POLITECNICO DI MILANO
DEPARTMENT OF ELECTRONICS INFORMATICS AND BIOENGINEERING
DOCTORAL PROGRAMME IN INFORMATION TECNOLOGY

AUGMENTING TRADERS WITH LEARNING MACHINES

Doctoral Dissertation of:
Edoardo Vittori

Supervisor:
Prof. Marcello Restelli

Tutor:
Prof. Nicola Gatti

2021 – Cycle XXXIV

Abstract

The financial markets are comprised of several participants with diverse roles and objectives. Asset management firms optimize the portfolios of pension funds, institutions and private individuals; market makers offer liquidity by continuously pricing and hedging their risks; proprietary traders invest their own capital with sophisticated methodologies. The approaches adopted by these actors are either manual or expert systems that rely on the experience of traders, and thus are subject to human bias and error.

This dissertation proposes innovative techniques to address the limitations of the current trading strategies. Specifically, we explore the use of algorithms capable of autonomously learning the aforementioned sequential decision-making processes. The development of these algorithms entails a careful reproduction of realistic environments, as well as the observance of trading objectives, *i.e.*, maximizing returns while maintaining a low risk profile and minimizing costs. These algorithms all share a common core structure, that is making a trading decision conditional on the current state of the financial markets.

Our main theoretical and algorithmic contributions include the extension of the online learning field, as we introduce transaction costs and conservativeness in online portfolio optimization, and the enhancement of Monte Carlo Tree Search algorithms to account for the stochasticity and high noise typical of the financial markets. In terms of experimental contributions, we apply Reinforcement Learning to learn profitable quantitative trading strategies and option hedging approaches superior to the standard Black & Scholes hedge. We also find that Reinforcement Learning combined with Mean Field Games enables the development of competitive bond market making strategies. Finally, we demonstrate that dynamic optimal execution methods can be learned through Thompson Sampling with Reinforcement Learning.

The use of such advanced techniques in a production environment may allow the achievement of a competitive advantage that will translate into economical benefits.

Acknowledgements

This work would not have been possible without the precious contributions of several people.

First of all, I would like to thank Marcello, who guided me throughout this journey.

I would like to thank my colleagues of the XVA Management desk, Michele, Marco, Francesco and Simone, who were always eager and ready to help out.

I'm grateful for the support of Andrea, who made the executive Ph.D. formula possible and who has been a mentor to me.

Working with the other Ph.D. students at Politecnico has been extremely valuable. Thank you Lorenzo, Luca, Pierre and Amarildo. I hope we will continue working together.

I appreciate the hours spent with Francesco and Martino, co-authors in several papers. Their expertise helped achieve important results.

I'm grateful for my family that is always close to me, ready to support, encourage and help out in every possible way.

A special thanks to my girlfriend Barbara, who read and checked the thesis in its entirety. She is always there for me, cheering me on.

I also appreciate invaluable feedback of the reviewers, Prof. Guéant and Prof. Kolm.

Finally, I would like to thank all those who helped me in the attempt to transform the technologies presented in this thesis into a entrepreneurial project. I hope this dream will soon become a reality.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ADMM Alternating Direction Method of Multipliers 63

AUM Assets Under Management 21

B&H Buy and Hold 87

B&S Black and Scholes 17

BBG Bloomberg 14

bps Basis Points 16

CDA Continuous Double Auction 13

CDS Credit Default Swaps 14

CET Central European Time 51

CME Chicago Mercantiles Exchange 17

COCO Conservative Online Convex Optimization 69

CP Conservative Projection 5

CRDG Constrained Reward Doubling Guess 75

CRP Constant Rebalancing Portfolio 57

CS Conservative Switching 75

DQN Deep Q Network 80

DVA Debt Valuation Adjustment 6

ETFs Exchange Traded Funds 13

List of Symbols and Notation

Markov Decision Processes and Online Learning

$\mathcal{S}$	state space
$\mathcal{A}$	action space
P	transition model
$\mathcal{R}$	reward model
r	reward function
μ_0	initial state distribution
γ	discount factor
π	policy
μ	initial state distribution
τ	trajectory
G_τ	return of a trajectory
V_π	state value function of policy π
Q_π	action value function of policy π
J_π	expected return of policy π
$f(a_t, y_t)$	loss function
L_t	cumulative loss function
R_T	regret

Option Portfolio Optimization

$\mathbf{a}_t$	portfolio allocation
Δ_{M-1}	$(M-1)$-simplex in $\mathbb{R}^M$
$\mathbf{y}_t$	price relatives
W_t	wealth
$\tilde{W}_t$	wealth including costs

Quantitative Trading

P_t	asset mid price
a_t	portfolio position
bid-ask	bid ask spread

Dealer Markets

$P_{t,buy}$	buy limit order price
$P_{t,sell}$	sell limit order price
P_t	asset mid price
v_t	trade size
z_t	current inventory
$\phi(z)$	inventory penalty

Option Hedging

S_t	underlying price
C_t	B&S call option price
P_t	B&S put option price
r_f	risk free rate
σ	volatility
$\tau(t,T)$	year fraction between two dates
K	strike price
$A_S(t)$	annuity
$P_S(t,\theta)$	survival probability
Pay_t	payer option price
Upf_t	CDS Upfront

Optimal Execution

X	number of shares to execute
T	total execution time
$N+1$	number of execution timesteps
P_t	mid price
Q^k_{ask}	volume of outstanding limit orders at *k*-th best ask
TD^k_h	total depth: cumulative sum of volume over multiple price levels
v_{imb}	volume imbalance
Z_{imb}	price imbalance

CHAPTER 1

Introduction

Trading robots are starting to dominate the financial markets (Gunia, 2019; Economist, 2019). Many quantitative funds, such as Renaissance Technologies (Keh, 2018) and Two Sigma, are managing capital via computerized trading strategies. Market making desks in top dealers are now mostly comprised of computer scientists who build software in collaboration with traders. These trading algorithms are, in most cases, expert systems, built by highly qualified individuals with a deep knowledge of the financial markets. Often times, the algorithms' objective is not to anticipate the markets, but to be faster than everyone else. Thus, it becomes necessary to delegate trading to computers with a low-latency infrastructure and colocated servers. This is exactly the functioning of High Frequency Trading (HFT) firms such as Optiver.

There are other usages of advanced algorithms, for example in the asset management world, the exploitation of Machine Learning (ML) to analyze enormous quantities of data, which would be out of reach even for an army of analysts. The analyses resulting from this approach, now referred to as quantamental (Tadoori, 2020), aid the asset managers in taking the investment decisions. Thus, to summarize, current technologies are used either to support the decision-making of traders or to systematize the traders' strategies to act in high frequency.

Recent ML technologies have demonstrated the capabilities of learning decision making tasks at such a level that they surpassed even the world champion in the field, for example in the game of Go (Silver et al., 2016). From this inspiring success stems the question if these technologies can rival humans also in the financial markets domain. The objective of this dissertation is to address this question by applying the same successful technologies

to the main decision making processes of the financial markets arena, namely, portfolio optimization, proprietary trading, market making, hedging, and optimal execution. These areas all share a common decision making core structure, that is making a trading decision conditional on the current status of the financial markets. Nonetheless, they differ in terms of the objective, which can be, for instance, to maximize profit, minimize risk, reduce transaction costs, maximize the number of transactions or a combination of the above.

Reinforcement Learning (RL) (Sutton and Barto, 1998) algorithms are the enabler for such an endeavor. RL is one of the main paradigms of ML, together with *Supervised Learning (SL)* (Bishop, 2006), which is based on the problems of classification and regression, and *Unsupervised Learning* (Rokach and Maimon, 2005), which focuses on finding patterns in unlabelled data, *e.g.,* clustering and feature extraction. RL enables learning a policy to optimize an objective by interacting with an environment, and can be applied in many different contexts such as games (Silver et al., 2016), robotic control (Meyes et al., 2017) and autonomous driving (Wang et al., 2018). Amongst the many players that interact in the markets, this dissertation concentrates on *quantitative traders, portfolio managers*, and *market makers*.[1] For each player, we analyze how to create an *autonomous agent* that independently learns how to replicate the main tasks required by these players.

1.1 Reinforcement Learning for the Financial Markets

The work of a trader can be modeled as a sequential decision-making process, acting on the financial markets by observing the information available and taking an investment decision. The available information can be of any type. The most relevant is the price of the assets considered for the investment, but it may also include related assets, the latest economic and political news, and fiscal and economic policies.

Currently, many traders like to follow a manual "instinctive" approach to trading. However, even traders with great experience may be subject to biases (Feng and Seasholes, 2005) and errors, underperforming due to human related limits. All these factors can be avoided through the use of systematic, hard coded trading strategies, which replicate the behavior of the traders. Furthermore, well designed learning algorithms can surpass the trading strategies of human traders by identifying trends and patterns unrecognizable to humans, thanks to the processing capabilities of computers. Devising such algorithms – one of the objectives of this dissertation – holds immense economic value, as it may enable the possibility of obtaining positive performance with low risk.

Designing quantitative trading strategies is not the only topic of this dissertation, as we also analyse other financial tasks such as portfolio optimization, market making, hedging, and optimal execution. In particular, market makers play an extremely important role as they are entitled by national regulators to generate liquidity, enabling the well-functioning and stability of the markets. Market making is a high frequency job, since it is necessary to monitor the price in a continuous fashion and for certain assets, also answer to Request For Quotes (RFQ). Thus, market makers, if not using automation software, are one of the financial players most impacted by biases or other human factors, as the high number of rapid decisions required makes any small distraction potentially impactful. In this dissertation, we propose to implement RL algorithms to learn market making, hedging and optimal execution strategies. The widespread adoption of such algorithms could eliminate

[1] All these different market players, may be generally referred to as traders.

human limitations and increase the efficiency of financial institutions and, thus, of financial markets.

1.2 Reinforcement Learning

In nature, both a baby who learns to walk and a dog that learns to sit adopt the same approach, *i.e.,* learning by trial and error. RL derives inspiration from biology: it lets an agent interact with an environment and provides it with a reward that conveys the effectiveness of its actions. By maximizing the rewards, the agent learns autonomously the optimal way to reach its objective. RL is an extremely generic approach that can, in principle, solve any sequential decision-making process, such as robot control, and playing games.

A sequential decision-making process can be labelled as a Markov Decision Process (MDP) (Puterman, 2014), defined as an *agent* interacting with an *environment*. The environment describes the world the agent can interact with. Moreover, the agent can make decisions, referred to as *actions*, and for these decisions receives a feedback from the environment, the *reward*. The action will cause a change in the environment with a certain probability, or *transition probability*. The information of the current condition of the environment is shown to the agent through the *state*. We often broadly refer to RL, even though it is more correct to distinguish between RL, online planning, and online learning. The differences between these approaches are outlined in the dissertation. However, it is important to recall that they all have the same objective of learning sequential-making decision processes.

Why Reinforcement Learning? The task of any market participant can be described as a sequential decision-making process, where the state is the current market information, the action is the investment or trade, and the reward, in general, is how well the trade performed, *i.e.,* the *Profit and Loss (P&L)* of the investment.

The adoption of RL techniques allows to solve sequential decision-making processes by learning the optimal policy that not only optimizes the immediate gain, but also plans the best strategies to handle future market dynamics. Furthermore, most RL algorithms can learn directly from market data, without making any assumptions on the dynamics of the financial assets. In RL this property is referred to as *model-free*.

Why not Supervised Learning? SL algorithms are conventionally used with the objective of solving classification or regression problems by optimizing the loss of labelled examples. These techniques have achieved impressive results in tasks such as image recognition (Lu and Weng, 2007) and natural language processing (Chowdhury, 2003). SL is used for several scopes in the financial context, these include building signals, forecasting volatility, classifying clients, and rapidly analyzing large quantities of data such as balance sheets, income statements, and financial ratios to provide an aid in stock picking (quantamental). The application of SL to the topics covered in this dissertation would provide limited value, since SL could be used only to predict what the market will do next, but would not be able to advise how to behave optimally. Contrarily, RL learns trading strategies that both maximize the immediate reward and plan ahead to design the path that optimizes the gain

over the entire trading period. Hence, it naturally appears as the optimal paradigm for the scope of this dissertation.

1.3 Original Contributions

This dissertation has the objective of illustrating and deep-diving in several applications of RL to the financial markets. Many of the problems considered, including portfolio optimization, hedging, and optimal execution, have previously been tackled from a financial mathematics perspective. In these scenarios, in the majority of cases, the market dynamics are given a stochastic characterization and the goal is to find an analytical solution that optimizes a utility function. The solutions could serve as guidance for traders, but the unrealistic assumptions required by these models, such as continuous time trading when pricing derivatives or the backward looking market characterization when optimizing portfolios, leave them unusable in practice. This dissertation explores and proposes an alternative approach for crafting trading, portfolio optimization, hedging, market making, and optimal execution strategies suitable for deployment in a production environment. In fact, in this work, when defining the financial environment, the focus is not on characterizing the dynamics of the assets, but on accurately modeling the characteristics of the financial instruments and the trading frictions. The proposed algorithms will deliver several benefits, including the following:

- management of portfolios with market exposure but with a low risk profile;
- achievement of profitable and low risk quantitative trading strategies with no correlation to market movements;
- achievement of profitable, low risk real-time pricing strategies also capable of answering to RFQs in the market making context;
- reduction of market risks, specifically in the case of the inventory risk of market makers;
- reduction of execution costs, a useful tool when managing portfolios, trading, and hedging.

The use of the proposed approaches would also have a positive impact in terms of process automation, which, in turn, leads to reduction of operational risk and associated costs.

The paragraph below presents a brief description of the different financial problems covered in this dissertation, emphasizing the contributions to the state-of-the-art. These topics are examined fully by means of dedicated chapters in the dissertation, namely Chapters 5, 6, 7, 8, and 9.

1.3.1 Online Portfolio Optimization

Online Portfolio Optimization (OPO) (Li and Hoi, 2014) is an intrinsically different approach from the currently well-known Modern Portfolio Theory (MPT) (Markowitz, 1952). Differently from MPT, it does not give a statistical characterization to the assets, but assumes the market is adversarial and gives *regret* guarantees of converging to the oracle. This dissertation proposes two main contributions to the OPO framework, in the

form of two novel algorithms. The first is the *Online Gradient Descent with Momentum (OGDM)* algorithm (Vittori et al., 2020a), a modified version of Online Gradient Descent (OGD) (Zinkevich, 2003), with the aim of dealing with transaction costs in the OPO framework. This algorithm is supported by a theoretical analysis of the total regret (regret in the presence of costs) and an empirical comparison with state-of-the-art OPO algorithms. The second is the *Conservative Projection (CP)* algorithm (Bernasconi de Luca et al., 2021), which gives theoretical guarantees of performing worse than a known benchmark with low probability, a typical requirement in the asset management world. These topics are covered in depth in Chapter 5.

1.3.2 Quantitative Trading

Most of the existing trading algorithms are expert systems, where experienced traders with computer scientists write hard coded rules to exploit arbitrages or implement trading ideas coming from the trader's experience. These trading algorithms can be crafted in different ways, from high (milliseconds) to low (minutes or hours) frequencies. In either case, they are designed to systematize the strategy of the trader. In this dissertation, we focus instead on trading strategies self-learnt by the RL algorithm, initially testing the use of Fitted Q Iteration (FQI) (Ernst et al., 2005), following Riva et al. (2021), and then Monte Carlo Tree Search (MCTS) (Kocsis and Szepesvári, 2006), following Vittori et al. (2021) to learn a realistic trading strategy. The methodology and the experimental results achieved, extensively elaborated in Chapter 6, are the main contributions to this theme.

1.3.3 Market Making

Market makers play a fundamental role in the financial markets, providing liquidity and thus stability. Nonetheless, as all market participants, market makers too have the objective of achieving a profit while minimizing risk. In this dissertation, we describe the market maker problem as an N-player stochastic game of which we want to find the equilibrium. To solve this game and find the equilibrium, we apply Mean Field Games (MFGs) (Huang et al., 2006) where each player is an RL agent.

Specifically, for the first time, we model the dealer market environment as a multi-agent game and solve it using model-free RL on discrete-time MFGs (Gomes et al., 2010). The proposed approach is extremely flexible, as the equilibrium is learned by self-play for any model that governs the exogenous stochasticity of the problem, *e.g.,* the price of the asset or the arrival times of the RFQs. We analyse the behavior of this approach with an extensive experimental campaign, where different trading agents compete against each other, as outlined in Chapter 7. Finally, we draw important conclusions from the experiments, shedding light on the inner working of dealer markets.

1.3.4 Hedging

An options market maker habitually prices simultaneously a large number of options and hedges the risk caused by the greeks (*e.g.,* delta and vega). In this thesis, we focus on hedging the first order risk caused by the underlying, referred to as the delta risk. To hedge the delta risk, the market maker is often aided by an automatic software that hedges the delta of each new trade and periodically (every few hours), hedges the delta risk of the entire

portfolio of options to keep it delta neutral. Trading the underlying asset creates transaction costs that can be relevant, depending on the liquidity of the underlying instrument. There are three main methods to optimize these costs: by optimizing the execution (and blindly following the delta hedge), by optimizing the hedging policy, or a combination of the two. In the *hedging* stream of this dissertation (see Chapter 8), we focus on optimizing the hedging policy using a risk-averse policy search algorithm. We analyze the case of equity option hedging, following Vittori et al. (2020b), and credit index option hedging.

Additionally, this dissertation also considers the Debt Valuation Adjustment (DVA) hedging problem (see Appendix C). DVA hedging is related to option hedging but is characterized by a higher degree of complexity, as DVA is a hybrid risk that depends on multiple risk drivers and cannot be hedged by means of a single highly correlated instrument.

Hedging with RL is illustrated in Chapter 8, the contribution is mainly methodological and experimental.

1.3.5 Optimal Execution

Financial institutions often find themselves having to execute such large orders that they cause the price of the instrument to move in a direction that increases the cost of the transaction. There are various techniques used to minimize this *market impact*, which generally consist in splitting the large order into smaller ones. One of the difficulties when studying this problem is that in a historical simulation it is not possible to reproduce the effect of the trade if not by making assumptions on the market impact. Thus, we tackle this problem by using the multi-agent market simulator ABIDES (Byrd et al., 2019) to train and test our approach. Another problem is the non-stationarity of the market. We propose an online approach to adapt in rapidly changing market conditions. The methodological and experimental contribution focuses on using RL to learn an optimal execution algorithm as described in Chapter 9.

