social Results

We now discuss the results from the social perspective. In this particular evaluation we use three of the most significant social welfare functions: the Utilitarian Social Welfare (USW), defined as the sum of the individual utilities; the Egalitarian Social Welfare (ESW), given by the utility of the agent which is worst off and regarded as a
good measure of the fairness of a system; and the Nash Product (NP), obtained from the product of the individual utilities and viewed as a good compromise between the USW and the ESW. Other options can be found in Chevaleyre et al. [27].

Figure 5.13 shows the average SW for experiments of type learner vs. learner, learner vs. static and static vs. static. It also shows the results for experiments of type learner vs. static from the previous section.

From Figure 5.13, the market’s performance is improved using the learning agents from the current scenario. Note that the social performance is always better when using only them. The use of one of the current learners and one of the static agents is also able to improve the SW. Not surprisingly, the learning agents from the previous experiments present a poor social performance.

A final analysis about the individual utility received by the learning agents from this
scenario and from the previous reveals a small decrease. The comparison is presented in Figure 5.14.

![Figure 5.14: Average individual utility of learning agents over previous and current experiments.](image)

In summary, the experiments show that selfis learning agents that are trained jointly are able to learn demand functions that perform quite well under both the individual and the social perspective. Compared to when they are trained against static agents, the jointly-trained agents achieve a much better social solution, improving the market’s social welfare.

## 5.6 Related Works

We now discuss some works that are relevant to the subjects developed in this chapter: learning in market-based mechanisms and utility-aware resource allocation.

There is a significant amount of work on learning in market-based mechanisms. While most of the approaches have not been developed specifically for resource allocation, they can be naturally extended for this end. One branch of research corresponds to the improvement of bidding and asking strategies in auction mechanisms. For example, Preist et al. [85] investigate the use of belief-based learning in simultaneous English Auctions. Agents build belief functions to model the valuations held by other participants and coordinate their bids accordingly in multiple auction houses. Gjerstad and Dickhaut [44] apply a similar approach but for single Double Auctions. The application of learning to optimize auction parameters has also been investigated under the general area of Mechanism Design [31, 81].

There is also some work on learning to price. Kephart and Tesauro [61] study the use of $Q$-Learning in a scenario where two competitive “pricebots” have to set the price of a commodity. Könönen [64] investigates a similar problem for the scenario in which one
agent has the power to enforce its strategy on the other. In a recent work, Yanagita and Onozaki [129] investigate the dynamics of a market in which firms use reinforcement learning to set price and production levels.

In the specific case of market-based mechanisms for resource allocation, a survey of 35 different systems is presented by Yeo and Buyya [130]. From those, learning is applied only in Catnets [93]. Catnets’ agents apply an evolutionary-like approach to learn negotiation strategies in a completely decentralized bargaining model. Most of work on learning for negotiation can be naturally transferred to the resource allocation field and, therefore, can be used to improve bargaining models.

With regards to the application of learning in market-based mechanisms is general, our approach differs from the above works in what concerns the application of utility functions to express the preferences of the agents. Previous works usually consider the utility of one agent as a function of the profit it makes in the market. In our approach, in contrast, the agents describe their preferences by explicitly modelling utility functions for attributes of the allocation, which gives them more power in developing strategic behaviour. With regards to the investigation of strategic behaviour in equilibrium-based markets, none of the above approaches addresses the case.

Closely related to the general subject addressed in this thesis is the work by Sandholm and Ygge [91]. The authors investigate the possible gains of speculative behaviour in equilibrium-based markets such as the ones we target. They mathematically derive the maximal gain a speculative agent can obtain when playing against competitive participants and develop strategies for constructing demand functions so that the speculative agent can obtain the maximal advantage from the mechanism. The analysis, however, addresses only one specific family of utility functions for the competitive participants and does not consider the case of multiple speculative agents, as we do in this work. Additionally, the social impacts of the speculative behaviour are not investigated. Moreover, learning techniques are not considered as the mechanism for the development of the speculative agent’s strategic behaviour.

Learning has also been explored to improve non-market-based resource allocation. In Galstyan et al. [43], agents learn which resource nodes to submit their jobs to, given that the nodes are managed by local schedulers. Abdallah and Lesser [1] propose a gradient ascent learning algorithm and applied it in distributed task allocation. Csáji and Monostori [33] modelled a resource allocation problem with precedence constraints as a Markov Decision Process and applied a distributed RL approach to solve it. Those
5.7 Conclusions

works also follow a different approach for the preferences of the agents, which are typically modelled as functions of scheduling parameters, such as reducing the time interval between a job submission and its completion. Utility functions are not considered.

Finally, the work of Chunlin and Layuan [28] is also related to ours. The authors develop a pricing mechanism to maximize the aggregated utility for resource allocation in the Grid. Apart from the fact that learning is not applied, they also follow a different approach for agents’ preferences. Their experiments use the maximization of resource usage as resource owners’ preferences and functions of concrete parameters (e.g. price of the resources and time to complete a task) as users’ preferences.

To the best of our knowledge, the approach presented herein is the first attempt to address the problem of learning demand functions based on the agents’ preferences.

5.7 Conclusions

In this chapter we have presented the IPA with RL, a market-based resource allocation mechanism that enhances the original IPA by introducing agents that use utility functions to describe preferences over different resources attributes and develop strategic behaviour by learning demand functions adapted to the market through RL.

The mechanism has been evaluated in two scenarios. Both scenarios considered a single IPA market with one type of resource and two self-interested agents. In the first scenario one of the agents was a strategic learning agent exhibiting selfish behaviour and the other was a static agent with a pre-define demand function. The results of this case have shown the learning agents being able to develop very good demand functions, achieving near-optimal individual performance. The results illustrate the feasibility of the Q-learning algorithm for the addressed scenario.

The second scenario investigated the case of a market composed of two self-interested learning agents trained jointly. These agents developed demand functions that performed quite well under both the individual and the social perspective. Compared with the agents of the first case, these learning agents achieved a much better social solution, being able to increase the market’s social welfare.

In summary, the investigation has revealed the interesting phenomenon in which selfish
learning agents trained jointly ended up developing demand functions that improve the social welfare of the market. In the next chapter we will perform further investigations and evaluate the impacts of the presence of *selfish* and *altruistic* learning agents in the market.
Chapter 6

Selfish and Altruistic Learning Participants in the IPA with RL

This chapter presents further experiments in the IPA with RL. It investigates the impacts of selfish and altruistic learning agents in the mechanism. The individual and social performances of the market in the presence of these two agent types are experimentally evaluated and the results compared.

6.1 Introduction

The results presented in the previous chapter have shown the interesting phenomenon in which the social welfare of the IPA with RL is improved with the application of selfish learning agents trained jointly. In this chapter we perform further investigations on the mechanism and present the comparison of its performance in the presence of selfish and altruistic learning agents. The objective is to investigate the impacts of these two agent types on the individual and social efficiency of the market.

The study is conducted in four parts. The first part analyses the performance of the market using the same amount of learning used in the previous chapter. The second part focuses on the performance obtained throughout the learning process. The goals for this particular investigation are to identify whether that amount of learning can be reduced and to observe the behaviour of the agents throughout the learning process.
The third part further explores the case in which only learning agents are present in the market. The goal is to investigate if the results found in the previous two parts hold when the number of agents is scaled. Finally, the fourth part addresses the case of agents with non-symmetric preferences.

6.2 The Individual and Social Reward Functions

We start by introducing the reward function used by the altruistic learning agents. The learning process of the altruistic agents is based on the social welfare of the market. Using global information, the agent receives a positive reward equal to the social welfare of the market when it reaches an equilibrium state and zero in all the other states:

\[ r = \begin{cases} 
  \text{SW} & \text{if equilibrium found,} \\
  0 & \text{otherwise} 
\end{cases} \]

where \( \text{SW} \) is the social welfare. This reward function will be referred to as the Social Reward Function (SRF). On contrary to the individual reward function, used by the selfis agents (see Section 5.2.3), this function generates a cooperative learning problem, where one agent’s goal is to learn a demand function that maximizes the discounted expected SW of the market.

To calculate the market’s SW we use the Nash Product (NP) function. The NP is given by the product of the individual utility of the agents:

\[ SW = NP = \prod_{i=1}^{n} U_i \]  

(6.1)

where \( U_i \) is the utility of agent \( i \). It is suitable for the resource allocation scenario because it emphasizes the improvement and the balance among the utility of the agents. In addition, this function is regarded as a good compromise between the Utilitarian Social Welfare (USW), obtained from the sum of the individual utilities, and the Egalitarian Social Welfare (ESW), given by the utility of the agent which is worst off [27].

Algorithm 6.1 describes the algorithm employed by the altruistic learning agents. Note that line 9 is the only difference between it and the algorithm used by the selfis learn-
ning agents (Algorithm 5.2). The line describes the reward received by the agent for equilibrium states. The altruistic agents receive a reward equivalent to the Nash Product of the allocation. The selfish agents, on the other hand, receive a reward equivalent to their individual utility.

Initialize Q-table with arbitrary values
while equilibrium is not found do
  Observe price $p$
  Select demand request $d$ using $\epsilon$-greedy
  Request $d$
  Observe new price $p'$
  Observe reward $r$:
  if equilibrium is found then
    $r \leftarrow$ social welfare $\prod_{i=1}^{n} U_i$;
  else
    $r \leftarrow 0$
  end
  Update Q-table:
  $Q(p, d) \leftarrow Q(p, d) + \alpha (r + \gamma \max_{p', d'} Q(p', d') - Q(p, d))$
  $p \leftarrow p'$
end

Algorithm 6.1: Altruistic Learning Agent (using the Social Reward Function)

6.2.1 Experimental Setup

The experimental investigation in this section evaluates the performance of the market using the same amount of learning used in the previous chapter, that is, with demand functions obtained at $450 \times 10^3$ learning episodes.

The setup for the experiments is similar to the setup used before. We consider the case of a single IPA Market with one type of resource. However, we increase the number of agents in the market to 3 and the amount of resources available to 15 units, 5 units per agent. The agents have preferences over price and amount of resources described by the utility functions $U_1(p)$ and $U_2(m)$, presented in the previous chapter.
and reproduced below in Figure 6.1. The total utility of an agent is given by the product of these functions, \( U(p, m) = U_1(p) * U_2(m) \), stressing the existence of dependency between the two properties. Neither the old nor the new amount of resource permit all the agents to have a complete satisfaction in the allocation, but they permit the analysis of the market and the learning under a condition of limited supply.

The learning experiments also include the use of the same static agents defined in the previous chapter. We run learning experiments with both reward functions, iterating the number of learning agents per static agent. The experiments are run 10 times of 500 000 learning episodes each. The learning parameters are set to \( \alpha = 0.1, \gamma = 0.9 \) and \( \epsilon = 0.4 \), and the market constant set to \( \alpha = 0.1 \).

### 6.2.2 Learning Results

The evolution of the demand functions over the episodes has shown to be dependent on the number of learning agents in the market. In experiments with 1 learning and 2 static agents, it is very consistent, developing constantly towards a well-defined shape (see Figure 6.2). There is some small instability, but an overall shape is always defined and visible. Experiments with 2 and 3 learning agents present a less consistent evolution, showing higher instabilities. This behaviour results from the same causes discussed in Section 5.5: co-adaptation and non-appropriate decay rule for the learning parameters.

Apart from that, the learnt demand functions present similar trends, with higher demands for lower prices, some middle demands in the central area and lower demands for higher prices. However, they differ in shape according to the number of learning agents in the market, the static agent and the reward function. Figure 6.2 shows examples of demand functions obtained at step 450 000 for experiments with S2. The pattern is the same for S1 and S3. In general, the application of the social reward func-
6.2. The Individual and Social Reward Functions

Figure 6.2: Examples of demand functions obtained by the learning agents, with fitted curves.

tion moves the function slightly to the left. The application of three learners lowers the upper part of the function, allowing a more smooth transition between the upper and the lower parts. The effects of these features in market are commented in the next subsection.

Even though the agents have not achieved exactly the same demand function, the functions obtained are consistent and presented a similar overall trend (see Figure 6.2).

6.2.3 Market Results

As per our approach, we evaluate the demand functions obtained at 450x10^3 learning episodes. This amount of learning was selected based on the visual analysis of the learnt demand functions, which present a very consistent shape at that stage of the
learning process. The visual analysis has shown an overall sigmoidal trend for the learnt demand functions. For this reason, we select the Sigmoidal-Boltzmann model for the curve-fitting

\[ y = a + \frac{b - a}{1 + e^{-x/c}} \]  \hspace{1cm} (6.2)

where \( a, b, c \) and \( d \) are the fitting parameters, \( x \) is the price (state) and \( y \) is the demand request (action).

Figure 6.2 shows examples of demand functions and respective fi curves for experiments with S2. The trends are similar for the other agents. The resulting demand functions are applied in the ordinary IPA market in the same configuration used during the learning stage.

6.2.3.1 Social Results

We first discuss the results from the social perspective. Figure 6.3 shows the comparison of the average NP of the market using both reward functions, by type of experiment. For comparison reasons, it also shows the NP of the market when only static agents are used. The first point to note in the figure is that, in the case of the individual reward function, the average NP of the market running only static agents is in general better than the average NP of the market running 1 or 2 learning agents. This result was obtained because the learning agents adapt their demand functions to improve their own utility, extracting the most from the interaction with the static agents but disregarding the social welfare. It is also interesting to note that the NP of the market running three S3 agents is better even than the NP of the market running only learning agents.

The good social performance of three S3 agents in the market can be explained. The analysis of their demand functions and the market configuration used in the experiments points that the equilibrium price for these agents will be around 2, which is the price area where all S3 agents will request 5 units of resource. For S1 and S2, the equilibrium will be achieved in the price area between 4 and 6. As the price S3 pays for the resources is lower than the price S1 and S2 pay, its total utility is better. It suggests that a good strategy is to wait until the price is lower enough and then request the desired demand. However, this strategy only works when the agents have complete information, which is not realistic. In the case of agents with the same demand func-
6.2. The Individual and Social Reward Functions

Figure 6.3: Average NP of the market using the individual and the social reward functions (results for the experiments with 3 learning agents are repeated for easier comparison).

This strategy can be adopted implicitly, this explains the case of S3. The implicit coordination contributes to the good social performance achieved when only learning agents are used in the market.