1.4 Overview of the Dissertation

The dissertation is organized in two parts. In the first part, (Chapters 2 through 4) we layout the financial knowledge and RL fundamentals that prepare the reader to understand problems analyzed in the second part (Chapter 5 through 9).

Part I: Financial markets and RL Fundamentals

Part I is divided in three introductory chapters:

- Chapter 2 defines the key mechanisms driving the financial markets and describes the main financial instruments taken into consideration in this dissertation. This chapter also addresses the major players in the financial markets, with a focus on asset managers, quantitative traders, and market makers.

- Chapter 3 is devoted to an introduction to the algorithms considered, initially defining MDPs, and then giving an overview on RL, online planning, and online learning.

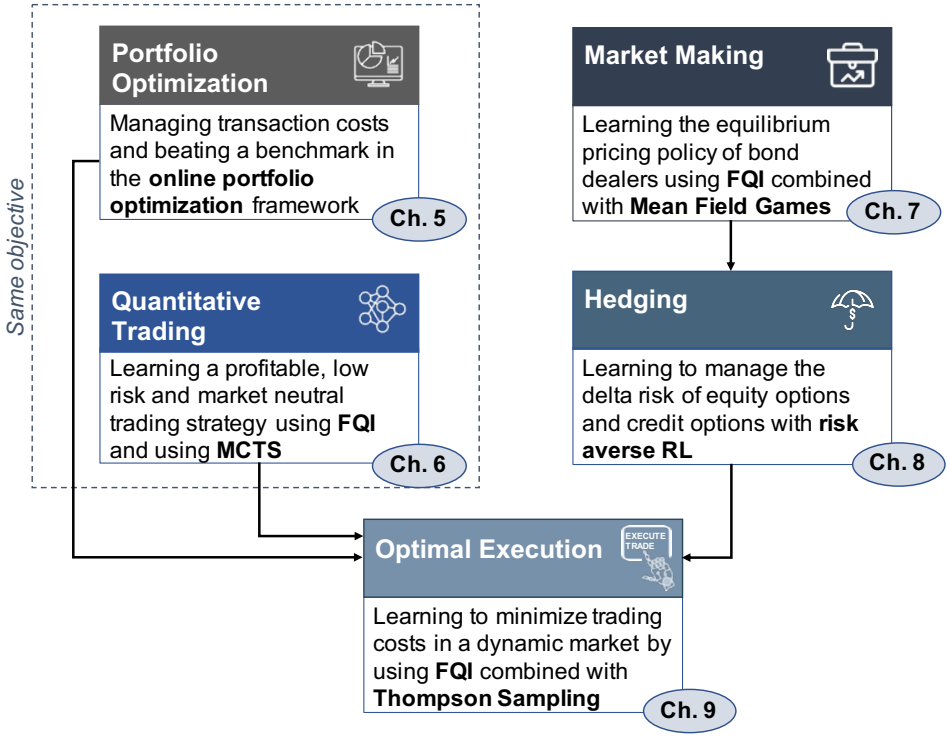

Figure 1.1: *Map of Part II.*

- Chapter 4 goes through the practical steps undertaken to achieve our results, thus comprehensively explaining the approach followed to collect and process the data, test and evaluate the learnt models.

Part II: Learning the Financial Markets through RL

Part II represents the core of the dissertation and contains the original contributions and experimental results. The reader can refer to Figure 1.1 as support to navigate Part II.

- Chapter 5 analyzes the *Portfolio Optimization* problem through the OPO framework and is composed of two main sections. The first section focuses on introducing transaction costs in the OPO framework and is based on published work by Vittori et al. (2020a). The second section centers on portfolio optimization with a benchmark: the typical asset management task of beating a specific market index. This section is based on published work in Bernasconi de Luca et al. (2021).

- Chapter 6 analyzes the *Quantitative Trading* problem, training an agent to go long, short or stay flat on foreign exchange data. This chapter is composed of two main parts. The first part uses the FQI algorithm and is based on the paper by Riva et al. (2021). The second part of the chapter uses MCTS and is based on the paper by Vittori et al. (2021).

- Chapter 7 emphasizes the bond market making problem, introducing a novel method, using MFGs and RL, to find the optimal price quoting strategy. This is a novel contribution that will be submitted for publication.
- Chapter 8 analyzes the hedging problem, assessing two different approaches in separate sections. The first section analyzes the equity option hedging problem based on the work by Vittori et al. (2020b). The second section concentrates on the credit option hedging problem, a novel contribution that will be submitted for publication. A third hedging problem, namely DVA hedging, can be found in Appendix C.
- Chapter 9 treats the optimal execution problem using the multi-agent market simulator ABIDES and analyzing how, through the use of FQI, it is possible to minimize the market impact of a trade. This is a novel contribution that will be submitted for publication.

Reading Part II As depicted in Figure 1.1, the topics described in the five chapters of Part II are strongly connected and comprehend the main money management techniques. Portfolio management and quantitative trading are studied in different streams of literature, but they are becoming continuously more intertwined as they follow the same objective of obtaining a positive return on invested capital. Furthermore, when managing a portfolio or implementing a trading or hedging strategy, it is beneficial to use an optimal execution approach to reduce trading costs.

A market maker's main task is to continuously price an asset both on the bid and ask sides, which is the topic tackled in Chapter 7 for the case of bonds. When accumulating inventory, market makers are exposed to market risk. In the case of bonds this means using bond futures to hedge against interest rate risk, whereas in the case of options market making, this means hedging the delta risk using the underlying (Chapter 8). This results in the fact that a market maker is always dealing at the same time with both the pricing and the hedging problems. Moreover, some specialist firms continuously price an asset while also taking a market position with trading strategies, hedging unwanted risks and optimizing the execution.

All the proofs of theorems and lemmas can be found in the Appendix A. Finally, Chapter 10 is dedicated to the conclusions of the dissertation.

Part I

Intro to Financial Markets and Reinforcement Learning

CHAPTER 2

Financial Markets Fundamentals

Although the adoption of RL to model players in the financial markets has only started recently, its use is extremely intuitive: market players take decisions in a sequential form, observing the current information available and using it to decide which trade they should execute to maximize their objective. This idea drives the entire dissertation. However, to create autonomous agents that are suitable for application in the real markets, it is necessary to model with great detail the simulators employed to train and test the algorithms. Thus, a solid understanding of the financial assets used and of the objectives of the market players is required.

This chapter serves as a guide to the reader, introducing and explaining in detail the financial vocabulary and primary concepts that help understand the problems addressed in Part II. For the reader interested in expanding the topics of this chapter, we recommend to refer to financial text books, such as Hull (2003).

Chapter outline This chapter is composed of four main sections. Section 2.1 provides a general overview of the financial markets. Section 2.2 covers the financial instruments embraced in the dissertation, whereas Section 2.3 describes the major players that interact in the markets. Finally, Section 2.4 introduces algorithmic trading.

2.1 Introduction to Financial Markets

Financial Markets (also referred to simply as “market” or “markets”) generally denote the collection of marketplaces where the exchange of financial securities occurs. Their main function is to connect investors looking for returns with companies or entities looking for financing; thus, they are the key enabler of the functioning of a capitalist society. Financial markets can be subdivided depending on the location and the assets exchanged, but, in general, they can be thought of a single place where everything is interconnected. For example, a news on a specific European company can influence the price of an option on an index that contains that company and is traded in the US.

There are three main types of financial instruments (also referred to as financial “assets”) that are exchanged in the markets: *stocks, bonds*, and *derivatives*. Stocks represent partial ownership of a company, whereas bonds are a debt instrument, similar to loans but with the possibility of being traded as small units of the total amount. Finally, derivatives are complex instruments, whose price depends on that of a simpler instrument (referred to as "underlying"). Financial instruments are described in greater detail in Section 2.2.

Lastly, there are the asset classes, which include equity, fixed income, credit, Foreign eXchange (FX), and commodities. Equity includes stocks and equity derivatives, and similarly fixed income and credit include bonds and derivatives. FX refers to the global market place for currencies and is exchanged either spot, or through derivatives. A commodity represents a basic good used in commerce, such as corn, gold or oil. Also commodities can be exchanged using different types of financial derivatives. Asset classes are described in greater detail in Section 2.2.5. The following paragraphs illustrate how financial assets are created and exchanged.

Primary vs Secondary Market In general, financial assets, before being exchanged on the markets must be originated, *i.e.,* created. Hence, we now introduce a distinction between two types of markets, namely, the *primary market*, where assets are originated, and the *secondary market*, where assets are exchanged.

Primary markets differ depending on the instrument in consideration. In the case of stocks, new shares are issued through an Initial Public Offering (IPO), while new bonds or derivatives are created through an issuance. An origination is usually carried out with the support of investment banks behaving as financial advisors and aiding in finding buyers for the new instrument. An IPO is a long process that allows a private company, owned from by a relatively small number of investors, to become public, so that anyone can buy a share and participate in the ownership of the company. Bond issuance has a similar, but more streamlined process that involves investment banks with the Debt Capital Markets unit. Finally, derivatives’ origination stands out as the most agile process, and is handled by the Capital Markets Structuring unit.

Once a financial asset has been originated, it can then start trading on the secondary market. In the case of stocks, the secondary market is the so-called stock market, such as the New York Stock Exchange, NASDAQ, London Stock Exchange, and many others. In the case of bonds, secondary markets are also called dealer markets (detailed in the following section).

Lastly, in the case of derivatives, the sale of an existing derivative to another counterparty is referred to as novation. For some standardized derivatives such as futures there is also a

Last	Last Vol	Total Vol	Close	Daily Low	Daily High
4045.00	2	367267	4097.50	4033.50	4101.50

Implied

Bid		Offer	
Volume	Price	Price	Volume
136	4044.50	4045.00	62
327	4044.00	4045.50	293
348	4043.50	4046.00	427
620	4043.00	4046.50	426
358	4042.50	4047.00	463
330	4042.00	4047.50	348
325	4041.50	4048.00	327
318	4041.00	4048.50	294
305	4040.50	4049.00	281
512	4040.00	4049.50	288

Figure 2.1: *Example of limit order book of the EuroStoxx 50 futures.*

liquid market. In this case, there is no distinction between primary and secondary markets, as a new contract is created every time a trade happens and a position is closed through an offsetting trade, not by novation.

2.1.1 Trading Venues

Regulated Exchanges Exchanges are marketplaces where stocks, Exchange Traded Funds (ETFs), commodities, and some derivatives, most commonly futures and options, are traded. In exchanges, buyers submit their *bids* and sellers submit their *asks*, creating a Continuous Double Auction (CDA) process. The submitted orders are also known as limit orders and the collection of all outstanding limit orders is referred to as the *Limit Order Book (LOB)* (see Figure 2.1), which we define as follows.

Definition 2.1 (Limit Order Book)**.** *A limit order book is a record of all the currently outstanding limit orders.*

Definition 2.2 (Bid-Ask Spread)**.** *The best bid price is the highest price of all limit purchase orders and the best ask price is the lowest price of all sell orders. The bid-ask spread is calculated as the difference between the best ask and best bid price (we refer to it as bid-ask or bid-ask spread).*

As is illustrated in Section 2.4.1, half of the bid-ask spread is considered as the transaction cost, when the entire trade can be absorbed by the volume present in the first level of the LOB. To trade on an exchange, it is necessary to either be an exchange member by purchasing a membership or, otherwise, to go through a broker. The regulated exchanges that have just been described are known as *lit* as it is possible to see the LOB for each asset as in Figure 2.1. There are also venues, referred to as *Dark Pools*, in which the LOB is hidden. The regulatory environment has caused a fragmentation of liquidity, in fact, a single asset can be traded both on multiple lit exchanges and dark pools as well as other venues.[2] To minimize execution costs it may be beneficial to split the order between the different venues, this is commonly labelled as *smart routing*.

[2] https://fragmentation.fidessa.com/.

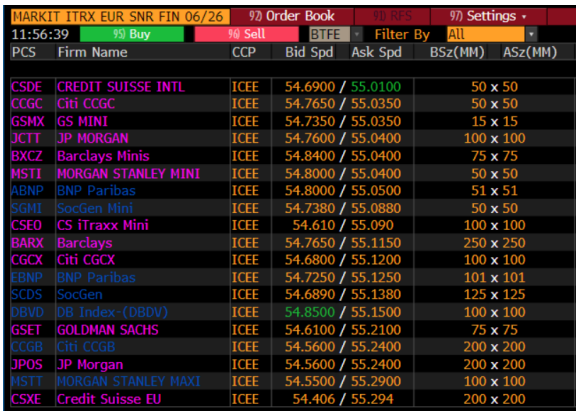

Figure 2.2: *Example of MD2C platform for the Itraxx Senior Financial Index.*

Over The Counter Bonds and derivatives such as Interest Rate Swaps (IRS) and Credit Default Swaps (CDS) (see Section 2.2) are not traded on regulated exchanges but on dealer markets. These types of financial instruments are identified as *Over The Counter (OTC)* assets. We define OTC derivatives as:

Definition 2.3 (OTC Derivative)**.** *An OTC derivative is a financial contract that does not trade on a regulated exchange, and that can be tailored to each party's needs.*

Derivatives are primarily used to manage exposure to the underlying financial risk. Trades of OTC instruments take place via recorded phone, through qualified chats (such as the Bloomberg (BBG) chat) or via *Multilateral Trading Facilities (MTFs)* also called Multi-Dealer-to-Client (MD2C) platforms, such as the Bloomberg Derivatives landing page (see Figure 2.2), the Bloomberg Fixed Income Trading, Tradeweb, or MarketAxess, that substitute traditional voice negotiations. While via chat and phone derivatives can be customized to fit the investor's needs, those traded on MD2C platforms are standardized. There are OTC markets in essentially every asset class, what is described in this thesis refers to OTC credit markets.

Differently from stock exchanges, where each participant can submit a limit order, two distinct types of actors play a role in OTC trading: the *dealers* (also called market makers, see Section 2.3), and the clients. Dealers continuously quote bid and ask prices. Figure 2.2 displays an example where various dealers, mostly investment banks, show their bid and ask prices and the size for which these prices hold. The dealer can exhibit two types of quotes: a *firm* quote, that means that it is the final price that can be executed, and an *indicative* quote, that instead means that it is an approximate price potentially subject to slight changes when the trader receives a Request For Quote (RFQ). Following the reception of the RFQ, the dealer communicates the firm quote. The second type of players, *i.e.,* clients, are all the other market participants who are not dealers in this instrument. If clients want to trade, they must pick the dealer they want to trade with, basing their decision on the best price displayed. Thanks to the MD2C platforms, clients can easily ask a quote to multiple dealers, incentivizing each dealer to quote the best price, since they do not know who the competitors are and what price they will show. A feature of MD2C platforms is the

limited visibility of the firm price of dealers to the client only, which implies that a market maker receives as feedback only the acceptance/rejection of her offer. Moreover, contrarily from the regulated exchanges where all the players are anonymous, clients trading OTC instruments know the identity of the dealer. In Chapter 7 we propose an approach using RL to optimize the RFQ pricing of dealers.

Brokers Brokers help the matching of demand and offer and may stand between the market participants and the exchanges or the dealers. The most well-known brokers are probably the large online brokerage firms providing market access to retail investors, such as Fidelity, Charles Swab, Interactive Brokers and TD Ameritrade. Most investment banks act as brokers for asset management firms, pension funds and other players without exchange memberships, thus providing mostly execution services. For OTC instruments, there are even inter-dealer brokers, that act as a mini-exchange through which dealers can anonymously trade with other dealers to offload positions (see Equation (2.25)), helping to smooth the market.

2.1.2 Regulation

Since the financial crises in the first decade of the century, the financial world has become heavily regulated. In the US, for example, the Dodd Franck Act, established in 2010, enforced a strict monitoring and control of the financial stability of the US markets, obligating banks higher reserve requirements.[3] Among the many provisions, it expanded the power of the Securities and Exchange Commission and introduced the Volker Rule, which prohibits banks from using their accounts for proprietary trading, protecting consumers who deposit their savings within these banks.

In Europe, two main regulations oversee the financial world, the Basel Accords and MiFID. The former regulate the leverage ratios and capital requirements that banks need to abide by.[4] Capital requirements aim to respond to the necessity to handle periods of financial distress. In particular, these requirements are generally defined as a percentage of the total risk weighted assets of the bank, *i.e.,* the amount of the bank's investments weighted by a coefficient defined by the regulatory authorities. The latter, MiFID, is a European regulation focusing on increasing trading transparency, fairness, and standardization of regulatory disclosures.[5] Initially MiFID applied mostly to stocks, but has later expanded, with MiFID II, to introduce increased regulation on OTC instruments and tighten restrictions on dark pools.

2.2 Financial Instruments

To appropriately train and test an RL agent it is fundamental to model the environment correctly, and thus, in this case, the financial instruments. In this section, we focus on describing the main financial instruments considered, delving in detail on those used in the applications of Chapters 5, 6, 7, 8, and 9. For more details on the financial instruments, the reader can refer to Appendix B.1.

[3] https://www.cftc.gov/LawRegulation/DoddFrankAct/index.htm.

[4] https://www.bis.org/bcbs/basel3.htm.

[5] https://www.esma.europa.eu/policy-rules/mifid-ii-and-mifir.

There are two main kinds of financial assets, the "simple" assets, also referred to as *cash*, which include stocks, bonds, and FX spot, and *derivative* instruments, whose price derives from one or more *underlying* assets. For instance, derivatives encompass futures, options, and swaps.

The size of a derivative is usually referred to as *notional*, the *sensitivity* represents how much the value of the derivative will change because of a movement of the underlying instrument. Generally, the sensitivity is defined as the derivative of the instrument relative to the underlying and it is calculated considering how much the underlying price moves percentage wise, or by how many *Basis Points (bps)* it moves, where 1 bp $= 0.01\%$. In general, the value of a derivative is referred to as the *fair value* or *Mark-to-Market (MtM)*. Depending on the type of derivative instrument, the contract may be opened without any cash exchange, by posting an initial margin, by paying an upfront or by paying a premium, but it is not necessary to pay the entire value of the notional. This characteristic can be referred to as being *unfunded*. Cash instruments are *funded*, since it is necessary to exchange the entire value of the instrument to obtain ownership.

2.2.1 Cash Instruments

Stocks Stocks are the most popular type of financial asset. They represent the ownership of a corporation and a single unit is referred to as a *share*. In this dissertation, we consider stocks when tackling online portfolio optimization in Chapter 5.

Bonds Bonds are debt instruments, similar to loans but shared among many different owners instead of a single entity. Bonds are OTC instruments, so they are generally traded on MTFs, even though some brokers also let retail clients trade them. In general, bonds are not very liquid, they exhibit very low volatility, have high transaction costs and are normally considered long term investments. In Chapter 7 we consider the problem of a bond dealer, specifically how to learn an optimal pricing and RFQ response policy.

2.2.2 ETFs and Futures

ETFs An ETF is an instrument that typically reproduces a known index, but can be traded on the stock market just like if it were a stock. ETFs have recently gained a lot of popularity because they give investors the opportunity to take exposure to a market index at a low price without resorting to futures or having to manually replicate the index by buying all the stocks that it contains.

Definition 2.4 (Market index)**.** *A market index is a collection of financial assets, commonly stocks. The returns of the market index are calculated as a weighted average of the returns of the constituents.*

It is possible to apply several weighting methods to calculate the indexes, the most common include market-cap weighting, revenue-weighting, float-weighting, and fundamental-weighting. As an example, one of the most famous indexes at a global level is the S&P 500 index, that is the market-capitalization-weighted index of the 500 largest publicly-traded companies in the U.S. The most renowned ETF that replicates this index is the SPY ETF. Similarly to stocks, also ETFs are considered when tackling online portfolio optimization in Chapter 5.

Futures Futures and forwards represent an agreement to buy or sell an asset for a certain price at a certain future time (whereas spot contract consist in an agreement to make a trade today). The payoff of a long position in a futures/forward contract is $S_T - K$ where K is the agreed upon price and S_T is the spot price of the asset at maturity. Forwards are OTC instruments, while futures actively traded on regulated exchanges, such as the Chicago Mercantiles Exchange (CME), Eurex, and Euronext. Typical futures have as underlying bonds (e.g., BTP, Bund), equity indexes (e.g., S&P 500 index), FX spot prices, and commodity spot prices (crude oil, natural gas, and many others). Depending on the type of futures, or on the counterparty agreement, the settlement can be either physical or cash. A physical settlement occurs when the underlying instrument has to be delivered, while a cash settlement means that only the monetary value is exchanged. In this dissertation, we assume all futures considered to be cash settled. Futures contracts can easily be shorted, so are compatible with the quantitative trading approach described in Chapter 6. Also, in Chapter 8, the underlying of equity options can be a futures contract.

2.2.3 Equity Options

Equity options are traded on exchanges like the American Stock Exchange (AMEX), the Chicago Board Options Exchange (CBOE), the New York Stock Exchange (NYSE), and several others. There are many types of options, for the sake of this dissertation, we focus on vanilla European options. These are contracts that offer the buyer the right to buy (call option) or sell (put option) a certain quantity of the *underlying asset* at a predefined price at a certain future time. Although not discussed in this work, there are more complex types of options such as American, Bermudan, as well as others. In the case of equity options, the underlying asset is in general a stock or an equity index .

In this section we describe the Black and Scholes (B&S) (Black and Scholes, 1973) pricing framework, the benchmark considered in this paper. The B&S model is used to convert option prices into implied volatilities. This framework entails the modelling of two elements, the underlying and the deriving option price. In the B&S framework, the underlying behaves as Geometric Brownian Motion (GBM) (see Section 4.3.1 for more details). Let S_t be the underlying price at time t, r_f the risk-free rate (we assume it to be 0 compatibly with current market conditions), σ the volatility, T the time of maturity, $\tau(t,T)$ the Time To Maturity (TTM) expressed as year fraction, K the strike price. The B&S call price C and put price P are:

$$\begin{aligned} C(t,S_t) &= \Phi(d_t)S_t - \Phi(e_t)Ke^{-r_f\tau(t,T)}, \\ P(t,S_t) &= \Phi(-e_t)Ke^{-r_f\tau(t,T)} - \Phi(-d_t)S_t, \\ d_t &= \frac{1}{\sigma\sqrt{\tau(t,T)}}\left[\log\left(\frac{S_t}{K}\right) + \left(r_f + \frac{\sigma^2}{2}\right)\tau(t,T)\right], \\ e_t &= d_t - \sigma\sqrt{\tau(t,T)}, \end{aligned} \tag{2.1}$$

where Φ is the cumulative distribution function of the standard normal distribution. We introduce

$$\frac{\partial C(t,S_t)}{\partial S} = \Phi(d_t), \tag{2.2}$$

which is known as the option delta and for our position (a long call of unitary notional) is bounded between 0 and 1. In particular when $\tau(t,T)$ is relatively small and $\frac{S_t}{K} \ll 1$,

$\frac{\partial C(t,S_t)}{\partial S} \to 0$ and $C(t,S_t) \to 0$; instead if $\frac{S_t}{K} \gg 1$, $\frac{\partial C(t,S_t)}{\partial S} \to 1$ and $C(t,S_t) \to S_t$. We consider equity options in Section 8.3, from the point of view of an options market maker, but focusing on optimizing the hedge of the underlying instrument.

2.2.4 CDS and CDS Index Options

Credit Default Swap A CDS is a financial derivative or contract that allows an investor to swap or offset her credit risk with that of another investor. To swap the risk of default, the lender buys a CDS from another investor who agrees to reimburse the lender in the case the borrower defaults. Most CDS will require an ongoing premium payment to maintain the contract, which is like an insurance policy. Similarly to a stock index, a CDS index is a weighted average of a number of single name CDS. In this dissertation, we concentrate on the Markit iTraxx Europe Senior Financial index (SNRFIN), which represents a basket of credit default swaps on 30 European financial institutions (banks and insurances), equally weighted, with standardized maturities, coupons and payment dates, but other CDS indexes or even single name CDS behave similarly.

Each CDS index has a premium leg and a protection leg, where the protection leg pays in case of a credit event. Since the premium leg has a standardized 1% coupon, the two legs are unbalanced by an amount that is exchanged at inception as a premium and is referred to as *upfront*. Even though the upfront amount is precisely the price of the derivative, the market does not quote it directly. Rather, following the standard single name CDS convention, what is traded is the running coupon of a par (*i.e.*, upfront equal to zero) CDS. The relation between the traded credit spread S_t (not to be confused with the bid-ask spread) and the upfront, assuming the latter to be received by the protection buyer from the protection seller, is:

$$\text{Upf}(t,S_t) = (1\% - S_t)\, A_S(t) + 1\% \tau(t_{acc}, t), \tag{2.3}$$

where t is the evaluation date, $\tau(t_{acc},t)$ is the year fraction, t_{acc} the coupon date immediately before t, and $A_S(t)$ the annuity at time t.[6] The latter quantity is defined as:

$$A_S(t) = \sum_{t^+ < \{t_i\} \le t_n} \tau(\max(t_{i-1},t), t_i) \frac{P_S(t,t_{i-1}) + P_S(t,t_i)}{2}, \tag{2.4}$$

where $t^+ = t + 1$ day, $\{t_i\}$ is the strip of index coupon dates, t_n is the index maturity, $P_S(t,\theta)$ the survival probability between the present time t, and any future time θ, given the current credit spread $S = S_t$ (notice that $A_S(t)$ does not depend on S_t directly but through $P_S(t)$).[7] The survival probability can be approximated as in Jarrow and Turnbull (1995):

$$P_S(t,\theta) = e^{-S_t \tau(t,\theta) \text{LGD}^{-1}}, \tag{2.5}$$

with Loss Given Default (LGD) usually set to 60% by convention. Making trading decisions based on the credit spread is convenient as the upfront amount has jumps at the coupon dates due to $\tau(t_{acc},t)$, while the credit spread maintains a smoother behavior. We consider the SNRFIN in Chapter 8.4, where it is used as underlying to hedge SNRFIN options, and in Appendix C, where it is one of the underlyings used in the DVA hedging problem.

[6] In the computation of the accrual term the year fraction is modified adding an extra day.

[7] Notice that if t is the day before a coupon date, this coupon is excluded from the strip.

Credit Index Options In this section, we consider options on CDS indexes. A *receiver* option gives the buyer the possibility of selling protection on the index at the expiry date at a credit spread equal to the strike. Conversely, a *payer* option gives the buyer the choice of buying protection at the expiry date at a credit spread equal to the strike. Upon exercise in case of a payer (receiver) option, the option seller (buyer) physically delivers the underlying CDS index. In terms of the strike K and the traded credit spread S_T at expiry, the payoff at expiry is:

$$\max((S_T A_S(T) - K A_K(T)), 0), \tag{2.6}$$
$$\max((K A_K(T) - S_T A_S(T)), 0), \tag{2.7}$$

respectively for a payer (Pay) and a receiver (Rec) option and where $A_K(T)$ is the same expression as $A_S(T)$ that considers $S_t = K$ in $P_S(t)$. In this work, for simplicity we consider the payoff:

$$\max((S_T - K)A_S(T), 0), \tag{2.8}$$
$$\max((K - S_T)A_S(T), 0), \tag{2.9}$$

which allows a treatment à la B&S on S_t, since the payoff of Equation (2.8) can be seen as a call on the underlying S_t.[8]

Similarly to Section 2.3.3, considering an option traded at time t with expiry T and strike K:

$$\begin{aligned}
&\text{Pay}(t, S_t) = \left[\Phi(d_t)S_t(T) - \Phi(e_t)K\right] A_S(T), \\
&\text{Rec}(t, S_t) = \left[\Phi(-e_t)K - \Phi(-d_t)S_t(T)\right] A_S(T), \\
&d_t = \frac{1}{\sigma\sqrt{\tau(t,T)}}\left[\log\left(\frac{S_t(T)}{K}\right) + \left(\frac{\sigma^2}{2}\right)\tau(t,T)\right], \\
&e_t = d_t - \sigma\sqrt{\tau(t,T)},
\end{aligned} \tag{2.10}$$

where $S_t(T)$ is the forward value of S_t, σ is the volatility, $\tau(t,T)$ the year fraction and we assumed the risk-free rate to be 0.[9] An adjustment of the forward credit spread $S_t(T)$ is necessary since the buyer of a payer (receiver) index option receives (pays) protection substantially from trading time t and not from expiry T, the option price needs to be adjusted consequently to consider any losses due to the default before T. Under our assumption of zero interest rates, the adjusted forward $S_t(T)$ is:

$$S_t(T) = S_t + \text{LGD}(1 - P_S(t,T))\frac{1}{A_S(T)}. \tag{2.11}$$

We can define the payer option delta as:

$$N_h(t) = \left(\frac{\partial P_S(t,T)}{\partial S}\right)\left(\frac{\partial\, \text{Upf}(t,S_t)}{\partial S}\right)^{-1}. \tag{2.12}$$

These types of options are only traded by financial institutions and are extremely illiquid. Their prices can be obtained by the dealers via selected channels, for example the Bloomberg chat. We consider in Chapter 8.4 the problem of dealers who need to hedge the risk generated by offering liquidity for these options.

[8] We focus on this simplification since the extension to the payoff of Equation (2.6) and (2.7), which is trivial from a numerical/RL perspective, complicates the analytical treatment in a way beyond our interest.

[9] We consider an ACT/365 convention; T for the annuity is the settlement date, T for d_t, e_t is the expiry date.

2.2.5 Asset Classes

To conclude this section on financial instruments, we would like to point out a subdivision in terms of asset classes. As a general rule, an asset class is defined in terms of its underlying risk driver and contains all the financial instruments that depend on that risk driver. While there is in general a strong correlation within an asset class, there is very little or negative correlation between different asset classes. The following paragraphs list and describe the most common asset classes.

Equity Equity is the most common and well known asset class. It consists of stocks and its derivatives such as futures, options, ETFs, equity swaps. Equity swaps behave similarly to IRS (see Appendix B.1.7) but the underlying risk is equity. The Equity realm includes also private companies so private equity deals, SPACS, IPOs, and equity capital markets.

Fixed income Fixed income is based on securities that pay fixed interest rates, thus it includes interest rate swaps, government bonds, and bond futures.

Credit Credit consists of the credit risk of a company. Instruments encompass corporate bonds and CDS.

Foreign Exchange FX can be traded either spot (also referred to as cash) or with futures, forwards, options, and Cross Currency Swaps (CCS). The FX market is one of the largest markets, with trillions of USD being exchanged every day (Debnath, 2019). We concentrate on FX cash trading and consider it in Chapter 6 to learn a quantitative trading strategy. FX cash markets are dealer markets and for the main currency pairs (e.g., EURUSD, USDGBP, etc.) they are extremely liquid. As an example, the EURUSD FX rate is currently around 1.2 (One can obtain 1.2 USD with 1 EUR) and bid-ask spread is in general 2×10^{-5} or 20 pips. When considering FX rates, we use a capitalized three letter symbol for each currency.

Commodities Commodities refers to physical goods such as grains, wheat, gold, copper, natural gas, and many others. These goods are mainly exchanged via futures or forwards, or also commodity swaps. When trading these instruments, it is important to remember to cash settle the trade, otherwise the physical good will be delivered to the trader on the day of expiry.

2.3 Market Players

In the previous section, we described the main financial instruments used to model the RL environment. Other crucial information for this purpose is understanding the different market players and their objectives, *i.e.,* how to calculate their rewards and how they interact with the markets to better specify the state information necessary. Market participants can be generally divided into three main categories: financial institutions building their business on the financial markets, non-financial companies that access the markets either to raise capital, by issuing equity or debt (bonds), or to reduce a risk, and, finally private individuals, who, through the use of increasingly popular online brokerage accounts, gain access to

the financial markets. In particular, non-financial companies can also be referred to as hedgers, and comprehend those that buy derivatives to mitigate a risk. Let us consider now an example. An airline company might want to mitigate the risk from a possible increase in fuel cost, and, therefore, purchase a forward on the fuel price to fix the price it will pay in the future. In this case, the airline company is acting as a hedger. The category of private individuals has been rapidly increasing in importance, thanks to the arrival of discount online brokerage firms such as Robinhood. Thanks to these low costs accounts and the influence of social media, specifically Reddit forums, private individual traders have been causing unprecedented effects in the financial markets (a renowned example is the GameStop case).[10]

Among the financial institutions, which are the focus of this dissertation, we can further distinguish in directional and non-directional ones. The former group encompasses players such as asset managers, pension funds, mutual funds, and most hedge funds, that have a "directional" view and invest hoping the market will then realize it. The other category contains market makers, brokers, proprietary trading shops and high frequency traders that want to make a profit by remaining market neutral. Banks can be considered as exhaustive market players in the sense that their internal structure includes the majority of the other typical players. Indeed, banks can have internal market makers, proprietary traders (outside of the US), and, most of them, also an asset management arm. On top of these functions, many banks can also count a retail arm, *i.e.,* provide several services such as savings accounts to private individuals. More details on banks and the corporate derivatives business can be found in Appendix C.1.1.

Directional The amount of assets managed by funds and private investors is currently more than 103 trillion USD, a quantity comparable to the global GDP.[11] There are several different types of directional players, perhaps the most renowned are asset managers, others include Private Equity (PE) firms, Venture Capital (VC) firms, and Hedge Funds (HFs).

PEs, VCs and HFs are also referred to as alternative investments and are sought for by investors who prefer a more aggressive type of return profile. These types of funds are accessible only to accredited investors, as they use riskier investment tactics than asset managers thanks to a more relaxed regulation, aiming at potential superior returns. In general, these funds ask for a fee on the Assets Under Management (AUM), and, in some cases (mostly PE and HFs) also a percentage (10% to 20%) on the capital gains.

Non-directional Non-directional market players are a structural type of market player, in the sense that they help the markets to function correctly. Among non-directional players, we focus on the role of market makers, who have the role of providing liquidity, and thus are continuously quoting both bid and ask prices, making a profit on the bid-ask spread. Specifically, their objective is to make a margin by buying at the bid and selling at the ask price, while mitigating the market risk associated to holding a large inventory of the traded instrument. Their business model is focused on the maximization of transaction volume. Some market makers may keep a non-flat inventory if they have a strong directional view, thus risking an adverse market movement that may easily wipe out the gains obtained.

[10] https://www.bbc.com/news/technology-56357526.

[11] https://www.bcg.com/publications/2021/global-asset-management-industry-report.

Therefore, in general, it is crucial to keep a zero inventory or hedge the risks so to avoid being subject to the market movements.

Market makers exist for most financial instruments and may behave differently depending on the type of financial instrument they are working with. In this dissertation we focus on market makers of equity options, credit options, and bonds. Generally, market makers are found in banks, given their role as financial intermediaries, but are now present also in proprietary trading firms that have found the possibility of profiting in this space, examples of such firms are Optiver, IMC, Susquehanna, DRW, and ADG. These firms have joined since the 1930s, contributing in tightening the competition. Such fierce competition has forced different players to improve the speed of access to the markets by means of sophisticated colocated servers and state-of-the-art fiber connections. The use of high speed infrastructure has also given these firms the title of high frequency traders.

2.3.1 Asset Managers

Asset management firms, such as Blackrock, Vanguard Group, Fidelity Investments, State Street Group and many others, are firms that allocate the savings of institutions and private individuals in the financial markets. Their objective is managing the funds of a client by optimizing the trade-off between risk and returns depending on risk aversion of the client. The biggest clients of asset management firms are usually pension funds, insurance companies, endowments and high net worth individuals. Asset management is typically a very prudent type of investment, with a time horizon of 5 to 10 years, and makes use of liquid assets.

Conventionally, the investment process can be divided in two steps, namely asset selection, *i.e.,* choosing of the most promising subset of available financial assets to invest in, and asset allocation, *i.e.,* deciding how much to invest in each asset (Grinold and Kahn, 2000). Asset allocation is traditionally decided based on Modern Portfolio Theory (MPT), which is a stream of literature concentrated on balancing risk and returns, started by Markowitz (1952); Sharpe (1963), it has now evolved to techniques such as Equal Risk Contributions (Roncalli, 2013) and Maximum Diversification (Choueifaty and Coignard, 2008). There are two main assumptions underlying MPT: the first is that we are considering a single-period portfolio, *i.e.,* the allocation is kept constant for the investment horizon, and the second is that assets behave stochastically according to a certain drift, volatility, and correlation structure. Merton (1969) was the first to consider a multi-period portfolio, though he maintained the backward looking assumption on the dynamics of the instruments. In Chapter 5, we explore a third approach, which considers a multi-period portfolio and assumes the market is an adversary, without giving any dynamic characterization to the assets. In practice, an asset manager can achieve two different objectives, *i.e.,* an *absolute return* objective maximizing the total amount of wealth, and a *benchmark* objective, that is beating a specified market index. In Chapter 5.2, we cover how it is possible to obtain an algorithm that gives theoretical guarantees of performing no worse than the pre-defined benchmark strategy with a high probability. We now formally define the task of an asset manager for the scope of this dissertation.