Still in Figure 6.3, the average NP of the market using the individual reward function increases with the number of learning agents. The same pattern is obtained using the social reward function, except for the experiments with S3, which will be commented later. The reason for the pattern is different in each case. For the case of the social reward function, the pattern is due to the fact that the learning agents actually learn to improve the social welfare, even if improving the NP means to decrease their own gains, as it is shown next. For the case of the individual reward function, on the other hand, it is due to the fact that the learning agents obtain a better social solution among them, as shown in the previous chapter. In this case, the more learning agents are present in the market, the better is the social welfare. It is important to emphasize, however, that the selfish learning agents develop some type of collective behaviour in which they together gain the most from the allocation, sometimes exploiting the static agents, but divide these gains more or less evenly among them, achieving a good social solution within the collective. While such a behaviour can decrease the social welfare of the market, since the non-learning participants can be exploited and left with few resources, it can also lead to its improvement as more selfish learning agents are added.

The exception found in the experiments with S3 and the social reward function is also generated in the effects of the good social performance achieved when the market runs more than one S3 agent. The same decrease is not seem when the individual reward function is applied because these agents learn to exploit S3, as explained above, leaving
Chapter 6. Selfish and Altruistic Learning Participants in the IPA with RL

to it few resources but increasing the social welfare with the increase of learning agents.

One important aspect shown in Figure 6.3 is the comparison between the average NP obtained by the two reward functions. The social reward function generates a better SW for the market in all cases. For experiments with S1 and S2 involving 1 and 2 learning agents, this function is also able to overcome the SW obtained by the market running only static agents. For the experiments with S3, the social welfare is almost the same as running only S3 agents. The experiments with 3 learning agents, however, have shown only a small improvement.

6.2.3.2 Individual Results

We now analyse the results from the individual perspective. Figure 6.4 compares the average utility of the agents. In experiments with 1 learning and 2 static agents, the application of the social reward function decreases the utility of learning agents and sharply increases the utility of the static agents. In experiments with 2 learning and 1 static agent, the utility of the learning agents changes very slightly, rising for experiments with S1 and S2 and dropping for experiments with S3. The utility of the static agents is also sharply augmented in this case. Finally, experiments using 3 learning agents have shown only a minor utility improvement.

Figure 6.4: Average utility received by the agents using the individual and the social reward functions (results for the experiments with 3 learning agents are repeated for easier comparison).

From the analysis of the learnt demand functions (Figure 6.2), it can be noted that the application of the social reward function causes a small movement of the demand functions to the left. This small movement is able to lower the equilibrium price of the
market. A lower equilibrium price means more resources to the static agents, increasing their utility in two components (price and amount of resource). In addition, the same small movement increases the learner utility in the price component, compensating a little bit the losses made in the resource component. The demand functions developed by the learning agents of the experiments with 3 learning agents have the same features of the demand function of S3 (discussed above): the equilibrium demand is found at a lower price, increasing the total utility in both price and amount of resources for all the agents.

6.2.4 Summary

In this section we have investigated the impacts of selfish and altruistic learning agents in the IPA with RL using the same amount of learning used in the previous chapter.

Under a social perspective, the introduction of altruistic learning agents is able to generate better results than the introduction of selfish learning agents. This outcome is consistent with expectation and intuition. In experiments including static agents, the increase in the social welfare is followed by a small decrease in the individual utility of the learning agents and a sharp increase in the utility of the static agents. In experiments including only learning agents, that increase is generated by a slightly rise in the individual utility of the agents.

One important observation to make is that, although being able to generate a better social performance, the improvement obtained with the altruistic learning agents over the selfish ones for the case with 3 learning agents was only marginal. The next two sections will further exemplify and explore this behaviour.

6.3 Performance over the Learning Episodes

In this section, we investigate how the market behaves using demand functions obtained throughout the learning process. Such an investigation is important for two reasons. First, it helps us to investigate how much learning is enough, aiming at to reduce the number of learning episodes needed. And second, it helps us to understand how RL affects the individual and social behaviour of the agents during the learning process.
6.3.1 Experimental Setup

The experiments are essentially the same performed in the previous section with the difference that demand functions are extracted and evaluated in intervals of 5000 learning episodes.

The market is set to 15 units of resources. The learning parameters are set to $\alpha = 0.1$, $\gamma = 0.9$ and $\epsilon = 0.4$, and the market constant set to $\alpha = 0.1$. We run learning experiments with both reward functions, iterating the number of learning agents for static agent. The experiments are run 10 times of 500 000 learning episodes each.

6.3.2 Market Results

The evaluation in the market, as in the previous sections, are made with the trend of the actual demand functions learnt by the agents. The trends are obtained using the Sigmoidal-Boltzmann model for the curve-fitting process (see section 6.2.3).

6.3.2.1 Social Results

Figure 6.5 shows the comparison of the average NP resulting from the use of the individual and the social reward functions over the learning episodes. The main point to note is that the NP achieves a level of relative stability very quickly, for both reward functions. This level was reached before 100 000 learning episodes in most of the experiments, suggesting that the learning process can be shorter than the 450 000 learning episodes used previously.

In general, the results presented in Figure 6.5 confirm the ones found in the previous section. The average NP of the market using the individual reward function increases with the number of learning agents. The application of the social reward function follows the same pattern, with the exception of S3 (see Section 6.2.3 for a discussion on this).

A quite interesting behaviour is found in the experiments with 3 learning agents. The results presented in the previous section have shown a minor improvement in the average NP when the social reward function is applied. The investigation of the same case but over the learning episodes (Figure 6.5-c) shows both reward functions delivering
6.3. Performance over the Learning Episodes

very similar results. The same feature is found on the individual utility of the agents (Figure 6.6-c).

Figure 6.5 also shows that the stability of the average NP over the learning episodes depends on the number of learning agents in the market. For experiments with 1 learning agent, a very stable level is achieved early and maintained along the learning episodes. Experiments with 2 and 3 learning agents present only a relative stability. The average NP evolves to a level, which was maintained, but fluctuate around it. This behaviour also results from the same causes discussed in Section 6.2.2: co-adaptation and non-appropriate decay rule for the learning parameters.

Another very interesting point is presented in Figure 6.5-g, which shows the average NP obtained for experiments using 1 learning and 2 S3 agents. The figure reveals the average NP of the market using the individual reward function decreasing over the learning episodes. This fall results from the fact that the learning agents developed demand functions able to exploit S3’s demand function. This exploitation has already been commented in the previous sections and is clearly understood from Figure 6.6-g, which shows a large difference when the individual utility received by those agents are compared. A decrease in the utility received by S3 along the learning episodes is also noticeable there.

6.3.2.2 Individual Results

We now turn to the individual utility of the agents. Figure 6.6 shows the comparison of the average utility obtained by learning and static agents over the learning episodes. As in the social level, there is a quick evolution of the individual utilities to a point of relative stability, which is achieved around or before the learning episode 100 000 for most of the agents. One exception is found for experiments with 1 learning and 2 S2 agents (Figure 6.6-d), where the average utility of the learning agents using the social reward function take around 200 000 learning episodes to reach the stable level. Another exception is found for the experiments with 1 learning and 2 S3 agents, where the average utility received by the static agents decreases slowly over the learning episodes.

In general, the results found at the individual level also confirm the ones presented in the previous section. The application of the social reward function decreases the utility obtained by the learning agents and sharply increases the utility of the static agents.
Figure 6.5: Comparison of Average NP obtained by the market using the individual and the social reward functions over the learning episodes (figure c, f and i show the results of the same experiments, with 3 learning agents, and are repeated for easier comparison).

6.3.3 Summary

The evaluation presented in this section has shown that the learning process can be shorter than the 450 000 learning episodes used before for the settings. In general, the quality of the demand functions has converged to a level of relative stability around or before the episode 100 000. After this point, no substantial individual or social gains were made to justify a longer learning process.
6.4 Multiple Learning Agents in the Market

In this section, we further investigate the case where the market is composed only of learning agents. The results presented in the previous section for this case, using 3 learning agents, are particularly interesting given the fact that the average SW of the market using the individual reward function, which uses only local information and transforms the agents into competitors, were similar to the average SW of the market using the social reward function, which uses global information and transforms the agents into cooperators. The objective of this section is to investigate if this result
Chapter 6. Selfis and Altruistic Learning Participants in the IPA with RL

holds for a higher number of agents.

6.4.1 Experimental Setup

We run learning experiments using 2, 4, 6 and 8 learning agents. Each configuration is run 20 times of 10,000 learning episodes. The demand functions are extracted and evaluated in intervals of 100 learning episodes. The learning parameters are set to $\alpha = 0.1$, $\gamma = 0.9$ and $\epsilon = 0.4$, and the market constant set to $\alpha = 0.05$.

6.4.2 Market Results

Figure 6.7 shows the evolution of the market’s average NP using the individual and the social reward functions. Again, there is a quick evolution to a level of relative stability. The average NP evolves to that level before 1000 learning episodes and fluctuate around it afterwards. The same type of instability was also commented in the previous section.

As in the previous section, both reward functions present similar results and approach the optimal social welfare. It is not distinctive which one performs better in general. The same type of result is found in the individual utilities received by the agents, shown in Figure 6.8.

![Figure 6.7: Average NP obtained by the market using the individual and social reward functions over the learning episodes.](image)

An important comment to make is that a number of experiments in this scenario have presented some problems in the curve-fitting not converging or generating demand...
functions that are unable to achieve equilibrium. These problems are due to the short learning process used here. Experiments performed using the same amounts of learning used in the previous section, with checkpoints at each 5000 learning episodes for a total of 500 000, presented less problems. Nevertheless, the experiments that have not presented problems, achieved good outcomes, as the graphs show.

The most important observation we can draw from these experiments is that, using both reward functions, there is a trend for the learning agents to divide the resources equally. While this strategy seems obvious for the case of the social reward function, since the optimal reward is received when the price is low enough and all the agents receive an equal share of the allocation, it is not so intuitive for the individual reward function.

6.4.3 Summary

The experiments presented in this section indicate that the interesting phenomenon where both reward functions deliver similar results when the market consists exclusively of learning agents holds when the number of agents is scaled.
6.5 Agents with Non-symmetric Preferences

The main outcome of the previous sections is the observation that, when the market is composed of learning agents only, the individual reward function can generate social results that are similar to the results obtained with the social reward function. Such an outcome is potentially important in domains where the social utility should be maximized but the agents are unwilling to reveal private preferences. The previous sections, however, have evaluated only the case of agents with symmetric preferences, i.e. when the agents use the same utility functions, generating a symmetric payoff table for the game. In this section we investigate the case of agents with non-symmetric preferences.

6.5.1 Experimental Setup

For the experiments in this section, we define 2 different utility functions for each attribute, the price and the amount of resource, and perform experiments using the possible combinations between them. Figures 6.9 and 6.10 show the utility functions. Table 6.1 presents the configuration of the experiments.

The experiments described in Table 6.1 are as follows. Experiment A reflects the situation of agents with symmetric preferences, which is the same configuration investigated in the previous sections. In Experiment B the agents have the same preferences for amount of resource but the second agent is willing to pay more. Experiment C investigates the situation in which both the agents are willing to pay the same price but the second agent needs more resources. In Experiment D, the first agent needs less resource and prefers to pay less for it. Finally, Experiment E studies the case where the first agent wants less resource but is willing to pay more for it.

![Utility functions for price.](image)

Figure 6.9: Utility functions for price.

The market is set with 10 units of resources, allowing for the analysis of the market under a condition of limited supply.
6.5. Agents with Non-symmetric Preferences

![Utility functions for amount of resource.](image)

**Figure 6.10:** Utility functions for amount of resource.

<table>
<thead>
<tr>
<th>Type</th>
<th>Agent 1</th>
<th>Agent 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Wants same $(U_{M1})$</td>
<td>Wants same $(U_{M1})$</td>
</tr>
<tr>
<td></td>
<td>Pays same $(U_{P1})$</td>
<td>Pays same $(U_{P1})$</td>
</tr>
<tr>
<td>B</td>
<td>Wants same $(U_{M1})$</td>
<td>Wants same $(U_{M1})$</td>
</tr>
<tr>
<td></td>
<td>Pays less $(U_{P1})$</td>
<td>Pays more $(U_{P2})$</td>
</tr>
<tr>
<td>C</td>
<td>Wants less $(U_{M1})$</td>
<td>Wants more $(U_{M2})$</td>
</tr>
<tr>
<td></td>
<td>Pays same $(U_{P1})$</td>
<td>Pays same $(U_{P1})$</td>
</tr>
<tr>
<td>D</td>
<td>Wants less $(U_{M1})$</td>
<td>Wants more $(U_{M2})$</td>
</tr>
<tr>
<td></td>
<td>Pays less $(U_{P1})$</td>
<td>Pays more $(U_{P2})$</td>
</tr>
<tr>
<td>E</td>
<td>Wants less $(U_{M1})$</td>
<td>Wants more $(U_{M2})$</td>
</tr>
<tr>
<td></td>
<td>Pays more $(U_{P2})$</td>
<td>Pays less $(U_{P1})$</td>
</tr>
</tbody>
</table>

Table 6.1: Experiments’ configuration

We run 20 learning experiments for each configuration and reward function. Each experiment consists of 500,000 learning episodes. The learning parameters are set to $\alpha = 0.1$, $\gamma = 0.95$ and $\epsilon = 0.35$, and the market constant set to $\alpha = 0.05$.

### 6.5.2 Market Results

Once again, the evaluation in the market is made with the trends of the actual demand functions learnt by the agents obtained by curve-fitting using the Sigmoidal-Boltzmann model (see section 6.2.3). We will next present the results.

#### 6.5.2.1 Social Results

We first present and discuss the results from the social perspective. Figure 6.11 shows the comparison of the median SW of the market over the learning episodes for both reward functions. The median approaches the optimal SW in all configurations.
achieves a level of relative stability before 200,000 learning episodes and fluctuate slightly afterwards.

![Figure 6.11: Median Nash Product (NP) of the market for the Individual and Social Reward Functions (RF) over the Learning Episodes (LE).](image)

It is also noticeable in Figure 6.11 that the social reward function generates better SW during the first learning episodes. This result is expected as those agents explicitly learn how to maximize the SW.

### 6.5.2.2 Individual Results

We now turn to the individual utility of the agents. Figure 6.12 shows the comparison of the median utility received by the agents over the learning episodes. It is important to recall that the utility of an agent in our scenario is actually the product of its utility for price and amount of resource, $U(p, m) = UP(p) * UM(m)$.