Definition 2.5 (Asset Management)**.** *Given a set of $M \in \mathbb{N}$ different assets, multi-period portfolio optimization is a sequential decision process in which at each (discrete) round $t \in \{1, \dots, T\}$ over an investment horizon $T \in \mathbb{N}$, an investor decides the portfolio*

allocation, i.e., *the proportion of the total budget to invest in each of the assets, to maximize her wealth. The portfolio allocation is represented by the vector* $\mathbf{a}_t \in \Delta_{M-1}$*, where* Δ_{M-1} *is the* $(M-1)$*-simplex in* $\mathbb{R}^M$ *and each element* $a_{j,t}$ *of* $\mathbf{a}_t$ *is the proportion of the asset* j *contained in the portfolio at round* t*.*

The sequence $\mathbf{a}_{1:T} = (\mathbf{a}_1, \ldots, \mathbf{a}_T)$ represents the investment strategy over T rounds with $\mathbf{a}_t = (a_{1,t}, \ldots, a_{M,t})$.[12] As common in the OPO framework Li and Hoi (2014), let us define the price relatives, $\mathbf{y}_t = (y_{1,t}, \ldots, y_{M,t})$ as $y_{j,t} = \frac{P_{j,t+1}}{P_{j,t}}$, where $P_{j,t}$ is the price of asset j at round t, and the price relatives sequence as $\mathbf{y}_{1:T} = (\mathbf{y}_1, \ldots, \mathbf{y}_T)$. The percentage profit in one time-step obtained by the asset manager can be defined as:

$$\rho_{t+1} = \langle \mathbf{a}_t, \mathbf{y}_t \rangle, \tag{2.13}$$

where $\langle \cdot, \cdot \rangle$ is the dot product. The cumulative wealth $W_T(\mathbf{a}_{1:T}, \mathbf{y}_{1:T})$ at round T, for an investment strategy $\mathbf{a}_{1:T}$ and a sequence of price relatives $\mathbf{y}_{1:T}$, is defined as:

$$W_T(\mathbf{a}_{1:T}, \mathbf{y}_{1:T}) = \prod_{t=1}^{T} \langle \mathbf{a}_t, \mathbf{y}_t \rangle, \tag{2.14}$$

The objective of any asset manager is to maximize wealth, keeping a low risk. The problem with multi-period portfolio optimization is that transaction costs can become quite relevant and penalize the overall performance. Adjusting the previous expression by adding costs gives:

$$\tilde{W}_T(\mathbf{a}_{1:T}, \mathbf{y}_{1:T}) = \prod_{t=1}^{T} \langle \mathbf{a}_t, \mathbf{y}_t \alpha_t \rangle, \tag{2.15}$$

where α_t is implicitly determined by the solution of the following equation (known in finance as turnover):

$$\alpha_t = 1 - \gamma ||\mathbf{a}'_{t-1} - \mathbf{a}_t \alpha_t||_1, \tag{2.16}$$

where α_t is the proportion of residual wealth after the transaction fees, γ is the transaction rate, which is equal for buying and selling and fixed throughout the investment horizon, $\mathbf{a}_t$ is the new portfolio and $\mathbf{a}'_{t-1} = \frac{\mathbf{a}_{t-1} \otimes \mathbf{y}_{t-1}}{\langle \mathbf{a}_{t-1}, \mathbf{y}_{t-1} \rangle}$ is the portfolio composition before it is updated but after the market movement $\mathbf{y}_{t-1}$.[13] It is important to notice that in the portfolio optimization problem, we are assuming that we are re-investing the gains (and losses), which is also why the costs in Equation 2.16 are implicitly defined. In Section 5.1 we portray how to account for these costs.

2.3.2 Quantitative Hedge Funds

Hedge funds invest through a variety of strategies, including those analyzing macroeconomic trends and invest according to the movements of interest rates, FX rates, the statements of central bankers, etc., and those that are event driven and speculate on companies that are in financial distress. Other hedge fund examples, more relevant from our

[12] The time duration of a step is discretionary and may be defined as a few hours, days, weeks, or months. Commonly, a daily discretization is considered when using OPO techniques.

[13] With $\mathbf{a} \otimes \mathbf{b}$ we denote the element-wise product between the two vectors $\mathbf{a}$ and $\mathbf{b}$.

perspective, are systematic investors, who base their strategies on specific rules that determine market entry and exit points. The more technologically advanced systematic investors are also referred to as quantitative investors, as they create strategies using mathematical and statistical modeling. Renaissance Technologies, created by the mathematician Jim Simons, is one of the most famous and successful quantitative hedge funds in the industry. In Chapter 6, we delineate how, using RL, it is possible to find a profitable intra-day quantitative trading strategy. Below is the formal definition of the task of such a hedge fund manager, as considered in this dissertation.

Definition 2.6 (Trading)**.** *Given an asset to trade, trading can be defined as a sequential decision process in which at each (discrete) round $t \in \{1, \dots, T\}$ over a trading horizon $T \in \mathbb{N}$, a trader decides whether to go long, short or stay flat with respect to the asset to maximize her wealth. The trader's position is represented by the action $a_t \in \{-1, 0, 1\}$.*

The profit in one time-step of the trader can be defined as:

$$r_{t+1} = \underbrace{a_t(P_{t+1} - P_t)}_{\text{market movement}} - \underbrace{\frac{\text{bid-ask}}{2}|a_t - a_{t-1}|}_{\text{transaction costs}}, \tag{2.17}$$

where a is the action which represents also the current portfolio, P_t is the price of the asset at time t and bid-ask is the bid-ask spread used to calculate the transaction costs. The first part of the reward consists in the gain (loss) derived from the current portfolio and the market movements, while the second one corresponds to the transaction costs originating when changing allocation. The expression of the transaction costs is simplified with respect to Equation 2.16, as we are not assuming the investment of the gains or losses, but we are keeping a constant action size. In this formulation we are not considering the market impact of a trade, *i.e.,* the trade does not cause the price of the asset to move in an adverse manner, but only proportional transaction costs.

While allocating a fixed amount of money could seem to be a limiting assumption, this is indeed sufficient when our goal is to maximize the expected return. To be more specific, if we believe the price of some asset is about to increase, then there is no reason to buy only a fraction of the asset. Thus, to conclude, the role of the hedge fund manager is to choose at each time-step which portfolio to hold, to maximize his cumulated wealth.

2.3.3 Options Market Makers

As we saw in Section 2.2.3 and Section 2.2.4, options are subject to several risk drivers, the most relevant is the underlying risk or delta. In the following paragraphs, we introduce firstly the equity options market maker and then the credit options market maker. The main distinction between the two is that, in the equity case, we are considering listed options whose underlying is also traded on an exchange, in the credit case, instead, we are in a dealer market scenario.

Equity Options Market Makers A single market maker often prices all the options present on a single underlying. To appropriately manage all these options (that have different strikes and maturities), she may be aided by an automatic software capable of hedging the delta risk, allowing the trader to concentrate on the other risks. The software

hedges the delta at each new operation and every so often or with the input of the trader, it hedges the delta generated by the entire portfolio of options. This can create quite relevant transaction costs, depending also on the liquidity of the underlying instrument. There are three main methods of optimizing the costs, either by optimizing the execution, trading reducing the portfolio turnover by means of smaller trades, or a combination of the two. Optimal execution is a general requirement when trading, we cover this topic in Section 2.4.3 and in Chapter 9. The reduction of portfolio turnover in hedging is tackled from a RL point of view in Chapter 8.

To formally define the role of an equity options market maker, let us recall the notation of Section 2.2.3.

Definition 2.7 (Option Hedging)**.** *Given an option with the respective underlying instrument, hedging the delta risk of the option can be defined as a sequential decision process in which at each (discrete) round $t \in \{1, \dots, T\}$ over the life of the option $T \in \mathbb{N}$, a trader decides how much to hold of the underlying instrument to minimize the price swings caused by the option. The hedge can be represented by the action $a_t \in [0, 1]$.*

A trader who has a long position in a call option will endure, for a time-lag of k, a P&L swing of $C_{t+k} - C_t$. A delta hedge is a strategy to limit this profit movement by buying or selling a certain quantity of the underlying, call this function $a(\frac{\partial C(t,S_t)}{\partial S}, \mathcal{E})$, which depends on the delta and the trader's experience $\mathcal{E}$; we refer to it as just a_t for ease of notation and because it represents the trader's action. Thus, the profit in one time-step of a trader who bought a call option and is hedging the delta risk can be defined as:

$$r_{t+1} = \underbrace{C_{t+1} - C_t}_{\text{option variation}} - \underbrace{a_t \cdot (S_{t+1} - S_t)}_{\text{market movement}} - \underbrace{c(a_t - a_{t-1})}_{\text{transact. costs}}. \tag{2.18}$$

where $c(a_t - a_{t-1})$ are the transaction costs caused by trading the underlying hedging instrument, which we define in Equation 2.23. Now, assume that we replicate the delta exactly, so $a_t = \frac{\partial C(t,S_t)}{\partial S}$ and there is no experience into play, then the B&S model assures a zero profit in continuous time and in the absence of transaction costs ($c(a_t - a_{t-1}) = 0$). With $k > 0$ and costs > 0, $a_t = \frac{\partial C(t,S_t)}{\partial S}$ ceases to be the optimal solution. It is important to notice that this information ($S_t, C_t, \frac{\partial C(t,S_t)}{\partial S}$) is always available from the market or through simple calculations deriving from market information. Several approaches have been proposed to extend the B&S model to account for transaction costs, starting with Leland (1985) and more recently Guéant and Pu (2017) which uses stochastic optimal control. In Chapter 8, we illustrate how the use of RL powers an efficent solution.

Credit Options Market Makers In the credit index options case, the scenario is fairly similar to what was described above and we can still refer to Definition 2.7. Using the same notation of Section 2.2.4, the profit in one time-step of a trader long a payer option and holding a_t in the hedging portfolio is:

$$r_{t+1} = \underbrace{\text{Pay}_{t+1} - \text{Pay}_t}_{\text{option variation}} - \underbrace{a_t \cdot (\text{Upf}_{t+1} - \text{Upf}_t)}_{\text{market variation}} - \underbrace{c(a_t - a_{t-1})}_{\text{transac. costs}}. \tag{2.19}$$

The trading costs can be defined as in Equation (2.24). Similarly to before, the B&S model assures a zero profit when we are in continuous time, $a_t = N_h(t)$, $N_h(t)$ defined in

Equation (2.12) and there are no transaction costs ($c(a_t - a_{t-1})) = 0$). In Section 8.4 we illustrate how the use of RL can be used to improve the solution.

2.3.4 Bond Dealers

Bond dealers are also referred to as market makers. A single dealer is usually continuously displaying both bid and ask quotes on a number of bonds, often specializing in a specific typology such as European government, high yield, investment grade, etc. In the MD2C platforms considered in this work, the clients see the indicative quotes of the dealers, and send a RFQ requesting the firm quotes to multiple dealers. This process creates a direct competition among the dealers, who want to satisfy the clients requests while keeping the inventory as small as possible. The clients' requests are answered by dealers, who show a firm price the client can execute.

As we have seen in Section 2.1.2, the high reserve requirements for banks imply keeping the inventory as small as possible. These reforms have impacted the market making process, by increasing the penalty of holding an inventory, which thus has become less profitable for the dealers. This lower profitability has forced some banks to exit the sector and others to reduce costs by automating the process of quoting prices. The lower liquidity of certain assets also paved the way for other financial entities, such as proprietary trading shops (e.g., XTX Markets), to enter the market making space using proprietary quoting algorithms.

A common market-making strategy that can be executed easily by a human operator requires looking at the average market spread and offering a lower ask in case of a positive inventory, and, vice versa, offering a higher bid in case of a negative inventory. Conversely, more advanced strategies use a hard coded pricing strategy for RFQs of small sizes, and direct to human traders the most relevant orders. We define the role of the market maker as considered in this dissertation.

Definition 2.8 (Market Making)**.** *Considering a specific bond, market making can be defined as a sequential decision process in which at each (discrete) round $t \in \{1, \dots, T\}$ the dealer updates her bid and ask prices $P_{t,buy}, P_{t,sell}$, to maximize P&L while minimizing inventory.*

The profit of a dealer can be divided in three parts: the gain in having bought (resp. sold) lower (resp. higher) than the mid (see Equation (2.25)), called spread P&L; the profit/loss given by the change in the market price, defined as inventory P&L and a penalty of holding inventory. Formally, the reward for each market maker is defined following Guéant and Manziuk (2019):

$$r_{t+1} = \underbrace{v_t(P_{t,h} - P_t)}_{\text{spread P\&L}} + \underbrace{z_{t-1}(P_t - P_{t-1})}_{\text{inventory P\&L}} - \underbrace{\phi(z_t)}_{\text{inventory penalty}}, \tag{2.20}$$

where $\phi : \mathbb{R} \to \mathbb{R}^+$ is a function encoding the penalty of owning a net inventory, $P_{t,h}$ is the price of the market maker with $h \in \{\text{buy}, \text{sell}\}$, P_t is the mid price, v_t is the size of the trade and z_t is the inventory. The penalty approximates the cost of capital requirements mentioned in Section 2.1.2 plus the risk-aversion of the dealer who wants to avoid market swings. The P&L is similar to Equation (2.20), but without the inventory penalization term, so:

$$\text{P\&L}_{t+1} = v_t(P_{t,h} - P_t) + z_{t-1}(P_t - P_{t-1}). \tag{2.21}$$

To summarize, the role of the dealer, is to price the "optimal" firm bid and ask prices, based on the current market information and his internal inventory, to optimize the reward function Equation (2.20). In Chapter 7, we describe how to use RL and Mean Field Games to optimally price and answer the RFQs.

2.4 Algorithmic Trading

Algorithmic trading is defined by Article 4(1)(39) of MiFID II (see Section 2.1.2) as "trading in financial instruments where a computer algorithm automatically determines individual parameters of orders such as whether to initiate the order, the timing, price or quantity of the order or how to manage the order after its submission, with limited or no human intervention [...]". In this section, we initially define trading/transaction costs, provide further details on the functioning of LOBs and then introduce the optimal execution problem.

2.4.1 Transaction Costs and Market Impact

Transaction costs are a fundamental concept, and can degrade the performance of a trading, portfolio optimization, or hedging strategy, in terms of both profits and risk, especially as the trading frequency and / or the sizes of the trades increase. The most common definition of transaction costs in practice is half of the bid-ask spread. This type of proportional transaction costs (in contrast with fixed costs) has been present in literature for a long time (Loeb, 1983; Bertsimas and Lo, 1998; Soner and Touzi, 2013). Specifically, given a trade of size n, we consider transaction costs as:

$$c(n) = \frac{\text{bid-ask}}{2} \cdot |n|, \tag{2.22}$$

this means we are assuming that all of the trade can be absorbed by the best bid or the best ask. We refer to $\frac{\text{bid-ask}}{2}$ also as mid-ask or mid-bid spread. If the best bid or best ask cannot absorb the entire trade, then it is necessary to consider market impact. To approximately model a market impact, it is possible to consider the formulation as in Kolm and Ritter (2019):

$$c(n) = \frac{\text{bid-ask}}{2} \cdot (|n| + 0.01n^2), \tag{2.23}$$

There are more sophisticated methods of modelling market impact as illustrated in Section 2.4.3, and through the use of agent based simulators such as ABIDES defined in Section 4.3.2.

Furthermore, as seen in Equation (2.16), when considering a *self-financing* portfolio, *i.e.,* a portfolio in which there is no exogenous infusion or withdrawal of money, the proportion of the remaining performance is an implicit equation defined as:

$$\alpha_t = 1 - \gamma ||\mathbf{a}'_{t-1} - \mathbf{a}_t \alpha_t||_1,$$

and, thus, the transaction costs are $(1 - \alpha_t)\langle \mathbf{a}_t, \mathbf{y}_t \rangle$.

Transaction costs play an important role throughout the whole dissertation, and, in particular, the aim is to minimize them. Given a trading action, there are two main ways of minimizing transaction costs, either by optimizing the execution through optimal execution

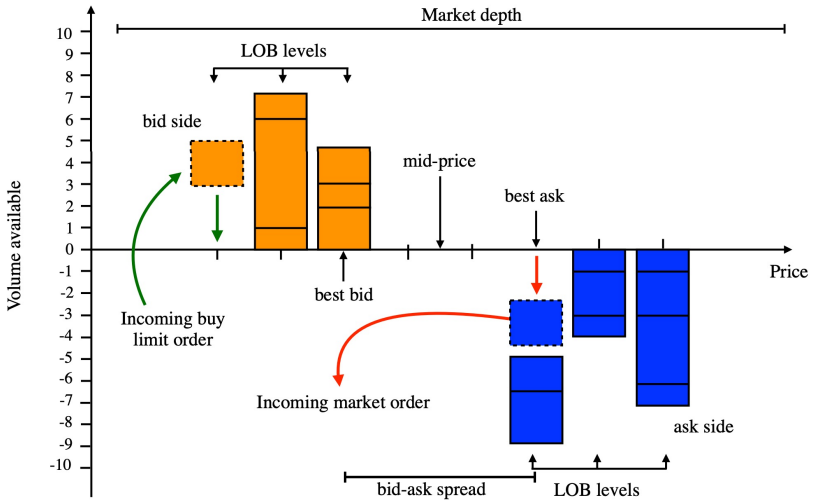

Figure 2.3: *Graphical representation of LOB (Briola et al., 2021).*

algorithms and smart routing procedures, or by reducing the portfolio turnover by means of smaller trades. These two approaches can also be adopted simultaneously. We analyze the former approach in Chapter 9 and the latter in Sections 5.1, 8.3 and 8.4.

Simulating market impact is a daunting task, especially on historical data. In this case, the only possible solution is to simulate a temporary impact that falls back to the historical time series. To tackle this problem, we decided to use ABIDES (Byrd et al., 2019), an advanced agent based market simulator, which models the interactions between market players and re-creates the order book. In Section 4.3.2 we explain the salient features of this simulator.

Trading Costs in OTC Instruments For OTC instruments, trading costs are defined in a more streamlined manner, as there is no real concept of market impact. To explain this further, we take as an example the CDS index defined in Section 2.2.4. The way in which applicable bid and ask quotes are built ensures that notionals up to hundreds of millions of the index can be traded without market impact. Thus, we can discard execution-related issues, assume that the trading costs can be computed from the bid-ask spreads (see Figure 4.2), and define the costs as:

$$c(\mathrm{N}) = \mathrm{N}\left|\mathrm{Upf}_t\left(S_t \pm \frac{\text{bid-ask}}{2}\right) - \mathrm{Upf}_t(S_t)\right|, \tag{2.24}$$

where the $+$ $(-)$ sign should be considered when buying (selling) protection, and N is the notional.

2.4.2 Limit Order Books

LOBs are the record of all the currently outstanding limit orders (see Definition 2.1). In Figure 2.3 we can see some of the mechanisms of LOBs, partly mentioned in Section 2.1.1.

In LOBs, price increments are discrete and the minimum increment is referred to as *tick size*. Each order is comprised of a price and a size. The combination of all the orders for a certain price is referred to as volume for which we use the letter Q.

An important concept in this field is that of *liquidity*. Liquidity can be defined as the total volume of all limit orders, the more limit orders are placed on the exchange and the more liquid the trading venue is. The concept of market impact is strongly related to that of liquidity, as, the more liquid a financial instrument is, the smaller the market impact. Cash is regarded as the most liquid instrument, as it can be immediately converted to any other financial instrument. On the opposite side, very illiquid financial instruments are for example real estate.

LOB Order Types When trading on a LOB, it is possible to employ different types of orders, listed below, that can be used to execute a trade:

- The *limit order* is the most adopted, and entails specifying the price at which we want to execute the trade. Such trade can execute only at this price or at a more advantageous one. There is no guarantee that the trade will happen, as the limit price may be never reached. These orders are executed considering a price-time priority, also known as First-In-First-Out (FIFO).
- The *market order* is the simplest typology, and consists in a request to carry out the order immediately at the best price available in the market. To be more specific, a market purchase (sell) order is matched with limit sell (buy) orders starting with the best ask price. This type of order is used throughout the dissertation.
- The *stop-loss* order is used to close a position if the market moves in an unfavorable direction. If the asset reaches the stop price, the order is transformed into a market order, thus, it may execute at a less favourable price than the stop price.
- Finally, there are other types of orders, such as fill-or-kill orders, which must be executed immediately on receipt or not at all. Or more complex orders, like icebergs, that break up a large order into several smaller orders.

LOB features In algorithmic trading, it is popular to use signals deriving from specific features, including the following:

- *Volume imbalance*: it describes the difference between the existing order volume on the bid and ask price levels. We consider this feature till the third level of the order book. It can be defined as:

$$v_{imb}^{k} = \frac{Q_{\text{ask}}^{k} - Q_{\text{bid}}^{k}}{Q_{\text{ask}}^{k} + Q_{\text{bid}}^{k}} \text{ for } k \in \{1, ..., 3\},$$

  where Q_{ask}^{k} (Q_{bid}^{k}) is the volume of outstanding limit orders at the k-th best ask (bid) price level.

- *Total depth*: it corresponds to the cumulative sum of the volume over multiple price levels. We consider this feature until the third level of the LOB:

$$TD_{h}^{k} = \sum_{j=1}^{k} Q_{h}^{j} \text{ for } k \in \{1, 2, 3\},\ h \in \{\text{bid, ask}\}.$$

- *Rolling volatility*: the standard deviation of a window of the returns of the asset.
- *Mid price*: it is the midpoint between the best bid and best ask prices:

$$P_{\text{mid}} = \frac{P_{\text{best ask}} + P_{\text{best bid}}}{2}. \tag{2.25}$$

- *Price imbalance*: it consists in the step-wise scaled imbalance between demand and supply. We consider this feature only for the first three levels of the LOB:

$$Z_{\text{imb}} = \frac{(P_j^{\text{ask}} - P_{j-1}^{\text{ask}}) - (P_j^{\text{bid}} - P_{j-1}^{\text{bid}})}{(P_j^{\text{ask}} - P_{j-1}^{\text{ask}}) + (P_j^{\text{bid}} - P_{j-1}^{\text{bid}})}. \text{ for } j \in \{2, 3\},$$

2.4.3 Optimal Execution

A single asset can be exchanged on multiple venues, both lit and dark. If we are considering a particularly large trade, to minimize market impact and, thus, the transaction fees incurred, it is convenient to divide up the order over multiple venues. In the lit venues, it is also possible to take further enhancements, always geared towards the objective of minimizing market impact, this is referred to as optimal execution.

Optimizing trade execution is one of the most important aspects when dealing with the financial markets. Once a specific trade has been decided, optimizing the execution means paying the least amount of trading costs possible. This is necessary when the order in consideration is greater than what is present in the first levels of the order book. To optimize the execution of a trade, direct access to the market is required. Thus, this task is usually performed by advanced market players such as banks or brokers that need to execute large client orders, as well as market makers. Indeed, the latter are continuously hedging the risk deriving from having an inventory, and minimizing the trading costs when hedging is fundamental to have a profitable strategy. Asset managers and proprietary traders are also very careful to optimize execution so that the performance is not depleted by the transaction costs.

Definition 2.9 (Optimal Execution)**.** *The optimal execution problem consists in executing a trade of X shares in a maximum amount of time T and number of time-steps $N+1$. It is a sequential decision process in which, at each discrete time-step $t_k = k\tau$ for $k \in \{0, ..., N\}$ where $\tau = \frac{T}{N}$, the trader decides the quantity to execute to minimize the difference between the arrival price and the actual execution price.*[14]

A trading trajectory is defined as a list $\{x_0, ..., x_N\}$, where x_k is the number of units held at time t_k. Clearly, $x_0 = X$ and liquidation at time T requires $x_N = 0$. Equivalently, it is possible to specify a "trade list" $\{n_1, ..., n_N\}$, where $n_k = x_{k-1} - x_k$ is the number of units sold between times t_{k-1} and t_k.

TWAP A baseline execution model, commonly used in practice, is the Time Weighted Average Price (TWAP) where a trade of size X is executed in sizes of $n_t = X/N$. The arithmetic average of prices collected yields the TWAP price:

$$\text{TWAP} = \frac{X}{N} \sum_{k=0}^{N} P_k, \tag{2.26}$$

[14] In literature there is often also a risk component to minimize (see Appendix B.2).

where P_k is the (average) at which each trade was executed through a market order sent at time t_k.

Another baseline present in literature is the Almgren and Chriss (2001), which is described in Appendix B.2. Optimal execution will be analyzed using RL in Chapter 9.

CHAPTER 3

Introduction to Reinforcement Learning

After having covered the main financial instruments and the market participants of interest, in this chapter we focus on introducing the algorithms or family of algorithms used in this work. This chapter helps the reader gain the fundamentals on RL, necessary to understand the following chapters.

The previous chapter highlighted that most market players take trading decisions in a sequential manner, basing their decisions on the currently available information. Once the market participants have executed the trade, they can observe how well decisions perform by monitoring the P&L. This sequential decision making can be modelled as a Markov Decision Process (MDP) (Puterman, 2014).

Chapter outline In this chapter, we firstly introduce RL and outline basic concepts and then briefly anticipate the algorithms that are at the core of the following chapters.

3.1 Markov Decision Process

Markov Decision Processes refers to the fact that the system obeys the Markov property, that is transitions only depend on the most recent state and action.

Definition 3.1 (Markov Decision Process)**.** *A discrete-time MDP is defined as a 6-tuple* $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu \rangle$*, where:*

- $\mathcal{S}$ *is a non-empty measurable space called state space;*

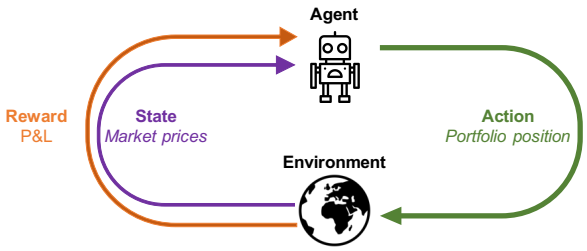

Figure 3.1: *Graphical representation of the interaction between the agent and the environment.*

- $\mathcal{A}$ *is a non-empty measurable space called action space;*
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to P(\mathcal{S} \times \mathbb{R})$ *is the transition model that assigns to each state-action pair* (s, a) *the probability measure* $\mathcal{P}(\cdot|s, a)$ *of the next state;*
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ *is a bounded reward model, which assigns for every triple* (s, a, s') *a probability measure* $\mathcal{R}(\cdot|s, a, s')$*;*
- $\gamma \in [0, 1)$ *is the discount factor, used to weight future rewards;*
- μ *is the initial state distribution, from which the starting state is sampled.*

State The state $s \in \mathcal{S}$ usually contains all the information the agent perceives from the environment. In the context of this dissertation, it encompasses market information such as a window of the latest prices of the financial instruments, and may also include also some internal agent information such as the current portfolio position. In all the environments considered, the state space is continuous.

Action The action $a \in \mathcal{A}$ represents how the agent interacts with the environment. In the considered applications, it usually represents the position to hold or the trade to make, and in the dealer scenario the action represents the price. In this dissertation, we consider both discrete and continuous action spaces.

Reward Function The reward function is the expected reward received when performing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ and arriving in state $s' \in \mathcal{S}$: $r(s, a, s') = \int_{\mathbb{R}} r\mathcal{R}(\mathrm{d}r|s, a, s')$. We can define the reward indipendently from the next state s' by computing an expectation over the next state $r(s, a) = \int_{\mathcal{S}} \mathcal{P}(ds'|s, a)r(s, a, s')$. In this work it mainly represents the P&L, in the dealers case also considering a penalty. In the optimal execution framework instead, it measures the distance between two prices.

Environment dynamics $\mathcal{P}(s'|s, a)$ describes the probability of reaching state s' given that we are in state s and take action a. The environment dynamics fulfill the Markov Property, which means that state transitions depend only on the most recent state and action and not on previous history. The environment dynamics, in this thesis, are given mainly by the asset prices.

MDP Interaction Considering Figure 3.1 and the previous definitions, it is possible to better understand the interaction between the agent and the environment. At time-step $t = 0$, the agent samples the initial state $s_0 \sim \mu$. Then, for each following decision step $t \in \mathbb{N}$, the agent selects an action $a_t \in \mathcal{A}$, from a policy $\pi(\cdot|s)$, that interacts with the environment. Given this action, the environment transitions to the next state $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$ and provides the agent with the reward $r_{t+1} \sim \mathcal{R}(\cdot|s_t, a_t, s_{t+1})$. In this dissertation, we restrict our applications to *finite horizon* MDPs a.k.a. *episodic* MDPs for which there is a terminal state from which no other states can be reached and all actions provide zero reward.

Remark 3.1. Focusing on the applications of interest in this thesis, in most cases the state can be divided into two parts $s = (s_m, s_i)$, where s_m are the market prices and s_i are the agent's internal information. We can then assume that the transition probability can be split into two independent parts $\mathcal{P}(s'|s, a) = \mathcal{P}(s'_i|s_i, a)\mathcal{P}(s'_m|s_m, a)$, where $\mathcal{P}(s'_i|s_i, a)$ is deterministic and $\mathcal{P}(s'_m|s_m, a)$ does not depend on the agent's action. Thus we can conclude that:

$$\mathcal{P}(s'|s, a) = \mathcal{P}(s'_m|s_m). \tag{3.1}$$

A similar reasoning can also be followed for the reward functions. We have already seen some potential reward formulations, for example Equations (2.17), (2.18), and (2.20). We can notice that the reward can be easily split into two independent parts, the first given by the market movement and the second given by the transaction costs:

$$r(s, a) = r(s_m, a) + r(s_i, a). \tag{3.2}$$

3.2 Value Functions and Bellman Equations

Provided that, in all the assessed applications in this dissertation, the objective of the agent is to reach a yearly (if not shorter) objective, we consider finite horizon problems in which future rewards are exponentially discounted with γ. Let us define a trajectory τ as a sequence of states, actions, and rewards:

$$\tau := (s_0, a_0, r_1, s_1, a_1, r_1, ..., s_T, a_T, r_{T+1}).$$

We then define the discounted sum of the rewards of a trajectory as the returns:

$$G_\tau = \sum_{t=0}^{T} \gamma^t r_{t+1}. \tag{3.3}$$

Each trajectory is generated by following a *policy*: $\pi(\cdot|s) : \mathcal{S} \to \mathcal{A}$, a mathematical formalization of the strategy the agent plays to select the action at each time-step. Given that both the policy and the transition probability may be stochastic, we are interested in the expected value of the return given all the possible trajectories, also known as the value function.

Definition 3.2 (State Value Function or V-Function)**.** *Let $\mathcal{M}$ be an MDP and π a policy. For every state $s \in \mathcal{S}$, the state value fuction $V_\pi : \mathcal{S} \to \mathbb{R}$ is defined as the expected return starting from state s and following policy π*

$$V_\pi(s) := \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_{t+1} \Big| s_0 = s \right], \tag{3.4}$$

which can be recursively determined by the following Bellman equation (Bellman, 1966):

$$V_\pi(s) = \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\left[r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim\mathcal{P}(\cdot|s,a)}\left[V_\pi(s')\right]\right].$$

Similarly, if we consider starting from a specific state and taking a specific action, we can define the state-action value function.

Definition 3.3 (State-Action Value Function or Q-Function)**.** *Let $\mathcal{M}$ be an MDP and π a policy. For every state-action pair $(s,a) \in \mathcal{S}\times\mathcal{A}$, the state-action value fuction $Q^\pi : \mathcal{S}\times\mathcal{A}\to\mathbb{R}$ is defined as the expected return starting from state s, playing action a and following policy π*

$$Q_\pi(s,a) \coloneqq \mathbb{E}_\pi\left[\sum_{t=0}^{T}\gamma^t r_{t+1} | s_0 = s, a_0 = a\right], \tag{3.5}$$

which can be recursively defined by the following Bellman equation (Bellman, 1966):

$$Q_\pi(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{\substack{s'\sim\mathcal{P}(\cdot|s,a)\\ a'\sim\pi(\cdot|s')}}\left[Q_\pi(s',a')\right]. \tag{3.6}$$

Finally, the objective we are interested in achieving in *risk neutral* RL is the maximization of the value function, given an initial state distribution.

Definition 3.4 (Objective Function or J-Function)**.** *Let $\mathcal{M}$ be an MDP and π a policy. Given the initial state distribution μ the objective function $J^\pi : \mathcal{S}\to\mathbb{R}$ we want to maximize, is defined as the expected return starting from state s_0 and following policy π:*

$$\begin{aligned} J_\pi &\coloneqq (1-\gamma)\mathop{\mathbb{E}}_{s_0\sim\mu}\left[\sum_{t=0}^{T}\gamma^t r_t\right], \\ &\equiv (1-\gamma)\int_{\mathcal{S}}\mu(s)V_\pi(s)ds. \end{aligned} \tag{3.7}$$

It may be useful to express J_π in dual form as:

$$\begin{aligned} J_\pi &\coloneqq \int_{\mathcal{S}} d_{\mu,\pi}(s)\int_{\mathcal{A}}\pi(a|s)r(s,a)\,\mathrm{d}a\,\mathrm{d}s, \\ &\equiv \mathop{\mathbb{E}}_{\substack{s\sim d_{\mu,\pi}\\ a\sim\pi(\cdot|s)}}\left[r(s,a)\right]. \end{aligned}$$

Where $d_{\mu,\pi}$ is the (discounted) state-occupancy measure induced by π:

$$d_{\mu,\pi}(s) \coloneqq (1-\gamma)\int_S \mu(s_0)\sum_{t=0}^{T}\gamma^t p_\pi(s_0 \xrightarrow{t} s)\,\mathrm{d}s_0,$$

where $p_\pi(s_0 \xrightarrow{t} s)$ is the probability of reaching state s in t steps starting from s_0 following policy π.

To conclude this section, we would like to focus again on the applications object of this dissertation, with the purpose of showing that if we do not include transaction costs in the reward formulation, maximizing the Q-function is equivalent to maximizing the immediate reward. Specifically, following ideas from factor MDPs (Koller and Parr, 1999), and recalling Equations (3.1) and (3.2), we can consider our MDP as divided into two independent parts.

Lemma 3.1. *In the applications considered in this dissertation, let $s = (s_m, s_i)$, where s_m are the market prices and s_i are the agent's internal information, assuming $\mathcal{P}(s'_i|s_i, a)$ is deterministic, $\mathcal{P}(s'_m|s_m, a) = \mathcal{P}(s'_m|s_m)$ i.e., it does not depend on the agent's actions and $r(s, a) = r(s_m, a) + r(s_i, a)$ then:*

$$Q_\pi(s, a) = r(s_m, a) + r(s_i, a) + \gamma \mathop{\mathbb{E}}_{\substack{s'_m \sim \mathcal{P}(\cdot|s_m) \\ a' \sim \pi(\cdot|s')}} \big[Q_\pi(s', a')\big].$$

Futhermore, assuming no transaction costs, then we can exclude s_i from the state, thus $s = s_m$, $r(s_i, a) = 0$ and we obtain:

$$\arg\max_a Q(s, a) = \arg\max_a r(s_m, a).$$

3.3 Learning the MDP

To solve the MDPs, we resort to several approaches, which can broadly be referred to as RL, even though it is more precise to distinguish between RL, online planning, and online learning. In this section, we outline a preliminary description of four main classes of algorithms considered in this work: value based and policy search algorithms belonging to the RL category, MCTS as an online planning algorithm and online learning algorithms including expert learning and multi-armed bandits.

3.3.1 Reinforcement Learning

Value based and policy search algorithms can be generally classified as *Batch RL*, where there is a training procedure based on a database of offline data. As RL algorithms are model-free, the data can be generated by any type of model, or it is real data collected through experience. After the training procedure, which can take from hours, to days or weeks, depending also on the available computational power, a general policy is learnt. This policy can then be used in real-time in a production environment to take the investment decisions. The drawback is that this policy is stationary, so if the market changes behavior drastically, compared to what is present in the training dataset, it is necessary to stop the algorithm (ideally with an automatic procedure) and re-train the policy with the new data.