**Experiment A** reflects the situation of agents with symmetric preferences. The optimal SW for this configuration is 0.64. The fairest Pareto-Optimal (PO) allocation that maximizes the SW produces a utility of 0.8 to each agent. To achieve it, the price has to be low enough ($p \leq 1$), so the agents’ utility for the price is optimal (i.e. $UP(p) = 1$), and they need to obtain 5 units of resources each. It can be seen in the graphs that both reward functions are able to direct the agents’ utility to the optimal levels. These results confirm the results found in sections 6.2 to 6.4 for the similar case.
Figure 6.12: Median Utility (U) of the agents for the Individual and Social Reward Functions (RF) over the Learning Episodes (LE).
In **Experiment B** the agents have the same preferences for amount of resource but the second agent is willing to pay more. The optimal social welfare in this case is also 0.64. The graphs show that both reward functions direct the agents to 0.8, which maximizes the SW. The graphs also show a difference between the utility of the agents 1 and 2 in the first learning episodes. It occurs because agent 2’s utility function for price is easier to be satisfied so this agent learns how to maximize it very early. Nevertheless, agent 1 also approaches the optimal utility after some episodes, indicating that the agents have learnt to coordinate the demand functions in order to decrease the price. This learning process was quicker for the social reward function because it carries information about the social utility in the reward signal. Such information is not carried by the individual reward function, making the coordination more difficult to emerge.

**Experiment C** investigates the situation in which both the agents are willing to pay the same price but the second agent needs more resources. The optimal social welfare in this case is around 0.25. In the fairest PO allocation, each agent receives a utility of 0.5, which corresponds to demand levels 3.5 for the first agent and 6.5 for the second (see Figure 6.10). The graph for the individual reward function shows the median utilities of both agents fluctuating around 0.5, indicating that they have learnt demand functions able to find the equilibrium around the optimal levels. The graph for the social reward function, however, shows a gap between the agents’ utility. This gap was generated because the agents developed demand functions that directed the equilibrium to demand levels around 4 and 6. We comment on this behaviour below.

In **Experiment D**, the first agent needs less resource and prefers to pay less for it. The results found for this case are a mixture between the ones found for experiments B and C. As in B, the agents coordinate to lower the price so both agents maximize $U_P$. As in C, there is a gap between the utility of the agents when the social reward function is used. Similar behaviour is observed in **Experiment E**, which studies the case where the first agent wants less resource but is willing to pay more for it.

To illustrate the cases of the experiments C, D and E, Figures 6.13 and 6.14 present the evolution of the individual components for experiment E. Figure 6.13 illustrates how the demand requests evolve over the learning episodes. Note that they stabilize around 6.5 and 3.5 for the individual reward function and around 6 and 4 for the social reward function. This behaviour generates a gap in the agents’ utility for amounts of resource, being the responsible for the gap between the final utility of the agents using the social reward function. An interesting aspect shown in Figure 6.14 is the evolution of the utility for the price. Note that the agent 1’s utility for this component is optimal from
6.5. Agents with Non-symmetric Preferences

the very first learning episodes while the agent 2’s utility starts lower but improves very quickly and stabilize at the optimal level. Similar behaviour was found in experiments B and D. Also note the price decreasing over the learning episodes, showing that the agents learn to coordinate the demand functions in order to decrease the price.

![Graph: Median demand (d) requests of the agents over the Learning Episodes (LE) for experiments of type E.]

![Graph: Median Utility (U) for amount of resource (UM) and price (UP) of the agents in experiments of type E.]

Figure 6.13: Median demand (d) requests of the agents over the Learning Episodes (LE) for experiments of type E.

Figure 6.14: Median Utility (U) for amount of resource (UM) and price (UP) of the agents in experiments of type E.

Figure 6.15 presents the relationship between the individual utility and the social welfare for the experiment E. The graph shows that the social reward function is noisy in the sense that small changes in the demand requests produce large differences in the individual utility of the agents but reflect very little changes in the social reward signal. For example if the equilibrium is found at the demand levels 3.5 and 6.5 for agent 1 and agent 2, respectively, the optimal social reward of 0.25 is achieved and the agents receive a utility of 0.5 each. On the other hand, if the equilibrium is found at 4 and 6, the social reward produced is very close to the optimal one, i.e. 0.24, but the utility of the agents is significantly different, 0.4 and 0.6. This illustrates that the individual results obtained by the social reward function in the experiments C, D and E are reasonable. However, there are two open issues. First, the relationship shown in Figure 6.15 is similar for the experiments A and B, but the results for these experiments do not
present the gap in the median utility of the agents. And second, the median demand requests at the equilibrium have evolved to 4 and 6, rather than to 3 and 7, which would produce the same social reward.

![Figure 6.15: Relationship between individual utility and social welfare](image)

To sum up, regarding the agents’ preferences for prices, the experiments show that the agents can learn to coordinate the demand requests in order to lower the price and, consequently, maximize $U_P(p)$. This behaviour held true for both reward functions even when different preferences are present. For the amount of resources, in experiments where the agents have the same preferences (A and B), both reward functions are able to approximate the solution to the optimal and fairest allocation. For experiments in which different preferences are applied (C, D and E), only the individual reward function is able to do it.

### 6.5.3 Summary

In this section we have investigated the application of the individual and the social reward functions in the case of agents with non-symmetric preferences. The experiments have shown that the individual reward function is able to generate social results that are similar to the results obtained with social reward function also in this case. Remarkably, the individual-based function was also able to approximate the solution to the fairest Pareto-Optimal allocation in situations where the social-based one failed.
6.6 Conclusions

In this chapter we have presented further experiments in the IPA with RL. In particular, we have investigated the impacts of selfish and altruistic learning agents in the mechanism. The impacts of both types of strategic learning agents on the individual and the social performance of the market were experimentally investigated and the results compared.

The main outcome of the experimental investigations presented in this chapter is that the market composed exclusively of selfish agents is able to achieve social performance similar to the performance obtained by the market composed exclusively of altruistic agents, both reaching near-optimal SW, measured by the Nash Product. In addition, the selfish agents are able to approximate the solution to the fairest PO allocation in situations where the altruistic agents fail.

Therefore, the experiments have shown that, in the IPA with RL, it is possible to simultaneously optimise both individual and social performances when the market is composed of strategic learning agents. In particular, they have shown that it is possible to optimise both efficiency measures through the application of selfish agents using only local information to learn. Such an outcome is significant not only for the market-based resource allocation domain but also for a series of other domains where individual and social utility should be maximized but agents are not guaranteed to act cooperatively in order to achieve it or they do not want to reveal their private preferences.

The experiments in this chapter have also served to evaluate the new technique introduced in Chapter 4 to support the application and scalability of RL in problems with continuous spaces. The approach was initially introduced to deal with characteristics found in the IPA with RL scenario. In particular, the discretization of the action and state spaces was introduced to deal with the continuity of these components and then enable the application of the Q-learning algorithm. The application of the curve-fitting was necessary to recover the continuity lost with the discretization, and also to avoid the local instabilities and non-monotonicity present in the actual demand functions learnt by the agents. The combination of these two techniques, however, has good potential for reducing the amount of learning required in RL and MARL problems.

The potential of the new technique, although not fully explored in this thesis, is evi-
denced by the experimental results obtained in this chapter. In particular, the results show the market’s performance converging to a level of relatively stability around or before 100 000 learning episodes, when, according to the visual analysis performed in the preliminary study (previous chapter, sections 5.4 and 5.5), the shapes of the actual demand functions learnt by the agents are not well-defined yet. This amount of learning is much shorter than the 450 000 learning episodes originally used. Therefore, the results show that the application of the new technique leads to the reduction of the required amount of learning.
Chapter 7

Theoretical Analysis of the IPA with RL

This chapter presents the theoretical analysis of the results found in the IPA with RL. It first performs a static analysis of the scenario using concepts from game-theory to highlight the properties of the solutions found by the agents. It then investigates the agents’ behaviour from the perspective of the dynamic process generated by the agents’ learning algorithm.

7.1 Introduction

The experimental investigation on the IPA with RL has revealed that the market composed exclusively of selfish learning agents can achieve social performance similar to the performance obtained by the market composed exclusively of altruistic learning agents, both achieving near-optimal social welfare measured by the Nash Product function. In this chapter we investigate the reasons for this result.

The study is conducted in two parts. In the first part we perform a static analysis using concepts from game-theory to highlight the properties of the solutions found by the agents. Based on a simplified version of the actual scenario addressed in the previous section, we develop two five-action two-player games that simulate the application of the reward functions and identify the Nash Equilibria and Pareto Optimal solutions of
these games. We then perform a series of learning experiments in the games to investigate what solutions the agents find and how coherent they are with the behaviour found in the previous chapter. In the second part we investigate the agents’ behaviour from the perspective of the dynamic process generated by the learning algorithm employed by them. We develop a theoretical model of the Multiagent $Q$-learning with $\epsilon$-greedy exploration algorithm and apply it to investigate the dynamic behaviour of the agents in the formulated games. Section 7.2 presents the static analysis and Section 7.3 presents the dynamic analysis.

### 7.2 Static Analysis

The static analysis of the IPA with RL in this section is based on the framework of Stochastic Game Theory. Stochastic games (SGs) extend the Markov Decision Process (MDP) framework to multiple agents, where the next state and one agent’s reward depend on the joint actions of the players. This framework has been advocated as a possible way to describe multiagent learning problems [14].

#### 7.2.1 The Market-based Resource Allocation Games

Modelled as an SG, the IPA with RL learning problem becomes the one of learning the general-sum SG described by a tuple $(n, P, D_1 \cdots D_n, T, R_1 \cdots R_n)$, where $n$ is the number of learning agents (players), $P$ is the set of possible resource prices (state space), $D_i$ is the set of possible demand requests available to agent $i$ (action space) with $D$ being the joint demand requests space $D_1 \times \cdots \times D_n$, $T$ is the IPA’s price update rule $P \times D \times P \rightarrow [0, 1]$ (state transition function), and $R_i$ is the reward function for the $i$th player $P \times D \rightarrow \mathbb{R}$.

For this game, the best-response function for agent $i$, $BR_i(\pi_{-i})$, is the set of all demand functions (policies) that are optimal given that the other agent(s) use the joint demand function $\pi_{-i}$. A demand function $\pi_i$ is optimal given $\pi_{-i}$ if and only if,

$$\forall \pi'_i \in \Pi_i \quad V_i(\pi_i, \pi_{-i}) \geq V_i(\pi'_i, \pi_{-i})$$

where $\Pi_i$ is the set of possible demand functions for agent $i$ and $V_i$ is the expected
reward for using demand function $\pi_i$, given $\pi_{-i}$.

A Nash Equilibrium (NE) occurs when all the players are using best-response policies. So, for us, a NE is a joint demand function, $\pi_i = 1, \ldots, n$, with

$$\forall i = 1, \ldots, n \quad \pi_i \in BR_i(\pi_{-i})$$

An important result shown by Fink [39] guarantees that every $n$-player general-sum discounted stochastic game possesses at least one NE point in stationary strategies. A stationary strategy is one that depends only on the current state.

According to Fulda and Ventura [42], from the system perspective, an optimal solution would be any one which is a NE and Pareto-Optimal (PO). A solution is Pareto-Optimal if there is no other solution that can improve one agents’ outcome without deteriorating others’. $Q$-learning has been developed to learn best-response policies in MDPs. It is also well-known that $Q$-learning has no proof of convergence in multiagent learning scenarios [14]. However, if $Q$-learning ever converges in those scenarios, it will do to a NE, since all the agents will be playing best-response policies [14]. The remaining questions are to which NE the agents will converge to and how optimal this NE is.

In research addressing those questions, Claus and Boutilier [30] investigated the dynamics of RL in cooperative multiagent systems. Their analysis of independent $Q$-learners for cooperative games, where the agents have the same payoff table, show that the agents tend to converge to the most profitabl equilibrium in simple games. Fulda and Ventura [42] also studied the cooperative case, formulating conditions in which the optimal solution will be chosen. Those results are important to the analysis of our social reward function. However, it should be noted that the use of the individual reward function generates a different game.

The full game-theoretical analysis of the IPA with RL’s scenario is quite complex because of its large number of states and actions and, due this very complexity, may not be very informative. Therefore, we use a simplifie version of the scenario. We consider the case of a single IPA market with 4 units of indivisible resources, so only discrete units can be sold or requested, and agents with preferences over the amount of resources only. To describe the preferences, the agents use an increasing utility function. The price of the resources is not important at this stage as we are more interested in explaining why the agents reach the equilibrium that optimizes the SW. So,
we consider the price to be constant.

Based on the simplified scenario, we can define a two-player, five-action stage game for each reward function. A stage game is a special case of a stochastic game where the number of states is 1. Tables 7.1 and 7.2 show the games. The actions represent the possible demand requests. Note that the joint-actions in the minor diagonal represent the possible equilibrium states for the market. Also note that the payoffs simulate the reward functions, generating a competitive game with symmetric payoff table for the individual reward function and a cooperative game for the social reward function. The selfish learning agents play the individual reward game and the altruistic learning agents play the social reward game. In the tables, pure Nash Equilibria (NEs) are presented in **bold** and Pareto-Optimal (PO) joint-actions in *italic*.

**Table 7.1: Individual Reward Game.**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td><strong>0,4</strong></td>
</tr>
<tr>
<td>1</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td><strong>1,3</strong></td>
<td>0,0</td>
</tr>
<tr>
<td>2</td>
<td>0,0</td>
<td>0,0</td>
<td><strong>2,2</strong></td>
<td>0,0</td>
<td>0,0</td>
</tr>
<tr>
<td>3</td>
<td>0,0</td>
<td><strong>3,1</strong></td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
</tr>
<tr>
<td>4</td>
<td><strong>4,0</strong></td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
</tr>
</tbody>
</table>

**Table 7.2: Social Reward Game.**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td><strong>0,0</strong></td>
</tr>
<tr>
<td>1</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td><strong>3,3</strong></td>
<td>0,0</td>
</tr>
<tr>
<td>2</td>
<td>0,0</td>
<td>0,0</td>
<td><strong>4,4</strong></td>
<td>0,0</td>
<td>0,0</td>
</tr>
<tr>
<td>3</td>
<td>0,0</td>
<td><strong>3,3</strong></td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
</tr>
<tr>
<td>4</td>
<td><strong>0,0</strong></td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
</tr>
</tbody>
</table>

Considering a RL process, the agents do not know the payoff tables, they choose actions independently and play the games infinitely times. In this case, the most reasonable solution for the social reward game is learning to play \( (2,2) \), which is the only Nash Pareto-Optimal equilibrium. For the individual reward game, however, all the joint-actions in the minor diagonal are NEs and PO. Suppose agent 1 plays action 4, whatever agent 2 plays, it will earn 0. Thus, agent 2 will select random actions, making agent 1 earn in average 4/5. Therefore, on this particular game, playing action 0 or action 4 seems to be the least reasonable strategy. Now suppose agent 1 plays action 3. In this case, agent 2 should play action 1 to earn 1, and agent 1 earns 3. Using Q-learning, however, each agent has a probability of choosing random actions in each turn. By exploring the game in this fashion, agent 2 will eventually discover that other actions are more attractive and may change its strategy, making agent 1 adapt to it. The same reasoning applies to agent 1.