Value based algorithms

In RL, the objective is to learn the optimal policy by direct interaction with the environment. There are several RL algorithms, we dwell on two of the main categories, namely value based and policy search algorithms. Value based algorithms focus on learning the state-action value function (Equation 3.5) for each state and action, by using the Bellman

equation (Equation 3.6) in an iterative manner. The most known algorithm of this kind is Q-learning (Watkins, 1989), that uses the following update equation:

$$Q_t(s,a) = (1-\alpha)Q_t(s,a) + \alpha\left(r(s,a) + \gamma \max_{a'} Q_t(s',a')\right). \tag{3.8}$$

The main drawback of this approach is that it does not scale well for large state and action spaces, so it is not ideal for the environments considered in this thesis, *i.e.* continuous state MDPs. To mitigate this problem, there are methods that discretize the state and action spaces, then fit a regressor to interpolate. Examples of such algorithms are Fitted Q Iteration (FQI) (Ernst et al., 2005), defined in Section 6.2.1 and Deep Q Learning (DQN) (Mnih et al., 2013). Typical regressors are extra trees (see Appendix B.4.2) or Neural Networks (NNs) (see Appendix B.4.3). We discuss in Chapters 6, 7 and 9 the use of FQI in the quant trading, market making and optimal execution scenarios, respectively.

Policy Search algorithms

Policy search algorithms embrace a different approach, and instead of learning the value function, they learn directly the optimal policy. The policy is parametric (usually a NN, see Appendix B.4.3) and by moving through the policy space, usually through gradient descent, it is possible to find the policy that maximizes the objective (Equation 3.7). Basic policy search algorithms are REINFORCE (Williams, 1992), GPOMDP (Baxter and Bartlett, 2001), and are based on the policy gradient theorem:

$$\nabla J_\pi = \mathop{\mathbb{E}}_{\substack{s\sim d_{\mu,\pi} \\ a\sim\pi_{\boldsymbol{\theta}}(\cdot|s)}} \left[\nabla \log \pi_{\boldsymbol{\theta}}(a|s) Q_{\pi_\theta}(s,a)\right],$$

where π_θ is the parametrized policy.

There exist also more sophisticated algorithms, which have achieved promising experimental results, such as TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017). In Chapter 8, we assess the use of a risk aversion version of TRPO, namely TRVO (Bisi et al., 2020b), to study risk management in hedging problems.

3.3.2 Online Planning

Online planning, which includes MCTS (Kocsis and Szepesvári, 2006) and Dynamic Programming (Efroni et al., 2020), is an online, model-based type of algorithm. Hence, there is no offline training procedure because the algorithm learns a local policy as new information is received, by planning with a *generative model* of the environment. Strong links between learning and planning methods have been observed and analyzed by researchers in recent years as paradigms solve similar problems. The key difference is the source of experience: whether it comes from real interaction with the environment (in learning) or from a generative model (in planning) (Vodopivec et al., 2017).

MCTS algorithms have achieved great results in games such as AlphaGo (Silver et al., 2016), where the state space is extremely large and it is challenging to learn a general policy with policy search or value based approaches.

Monte Carlo Tree Search

MCTS consists in planning interleaved with acting. Planning is performed using a generative model of the future, which can be for example "traditional" stochastic differential equations (see Section 4.3.1). This local planning procedure, depending on the time necessary (thus computational power and budget), can cause a delay between the reception of new information and the decision, ranging from a few seconds to a few minutes. The positive aspect is that, if the market changes behavior drastically, it is only necessary to update the generative model (it can be done online) and the algorithm can continue working in any scenario, thus making MCTS more adaptable in non-stationary scenarios. The original MCTS algorithm is Upper Confidence Tree (UCT) (Kocsis and Szepesvári, 2006).

The MCTS family of algorithms combines tree search algorithms with Monte Carlo sampling to iteratively build a search tree of possible future scenarios, which is in turn used to build an estimator of the optimal value of each action in the current state of the environment. These algorithms are characterized by 4 phases:

1. **Selection:** Starting from the root of the planning tree, a *tree policy* is iteratively applied until an unexpanded (a node with unvisited children) node is reached.

2. **Expansion:** One or more the successors of the reached node are added to the tree. A common best practice is to add only the first newly visited node.

3. **Simulation:** A Monte Carlo simulation (*rollout*) is started from the expanded node to provide an initial estimate of the nodes' value.

4. **Backpropagation:** The values of the states visited during the tree traversal and the simulation are backpropagated up the tree until the root, updating the relevant statistics.

UCT applies as tree policy the well-known Multi-Armed Bandit (MAB) algorithm, Upper Confidence Bound (UCB1) (Auer et al., 2002), described in Equation (3.10).

UCT is suitable for deterministic states and discrete actions, and thus, to be compatible with the trading environment, it is necessary to extend it to stochastic states. In Section 6.3, we examine how UCT was modified to be compatible with the trading problem.

3.3.3 Online Learning

The online learning category, just like online planning, has no training phase. The main difference with the previously described categories lies in the fact that there is no transition probability from one state to the next, in fact we are always in the same state and we only want to optimize for the next action. Online learning algorithms usually have low computational complexity, thus the investment choice is in quasi real-time.

In online learning, an agent (also referred to as learner) has to guess the outcome $y_t \in \mathcal{Y}$ based on the past sequence $y_1, y_2, ..., y_{t-1}$ of events that occurred in the outcome space $\mathcal{Y}$, at each time-step the agent will play a_t, that is an element of the prediction space (also known as decision space) $\mathcal{A}$ and the environment will choose a loss function $f(a_t, y_t)$ by determining the outcome y_t. Formally, online learning can be defined as:

Algorithm 1: Online Learning

Initialize: decision space $\mathcal{A}$, outcome space $\mathcal{Y}$, loss function $f : \mathcal{A} \times \mathcal{Y} \to \mathbb{R}$
set $L_0 = 0$ **for** $t \in \mathbb{R}^N$ **do**
  The agent chooses an element of the decision space $a_t \in \mathcal{A}$
  The environment chooses $y_t \in \mathcal{Y}$ and then determines the loss function $f(\cdot, y_t)$
  The agent incurs in a loss $f(a_t, y_t)$
  The agent updates its cumulative losses $L_t = L_{t-1} + f(a_t, y_t)$

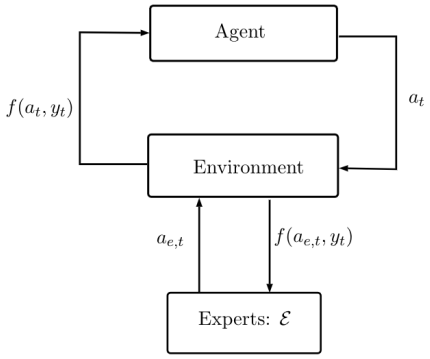

Figure 3.2: *Online learning with Expert Advice as an Agent-Environment interaction.*

Definition 3.5 (Online Learning)**.** *Let $\mathcal{Y}$ be the outcome space, $\mathcal{A}$ the prediction space, and $f : \mathcal{A} \times \mathcal{Y} \to \mathbb{R}$ is a loss function, an online game is the sequential game played by the forecaster and the environment, described in Algorithm 1.*

The following sections describe two specific characterizations of online learning, namely, online learning with expert advice, and multi-armed bandits.

Online Learning with Expert Advice

In online learning with expert advice, a.k.a expert learning (Cesa-Bianchi and Lugosi, 2006), we refer to the environment as *adversarial* since no stochastic characterization is given to the outcome sequence y_t and the analysis of the regret is conducted assuming a worst case scenario. Since the adversary knows the prediction a_t before deciding the outcome y_t, the design of an algorithm that tries to minimize the loss is a hopeless task and so we have to set an easier objective. In fact, the adversary can always choose the outcome y_t that maximizes the loss $f(a_t, y_t)$ regardless of the decision $a_t \in \mathcal{A}$ taken by the learner. Let us now consider, on top of the agent also a set $\mathcal{E}$ of other, expert, players. At each step in the prediction game, every expert $e \in \mathcal{E}$ predicts an element $a_{e,t} \in \mathcal{A}$ and incurs in a loss $f(a_{e,t}, y_t)$ (see Figure 3.2). The goal of the learner is to obtain small losses with respect to the best expert in the class $\mathcal{E}$. This concept can be expressed by using regret. Formally, the

regret $R_{e,T}$ for the agent with respect to the expert $e \in \mathcal{E}$ is defined as:

$$R_{e,T} = \sum_{t=1}^{T} \left[f_t(a_t, y_t) - f_t(a_{e,t}, y_t) \right],$$

The regret observed by the agent with respect to the entire class of experts $\mathcal{E}$ is defined as:

$$R_T = \sup_{e \in \mathcal{E}} R_{e,T} = \sum_{t=1}^{T} f_t(a_t, y_t) - \inf_{e \in \mathcal{E}} \sum_{t=1}^{T} f_t(a_{e,t}, y_t). \tag{3.9}$$

The task of the agent is to find a sequence a_t to obtain small regret R_T with respect to any sequence of outcomes $y_1, y_2, ..., y_T$ chosen by the environment. Specifically, we aim at achieving sublinear regret $R_t = o(T)$, meaning that the per-round regret R_T/T will asymptotically vanish:

$$R_T = o(T) \implies \lim_{T \to \infty} \frac{R_T}{T} = 0.$$

In the case of uncountable experts, expert learning also "coincides" with Online Convex Optimization (OCO) (Zinkevich, 2003). As we will see in Chapter 5, expert learning and OCO can be used to describe the OPO framework.

Multi-Armed Bandits

The Multi Armed Bandit (MAB) framework is also an online learning framework, it differs from expert learning as we do not have full feedback, *i.e.,* we can only calculate the reward (in the MAB setting it is common to refer to rewards instead of losses) for the action taken by agent. The actions are also referred to as arms. In MAB, if we want to have information about an arm, we need to try it, this makes the problem intrinsically more difficult than the full feedback one, namely, expert learning. Regret is defined as for the expert learning case:

$$R_T = \sum_{t=1}^{T} \left[f_t(a_t, y_t) - f_t(a^*, y_t) \right],$$

with the difference that we are considering an oracle a^* as there is no concept of "experts". To keep the regret limited, it is important to balance between exploration and exploitation. One of the most well known MAB algorithm is (UCB1) (Auer et al., 2002). Given a finite number of arms K, at iteration n, UCB1 chooses the action that maximizes a high probability upper bound of the value of the actions according to:

$$a_n = \arg\max_{i=1..K} \overline{X}_i + C\sqrt{\frac{2 \log n}{T_i}}, \tag{3.10}$$

where C is a constant that regulates the exploration-exploitation trade-off, T_i is the number of times action i has been played up to time $n-1$ and $\overline{X}_i$ is the average payoff observed from arm i.

UCB1 is a frequentist approach, there are also Bayesian approaches such as Thompson Sampling (TS) (Thompson, 1933), used in Chapter 9.

CHAPTER 4

Data Preparation and Testing

In Chapter 2 we learnt about the main financial market participants and gained an overview of the main financial instruments. To appropriately model the mentioned market participants using RL algorithms, it is necessary to establish a pipeline that starts with downloading and pre-processing historical data or simulating data, to train the RL algorithm, and ends with testing. Testing is generally conducted on historical, out-of-sample data (backtesting) but should also be run on real-time market data.

There is no optimal size for the training set on which to train the algorithms but, as a general rule, it would be preferable to have tens, if not hundreds of thousands of data points. Although we live in a big data era, such an extensive dataset is not easily available in the financial market setting. Indeed, value investors, for instance, re-balance their portfolios quarterly, at the release of the financial statements of the relevant companies. This scarcity of data holds true also if we consider the historical price daily time series, which means one price-point per day, as even on a period of 20 years, with 250 trading days per year, we only collect 5000 data points, which are hardly enough and well below the ideal amount. Another fundamental aspect is data quality. In general, data quality worsens with less liquid instruments. For these reasons, it is more common to consider highly liquid exchange traded instruments; however, with a well designed data pre-processing approach, also other assets may be considered.

This chapter guides the reader through the data collection process carried out for the applications in the following chapters.

Chapter Outline

This chapter is divided into five sections. In the first section, we describe the various types of LOB or OTC data that can be collected. Then, in Section 4.2, we define the data collection effort completed for the analyses of this dissertation. In Section 4.3, we present some data generation models: classic Stochastic Differential Equations (SDEs) and a multi-agent market simulator. In Section 4.4 we delineate the data cleaning approach undertaken. We then present the main performance metrics considered in Section 4.5 and end the chapter by illustrating the framing of the testing stage in Section 4.6.

4.1 Data Types

Data can have several frequencies. As anticipated in the previous section, most companies tend to release financial statements quarterly. This data is generally public for listed companies and can be collected by anyone, going several years in the past.

Daily data of closing asset prices is also very popular in the financial markets. For liquid exchange traded instruments and major indexes, daily prices can also be easily found online. Daily price data can generally be downloaded for the past 20 to 30 years and consists of one price point per day at closing.

Intraday prices, usually snapshots of the price with a constant time interval, are, instead, harder to find in public resources. Through the Bloomberg terminal, it is possible to download historical intraday data going back a few months for most exchange traded instruments.

More frequent and complex than the types of data we have seen so far is *tick-by-tick* data, i.e. the record of each transaction. There is thus a variable data frequency depending on the number of transactions, so keeping track of the timestamps is fundamental. Tick-by-tick is in general not available publicly except for FX data, accessible through HistData.[15] It is possible to download tick-by-tick data through the Bloomberg terminal, but with a very limited history, or through some online brokerage accounts upon registration.

While tick-by-tick is the highest frequency type of price data, it is possible to download further information than just the price, for example, LOB data. Even harder can be finding OTC data, we analyse and compare multiple data types in the following section.

LOB data

LOBs (see Section 2.4.2) record all the limit orders currently active, which means that, given a specific price, it is possible to see the order volume, which summarizes the current number and size of the orders. Three levels of order book information can be analyzed:

- level I means the best bid and best ask prices and respective volumes;
- level II represents the order book with 5 to 10 price and volume levels for both the bid and the ask side;
- level III contains all the price and volume levels, and it is possible to distinguish individual orders, even if in an anonymous form.

[15] http://www.histdata.com/.

Figure 4.1: *Graphical representation of the three limit order book levels.*

LOB level II and III data is available for purchase directly from exchanges or from specialized market data vendors like BMLL.[16] In our analyses, we considered LOB level II data for optimal execution in Chapter 9. Given the difficulties in collecting the data and the static nature of the data – making it not so appropriate to study market impact – we decided to use the multi-agent market simulator ABIDES (Byrd et al., 2019), which is capable of reproducing a level II LOB data. ABIDES is described further in Section 4.3.2.

OTC data

In Section 2.1.1, we have seen how OTC data differs from regulated exchanges, and specifically that OTC instruments are used mostly by financial institutions. This makes OTC data extremely challenging, if not impossible, to find from public sources. In our case, a Bloomberg license was necessary to download most of the OTC data.

In dealer market scenarios, that is the majority of the cases when talking about OTC instruments, there are two interesting types of data: on the one hand, there are the quotes of the dealers, and, on the other hand, the RFQs of the clients. Specifically, while, as just mentioned above, dealers' quotes are available in Bloomberg, RFQs are individual and depend on the dealer in consideration. Larger dealers will certainly have a larger RFQ influx given their larger set of clients. To our knowledge, only Fermanian et al. (2016) analyzed and calibrated a distribution to dataset of RFQs received by BNP Paribas for a selection of corporate bonds.

4.2 Data Collection

This section describes the data collection process pursued for the experiments of this dissertation. Data sources include the Bloomberg terminal and online sources.

A data license for the Bloomberg terminal enabled intraday market data collection, by taking a periodic real-time snapshot of the market. For many of the instruments, the collection through this license started on the 29^{th} of May 2017 with snapshots from Bloomberg every 5 minutes starting at 9:30am and ending at 5:25pm. These instruments

[16] `https://bmlltech.com/`.

Instrument	Downloaded features	Frequency	Processed features
SNRFIN	credit spread, upfront, coupon, sensitivity	5 min	time to roll
Intesa CDS	credit spread levels of the entire curve	5 min	time to roll
BTP futures	price, sensitivity, yield, LTD	5 min	time to roll
Bund futures	price, sensitivity, yield, LTD	5 min	time to roll
SX7E futures	price, sensitivity, LTD	5 min	time to roll
VIX index	price	5 min	
V2X index	price	5 min	
EURUSD spot	price	tbt	
USDGBP spot	price	tbt	
SPY ETF	price	daily	
BNDX ETF	price	daily	
DAX ETF	price	daily	
VXX ETF	price	daily	

Table 4.1: *Summary of the instruments considered in this dissertation.*

can be recognized in Table 4.1 as they have a "5 min" frequency. Finally, tick-by-tick (tbt) FX data was downloaded from HistData.com. The data is specified in Table 4.1, which includes also the processed features. Once this data is downloaded and stored on an offline database, it must then be extracted, parsed to create a CSV and cleaned. In the following paragraphs, we describe with more detail the characteristics of the main instruments considered.

CDS Single Name and Indexes For CDS and CDS indexes, which includes SNRFIN and Intesa CDS, yearly maturities are quoted on the markets: 1 year, 2 years, 3 years, 4 years, 5 years, 7 years, 10 years, 15 years, 20 years, and 30 years. We downloaded the 5Y maturity with a 5 minute frequency, specifically the bid and ask credit spreads of several dealers (about 20). Dealers contribute continuously a bid and an ask credit spread for a given notional, ranging from €10mln (million) to up to €400mln, with most of the contributions ranging between €50mln and €200mln. Trading times are not strictly regulated but generally go between 8am CET and 6pm CET. Each contributor has a typical bid-ask spread, which depends on the market conditions and on the level of the index credit spread. Figure 4.2 shows the mid credit spread and bid-ask spreads of the downloaded SNRFIN data, on a one-year time horizon, considering intra-day data with 30-minute time-steps. The market quote published by the dealers can be applied by the clients, but acceptance from the dealer is not always certain: some dealers ensure that the quote will be always confirmed, some retain the right to review it. Moreover, dealers can also decide to publish bid and ask credit spreads that cannot be executed at all. These features make it difficult to define an order book for OTC traded objects. It is often even difficult to ascertain the applicable credit spread, from the downloaded data. Section 4.4 explains the approach taken to process the dataset. It is possible to calculate the upfront, as we saw in Section 2.2.4, or download it from Bloomberg together with the sensitivity, coupon, and coupon date.