To better understand how that dynamics proceeds, the next section presents a series of learning experiments in the games.
7.2. Learning Experiments

We set the learning parameters to $\alpha = 0.1$, $\gamma = 0.9$ and $\epsilon = 0.6$, and modify the update rule to $Q(s, a) = Q(s, a) + \alpha(r(s, a) - (\gamma * Q(s, a)))$ since only one state is available. Figures 7.1, 7.2 and 7.3 present the results obtained out of 1000 experiments.

Figures 7.1 and 7.2 show the number of times each action was chosen by the players over the learning episodes for the individual and the social reward games, respectively. The main point to note is that action 2 is the most chosen in both games. The agents develop the preference for this action very early. For the individual reward game, note that action 4, which is the one with best payoff, is only the fourth action in preference. Not surprisingly, action 0 is quickly ignored in both cases.

Figure 7.3 shows the number of times each joint-action was chosen. Once again it can be noted that joint-action $\langle 2, 2 \rangle$ is the solution preferred by the agents. It can also
be noted an equilibrium between the joint-actions $(1,3)$ and $(3,1)$. In the individual reward game, this equilibrium is maintained over the episodes, suggesting that the agents keep trying to achieve a higher payoff, even if the joint-action $(2,2)$ is more attractive. In the social reward game, the likelihood of choosing $(1,3)$ and $(3,1)$ lose its strength over the episodes, which is quite obvious since both players can do better by choosing $(2,2)$.

![Figure 7.3: Joint-actions chosen by the players over the learning episodes](image)

The results from these experiments are coherent with the behaviours found in the investigated IPA market. It can be clearly seen that choosing to share the resources equally is Pareto-Optimal and also a Nash Equilibrium. Therefore, the convergence (or pseudo-convergence as no decay rules have been applied) to such a strategy via Q-learning is rational for both reward functions. As commented above, similar results have been found in cooperative games [30, 42, 94], which is the case of the social reward game. For the individual reward game, however, further investigations are necessary.

### 7.2.3 Summary

In this section we have performed a static analysis of the the IPA with RL’s scenario. Based on a simplified version of the scenario, we developed two five-action two-player games that simulate the application of the individual and the social reward functions and performed a game-theoretical analysis of these games. We then performed a series of learning experiments to investigate the coherence between the results found in the games and the results found in the actual IPA with RL.

The game-theoretical analysis has revealed the presence of Nash Equilibrium and Pareto-Optimality at the joint-actions in which the resources are shared equally, in both
games, which maximizes the social welfare. The learning experiments have shown the 
*pseudo*-convergence of the agents to those joint-actions, reproducing the behaviour 
found in the actual IPA with RL.

7.3 Dynamic Analysis

The game-theoretical analysis performed in the previous section is useful to highlight 
the properties of the solutions found by the learners in the IPA with RL. However, it 
does not capture the dynamic essence of the learning process. In this section we study 
the behaviour of the agents from the perspective of the dynamics of the learning al-
gorithm employed by them. For this, we develop a theoretical model of Multiagent 
$Q$-learning with $\epsilon$-greedy exploration and apply it in the market-based resource alloca-
tion games formulated in the previous section.

7.3.1 A Model of Multiagent $Q$-learning with $\epsilon$-greedy Exploration

The development of mechanisms to understand and model the expected behaviour of 
multiagent learners is becoming increasingly important as the area fin application 
in wide variety of domains. The advantages of having such mechanisms are many. 
For instance, MAL systems usually have parameters that need to be adjusted so the 
overall behaviour of the system can be optimized. The usual approach to setup those 
parameters is to execute extensive experimentation with different configuration and to 
aggregate the outcomes in the hope of finding some useful information [116]. A better 
understanding of the expected behaviour can help the system’s designer in the task.

The $Q$-learning is certainly one of the most studied Reinforcement Learning algo-
rithms and has been applied with success in several domains, from relatively simple 
toy problems, such as Cliff-Walking [108], to more complex ones, such as web-based 
education [52] and face recognition [47]. Initially proposed for single-agent environ-
ments, the simplicity and effectiveness of this algorithm has led to its application also 
in multiagent configurations for example Galstyan et al. [43], Servin and Kudenko 
[95] and Ziogos et al. [132]. In this case, however, its supporting theoretical framework and convergence guarantees are lost.

One of the difficultie of the multiagent case is to cope with the very dynamic environ-
ment generated by multiple learners. There is also the co-adaptation effect, in which one agent adapts its strategy to the others’, and vice-versa, in a cyclic fashion. In addition, the rewards that one agent receives depend on the actions of the other agents. All these features make it especially difficult to predict and to model the learning behaviour [79].

An important research in the area is the work of Tuyls et al. [114]. The authors studied the case of Q-learning agents with Boltzmann exploration. They developed a continuous time model for the learning process and have shown a link between the model and the Replicator Dynamics (RD) of Evolutionary Game Theory [49]. The main principle of the RD is that the growth in the probability of playing a given action is directly proportional to the performance of that action against the others. The ε-greedy mechanism, however, produces different dynamics. This mechanism defines a semi-uniform probability distribution in which the current best action is selected with probability $1 - \epsilon$ and a random action with probability $\epsilon$. Hence, that research cannot be directly applied in our case.

Apart from the necessity of explaining our results in the IPA with RL, the importance of obtaining such a model is justified through its large number of applications. For example, Galstyan et al. [43] applies the algorithm to develop a decentralized resource allocation mechanism, Ziogos et al. [132] to develop bidding strategies in auction mechanisms and Li and Yahyapour [71] to develop negotiation strategies for Grid resource allocation.

To develop our model, we analyse a continuous-time version of the Q-learning update rule and study how the ε-greedy mechanism and the presence of other agents affect it. We then use this analysis to model the problem as a system of difference equations which is used to calculate the expected evolution of the Q-values and, consequently, the expected behaviour of the agents.

The study is organized as follows. In Section 7.3.1.1 we present a brief review of the Q-learning algorithm with ε-greedy exploration and its extension to multiagent scenarios. In Section 7.3.1.2 we present the analysis and the equations to model the behaviour of the agents. In section 7.3.1.3 we provide the evaluation of the framework by comparing the theoretical behaviour obtained by the model with the behaviour found in real experimentation with Q-learning in typical games selected from the literature. Finally, in Section 7.3.1.4 we discuss the works directly related to the model.
7.3. Dynamic Analysis

7.3.1.1 Background

In this section we briefly review the Q-learning algorithm, the $\epsilon$-greedy action-selection mechanism and the extension of Q-learning to multiagent scenarios.

**Single-agent Q-learning**  The task of a Q-learning agent is to learn a mapping from environment states to actions so as to maximize a numerical reward signal [108]. The model is formalized by a tuple $(S, A, T, R)$, where $S$ is a discrete set of environment states, $A$ is a discrete set of actions, $T$ is a state transition function $S \times A \times S \rightarrow [0, 1]$, and $R$ is a reward function $S \times A \rightarrow \mathbb{R}$. One of the attractors of Q-learning is that it assumes no knowledge about state transitions and reward functions, which must be learned from the environment. In each step, the agent receives a signal from the environment indicating its state $s \in S$ and chooses an action $a \in A$. Once the action is performed, it changes the state of the environment, generating a reinforcement signal $r \in R$ that is then used to evaluate the quality of the decision by updating the corresponding $Q(s, a)$ values.

The $Q(s, a)$-values are estimations of the $Q^*(s, a)$-values, which represent the sum of the immediate reward obtained by taking action $a$ at state $s$ and the total discounted expected future rewards obtained by following the optimal policy thereafter. By updating $Q(s, a)$, the agent eventually makes it converge to the $Q^*(s, a)$. The optimal policy is then followed by selecting the actions where the $Q^*$-values are maximum. The formula used to update the $Q$-values is:

$$Q(s, a) := Q(s, a) + \alpha (r(s, a) + \gamma \max_{a'} Q(s', a') - Q(s, a))$$  \hspace{1cm} (7.1)

where $0 < \alpha < 1$ is the learning rate and $0 < \gamma < 1$ is the discount rate.

Considering that the probabilities of making state transitions $T$ and receiving specific reinforcement signals $R$ do not change over time, i.e. a stationary environment, if each action is executed in each state an infinite number of times and $\alpha$ is decayed appropriately, the $Q$-values will converge with probability 1 to the optimal ones [55, 108, 113].

**The $\epsilon$-greedy Mechanism**  An important component of Q-learning is the action selection mechanism. This mechanism is responsible for selecting the actions that the
agent will perform during the learning process. Its purpose is to harmonize the trade-off between exploitation and exploration such that the agent can reinforce the evaluation of the actions it already knows to be good but also explore new actions.

In our experiments in the IPA with RL we have applied the $\epsilon$-greedy exploration. This mechanism selects a random action with probability $\epsilon$ and the best action, i.e., the one that has the highest $Q$-value at the moment, with probability $1-\epsilon$. As such, it can be seen as defining a probability vector over the action set of the agent for each state. If we let $x = (x_1, x_2, ..., x_j)$ be one of these vectors, then the probability $x_i$ of playing action $i$ is given by:

$$x_i = \begin{cases} (1 - \epsilon) + (\epsilon/n), & \text{if } Q(s, a) \text{ is currently the highest } \\ \epsilon/n, & \text{otherwise} \end{cases}$$

where $n$ is the number of actions in the set.

### Multiagent Q-learning

Multiagent $Q$-learning is a natural extension of single-agent $Q$-learning to multiagent scenarios. In this approach, the agents are equipped with a standard $Q$-learning algorithm each and learn independently without considering the presence of each other in the environment. The rewards and the state transitions, however, depend on the joint actions of all agents. The problem is formalized as the Stochastic Game defined by the tuple $(n, S, A_1 \cdots A_n, T, R_1 \cdots R_n)$, where $n$ is the number of players, $S$ is the set of states, $A_i$ is the set of actions available to agent $i$ with $A$ being the joint action space $A_1 \times \cdots \times A_n$, $T$ is the transition function $S \times A \times S \rightarrow [0, 1]$, and $R_i$ is the reward function for the $i$th player $S \times A \rightarrow \mathbb{R}$. Note that both $T$ and $R$ are defined over the joint action space.

#### 7.3.1.2 The Model

We now present our model for multiagent $Q$-learning with $\epsilon$-greedy exploration. To develop this model, we study how the $\epsilon$-greedy mechanism and the presence of other agents affect the learning process of one agent. For this, we first show the derivation of a continuous time equation for the $Q$-learning rule. We then analyse the limits of this equation for the case of a single learner and show how they change dynamically when multiple learners are considered. Finally, we show how the $\epsilon$-greedy mechanism
affects the shape of the modelled function. The observations and results from this study are used to develop a system of difference equations to model the behaviour of the learners.

For simplicity of explanation, to develop the model we consider scenarios composed of 2 agents with 2 actions each\(^1\) and a single state. The reward functions of the agents in this case can be described using payoff tables of the form:

\[
A = \begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix} \quad B = \begin{bmatrix}
b_{11} & b_{12} \\
b_{21} & b_{22}
\end{bmatrix}
\]

where A describes the rewards, or payoffs, for the first agent and B the rewards for the second. Given the existence of only one state, the Q-learning update rule can be simplified to

\[
Q_{a_i} := Q_{a_i} + \alpha (r_{a_i} - Q_{a_i})
\]

(7.3)

where \(Q_{a_i}\) is the Q-value of agent a for action \(i\) and \(r_{a_i}\) is reward that agent a receives for executing action \(i\). Please note the change in notation from the one applied throughout the thesis.

**Analysis** We start the study by rewriting the update rule for the first agent as follows:

\[
Q_{a_i}(k+1) - Q_{a_i}(k) = \alpha (r_{a_i}(k+1) - Q_{a_i}(k))
\]

(7.4)

This difference equation describes the absolute growth in \(Q_{a_i}\) between times \(k\) and \(k+1\). To obtain its continuous time version, consider \(\Delta t \in [0, 1]\) to be a small amount of time and

\[
Q_{a_i}(k + \Delta t) - Q_{a_i}(k) \approx \Delta t \times \alpha (r_{a_i}(k + \Delta t) - Q_{a_i}(k))
\]

\(^1\)this constraint is relaxed in our evaluation of the market-based resource allocation games (Section 7.3.2)
Chapter 7. Theoretical Analysis of the IPA with RL

to be the approximate growth in $Q_a$ during $\Delta t$. Note that this equation becomes: an identity when $\Delta t = 0$; Equation 7.4 when $\Delta t = 1$; and a linear approximation when $\Delta t$ is between 0 and 1. Dividing both sides of the equation by $\Delta t$,

$$\frac{Q_a(k + \Delta t) - Q_a(k)}{\Delta t} \approx \alpha (r_a(k + \Delta t) - Q_a(k))$$

and taking the limit for $\Delta t \to 0$,

$$\lim_{\Delta t \to 0} \frac{Q_a(k + \Delta t) - Q_a(k)}{\Delta t} \approx \alpha (r_a(k) - Q_a(k))$$

we obtain

$$\frac{dQ_a(k)}{dt} \approx \alpha (r_a(k) - Q_a(k)) \quad (7.5)$$

which is an approximation for the continuous time version of Equation 7.4. This result is in line with Tuyls et al. [114].

The general solution for Equation 7.5 can be found by integration:

$$Q_a(k) = Ce^{-\alpha t} + r_a$$

where $C$ is the constant of integration. As $e^{-x}$ is a monotonic function and $\lim_{x \to \infty} e^{-x} = 0$, it is easy to observe that the limit of Equation 7.6 when $t \to \infty$ is $r_a$:

$$\lim_{t \to \infty} Q_a(k) = \lim_{t \to \infty} Ce^{-\alpha t} + \lim_{t \to \infty} r_a = r_a$$

If we consider that only the first agent is learning and that the second is using a pure strategy, and assuming that the rewards are noise-free, playing a particular action will always generate the same reward for the first agent. In this case, the derivation above is enough to confirm that $Q_a$ will monotonically increase or decrease towards $r_a$, for any initial value of $Q_a$. More specifically, the function is monotonically increasing if $Q_a(0) < r_a$ and monotonically decreasing if $Q_a(0) > r_a$. Examples of this
behaviour are shown in Figure 7.4, which also shows the slope field associated with Equation 7.5. The figure plots the slope field obtained when $\alpha = 0.2$ and $r_{a_i} = 5$, and the sample paths for $Q_{a_i}(0) = 0, 2, 8$ and $10$. The line at $Q_{a_i} = 5$ is the equilibrium point for the slope field and the limit for the sample paths.

If the second agent is using a mixed strategy and the game is played repeatedly, then $r_{a_i}$ can be replaced by

$$E[r_{a_i}] = \sum_j a_{ij} y_j$$

which is the expected payoff of the first agent given the mixed strategy $y$ of the second. Note that a pure strategy is the specific case of a mixed strategy in which probability 1 is given to one of the actions. We then rewrite Equation 7.5 and 7.6 respectively as

$$\frac{dQ_{a_i}(k)}{dt} \approx \alpha (E[r_{a_i}(k)] - Q_{a_i}(k))$$

$$Q_{a_i}(k) = Ce^{-\alpha t} + E[r_{a_i}]$$

Thus, if the adversary is not learning, $Q_{a_i}$ will move in expectation towards $E[r_{a_i}]$ in a monotonic fashion. With a learning adversary, however, the situation is more complex. In this case, there is a possibility that the expected reward will change over time. A learning adversary can change its probability vector, which affects the expected re-
ward. If we first look at Equation 7.8, changes in the expected reward will modify the associated direction field and, consequently, the equilibrium points of it. At this level, every time the expected reward changes, a new direction field is generated. If we now look at Equation 7.9, the changes will modify the limit and, possibly, the direction of $Q_{a_i}$. Hence, it is important to identify when they will occur.

The $\epsilon$-greedy mechanism updates the probability vector whenever a new action becomes the one with the highest $Q$-value. Thus, we need to identify the intersection points in the functions of the adversary. It follows that the overall behaviour of the agent depends on these intersection points as they define which values $Q_{a_i}$ will converge to.

From the analysis point of view, the fact that the expected rewards can change over time implies that Equation 7.8 cannot be solved in the same way we solved Equation 7.5. However, one can easily derive the paths given the initial $Q$-values.

Another important aspect to be considered in the model is the speed in which the $Q$-values are updated. During the learning process, the actions have different probabilities of being played. For example, if $\epsilon = 0.2$, the $Q$-value of the current best action has a probability of 0.9 of being updated, while the other has a probability of 0.1 (considering a 2-actions game). It means that the $Q$-values are updated at different speeds. To simulate this behaviour, we define the growth in the $Q$-values as directly proportional to the probabilities. Then, Equation 7.8 becomes

$$\frac{dQ_{a_i}(k)}{dt} \approx x_i(k)\alpha(E[r_{a_i}(k)] - Q_{a_i}(k)) \quad (7.10)$$

where $x_i(k)$ is the probability of playing action $i$ at time $k$.

It is important to emphasize that the speed of the updates affects the shape of the functions and, as a consequence, the points at which they will intersect each other. As such, this component plays a very significant role in the model. Roughly speaking, the expected reward indicates the values $Q_{a_i}$ will converge to, the speed of the updates define the paths that it will follow to get there and the presence of intersection points in the functions of the adversary determines if it is ever going to get there.

It should be clarified however, that while the presence of intersection points in one agent’s function does not affect the limits of its equations and the equilibrium points
of the associated slope fields it does affect the speed of the convergence and the slope field itself. To illustrate it, suppose that $x_i$ and $E[r_{ai}]$ are constants. Then, by integration we can find the general solution for Equation 7.10:

$$Q_{ai}(k) = Ce^{-x_i \alpha t} + E[r_{ai}]$$

(7.11)

Note that the only difference between this equation and Equation 7.9 is the exponential term. Because the limit of this term is 0 for $t \to \infty$, the limit of the equation remains $E[r_{ai}]$, regardless of the value of $x_i$. On the other hand, different values of $x_i$ generate different slope fields. This can be seen in Figure 7.5 where we plotted the slope field obtained when $E[r_{ai}] = 5$ and $\alpha = 0.2$ for $x_i \in \{0.1, 0.9\}$. For the sake of comparison, we have also plotted the sample paths for $Q_{ai}(0)$ equal to 0, 2, 8 and 10.

Next, we show how the observations above come together to model the behaviour of the $Q$-values during the multiagent learning process.

**The model** For the first and the second players, respectively, let $A$ and $B$ be the payoff matrices, $x$ and $y$ be the probability vectors, and $Q_a$ and $Q_b$ be the vectors of $Q$-values. Then, based on the analysis above, the expected behaviour for the $Q$-values can be modelled by the system of equations:

$$Q_{ai}(k + 1) = Q_{ai}(k) + x_i(k)\alpha(\sum_j a_{ij}y_j(k) - Q_{ai}(k))$$
Chapter 7. Theoretical Analysis of the IPA with RL

\[ Q_{b_i}(k + 1) = Q_{b_i}(k) + y_i(k)\alpha \left( \sum_j b_{ij}x_j(k) - Q_{b_i}(k) \right) \]

\[ x_i(k) = \begin{cases} (1 - \epsilon) + \left( \frac{\epsilon}{n} \right), & \text{if } Q_{a_i}(k) \text{ is currently the highest} \\ \frac{\epsilon}{n}, & \text{otherwise} \end{cases} \]

\[ y_i(k) = \begin{cases} (1 - \epsilon) + \left( \frac{\epsilon}{n} \right), & \text{if } Q_{b_i}(k) \text{ is currently the highest} \\ \frac{\epsilon}{n}, & \text{otherwise} \end{cases} \]  

(7.12)

Having the above model for the \(Q\)-values, the expected behaviour of the agents can be derived by tracking the actions with highest \(Q\)-value over the learning process of each agent. In the next sub-sections the applicability of the framework is tested through experiments in typical games from the literature and in the games formulated with basis on the IPA with RL.

7.3.1.3 Evaluation of the Model in Typical Games

We illustrate the application of the framework in three typical games selected from the literature: the Prisoners Dilemma, the Battle of the Sexes and an interesting game from Tuyls et al. [114]. For each game we compare the theoretical behaviour obtained with the model with the behaviour found in real experimentation with \(Q\)-learning. The experiments were performed with the same configuratio as for the model and the results aggregated with the statistical median. The median is employed because it is more robust in the presence of outlier values than the mean. Therefore, it is more informative in showing the typical \(Q\)-values found during the experiments.

**The Prisoners Dilemma** The first game we consider is the Prisoners Dilemma (PD). This game has a single Nash Equilibrium in which both players play their dominant strategies (action 1). The payoff matrices for the first and second players are respectively

\[
A = \begin{bmatrix} 1 & 5 \\ 0 & 3 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 \\ 5 & 3 \end{bmatrix}
\]
7.3. Dynamic Analysis

In Figure 7.6 we plot the graphs obtained for this game when the initial $Q$-values are set to $Q_a = [0, 1]$ and $Q_b = [1, 0]$, and the learning parameters set to $\alpha = 0.1$ and $\epsilon = 0.4$. The starting strategies of the agents given these configuration are $x = [0.2, 0.8]$ and $y = [0.8, 0.2]$. The graphs on the left-hand side of the figure compare the theoretical curves of $Q$-values obtained by the model with the curves found in the experiments. The experimental curves show the median $Q$-values over 5000 learning experiments. The graphs in the center and on the right-hand side show respectively the theoretical and the observed dynamics for the strategies of the agents. The first is obtained from the analysis of the theoretical $Q$-values and the second from the analysis of the median $Q$-values.

![Graphs for the Prisoner’s Dilemma](image)

Figure 7.6: Graphs for the Prisoner’s Dilemma: comparison between the theoretical $Q$-values derived by the model and the median $Q$-values observed in the experiments (left); the expected dynamics for the agents’ strategies according to the model (center); and the dynamics observed in the experiments (right).

We first analyse the learning dynamics from the perspective of agent 2. The $Q$-values of this agent in the beginning of the learning process describe curves that would converge to 4.2 and 2.4 if no intersection point had been found in the curves of agent 1 (see Figure 7.7). The values 4.2 and 2.4 are the expected rewards of agent 2 given the strategy of agent 1 in that period. It can be seen that the curve of action 1 is quicker than the curve of action 2. This behaviour is linked to the starting strategy of this agent, which allocates probability 0.8 for the first action and 0.2 for the second. Just after time 20, there is a change in the direction of the curves. This change results from a change in the expected rewards, generated by the new strategy adopted by agent 1 after the intersection point found in its curves. From that point on, the curves of agent 2 start
to converge towards 1.8 and 0.6, the new expected rewards, and eventually stabilize around these values. Meanwhile, the $Q$-values of agent 1 evolve constantly towards 1.8 and 0.6. The intersection point does not affect its expected rewards but changes the *speed* of the convergence and consequently the shape of its curves.

![Figure 7.7: Example of theoretical $Q$-values obtained by the model in the Prisoners Dilemma. The *dashed* curves represent the theoretical evolution that would have been obtained if the agents had not changed their strategies.](image)

As seen in the graphs, the model is able to capture all the major trends found in the experiments. One particular point to note, however, is that while the changes in the theoretical curves are very crisp, in the observed ones they are actually smoother. The explanation for this behaviour is that the intersections in the experiments do not take place all the time in the exact point found by the model. The main aspect affecting the location of the intersection point is the *speed* of the updates, which is in fact a result of the stochastic process. It follows that, in our example, the strategy of agent 1 can change before or after the theoretical point, smoothing the curves when the median is calculated.

Figure 7.8 shows an example of the typical behaviour found in the experiments. Note that the intersection in the curves of the agent 1 was found slightly after the theoretical point. Also note that, apart from the local variabilities generated by the stochasticity in the actions of both agents, the general trends of the curves match the trend found by the model very well.

**The Battle of the Sexes**  The second game we study is the Battle of the Sexes (BS). This game has two pure Nash Equilibria, one where both players play the first action and the other where both play the second one. The game has also a mixed Nash Equilibrium where the first and the second players play action 1 with probabilities $2/3$ and $1/3$ respectively. The payoff matrices are as follows:
7.3. Dynamic Analysis

Figure 7.8: Example of an individual run of Q-learning in the Prisoners Dilemma.

\[ A = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \]

Figure 7.9 shows the graphs for the BS when the initial configuration are: \( Q_a = [2, 1], \ Q_b = [2, 4], \ \alpha = 0.1 \) and \( \epsilon = 0.1 \). The starting strategies of the agents given these configuration are \( x = [0.95, 0.05] \) and \( y = [0.05, 0.95] \). The experimental results aggregate the outcomes of 5000 rounds.

Figure 7.9: Graphs for the Battle of the Sexes: comparison between the theoretical Q-values derived by the model and the median Q-values observed in the experiments (left); the expected dynamics for the agents’ strategies according to the model (center); and the dynamics observed in the experiments (right).

Once again the model was able to capture all the major trends found in the experiments. The interesting behaviour observed in this case is the series of intersections points found in the early stages of the learning process. These intersections are reflective...
the expected strategies of the agents, shown by the graphs in the center of Figure 7.9. The graphs on the right-hand side of that figure show that a similar behaviour was also found in the experiments.

Figure 7.10 shows an example of the typical behaviour found in the learning experiments. Once again the coherence between the model and the experiments can be seen.

![Graphs showing the typical behaviour found in the learning experiments.](image)

Figure 7.10: Example of an individual run of Q-learning in the Battle of the Sexes

**A Game with no Pure Equilibrium** The third game has been selected from Tuyls et al. [114]. This game has no Nash Equilibrium in pure strategies and a unique Nash Equilibrium in mixed strategies, where both players play the first action with probability 0.5. The payoff tables are as follows:

\[
A = \begin{bmatrix} 2 & 3 \\ 4 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}
\]

In Figures 7.11 we plot the graphs obtained when the initial Q-values are \( Q_a = [0, 1] \) for agent 1 and \( Q_b = [2, 3] \) for agent 2. The learning parameters are \( \alpha = 0.1 \) and \( \epsilon = 0.1 \). The experimental results show the median Q-values over 5000 learning experiments.

The results for this example can be divided in two parts. The first part is characterized by major changes in the curves of both agents, which the model is able to capture very well. In particular, note that the Q-values of agent 2 stabilize in the very beginning of
7.3. Dynamic Analysis

Figure 7.11: Graphs for the 3rd game: comparison between the theoretical Q-values derived by the model and the median Q-values observed in the experiments (left); the expected dynamics for the agents’ strategies according to the model (center); and the dynamics observed in the experiments (right).

the learning process. Around time 100, however, there is an intersection point in the curves of agent 1 that violates the equilibrium and triggers the process of adaptations.

In the second part there seems to be a discrepancy between the model and the experiments. According to the model, this part is characterized by a cycle-like behaviour, indicating that the strategies will not stabilize. Instead, the experimental results show the convergence of the system.

To further illustrate this case, Figure 7.12 shows an example of an individual run of the Q-learning algorithm for this scenario. The graphs reveal that the experiments actually present a cyclic behaviour similar to the one described by the model. The convergence shown in Figure 7.12 is the result of the aggregation obtained with the statistical median.

7.3.1.4 Related Works

As commented before, the work most closely related to our work to model the behaviour of Q-learning agents that use the e-greedy mechanism is the research of Tuyls et al. [114]. The authors have developed a continuous time model for Multiagent Q-learning with Boltzmann exploration and have shown a link between it and the Replicator Dynamics (RD) of Evolutionary Game Theory [49]. The same type of link has also been explored by Borgers and Sarin [11] and Panait et al. [80]. The former investigated the behaviour of the agents in the Cross learning algorithm of Bush and
Mosteller [18]. The later proposed and analysed a variation of the Boltzmann-based Multiagent Q-learning to improve the cooperative behaviour of the agents.

In the RD, the probability of each action grows at a rate which is directly proportional to its performance against the others. Similar principles are applied in several Reinforcement Learning algorithms, including Q-learning with Boltzmann exploration. So the inspiration to develop RD-like analysis for those algorithms is quite natural. The ϵ-greedy mechanism, however, generates a different dynamics. More specifically, the mechanism defines a semi-uniform probability distribution in which the current best action is selected with probability $1 - \epsilon$ and a random action with probability $\epsilon$. Such a distribution is non-continuous and defined by a conditional function. Hence, the link cannot be directly applied in our case.

In another approach, Vidal and Durfee [116] present a framework to track the error in one agent’s decision during the multiagent learning process. The framework is generic enough to cover several different algorithms. However, it requires the tuning of some parameters that might not be known a priori or even impossible to obtain without extensive simulations. Our approach, in contrast, does not use any parameters other than the ones that are required by the Q-learning algorithm.

As in our research, all the above works share the property of being based on the analysis of differential or difference equations [83]. The topic has a long research tradition in the mathematical disciplines, a considerable theoretical framework and forms the standard approach to the study of dynamical systems. Other examples of the application of the approach to analyse multiagent reinforcement algorithms are the works of Abdallah and Lesser [2], who applied differential equations to study the dynamics of
the Weighted Policy Learner algorithm [2], and Leslie and Collins [70], who studied the asymptotic behaviour of variants of the Boltzmann-based multiagent Q-learning. The approach has also been used for the analysis of single-agent reinforcement learning algorithms [3, 12].