For maturities other than 5Y, which tend to have a very low liquidity, usually only the

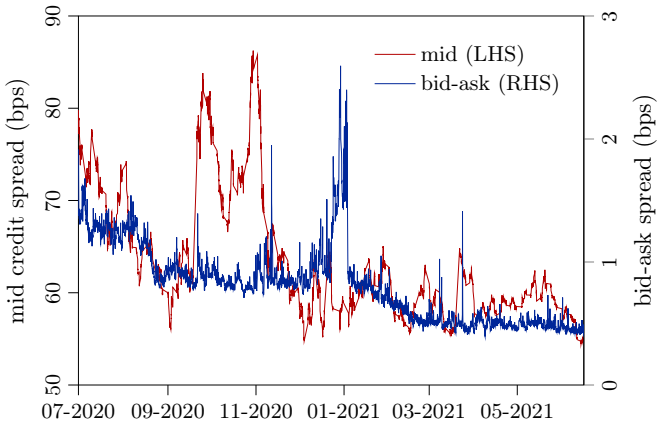

Figure 4.2: *The evolution of the SNRFIN 5y mid credit spread (left axis) and bid-ask spread (right axis) from mid 2020 to mid 2021.*

end of day value is available.

Futures Futures are described in Section 2.2.2. BTP and Bund futures have a physical delivery, while the EURO STOXX Banks Index Futures (SX7E) is cash settled. They all have liquidity concentrated on four maturities, which are typically on the eighth day of March, June, September, and December. Of the bond instruments, we also downloaded other information like the yield, to calculate the *BTP-Bund spread* (difference in yield). These instruments are described in further detail in Appendix B.1.4 and used in Appendix C.

VIX and V2X Indexes and ETF The Chicago Board Options Exchange Volatility Index (VIX) and the EURO STOXX 50 Volatility Index VSTOXX (V2X) are two volatility indices. The ETF version of the VIX was used for a portfolio optimization exercise (see Chapter 5), while the indices are used as a feature to the agent in Appendix C. They represent the market's expectation of 30 day forward looking volatility and provide a measure of market risk and investors' sentiments. In the case of the VIX, the price derives from S&P 500 index options, whereas in the case of V2X, from the EURO STOXX 50 index options.

FX Data: EURUSD & USDGBP FX data is used extensively in Chapter 6 to learn a quantitative trading strategy. FX tick-by-tick data was downloaded from the website HistData.com with bid and ask for the EURUSD and USDGBP FX pairs from 2017 to 2020.

Corona Dataset The *Corona dataset*, used in Section 5.1, was designed using data coming from the recent Covid-19 crisis period to analyze the behavior of the OPO algorithms in times of high volatility (Chapter 5). This dataset contains the ETFs of Table 4.1: the SPY ETF, the Vanguard Bond Index Fund (BNDX ETF), the Global X DAX Germany ETF and the iPath Series B S&P 500 VIX Short-Term Futures ETN (VXX).

Datasets				
Name	Market	Year Span	Days	Assets
NYSE	New York Stock Exchange	1962 - 1984	5651	36
TSE	Toronto Stock Exchange	1994 - 1998	1258	88
SP500	Standard Poor's 500	1998 - 2003	1276	25
Corona	Global	2019 - 2020	280	4

Table 4.2: *Datasets used in the experimental campaign of Chapter 5.*

Benchmark Data For the portfolio optimization analyses of Section 5.1, a pre-defined dataset was used, shown in Table 4.2. Specifically, the NYSE, SP500, and TSE datasets are well-known benchmarks and are used in several research papers on portfolio optimization to provide the daily prices for a fixed set of asset.[17]

S&P 500 Index Components For the portfolio optimization with benchmark analyses of Section 5.2, we used a public dataset of 502 stocks, which make the S&P 500 index, collected with minute frequency from $2017/09/11$ to $2018/02/16$ for a total of 43148 points.[18]

4.3 Data Simulation

Although downloading real data is clearly preferable, the difficulties in finding high frequency data make it necessary to explore also data generation methods. While for learning quantitative trading strategies (Chapter 6) and portfolio optimization (Chapter 5), historical data is fundamental, for other approaches, for example hedging (Chapter 8), using simulated data might be a good way to create a proof of concept or initialize the policy. We now briefly describe classical approaches using stochastic differential equations (SDEs), while an example of an econometric model can be found in Appendix B.3.1. Examples of advanced ML approaches include Quant Gans (Wiese et al., 2020) and The Market Generator (Kondratyev and Christian, 2019). These topics are extremely vast, we only describe what is necessary to understand how they were applied in this dissertation and eventually how they can be used for other projects.

4.3.1 Stochastic Differential Equations

For applications in the dealer markets framework (see Chapter 7) and the option hedging framework (see Chapter 8), we concentrated on the use of SDEs to simulate the price process of the financial instruments. We also initially tested the quantitative trading approaches on simulated data before using real data.

While potentially it is possible to make an educated guess for the SDE parameters that seem to have a "good behavior", it is optimal to calibrate the SDE to the real data. Even though it is necessary to have real data for the calibration, differently to ML approaches, only a few data points are required. Once we have decided the SDE parameters, then it is possible to generate Monte Carlo trajectories that can be used to train the RL algorithm. The

[17] These datasets are available at `http://www.cs.technion.ac.il/~rani/portfolios/`.

[18] This dataset is available at `https://www.kaggle.com/nickdl/snp-500-intraday-data`.

trajectories can be generated using the Euler-Maruyama scheme, or by using the analytic solution when available.

Geometric Brownian Motion Let S_t be the price of the underlying asset at time t, then a GBM process is defined as:

$$\mathrm{d}S_t = \mu S_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t,$$

where W_t is Brownian motion, μ the drift (in general, we assume it to be 0 throughout without loss of generality) and σ the volatility. This process is simple and effective, but a constant volatility is a strong assumption in finance. For an initial value S_0, the SDE has the analytic solution:

$$S_t = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right),$$

where: W_t is the Wiener process, thus $W_{t+u} - W_t \sim \mathcal{N}(0, u) = \mathcal{N}(0, 1) \times \sqrt{u}$, $\mathcal{N}$ being the normal distribution.

If we assume that the price of the asset behaves as a GBM, then the log returns $x_1, ..., x_n$ form normal iid random variables. We can thus compute the parameters that maximize the log likelihood as: $\hat{m} = \sum_{i=1}^{n} x_i/n$ and $\hat{v} = \sum_{i=1}^{n}(x_i - \hat{m})^2/n$, with which we can find estimated parameters of the GBM: $\hat{\mu} = (\hat{m} - \frac{1}{2}\hat{v})\frac{1}{\Delta t}$ and $\hat{\sigma}^2 = \frac{\hat{v}}{\Delta t}$.

Vasicek Model The Vasicek model is normally used in finance to describe the evolution of interest rates. It is a process with mean reversion and follows the stochastic differential equation:

$$\mathrm{d}S_t = a(\mu - S_t)\mathrm{d}t + \sigma \mathrm{d}W_t,$$

where a is the speed of mean reversion, μ the long term mean level and σ is the volatility. It is possible to find the maximum likelihood parameters by calculating:

$$\begin{aligned}
\hat{b} &= \frac{n\sum_{i=1}^{n} x_i x_{i-1} - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_{i-1}}{n\sum_{i=1}^{n} x_{i-1}^2 - (\sum_{i=1}^{n} x_{i-1})^2}, \\
\hat{\mu} &= \frac{\sum_{i=1}^{n}[x_1 - \hat{b}x_{i-1}]}{n(1-\hat{b})}, \\
\hat{\sigma}^2 &= \frac{1}{n}\sum\left[x_i - \hat{b}x_{i-1} - \hat{\theta}(1-\hat{b})\right]^2 / \sqrt{\frac{(\hat{b}^2-1)\Delta t}{2\log\hat{b}}}, \\
\hat{a} &= -\log(\hat{b})/\Delta t.
\end{aligned}$$

Heston Model The Heston model can be intuitively thought of as a GBM but with a process for the volatility. The volatility ν behaves like a Cox-Ingersoll-Ross (CIR) model, which is an extension of the Vasicek model:

$$dS(t) = \sqrt{\nu(t)}\, S(t)\, dW^S(t), \tag{4.1}$$

$$d\nu(t) = \kappa\left(\theta - \nu(t)\right) dt + \xi\sqrt{\nu(t)} dW^\nu(t), \tag{4.2}$$

where κ the speed on mean reversion, θ the long run variance, *i.e.,* $\lim_{t\to\infty}\mathbb{E}[\nu(t)] = \theta$, ξ is the volatility of the volatility. The Weiner processes W^S and W^ν may be correlated.

4.3.2 Multi-agent Market Simulation

The method of agent-based financial markets simulators is radically different from the previously described ones. The focus0 is not on trying to reproduce a realistic price process, or predict future price movements, but on simulating the interactions between market participants in order to reproduce a limit order book. We considered the simulator ABIDES (Byrd et al., 2019), of which the main advertised characteristics are:

- Support for continuous double-auction trading at the same nanosecond time resolution of real markets such as NASDAQ;
- Variable electronic network latency and agent computation delays;
- Requirement that all agents intercommunicate solely by means of standardized message protocols;
- Easy implementation of complex agents through a full-featured hierarchy of base agent classes.

The price process is described by a "fundamental value", which can be a historical time series or a process – in our case, a Vasicek model (described in Section 4.3.1). This exogenous price series is to be interpreted as the global consensus value for the asset under consideration, coming from the aggregation of all accessible news and information. Once the characteristics of the fundamental are defined, it is possible to create an arbitrary number of market players, with definable characteristics. The interactions of the market players create an order book and cause variations to the real price process from the fundamental value. The main types of traders we considered are the following:

- *Zero intelligence traders* a.k.a noise traders, who trade randomly;
- *Value investors*, who create a future projection of the price of the traded asset, and buy if the current price is below (or sell it if is above) their projection;
- *Momentum traders*, who go long (short) as the price of the asset is increasing (decreasing) following heuristic rules.

The combination of a different number of these traders and changes in the characteristics of the fundamental price can result in different behaviors of the limit order book. ABIDES is used in Chapter 9 to study optimal execution.

4.4 Data Processing

Once the data has been downloaded, or simulated, it is necessary to process it, to make it compatible with the RL algorithm (for example with FQI, see Section 6.2.1), or to create additional relevant financial features as described for example in Section 2.4.2, at times used only as state variables, other times necessary to calculate the reward. The most useful features are then chosen through a *feature selection* procedure. Finally, it is fundamental to normalize the data, to avoid biases to the policy. Especially when working with neural networks, the objective is to make sure that all the features are of similar size, so no feature overshadows the others.

Processing CDS data

As described in Section 4.2, the starting point is a dataset containing the most recent bid and ask credit spreads quoted by all the dealers (about 20 in the dataset) every 5 minutes, during the most liquid trading hours (9:30am CET and 5:30pm CET), as specified in Table 4.1. To use this data, it is necessary to discard credit spreads that are most likely typos, not executable or technological problems. Thus, for each time-step and for both the bid and ask we consider the mean and standard deviation of the credit spreads of all dealers and discard from the set the ones that differ from the mean by more than two standard deviations. Considering the processed data, we define as *applicable bid* the average of the remaining bid credit spreads, and as *applicable ask* the average of the remaining ask credit spreads. Finally, we obtain the mid credit spread as the average of the applicable bid and applicable ask, and the bid-ask spread as the difference between them. Figure 4.2 shows an extract of the cleaned data.

Feature Selection

Feature selection is a fundamental process to reduce the number of variables in the state space. This can help to make the training process quicker, and also enhance the results, especially when using value based approaches. Feature selection can be performed with different approaches, in this dissertation we use extra tree regressors (see Appendix B.4.2). Specifically, we create a dataset of experience using a random policy, we then train extra tree regressors on the supervised problem where state and action are the input and the reward is the output. From the trained extra trees, we then extract the impurity-based feature importance, selecting the features with the highest relevance. Feature selection is applied in Chapter 6 and 9. For the sake of completeness, it is also popular to conduct feature selection by means of well-known libraries, such as SHAP (Lundberg and Lee, 2017).

4.5 Performance Metrics

Performance metrics can be distinguished between those typical in a financial setting and those more relevant in a RL setting, even though in many cases they overlap.

Financial The main financial performance metrics considered in this dissertation are listed below:

- *Profit & Loss*. P&L represents the monetary gains of the strategy and can be calculated in slightly different manners. Equations (2.13), (2.17), (2.18) and (2.21) are examples of P&L formulations.
- *Cumulated P&L, or wealth*. Wealth has two main formulations. The first, when re-investing the gains/losses as in Equation (2.13), it is calculated as:

$$W_T = \prod_{t=1}^{T} \rho_t.$$

  Typically, performance is expressed in an annualized fashion, for ease of comparison:

$$W_A = (1 + W_T)^{250/t_w} - 1, \tag{4.3}$$

where t_w is the time period in which the wealth was generated, and 250 is the number of trading days in a year.

The second formulation, when considering a fixed action size, and without re-investing gains/losses as in Equation (2.17), the cumulated P&L becomes a sum:

$$W_T = \sum_{t=1}^{T} r_t.$$

In this case, the annualized wealth is

$$W_A = W_T \frac{250}{t_w}. \tag{4.4}$$

- *P&L variance*. Once we have collected a number of successive daily P&L (the wealth of one day), defined as $\{r_t^d\}_{t\in[1,T]}$, it is possible to calculate the daily variance as:

$$v(r_t^d) = \frac{1}{T}\sum_{t=1}^{T}(r_t^d - \hat{r}^d)^2,$$

where $\hat{r}^d$ is the average daily P&L, and T is the number of days. The variance can be annualized as $v_A = 250 \cdot v$.

- *Volatility*. The volatility is defined as the standard deviation so

$$\text{vol} = \sqrt{v(r_t^d)}.$$

Similarly, the annualized vol: $\text{vol}_A = \sqrt{250} \cdot \text{vol}$.

- *Sharpe ratio.* The Sharpe ratio, usually considered in an annualized fashion, is defined as:

$$sr = \frac{W_A - r_f}{\text{vol}_A}, \tag{4.5}$$

where, r_f is the risk-free rate *i.e.,* what an investor would gain by leaving her money in a savings account for the investment period – at the time of writing, it is generally 0% in Europe and around 1% in the United States.

- *Maximum Drawdown (MDD)*. The maximum drawdown is a measure of risk defined as the maximum observed loss from a peak performance to the trough, before a new peak is attained, formally

$$MDD = \frac{\text{Trough Value} - \text{Peak Value}}{\text{Peak Value}}. \tag{4.6}$$

- *Implementation shortfall.* The Implementation Shortfall (IS) is a measure of market impact, and was defined in Section 2.4.3. We will use this performance metric in Chapter 9 on optimal execution:

$$IS = XP_{\text{arrival}} - \sum_{k=1}^{N} n_k P_k, \tag{4.7}$$

where X is the total size to execute, n_k is the size traded at price P_k, at each time k.

Reinforcement Learning The main RL performance metrics considered in this dissertation, recalling the notation from Section 3, are described in the following paragraph:

- *Regret*. The regret of an online algorithm $\mathfrak{U}$, also defined in Section 3.3.3, can be generically expressed as:

$$R_T(\mathfrak{U}) = \sum_{t=1}^{T} \left[f_t(a_t) - f_t(a^*) \right]. \tag{4.8}$$

  where a_t is the prediction of the agent, a^* is the prediction of the best expert, and f_t is the loss function.

- *Returns*. The returns obtained by following a trajectory τ are defined as in Equation (3.3):

$$G_\tau = \sum_{t=0}^{T} \gamma^t r_{t+1}.$$

  If the reward is the P&L, the return roughly coincides with the wealth (depending on γ).

- *Return variance*. The return variance, defined in (Tamar and Mannor, 2013) is expressed as:

$$\sigma_\pi^2 \coloneqq \mathop{\mathbb{E}_\pi}_{s_0 \sim \mu} \left[\left(G_\tau - \frac{J_\pi}{1-\gamma} \right)^2 \right], \tag{4.9}$$

  where J_π is defined in Equation (3.7).

- *Reward volatility*. The reward volatility a.k.a reward variance, defined in (Bisi et al., 2020b) is expressed as:

$$\nu_\pi^2 \coloneqq \mathop{\mathbb{E}}_{\substack{s \sim d_{\mu,\pi} \\ a \sim \pi(\cdot|s)}} \left[\left(r(s,a) - J_\pi \right)^2 \right], \tag{4.10}$$

$$= (1-\gamma) \mathop{\mathbb{E}_\pi}_{s_0 \sim \mu} \left[\sum_{t=0}^{T} \gamma^t \left(r(s_t, a_t) - J_\pi \right)^2 \right]. \tag{4.11}$$

  The reward volatility is the variance of the immediate rewards, and is comparable to the financial variance $v(r_t^d)$. It is common to label the standard deviation as volatility, we use the name reward volatility in continuity with the original name given in the paper.

4.6 Testing

Once we have defined the dataset and calculated the relevant features as described in the previous sections, it is necessary to optimize the policy on the training set and then test the performance on out-of-sample data. Assuming that the hyper-parameters have already been tuned, there are several levels of "stochasticity" that need to be analyzed:

- training with a different seed may produce a different policy, so it is necessary to train multiple times and test the different policy on the same test set, calculating confidence intervals;
- the policy itself may be stochastic, so it is necessary to test the same policy on the same test set multiple times and calculate confidence intervals;
- a single test set is not sufficient, it is necessary to test how the policy behaves on different out of sample realizations.

Backtesting Backtesting is a common term used in financial settings to indicate a test on historical data, whereas testing may also refer to a real-time paper money environment or a real-time real money framework. In this dissertation we only backtested our policies.

When creating a backtesting framework, it is important to be as realistic as possible. Thus, in this dissertation we considered transaction costs for this purpose. The concept of transaction costs can also be made even more realistic by adding the possibility of market impact, when trading on regulated exchanges, as described in Section 2.4.1. Considering market impact on historical data is a challenging task, which can only be conducted by modeling a transitory impact. In most of the examples examined in this dissertation, we assume that the trading size is small enough so that it causes negligible market impact. To study this impact in an optimal execution setting we decided to use the agent-based financial simulation library ABIDES.