A different line of investigation corresponds to the identification of factors that lead the agents to develop some types of behaviours. Fulda and Ventura [42], for example, presented a set of conditions, on the environment and the payoff tables, which are sufficient to guarantee optimal performance of cooperative agents using Q-learning with Boltzmann exploration. Claus and Boutilier [30] also studied the cooperative case. Their analysis of Multiagent Q-learning with Boltzmann exploration has shown that the agents tend to converge to the most profitable equilibrium in simple games.

As far as we are aware, none of the existing approaches has explored the specific case of Multiagent Q-learning with ε-greedy exploration.

### 7.3.2 Analysis of the Market-based Resource Allocation Games

We now apply the model to analyse the dynamics of the learning agents in the market-based resource allocation games defined in Section 7.2.1. For reviewing purposes, tables 7.3 and 7.4 reproduce the games. Recall that the actions represent the possible demand requests and the payoffs simulate the reward functions, generating a competitive game with symmetric payoff table for the individual reward function and a cooperative game for the social reward function. The selfish learning agents play the individual reward game and the altruistic learning agents play the social reward game. In the tables, pure Nash Equilibria (NEs) are presented in **bold** and Pareto-Optimal (PO) joint-actions in *italic*.

<table>
<thead>
<tr>
<th>Table 7.3: Individual Reward Game.</th>
<th>Table 7.4: Social Reward Game.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0,0 0,0 0,0 0,0 <strong>0,4</strong></td>
<td>0 <strong>0,0</strong> 0,0 0,0 0,0 <strong>0,0</strong></td>
</tr>
<tr>
<td>1 0,0 0,0 0,0 <strong>1,3</strong> 0,0</td>
<td>1 0,0 0,0 0,0 <strong>3,3</strong> 0,0</td>
</tr>
<tr>
<td>2 0,0 0,0 <strong>2,2</strong> 0,0 0,0</td>
<td>2 0,0 0,0 <strong>4,4</strong> 0,0 0,0</td>
</tr>
<tr>
<td>3 0,0 <strong>3,1</strong> 0,0 0,0 0,0</td>
<td>3 0,0 <strong>3,3</strong> 0,0 0,0 0,0</td>
</tr>
<tr>
<td>4 <strong>4,0</strong> 0,0 0,0 0,0 0,0</td>
<td>4 <strong>0,0</strong> 0,0 0,0 0,0 <strong>0,0</strong></td>
</tr>
</tbody>
</table>

The experiments presented in the previous section have shown the agents learning to request 2 units of resources (joint-action < 3, 3 >) in both games, which maximizes the
Chapter 7. Theoretical Analysis of the IPA with RL

SW. This strategy is coherent with the behaviour found in the actual IPA market with RL investigated in the previous chapter. The analysis of the social reward game (Table 7.4) shows that this strategy is the only one that is PO and a NE. Furthermore, the strategy is the most profitable so the convergence to this strategy is not surprising. The analysis of the individual reward game (Table 7.3), however, shows Pareto-Optimality and NE in all the strategies in the minor-diagonal. Supported by the model, in the next-subsections we investigate why and how the agents develop those strategies.

### 7.3.2.1 The Social Reward Game

We first focus on the analysis of the social reward game. This game has seven pure NEs: 

- \(<1, 5>\), \(<2, 4>\), \(<3, 3>\), \(<4, 2>\), \(<5, 1>\), \(<5, 5>\) and \(<1, 1>\).

And only one PO solution: \(<3, 3>\). The payoff matrices for the first and the second players are respectively:

\[
A = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 \\
0 & 4 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \quad B = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 \\
0 & 4 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

The results presented in the previous chapter and Section 7.2.2 were obtained with all the initial \(Q\)-values set to 0, so \(Q_a = [0, 0, 0, 0, 0]\) and \(Q_b = [0, 0, 0, 0, 0]\). This configuration generates a uniform probability distribution in which both agents play each action with probability \(1/5\) in the first round of the learning process. Figure 7.13 presents the graphs obtained in this situation when \(\alpha = 0.1\) and \(\epsilon = 0.45\). The graphs on the left-hand side of the figure show the theoretical evolution of the \(Q\)-values. The graphs on the center show the median \(Q\)-values aggregated from 100 learning experiments. The graphs on the right-hand side show the frequency in which each action has been adopted by the agents. By adopted action or strategy we mean the action or strategy with the highest \(Q\)-value at that particular time.

As seen in Figure 7.13, the dynamics of the agents’ \(Q\)-values follow the dynamics obtained by the model relatively well. There is, however, a small difference in the shape of the curves of action 3 in the initial part of the learning process. This difference
7.3. Dynamic Analysis

Figure 7.13: Graphs for the Social Reward Game when the initial $Q$-values are $Q_a = Q_b = [0, 0, 0, 0, 0]$: the theoretical $Q$-values derived by the model (left); the median $Q$-values observed in the experiments (center); the observed frequency of the actions adopted by the agents (right) during the experiments.

results from the uniform probability distribution applied by the agents in the first round, which has different impacts in the model and the experiments. For the model, the uniform distribution gives to action 3 the highest expected reward and, therefore, the highest $Q$-value from the second round on. Consequently, from the model point of view, the evolution of the $Q$-values is quite similar to the evolution obtained when strategy $<3,3>$ is the initial strategy of the agents, as shall be seen later. For the experiments, on the other hand, the uniform distribution gives to any strategy the same probability of being played. However, only strategies $<2,4>$, $<3,3>$ and $<4,2>$ are able to modify the $Q$-values as they are the only ones that return a payoff other than 0. Therefore, the first time any of these strategies is played, it will become the one with highest $Q$-values and the learning process from that point on will be similar to the process obtained when the initial strategy of the agents is set to $<2,4>$, $<3,3>$ or $<4,2>$, depending on which one is played first. Therefore, the impact that the initial uniform probability distribution has in the model is different from the impact that it has in the experiments. While in the model the strategy $<3,3>$ will be certainly adopted by the agents from the second round on, in the experiments strategies $<2,4>$, $<3,3>$ and $<4,2>$ have the same probability of being adopted. This difference generates the discrepancy between model and experiments in the initial part of the learning process. This behaviour is further illustrated by the graphs showing the frequencies of the actions, where it can be seen the equilibrium between actions 2, 3 and 4 during the initial part of the learning process.
Still in Figure 7.13, despite the different starting points between model and experiments, the observed $Q$-values end up stabilizing around the expected levels. In order to explain this behaviour, it is necessary to investigate the dynamics of the agents when they start the learning process playing specific strategies. For this, we modify the starting $Q$-value vectors in order to reflect the desired starting strategy. In particular we investigate the starting strategies $<1,5>$, $<2,4>$ and $<3,3>$ as they are the strategies that would be able to equilibrate demand and supply in the actual IPA with RL scenario. The analysis of initial strategies $<5,1>$ and $<4,2>$ are not shown because they are symmetric to strategies $<1,5>$ and $<2,4>$, respectively.

Figure 7.14 plots the graphs obtained when the agents start the process playing strategy $<1,5>$. The initial $Q$-values are set to $Q_a = [0.01, 0, 0, 0, 0]$ and $Q_b = [0, 0, 0, 0, 0.01]$, the learning parameters are set to $\alpha = 0.1$ and $\epsilon = 0.45$. The results found for this case are quite similar to the results found in the previous case. The dynamics of the agents follow the dynamics obtained by the model with a small difference in the initial shapes of the action 3’s curves. From the model perspective, as the payoff of actions 1 and 5 is 0, action 3 will have the highest expected reward and, consequently, the highest $Q$-value from the second round on, making the process similar to the one obtained in the first case. From the experimental perspective, as strategy $<1,5>$ returns 0, the first time any of the strategies $<2,4>$, $<3,3>$ and $<4,2>$ is played, it will be adopted by the agents from that point on. As the three strategies have the same probability of being played, then the process from the experimental perspective is also similar to the process obtained in the first case.

Figure 7.14: Graphs for the Social Reward Game when the initial strategy is $<1,5>$: the theoretical $Q$-values derived by the model (left); the median $Q$-values observed in the experiments (center); the observed frequency of the actions adopted by the agents (right) during the experiments.
Figure 7.15 plots the graphs obtained when the agents start the process playing strategy $< 2, 4 >$. The initial $Q$-values are set to $Q_a = [0, 0.01, 0, 0, 0]$ and $Q_b = [0, 0, 0, 0.01, 0]$, the learning parameters are set to $\alpha = 0.1$ and $\epsilon = 0.45$. The results obtained for this case are quite interesting. The dynamics of the agents in the initial part of the learning process is similar to the dynamics predicted by the model. After episode 2000, however, there is a sudden increase in the median $Q$-values of action 3 and a decrease in the median $Q$-values of actions 2 and 4 of the first and second agents respectively. This strategy change, from $< 2, 4 >$ to $< 3, 3 >$, is not captured by the model, which predicts that strategy $< 2, 4 >$ will be kept throughout the learning process. The graphs that show the frequency of the strategies further illustrate the fact that the agents eventually converge to strategy $< 3, 3 >$ during the learning process. We will get back to this example later when we show the typical behaviour found in the analysis of the individual runs of the learning experiments.

In Figure 7.16 we plot the graphs obtained when the agents start playing strategy $< 3, 3 >$. The initial $Q$-values in this case are set to $Q_a = Q_b = [0, 0, 0.01, 0, 0]$ and the learning parameters set to $\alpha = 0.1$ and $\epsilon = 0.45$. The main point to note in this case is that the agents follow the dynamics described by the model remarkably well. The median $Q$-values of action 3 increases quite quickly and stabilizes around the expected levels. As commented above, the curves of theoretical $Q$-values are quite similar to the curves shown in Figure 7.13. Recall that in that case the agents apply a uniform probability distribution in the first round of the learning process, which gives to action 3 the highest expected reward and, therefore, the highest $Q$-value from the
second round on. Consequently, both processes have similar starting points from the model perspective. From the experimental perspective, however, the starting points are significantly different. The uniform probability distribution of that case gives to strategies $<2, 4>$, $<3, 3>$ and $<4, 2>$ the same probability of being adopted in the second round. The pre-define distribution of this case, on the other hand, will always give to strategy $<3, 3>$ the highest probability. Such a difference can be seen in the graphs that show the frequency of the agents’ strategies. Note that, while in Figure 7.13 there is an initial equilibrium between the frequencies, which is generated by the uniform probability distribution, in Figure 7.16 the frequency of action 3 is always higher than the others and kept stable.

![Figures 7.16: Graphs for the Social Reward Game when the initial strategy is $<3, 3>$: the theoretical Q-values derived by the model (left); the median Q-values observed in the experiments (center); the observed frequency of the actions adopted by the agents (right) during the experiments.]

To conclude the discussion on the Social Reward Game, Figure 7.17 and 7.18 present the typical behaviours found during the learning experiments. The analysis of the individual runs revealed the presence of two types of typical behaviour. In the first type (Figure 7.17) the agents converge quite quickly to strategy $<3, 3>$ and keep this strategy throughout the learning process. In the second type (Figure 7.18) the agents converge initially to strategies $<2, 4>$ or $<4, 2>$ and then change to strategy $<3, 3>$, keeping it during the rest of the learning process. These behaviours were obtained from the analysis of experiments performed with initial Q-values set to 0, which is the configuration used in the actual IPA with RL. Nevertheless, they are illustrative of the typical behaviours found in the other configuration adopted in this section and can be used to further explain the results found for them. In particular, the second type of behaviour shows that when agents start playing strategy $<2, 4>$
7.3. Dynamic Analysis

they will eventually converge to strategy \(<3, 3\rangle\), which is the general behaviour presented in Figure 7.15. One point to note is that when the agents are playing strategy \(<3, 3\rangle\), the behaviour of the observed \(Q\)-values is similar to the behaviour predicted by the model when the starting strategy of the agents is modified to \(<3, 3\rangle\), shown in Figure 7.16. Likewise, the behaviour when the agents are playing strategy \(<2, 4\rangle\) is similar to the behaviour predicted when the starting strategy is modified to \(<2, 4\rangle\), shown in Figure 7.15.

Figure 7.17: Example of the typical behaviour found in the learning experiments with the Social Reward Game when the initial \(Q\)-values are \(Q_a = Q_b = [0, 0, 0, 0, 0]\).

Figure 7.18: Example of the typical behaviour found in the learning experiments with the Social Reward Game when the initial \(Q\)-values are \(Q_a = Q_b = [0, 0, 0, 0, 0]\).

7.3.2.2 The Individual Reward Game

We now focus on the analysis of the individual reward game. This game has six pure NEs: \(<1, 5\rangle\), \(<2, 4\rangle\), \(<3, 3\rangle\), \(<4, 2\rangle\), \(<5, 1\rangle\) and \(<5, 5\rangle\). And five PO
solutions: \(< 1, 5 >, < 2, 4 >, < 3, 3 >, < 4, 2 >, < 5, 1 >\). The payoff matrices for the firs and the second players are respectively:

\[
A = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0
\end{bmatrix}
\quad B = \begin{bmatrix}
0 & 0 & 0 & 0 & 4 \\
0 & 0 & 3 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

As for the social reward game, we firs show the results obtained when \(Q_a = [0, 0, 0, 0, 0]\) and \(Q_b = [0, 0, 0, 0, 0]\), which has been the configuratio applied in the previous chapter and Section 7.2.2. This configuratio generates a uniform probability distribution in which both agents play each action with probability 1/5 in the firs round of the learning process. Figure 7.19 presents the graphs obtained in this situation when \(\alpha = 0.1\) and \(\epsilon = 0.45\). The graphs on the left-hand side of the figur show the theoretical evolution of the \(Q\)-values. The graphs on the center show the median \(Q\)-values aggregated from 100 learning experiments. The graphs on the right-hand side show the frequency in which each action is adopted by the agents during the experiments. Again, by adopted action or strategy we mean the action or strategy that has the highest \(Q\)-value at that particular time.

As seen in Figure 7.19, the dynamics of the agents follow the dynamics predicted by the model relatively well in the beginning of the learning process. However, there is an
increase in the median $Q$-value of action 3 around the learning episode 4000 that is not predicted. According to the model, the $Q$-values would converge constantly towards $Q_a = Q_b = [0, 0.09, 0.18, 0.27, 0.36]$. These values are the expected rewards when the agents play strategy $<5, 5>$. The development of this strategy is a consequence of the uniform probability distribution applied in the first round of the process, which gives to action 5 the highest expected reward and, therefore, the highest $Q$-value in the second round. As there are no intersection points in the curves of either agent, the $Q$-values would stabilize around those values. The experimental results show the median $Q$-values following this dynamic relatively well during the first learning episodes. Around episode 4000, however, there is a sudden increase in the median $Q$-value of action 3, which is not captured.

As commented above, the uniform distribution gives to action 5 the highest $Q$-value in the second round of the learning process according to the model. Therefore, it is necessary to investigate the dynamics of the agents when they start the learning process playing strategy $<5, 5>$. The graphs for this case are presented in Figure 7.20. The initial $Q$-values are set to $Q_a = Q_b = [0, 0, 0, 0, 0.01]$ and the learning parameters set to $\alpha = 0.1$ and $\epsilon = 0.45$. As it can be seen, the results obtained in this case are similar to the results obtained in the previous case. The main difference between them is in the observed frequency of the actions during the very beginning part of the learning process. While in the first case the frequency of action 5 starts at the same level as the frequency of actions 2, 3 and 4, which is a result of the uniform probability distribution, in this case it starts higher. In both situations, however, the frequency of action 5 decreases quite quickly and action 3 ends up dominating the learning process.

Figure 7.21 plots the graphs when the agents starting the learning process playing strategy $<1, 5>$. The initial $Q$-values are set to $Q_a = [0.01, 0, 0, 0, 0]$ and $Q_b = [0, 0, 0, 0, 0.01]$ and the learning parameters set to $\alpha = 0.1$ and $\epsilon = 0.45$. The results obtained in this case are also quite similar to the results obtained in the previous two cases. The median $Q$-values follow the theoretical $Q$-values relatively well in the first part of the learning process. In the second part there is a sudden increase in the $Q$-values of action 3, which is not predicted by the model. The frequency of the actions also reveals an interesting behaviour. For the first agent, the frequency of action 1 starts higher than the others but decrease rather quickly because the payoff of playing $<1, 5>$ is 0. Since the other actions have the same probability $\epsilon/n$ of being played, their frequencies start at the same level. As the learning process progress, however, the frequency of action 5 decreases quickly while the frequency of actions 2, 3 and 4
Chapter 7. Theoretical Analysis of the IPA with RL

Figure 7.20: Graphs for the Individual Reward Game when the initial strategy is \(< 5, 5 >\): the theoretical Q-values derived by the model (left); the median Q-values observed in the experiments (center); the observed frequency of the actions adopted by the agents (right) during the experiments.

are kept in equilibrium until action 3 dominates the process. For the second agent, the behaviour is similar to the behaviour found when the starting strategy is set to \(< 5, 5 >\): the frequency of action 5 starts higher than the others but decreases quickly and action 3 dominates the process.

Figure 7.21: Graphs for the Individual Reward Game when the initial strategy is \(< 1, 5 >\): the theoretical Q-values derived by the model (left); the median Q-values observed in the experiments (center); the observed frequency of the actions adopted by the agents (right) during the experiments.

The previous three examples have presented the similar behaviour in which the median Q-values follow the dynamics derived by the model relatively well during the first part of the learning process, when a sudden increase in the Q-values of action 3 is not captured. In the three cases the model predicts the convergence of the agents to
7.3. Dynamic Analysis

strategy $<5,5>$. When this strategy is adopted, however, the experiments show the frequency of action 5 decreasing quite quickly and the convergence to strategy $<3,3>$. Additionally, the experiments show the frequency of actions 2, 3 and 4 in equilibrium for some time before the frequency of action 3 starts to raise. We will next present the analysis of the dynamics when the starting strategy are $<2,4>$ and $<3,3>$. Before that, however, we will present the analysis of the individual runs of the learning experiments as it will help in identifying why the agents always converge to strategy $<3,3>$.

The analysis of the individual runs of the learning experiments has revealed the existence of two types of typical behaviour. In the first type the agents converge to joint-action $<3,3>$ in the beginning of the learning process and keep this strategy quite stable during the rest of it. In the second type the agents converge initially to joint-actions $<2,4>$ or $<4,2>$, which are kept quite unstable, particularly by the agent that is playing action 2, until they eventually converge to strategy $<3,3>$. In some cases the agents alternate between strategies $<2,4>$ and $<4,2>$ before converging to $<3,3>$.

Figures 7.22 and 7.23 show examples of the typical behaviours. Figure 7.22 plots the example for the first type. Note the quick convergence to action 3 and the very few changes of strategy that occurs throughout the learning process. Figure 7.23 plots the example for the second type. Note the initial convergence to strategy $<2,4>$ and then the convergence to strategy $<3,3>$. Also note the high number of changes that occurs in the strategy of the first agent while it is playing action 2. From these observations we can infer that strategy $<3,3>$ is more stable than strategies $<2,4>$ and $<4,2>$. We can then justify the rise in the $Q$-values of action 3: if the typical behaviour is to eventually converge to strategy $<3,3>$ and if such a strategy is more stable than the others, then it is reasonable to say that it will dominate the learning process in the long run.

In Figure 7.24 we plot the graphs obtained when the agents start the process playing strategy $<2,4>$. The initial $Q$-values are set to $Q_a = [0, 0.01, 0, 0, 0]$ and $Q_b = [0, 0, 0, 0.01, 0]$, the learning parameters are set to $\alpha = 0.1$ and $\epsilon = 0.45$. The results for this case are quite interesting. According to the model the $Q$-values would converge to $Q_a = [0, 0.63, 0.18, 0.27, 0.36]$ and $Q_b = [0, 0.09, 0.18, 1.92, 0.36]$, so the agents would keep playing strategy $<2,4>$. This dynamic is followed by the experiments during the first learning episodes, but then there is a decrease in the median $Q$-values of actions 2 and 4 for the first and second agent respectively. At the same time, there
Figure 7.22: Example of the typical behaviour found in the learning experiments with the Individual Reward Game when the initial $Q$-values are $Q_a = Q_b = [0, 0, 0, 0, 0]$.

is an increase in the median $Q$-value of action 3. This behaviour is also noticed in the graphs for the frequency of the strategies, which show the frequency of actions 2 and 4 decreasing while the frequency of action 3 is increasing. These observations illustrate the instability of strategy $<2,4>$. It is important to mention that the values found by the model are coherent with the values shown in Figure 7.23 for the periods where the agents are playing strategy $<2,4>$. Furthermore, the typical behaviour found is similar to the type of typical behaviour shown in Figure 7.23: the agents start playing strategy $<2,4>$ and eventually converge to $<3,3>$, which is then kept very stable.

In Figure 7.25 we plot the graphs obtained when the agents start the process playing strategy $<3,3>$. The initial $Q$-values are set to $Q_a = Q_b = [0, 0, 0.01, 0, 0]$ and the learning parameters set to $\alpha = 0.1$ and $\epsilon = 0.45$. The first point to note in the graphs
Figure 7.24: Graphs for the Individual Reward Game when the initial Q-values strategy is \(<2,4>\): the theoretical Q-values derived by the model (left); the median Q-values observed in the experiments (center); the observed frequency of the actions adopted by the agents (right) during the experiments.

is that, in this case, the dynamics found in the experiments is captured very well by the model. It can also be seen that the frequency of strategy \(<3,3>\) is kept constant throughout the learning process, which illustrates the stability of this strategy. Another point to note is that the theoretical Q-values described by the model is coherent with the values found in the individual runs of the learning experiments for the periods where the agents are playing strategy \(<3,3>\) (see Figure 7.22). As in the previous example, the typical behaviour found in this case is similar to the first type of typical behaviour shown in that Figure: the agents converge to strategy \(<3,3>\) in the beginning of the process and keep this strategy quite stable during the rest of it.

The experiments indicate the existence of certain relationships between the payoffs tables, the learning rate and the exploration rate of the \(\epsilon\)-greedy mechanism. In the general case, these relationships may be responsible for inaccuracies of the model in the sense that the model is not able to capture the actual stochasticity present in the experiments. For instance, consider the case where the Q-values of two actions are quite close to each other (the theoretical Q-values shown in Figure 7.19 can serve as an example). With a high exploration, there is a high chance for the occurrence of intersections between these two curves during the real experimentation, even if the model does not predict it. The occurrence of non-predicted intersections can decrease the quality of the model as they can change the expected rewards, changing the curves in general. In our particular case of the individual reward game, the relationships are responsible for the instability presented by strategies \(<2,4>, <4,2>, <1,5>\) and \(<5,1>\), and contribute to the convergence of the system to strategy \(<3,3>\), once
this strategy is the most stable of them. To identify those relationships can improve the general quality of the model and constitute an important future work for this research.

7.3.3 Summary

In this section we have analysed the behaviour of the agents in the IPA with RL from the perspective of the dynamic process generated by the learning algorithm employed by them. For this, we first developed a model for the dynamics of Multiagent Q-learning with $\epsilon$-greedy exploration and evaluated its applicability on typical games selected from the literature. This initial investigation has shown the general feasibility of the model, which was able to capture all the major trends found in the experiments. We then applied the model to theoretically analyse the two market-based resource allocation games formulated in the previous section.

The investigation has revealed some interesting properties of the IPA with RL’s scenario. In particular, the game-theoretical analysis of the games, performed in section 7.2, has shown Pareto-optimality and Nash Equilibrium at the joint-strategies in which the agents request the same amount of resources, maximizing the social welfare. While in the social reward game such a strategy returns the most profitable payoff, in the individual reward game there are strategies that are more profitable from a single agent’s perspective. The analysis supported by the model, however, has shown that those strategies are less stable, therefore, justifying the interesting results in which the selfish
agents also end up optimising the social welfare.

7.4 Conclusion

In this chapter we have investigated the reasons for the experimental results found in the IPA with RL. The study was conducted in two parts. In the first part we have performed a static analysis using concepts from game-theory to highlight the properties of the solutions found by the agents. Based on a simplified version of the actual scenario addressed in the previous chapter, we developed two five-action two-player games that simulate the application of the reward functions and identify their Nash Equilibria and Pareto Optimal solutions. We then performed a series of learning experiments in the new games to investigate what solutions the agents find and how coherent are they with the behaviour found in the previous chapter. The analysis has revealed the presence of Nash Equilibrium and Pareto-Optimality at the joint-actions in which the resources are shared equally, in both games, which maximizes the social welfare. The learning experiments have shown the pseudo-convergence of the agents to those joint-actions, reproducing the behaviour found in the actual IPA with RL.

In the second part of the study we have investigated the agents’ behaviour from the perspective of the dynamic process generated by the learning algorithm employed by them. We developed a theoretical model of the Multiagent $Q$-learning with $\epsilon$-greedy exploration algorithm and applied it to analyse the dynamic behaviour of the agents in the formulated games. For the individual reward game, the analysis supported by the model has shown that the strategies which are more profitable from a single agent’s perspective are less stable than the strategy that maximizes the social welfare. This observation justify the interesting results in which the selfish agents end up optimising the social welfare.
Chapter 8

Conclusion

In this thesis we have proposed and developed a framework to optimise both the individual and social allocation efficiency in equilibrium-based commodity market mechanisms composed of strategic learning participants. Classical equilibrium-based mechanisms assume the existence of rational price-taking participants, which will not attempt to strategically influence the mechanism to obtain higher gains. This thesis, on the other hand, assumes that the participants can behave like entities composing real economies and will engage in strategic behaviour in order to satisfy their preferences. Such a premise enables the development of a more realistic and robust scenario for resource allocation in distributed systems and the improvement of the applicability of commodity market resource allocation mechanisms in these systems.

The strategic participants in this thesis are intelligent learning agents that use utility functions to describe preferences over different resource attributes and develop strategic behaviour via Reinforcement Learning. The hypothesis driving the research was that the individual and social allocation efficiency of equilibrium-based commodity market resource allocation mechanisms with strategic learning participants can be optimised.

In order to test the hypothesis, we have first defined a conceptual framework that enables the realisation of commodity-market resource allocation with reinforcement learning. We have identified the relationships between the two areas and proposed a conceptual architecture for the integration of strategic learning participants into equilibrium-based mechanisms for market-based resource allocation. The framework forms the basis for the experimental and theoretical investigation performed in the thesis.
As part of the framework, we have also introduced a new technique to support the application and scalability of Reinforcement Learning in problems with continuous spaces. The technique takes advantage of features found in the commodity-market resource allocation scenario. It consists in the discretization of the state and the action spaces during the learning process and the application of curve-fitting or interpolation procedures in the learnt discrete policy to obtain the agent’s continuous policy. Its application leads to the reduction of the required amount of learning since not all the actions and states need to be exhaustively explored to obtain a reasonable policy.

The conceptual framework was employed to introduce and investigate the impacts of strategic learning agents in a specific type of commodity market resource allocation mechanism called Iterative Price Adjustment (IPA). The result is the IPA with RL. The experiments in the new mechanism have considered two types of strategic learning agents: selfish learning agents, whose learning goal is to improve their individual utility; and altruistic learning agents, whose learning objective is to improve the social welfare.

The main outcome of the experimental investigation in the IPA with RL is that the market composed exclusively of selfish agents is able to achieve social performance similar to the performance obtained by the market composed exclusively of altruistic agents, both reaching near-optimal SW, measured by the Nash Product. In addition, the selfish agents are able to approximate the solution to the fairest PO allocation in situations where the altruistic agents fail.

Following the experimental investigation, a theoretical study has been conducted in order to identify the reasons for the interesting results found in the IPA with RL. The study was conducted in two parts. In the first part we have performed a static analysis using concepts from game-theory to highlight the properties of the solutions found by the agents. Based on a simplified version of the actual scenario addressed in the experiments, we have developed two five-action two-player games that simulate the application of the reward functions and identify their Nash Equilibria and Pareto Optimal solutions. We then performed a series of learning experiments in the new games to investigate what solutions the agents find and how coherent are they with the behaviour found in the previous chapter. The analysis has revealed the presence of Nash Equilibrium and Pareto-Optimality at the joint-actions in which the resources are shared equally, in both games, which maximizes the social welfare. The learning experiments have shown the pseudo-convergence of the agents to those joint-actions, reproducing the behaviour found in the actual IPA with RL.
Chapter 8. Conclusion

In the second part of the theoretical study we have investigated the agents’ behaviour from the perspective of the dynamic process generated by the learning algorithm employed by them. We have developed a theoretical model of the Multiagent $Q$-learning with $\epsilon$-greedy exploration algorithm and applied it to analyse the dynamic behaviour of the agents in the formulated games. For the individual reward game, the analysis supported by the model has shown that the strategies which are more profitable from a single agent’s perspective are less stable than the strategy that maximizes the social welfare. This observation justify the interesting results in which the selfish agents end up optimising the social welfare.

The research has lead to the confirmation of the hypothesis that the individual and social allocation efficiency of equilibrium-based commodity market resource allocation mechanisms with strategic learning participants can be optimised. The conclusion is that, in the IPA with RL, it is possible to simultaneously optimise both individual and social performances when the market is composed of strategic learning agents. In particular, the investigations have shown that it is possible to optimise both efficiency measures through the application of selfish agents using only local information to learn. Such an outcome is significant not only for the market-based resource allocation domain but also for a series of other domains where individual and social utility should be optimized but agents are not guaranteed to act cooperatively in order to achieve it or they do not want to reveal their private preferences.

This thesis advances the knowledge base in a number of areas of Market-based Resource Allocation, Multiagent Learning and Agent-based Computational Economics. First, it formalizes a new conceptual framework involving multiagent reinforcement learning in commodity market resource allocation mechanisms. Second, it introduces strategic learning agents in market-based resource allocation mechanisms founded on general equilibrium theory. So far, these mechanisms have assumed the existence of rational price-taking participants. The approach proposed in the thesis, instead, is more realistic as it explicitly addresses the existence of strategic participants that can try to exploit the mechanism. Third, it simultaneously optimises both the individual and social efficiency of the allocation. Existing approaches typically focus on the achievement of a PO allocation only, usually disregarding the individual utility and social welfare resulting from it. Fourth, it develops a theoretical model for the dynamics of Multiagent $Q$-learning with $\epsilon$-greedy exploration. Despite the popularity of this algorithm, such a model has not been developed before. Finally, it develops a new technique to support the application and scalability of reinforcement learning in problems.
8.1 Answers to the Research Questions

This section provides the answers to the research questions proposed in the thesis.

**How to integrate strategic learning participants into equilibrium-based commodity market mechanisms?**

To answer this question we have defined a conceptual framework for the realisation of commodity-market resource allocation with multiagent reinforcement learning. We have identified the relationships between the two areas and developed an integrated conceptual framework that enables the integration of strategic learning participants into equilibrium-based market mechanisms for resource allocation.

**What are the impacts of the presence of strategic learning participants on the individual and social performances of equilibrium-based commodity market mechanisms?**

To answer this question we have employed the conceptual framework to introduce and investigate the impacts of strategic learning agents in a specific type of equilibrium-based market mechanism called Iterative Price Adjustment (IPA). The experimental investigation has considered two types of strategic learning participants: selfish participants, whose learning goal is to improve their own utility; and altruistic participants, whose learning goal is to improve the social welfare of the market. The experiments included the application of strategic learning participants against other strategic learning participants and against participants with no learning capabilities. The results found in the investigation are summarized below.

When learning in the presence of non-learning participants, the selfish agents have developed some type of collective behaviour in which they together gain the most from the allocation but divide these gains more or less evenly among themselves, therefore achieving a good social solution within the collective. If in the one hand such a behaviour can decrease the social welfare of the market, once the non-learning participants can be exploited and left with few resources, on the other hand it leads to the improvement of the social welfare as more selfish learning agents are added.
The application of altruistic learning agents in the market containing non-learning participants has in general also lead to the increase of the social welfare. In this case, however, the improvement is obtained at the expenses of the strategic learning agents, which have actually learnt to improve the social welfare, even if in some cases it meant to decrease their own gains.

For the cases where only strategic learning agents were present in the market, both selfis and altruistic types have achieved similar social results. In both cases solutions with near-optimal social welfare have been found. The selfis learning agents, however, have been able to approximate the fairest Pareto-Optimal allocation in all the configuration evaluated, including some in which the altruistic ones have failed.

The question has been further addressed by a theoretical investigation on the behaviour of the strategic learning participants in the market. First, a game-theoretical analysis on a simplify version of the addressed scenarios has revealed the presence of Nash Equilibrium and Pareto-Optimality at the joint-actions in which the resources are shared equally and the social welfare is maximized. This result held true in both the cases, with altruistic and with selfis learning agents. Learning experiments performed on the simplify scenarios have revealed the pseudo-convergence of the agents to those joint-actions, reproducing the behaviour found in the actual IPA with RL. Then, a dynamic analysis of the behaviour of the agents, supported by a theoretical model of the reinforcement learning algorithm employed by them, has revealed that those joint-actions are actually more stable than the others. This result justifie why the selfis agents have converged to those strategies, even if others with more profitabl outcomes from a single agent’s perspective were present.

**What mechanism can optimize the individual and social efficiency in equilibrium-based commodity market mechanisms with strategic learning participants?**

To answer this question we have developed the IPA with RL, which is the mechanism resulting from the introduction of strategic learning participants in the IPA. The experimental investigation in this new mechanism has shown that: i) when non-learning participants are present in the market, the introduction of altruistic learning agents will generally lead to the improvement of the social welfare but may incur in a poor individual utility for the altruistic agents; ii) if selfis agents are used in the same case, then the individual utility of the non-learning participants may be negatively affected; iii) if only altruistic learning participants are present in the market, then near-optimal social
solutions can be found but the agents may be unable to approximate the individual solutions to the fairest Pareto-Optimal outcomes; iv) if only selfish learning participants are applied, near-optimal social solutions can be found and the agents may be able to approximate the individual solutions to the fairest Pareto-Optimal outcomes.

Therefore, the experimental investigation in the IPA with RL has shown that the individual and social allocation efficiency of the mechanism can be optimized through the application of selfish learning agents that use only local information to learn. Such an outcome is significant not only for the market-based resource allocation domain but also for a series of other domains where individual and social utility should be optimized but agents are not guaranteed to act cooperatively in order to achieve it or they do not want to reveal their private preferences.

**How to model the dynamics of the agents and to predict the expected behaviour of equilibrium-based commodity market mechanisms with strategic learning participants?**

To answer this question we have developed a theoretical model for the dynamics of the algorithm employed by the agents, the Multiagent \( Q \)-learning with \( \epsilon \)-greedy exploration. Despite the popularity of this learning algorithm, the problem of analysing and modelling the expected behaviour of agents using it has not been addressed before. To develop the model, we have first analysed a continuous-time version of the \( Q \)-learning update rule and studied how the \( \epsilon \)-greedy mechanism and the presence of other agents affect it. We then used this analysis to model the problem as a system of difference equations which was used to calculate the expected evolution of the \( Q \)-values and, consequently, the expected behaviour of the agents. The feasibility of the model has been evaluated in typical games selected from the literature. We then applied it to understand and justify the results found in our experimental investigation in the IPA with RL.

**How to support the application and scalability of reinforcement learning in the context of market-based resource allocation?**

One of the aspects of the market-based resource allocation scenario is that the spaces of the problem are usually continuous, such as price and amount of resource. Traditional reinforcement learning algorithms, however, are based on tabular representations of the action and state spaces and, therefore, cannot be directly applied in problems
with continuous spaces. Continuous spaces also have a direct effect on the scalability aspects of the algorithms, which usually rely on exhaustive trial-and-error searches.

To answer the question we have developed a new technique that consists in the discretization of the state and the action spaces during the learning process and the application of curve-fitting or interpolation procedures in the learnt discrete policy to obtain the agent’s continuous policy. The approach enables the application of reinforcement learning in problems with continuous spaces and leads to the reduction of the required amount of learning as evidence by the experimental investigation in the IPA with RL.

### 8.2 Summary of Contributions

In summary, the main contributions of this thesis are:

- The conceptual framework for commodity-market resource allocation with multiagent reinforcement learning.
- The IPA with RL, a utility-aware market-based resource allocation mechanism supported on the multiagent learning approach;
- The analysis and comparison of individual and social-based rewards for market-based resource allocation;
- The theoretical model for the dynamics of Multiagent $Q$-learning with $\epsilon$-greedy exploration;
- A new technique to support the application and scalability of Reinforcement Learning in problems with continuous spaces.

### 8.3 Future Works

There are many aspects in which this work could be extended. In this section we discuss some of the immediate open opportunities. We divide them in three main categories, the first containing aspects related to *equilibrium-based markets with strategic...*
learning participants, the second with aspects related to multiagent reinforcement learning and the third with aspects related to the integration of the proposed approach into Grid and Cloud computing contexts.

8.3.1 Equilibrium-based Markets Aspects

With regards to the application of strategic learning participants in equilibrium-based markets, one of the immediate future works for this research is to extend the approach and scenarios to include many agents, aiming at 100-500. The experimental investigation in the IPA with RL have shown very promising results. It should be noted, however, that only configuration with a few agents have been considered. To scale it is important for two reasons. First, it is necessary to investigate the impacts of the scaling factor on the individual and social results of the mechanism. According to the classical theory on equilibrium-based markets, the gains from strategic attempts decrease as the number of participants increase, justifying the perfect competition assumption and making it very important to investigate what happens in our approach when the number of agents is largely scaled. And second, to scale the approach is important so as to enable its application in large-scale distributed systems, which are perhaps the most interesting application for market-based resource allocation mechanisms.

Scaling the approach will also involve research on aspects related to scalability issues of multiagent reinforcement learning algorithms, which will be commented in the next section.

Another important future work is to consider the cases where the market is composed of buyers and sellers. For this, it is necessary to extend the approach applied in the IPA with RL to other equilibrium-based market mechanisms. This step is critical for the development of this research because resource sellers are important entities in real economies. Like buyers, these agents also have different preferences which they will try to satisfy during the allocation process. Therefore, it is reasonable to believe that they will also engage in strategic behaviour in order to attempt better outcomes. Besides, one of the features of the IPA is that its goal is to maximize resource usage and not to increase the profit of the resource owners. Our investigations in the IPA with RL have shown that the agents were able to strategically take advantage of this feature and decrease the price to a very low level in order to maximize their price utility gains. The existence of learning sellers in the market will represent a more
complex scenario for the buyers as they will need to deal with strategic attempts of those to increase the price. At the moment, two prospective choices for this study are the WALRAS [125] and the G-commerce [126] mechanisms. They are both based on the Walrasian economic equilibrium but use different methods to find it. They both consider the presence of resource sellers and have experimental results available, so we could easily compare the performances of the original mechanisms and the new learning-enhanced ones.

Finally, it is also necessary to extend the approach and scenarios to cases including more than one type of resource. For this, the agents will need to be able to participate in different markets, as each market is associated to one resource type, and, therefore, learn different but inter-dependent demand functions, one function per market/resource type.

### 8.3.2 Multiagent Reinforcement Learning Aspects

With regards to the multiagent reinforcement learning area, one interesting opportunity for future work is to extend our model of the Multiagent $Q$-learning with $\epsilon$-greedy exploration algorithm. Our evaluations of the current model have shown promising results in simple games selected from the Game-Theory literature. However, further developments are necessary in order to apply it to more complex scenarios, such as the market-based resource allocation mechanism proposed in this thesis. In particular, the model is currently able to cope only with single-state games. Since the actual scenario faced by the agents in multiagent contexts is likely to contain several agents and states, it is necessary to extend the model to multi-state multi-player games. Moreover, the dynamics of the algorithm is currently analysed in 2-dimensional graphs that show the development of the expected $Q$-values and strategies of the agents over the time, two graphs per agent. Therefore it is also necessary to develop new techniques to allow the analysis of scenarios composed of multiple states and multiple agents.

In addition, the application of the model in the market-based resource allocation games has revealed some interesting aspects with regards to the convergence stability of the Multiagent $Q$-learning with $\epsilon$-greedy exploration algorithm. To investigate such aspects more deeply is also an interesting path for future works.

Finally, a critical aspect for the further development of this research is to deal with the scalability issues of the application of reinforcement learning algorithms in multiagent
8.3. Future Works

scenarios. In this thesis we have proposed a novel technique for it but have not fully investigated its applicability and potential. To perform such an investigation consists in another important future work.

8.3.3 Grid and Cloud Computing Contexts

One important future work concerns the integration of the proposed approach into Grid and Cloud computing contexts. This integration can be performed through the leverage of existing market-oriented architectures [4, 22, 97].

In the cloud context, for example, Buyya et al. [22] describe a high-level architecture that supports the realisation of market-oriented resource allocation. The main entities composing the architecture are Users/Brokers, Service Level Agreement (SLA) Resource Allocator, Virtual Machines (VMs) and Physical Machines. The approach proposed in this thesis can be easily mapped and integrated into that architecture. The integration, illustrated in Figure 8.1, can be realised as follows:

- **Users/Brokers** are the entities responsible for submitting service requests to the Cloud. Brokers act on behalf of their users and negotiate access to services with the SLA Resource Allocator according to the users’ preferences. In the integrated architecture, the brokers are equipped with a reinforcement learning component. Being able to learn from past negotiation rounds, they become Strategic Learning Participants. They receive the users’ preferences in the form of utility functions and use the reinforcement learning approach proposed in the thesis to optimize these functions or a combination of them. Raw resources types, such as the ones studied in this thesis, are translated into Quality of Service (QoS) parameters that describe the services’ characteristics.

- The **SLA Resource Allocator** acts as an interface between the Users/Brokers and the Cloud service providers. Among other functionalities, it provides a pricing mechanism responsible for deciding how users/brokers are charged. In the integrated architecture, the SLA Resource Allocator incorporates the role of the Market entity and its pricing mechanism implements the Iterative Price Adjustment used in the thesis. The SLA Resource Allocator receives service requests from brokers, with respective QoS parameters, and adjusts the prices accordingly until a suitable allocation is found.
• **VMs** enable flexible partition of resources on the same physical machine to specific service requirements of different users. Once service requests are accepted by the SLA Resource Allocator through the IPA mechanism, the respective VMs are instantiated to accommodate the services.

• **Physical Machines** provide the resources to meet service demands. Multiple VMs can run on a single machine, accommodating different service needs.

Figure 8.1: Integrated high-level architecture for the application of the IPA with RL in Cloud computing contexts (adapted from Buyya et al. [22])

Before implementing the proposed approach in real scenarios, it may be interesting to evaluate it using simulation libraries such as the **GridSim** [107]. GridSim is an event-driven Java-based toolkit that supports the modelling and simulation of heterogeneous Grid resources, users and application models. It enables the derivation of realistic results in a rapid, flexible, repeatable and controllable manner. Buyya and Murshed [20], for example, describe the application of GridSim for the simulation of the economic-driven Grid resource management system Nimrod-G [4].
The main entities in a GridSim simulation are Grid Resources, Users and Resource Brokers. A Grid Resource is composed of a set of machines representing physical computing nodes under the same administrative domain. Each Grid Resource implements its own strategy for the internal scheduling of jobs. Users interact with the Grid through their Resource Brokers. A Resource Broker is responsible for discovering suitable Grid Resources and submitting jobs to them so as to satisfy the user’s preferences and requirements. Such an architecture can be easily adapted for the application of the approach proposed in this thesis. For this, each Grid Resource needs to perform the role of the Market entity, implementing the Iterative Price Adjustment mechanism, and Resource Brokers need to perform the role of strategic learning participants, implementing the reinforcement learning-based strategy.
Bibliography


Bibliography


Bibliography