Delay is another way to render the framework more realistic, as the price of an asset can change from the time when the price is observed to the time when the trade is executed. For this reason, it is important to consider algorithms with good computational properties to add as little delay as possible. From this point of view, batch RL algorithms, introduced in Section 3.3, are optimal, as once the policy is trained, obtaining the new action is extremely quick. On the other hand, online approaches, especially MCTS, need to be appropriately tuned and configured to reduce the search time. It is important also to optimize the rest of the pipeline, like the order management system that executes the order.

Paper Money After backtesting the learnt policy, the correct approach would be to test on a paper money account. Such account is made available by several online brokerages, like Interactive Brokers, or trading websites such as Quantconnect.[19] The APIs of these accounts enable the algorithm to send orders, while the online brokerage account keeps track of the P&L. Although the use of paper money accounts integrates transaction costs and delay, even in this case, it is not possible to model market impact.

Real Money The ideal way to test an algorithm is through a brokerage account with real money. However, this entails a long procedure and is only possible when trading as a financial institution. Indeed, given the trading frequency of this dissertation, it is clear that, in order for the algorithm to be profitable, it is necessary to have very low trading costs, only achievable through a privileged market access. Moreover, installing these types of autonomous algorithms in a production environment requires both lengthy evaluations from internal risk management and compliance functions and a reliable infrastructure connecting the agent with the markets.

[19] `https://www.interactivebrokers.com/en/home.php`, `https://www.quantconnect.com`.

Part II

Learning the Financial Markets with RL

CHAPTER 5

Online Portfolio Optimization

The portfolio optimization task is a problem faced by asset managers, as described in Section 2.3.1 and in Definition 2.5. Started by Markowitz (1952) for single period portfolios, it was then extended by Merton (1969) to a continuous-time optimal control problem. Contrarily to the approach presented in this chapter, the Merton problem is based on the fact that assets behave with a specified dynamic, usually a GBM. Under this assumption and considering a stock and a risk-free asset, where the objective of the investor is to maximize the terminal wealth under a power utility function, there is a closed-form solution that suggests the optimal portfolio allocation based on the variance and the returns of the assets. In general, following the Merton approach, the objective is to find the analytic solution of an optimal control problem that defines an inter-temporal portfolio allocation. Our approach is related to Merton's, but with the relevant difference that it does not make explicit distributional assumptions on the assets' dynamics.

We focus on the OPO framework, which derives from the online learning literature (see Definition 3.5), specifically, the expert learning framework, anticipated in Section 3.3.3. Many algorithms from the expert learning literature have been applied to the OPO framework since they provide both strong theoretical guarantees in the adversarial setting and good empirical results. The most interesting ones have been described in detail by Li and Hoi (2014) and by Dochow (2016). The objective in the OPO framework is to define an algorithm capable of minimizing regret with respect to the optimal Constant Rebalancing Portfolio (CRP). The CRP is defined as the portfolio which keeps constant weights and obtains the maximum performance at the end of the investment period.

The first part of this chapter sheds light on the negative impact of transaction costs on

the trading strategy and how to reduce it, the second focuses on the asset manager's problem of beating a benchmark. OPO algorithms could be used together with an optimal execution approach, such as that presented in Chapter 9 to reduce costs, increasing performance.

Chapter outline This chapter is divided into two parts, Section 5.1 focuses on dealing with transaction costs in the OPO framework and Section 5.2 deals with the typical asset management task of beating a specific market index (see Definition 2.4), *i.e.,* to perform better than the chosen index and, concurrently, maximize the collected wealth. Both sections begin with the state-of-the-art, followed by the section-specific context, that is regret including transaction costs in the first case, and the conservative constraint in the second. Afterwards, we introduce our approach and a novel algorithm (respecively OGDM in Section 5.1.3 and CP in Section 5.2.3), which satisfy the new requirements, concluding each section with the experimental results.

5.1 Online Portfolio Optimization with Transaction Costs

The OPO framework has been largely studied but it is rarely used in a production framework. One of the reasons may be that the assumptions of OPO appear as too unrealistic, like the non-existence of transaction costs. Thus, this section focuses on extending the OPO framework and the definition of regret to include transaction costs, largely ignored in the OPO literature, by using the *total regret*, and proposes an algorithm Online Gradient Descent with Momentum (OGDM) that achieves a total regret order of $\mathcal{O}(\sqrt{T})$.

5.1.1 Background on OPO with Transaction Costs

This section provides an overview of the state-of-the-art on OPO algorithms and previous attempts to account for transaction costs. Universal Portfolios with Costs ($\mathrm{U}_C\mathrm{P}$) Blum and Kalai (1999) and Online Lazy Updates (OLU) (Das et al., 2013) are the algorithms closest to this work, and are discussed in depth in Section 5.1.4. Notably, the Online Newton Step (ONS) (Agarwal et al., 2006; Hazan et al., 2007) algorithm has been shown to provide good performance in terms of regret on the wealth when empirically tested, as well as feasible computational complexity. There are also heuristic algorithms designed to solve the OPO problem, *e.g.,*, Anticor (Borodin et al., 2004), PAMR (Li et al., 2012), OLMAR (Li et al., 2015), and MRTC (Yang et al., 2018b), which outperform the algorithms described above in terms of empirical performance. Remarkably, none of the above algorithms provide guarantees on the total regret.

In addition, Li et al. (2018a) extend both traditional and heuristic algorithms including an additional term to the optimization function to handle transaction costs, but only provide an empirical analysis and no type of regret guarantees. Moreover, what is presented by Ito et al. (2018) is related to the objective of controlling transaction costs. Indeed, the assumption that the portfolio is composed of a small set of assets indirectly addresses such a problem. However, the authors do not present theoretical guarantees on the potential costs incurred by such an algorithm.

The problem of dealing with transaction costs has also been tackled in sequential decision-making settings similar to the OPO one, *i.e.*, in the expert learning and MAB fields (Cesa-Bianchi et al., 2013; Trovò et al., 2016) and the Metrical Task Systems (MTS)

literature (Li et al., 2018b; Lin et al., 2012; Goel et al., 2019). In the expert and MAB literature the problem has been analyzed either purely theoretically by Cesa-Bianchi et al. (2013) or under the bandit feedback by Trovò et al. (2016), which is not realistic in our financial application. In the MTS literature, the notion of regret has been extended to include the cost of changing the prediction of the algorithm over time, but the framework allows the learner to know the future realizations of the environment, *i.e.*, the price of the assets for the next day, which is unreasonable in our application.

Finally, the algorithm we propose for dealing with online optimization is inspired by the effectiveness of the momentum technique by Polyak (1964). The momentum term smooths the estimation of the gradient and it has been used successfully in the optimization of complex non-linear functions, such as Neural Networks (Qian, 1999; Sutskever et al., 2013).

5.1.2 Formulating Transaction Costs in OPO

Definition 2.5 illustrates the asset manager's task. Recalling the notation of Section 2.3.1, $\mathbf{a}_t := (a_{1,t}, \dots, a_{M,t})$ is the portfolio allocation and $\mathbf{y}_t := (y_{1,t}, \dots, y_{M,t})$ are the price relatives. The wealth is defined in Equation (2.14), and extended to include transaction costs in Equation (2.15). As common in the portfolio allocation literature (Agarwal et al., 2006), we assume the price of the assets does not change too much during two consecutive rounds, or, formally:

Assumption 5.1. *There exist two finite constants $\epsilon_l, \epsilon_u \in \mathbb{R}^+$ s.t. the price relatives $y_{j,t} \in [\epsilon_l, \epsilon_u]$, with $0 < \epsilon_l \leq \epsilon_u < +\infty$, for each round $t \in \{1, \dots, T\}$ and each asset $j \in \{1, \dots, M\}$.*

It is necessary to characterize the loss function to calculate regret as generically defined in Equation (4.8). In the OPO setting, the log loss is commonly used:

$$f_t(\mathbf{a}_t) := -\log(\langle \mathbf{a}_t, \mathbf{y}_t \rangle). \tag{5.1}$$

Furthermore, it is imperative to define the experts. In the OPO framework, they are CRPs, or, in other words, portfolios that keep the investment strategy $\mathbf{a}_{1:T}$ constant for all $t \in \{1, \dots, T\}$. Thus, the best expert is naturally the best CRP:

$$\mathbf{a}^* = \mathbf{a}_t^* = \operatorname*{arg\,max}_{\mathbf{a} \in \Delta_{M-1}} \prod_{i=1}^{T} \langle \mathbf{a}, \mathbf{y}_i \rangle.$$

Hence, we are in a context with an infinite number of experts.

The *regret on the wealth* $R_T(\mathfrak{U})$ at round T is the difference between the cumulative losses of the best CRP and those of the algorithm $\mathfrak{U}$, formally, recalling Equation (4.8):

$$\begin{aligned} R_T(\mathfrak{U}) &= \sum_{t=1}^{T} f_t(\mathbf{a}_t) - \sum_{t=1}^{T} f_t(\mathbf{a}^*) \\ &\equiv \sum_{t=1}^{T} -\log(\langle \mathbf{a}_t, \mathbf{y}_t \rangle) - \sum_{t=1}^{T} -\log(\langle \mathbf{a}^*, \mathbf{y}_t \rangle) \\ &= \log(W_T(\mathbf{a}_{1:T}^*, \mathbf{t}_{1:T})) - \log(W_T(\mathbf{a}_{1:T}, \mathbf{t}_{1:T})), \end{aligned}$$

Regret on the Costs Equations (2.16) and (2.15) respectively define the transaction costs as an implicit formulation and the wealth considering those costs. To find a regret expression that includes costs, it is necessary to find an approximate but explicit cost formulation. If we assume that the price relatives $\mathbf{y}_t$ are small, we have that $\mathbf{a}'_{t-1} \approx \mathbf{a}_{t-1}$ and $\alpha_t \mathbf{a}_t \approx \mathbf{a}_t$, where $\mathbf{a}'_{t-1} = \frac{\mathbf{a}_{t-1} \otimes \mathbf{y}_{t-1}}{\langle \mathbf{a}_{t-1}, \mathbf{y}_{t-1} \rangle}$ and $\alpha_t = 1 - \gamma ||\mathbf{a}'_{t-1} - \mathbf{a}_t \alpha_t||_1$. Therefore, the proportion of remaining wealth becomes $\alpha_t = 1 - \gamma ||\mathbf{a}_{t-1} - \mathbf{a}_t||_1$. Using the above approximations and considering $\alpha_1 = 1$, the wealth $\tilde{W}_T(\mathbf{a}_{1:T}, \mathbf{y}_{1:T})$ can be transformed as follows:

$$\log(\tilde{W}_T(\mathbf{a}_{1:T}, \mathbf{y}_{1:T})) = \log\left(\prod_{t=1}^{T} \langle \mathbf{a}_t, \mathbf{y}_t \alpha_t \rangle\right) \tag{5.2}$$

$$\approx \log(W_T(\mathbf{a}_{1:T}, \mathbf{y}_{1:T})) + \log\left(\prod_{t=2}^{T} \alpha_t\right) \tag{5.3}$$

$$\approx \log(W_T(\mathbf{a}_{1:T}, \mathbf{y}_{1:T})) - \sum_{t=2}^{T} \gamma ||\mathbf{a}_t - \mathbf{a}_{t-1}||_1, \tag{5.4}$$

where we used a first term expansion $\log(1 - y) \sim -y$ to get Equation (5.4), given that the transaction rate is $\gamma \ll 1$. Using the second term of Equation (5.4), we define the proportional costs $C_T(\mathfrak{U})$ incurred by an algorithm $\mathfrak{U}$ during the investment horizon of T, as is done in Das et al. (2013):

$$C_T(\mathfrak{U}) := \gamma \sum_{t=1}^{T-1} ||\mathbf{a}_{t+1} - \mathbf{a}_t||_1. \tag{5.5}$$

The *total regret* $R_T^C(\mathfrak{U})$, *i.e.*, the regret computed considering the transaction costs, is defined as:

$$R_T^C(\mathfrak{U}) = R_T(\mathfrak{U}) + C_T(\mathfrak{U}).$$

The financial interpretation is that regret on the costs consists in the (approximated) turnover gap and it decreases the final wealth of the investor. While total regret is the combination of the regret coming from the suboptimal choice of the portfolio and the one from the turnover, *i.e.*, the wealth gap considering transaction costs. Notice that the best CRP investment strategy $\mathbf{a}^*_{1:T}$ generates no costs under this model, therefore the costs $C_T(\mathfrak{U})$ also represent the *regret on the costs* of the algorithm $\mathfrak{U}$ due to the transaction fees paid over the investment horizon T.

5.1.3 Online Gradient Descent with Momentum

The main novelty, presented in Vittori et al. (2020a), is the Online Gradient Descent with Momentum (OGDM) algorithm, which is an extension of the well-known OGD algorithm. The definition of the OGDM update rule for a generic convex loss function $f_t(\mathbf{a}_t)$ over a generic convex set X is the following:

$$\mathbf{a}_{t+1} = \Pi_X \left(\mathbf{a}_t - \eta_t \nabla f_t(\mathbf{a}_t) - \frac{\lambda_t}{2}(\mathbf{a}_t - \mathbf{a}_{t-1}) \right), \tag{5.6}$$

Algorithm 2: OGDM in OPO with Transaction Costs

1 **Initialize:** learning rate sequence $\{\eta_1, \dots, \eta_T\}$, momentum parameter sequence $\{\lambda_1, \dots, \lambda_T\}$
2 Set $\mathbf{a}_1, \mathbf{a}_2 \leftarrow \frac{1}{M}\mathbf{1}$
3 **for** $t \in \{2, \dots, T\}$ **do**
4 Select $\mathbf{a}_{t+1} \leftarrow \Pi_{\Delta_{M-1}}\left(\mathbf{a}_t + \eta_t \frac{\mathbf{y}_t}{\langle \mathbf{y}_t, \mathbf{a}_t \rangle} - \frac{\lambda_t}{2}(\mathbf{a}_t - \mathbf{a}_{t-1})\right)$
5 Observe $\mathbf{y}_{t+1}$ from the market
6 Get wealth $\log(\langle \mathbf{y}_{t+1}, \mathbf{a}_{t+1} \rangle) - \gamma ||\mathbf{a}_{t+1} - \mathbf{a}_t||_1$

where $\Pi_X(y) := \arg\min_{x \in X} ||y - x||_2^2$ is the standard projection of the vector y onto X, $\eta_t > 0$ is the learning rate at round t, λ_t is a parameter controlling the momentum influence at round t, and $\nabla(\cdot)$ denotes the gradient operator. Recalling that in the OPO framework the function to be minimized is the loss $f_t(\mathbf{a}_t) = -\log(\langle \mathbf{a}_t, \mathbf{y}_t \rangle)$, the portfolio update rule becomes:

$$\mathbf{a}_{t+1} = \Pi_{\Delta_{M-1}}\left(\mathbf{a}_t + \eta_t \frac{\mathbf{y}_t}{\langle \mathbf{a}_t, \mathbf{y}_t \rangle} - \frac{\lambda_t}{2}(\mathbf{a}_t - \mathbf{a}_{t-1})\right). \tag{5.7}$$

The pseudo-code corresponding to the OGDM algorithm in the OPO framework, including transaction costs, is presented in Algorithm 2. The algorithm starts with a portfolio $\mathbf{a}_1$ of weights equally allocated among the M available assets (Line 2). Then, for each round $t \in \{1, \dots, T\}$ it rebalances the assets according to Equation (5.7) (Line 4), it observes the market outcomes $\mathbf{y}_{t+1}$ (Line 5), and gains a per-round wealth, including costs, of $\log(\langle \mathbf{y}_{t+1}, \mathbf{a}_{t+1} \rangle) - \gamma ||\mathbf{a}_{t+1} - \mathbf{a}_t||_1$ (Line 6).

Regret Analysis

We now present the main result on the total regret of the OGDM algorithm.

Theorem 5.1. *The OGDM algorithm with $\eta_t = \frac{K_\eta}{\sqrt{t}}$, and $\lambda_t = \frac{K_\lambda}{t}$, for each value of $K_\eta, K_\lambda \in \mathbb{R}^+$, has a total regret of:*

$$R_T^C \leq \left[\frac{D^2}{K_\eta}\left(\frac{1}{2} + K_\lambda\right) + K_\eta \tilde{G}\left(2\gamma\sqrt{M} + \tilde{G}\right)\right]\sqrt{T}, \tag{5.8}$$

where $D = \sup_{\mathbf{a},\mathbf{y} \in X} ||\mathbf{a} - \mathbf{y}||_2$, and $\tilde{G} = \sup_{\mathbf{a} \in X} ||\nabla f_t(\mathbf{a})||_2 + \frac{DK_\lambda}{2K_\eta}$.

If we assume that the price of the assets does not change too much during two consecutive rounds, as stated in Assumption 5.1, we have:

Corollary 5.1. *If Assumption 5.1 holds, the OGDM algorithm $\eta_t = \frac{K_\eta}{\sqrt{t}}$, and $\lambda_t = \frac{K_\lambda}{t}$, for each $K_\eta > 0$ and $K_\lambda > 0$, has total regret of:*

$$R_T^C(OGDM) \leq \sqrt{T}\left[\frac{K_\lambda^2 + 4K_\lambda + 2}{2K_\eta} + K_\eta M \frac{\epsilon_u}{\epsilon_l}\left(\frac{\epsilon_u}{\epsilon_l} + 2\gamma\right) + \sqrt{2}\left(\frac{\epsilon_u}{\epsilon_l} + 2\gamma\right) K_\lambda \sqrt{M}\right].$$

Using the previous bound we can optimize the regret bound with respect to the parameters K_η and K_λ as follows:

Corollary 5.2. *If Assumption 5.1 holds, the OGDM algorithm with* $\lambda_t = 0$ *and with* $\eta_t = \frac{1}{\sqrt{t}}\left[\frac{M\epsilon_u}{\epsilon_l}\left(\frac{\epsilon_u}{\epsilon_l} + 2\gamma\right)\right]^{-1/2}$*, has a total regret of:*

$$R_T^C(OGDM) \leq 2\sqrt{\frac{M\epsilon_u}{\epsilon_l}\left(\frac{\epsilon_u}{\epsilon_l} + 2\gamma\right)T}. \tag{5.9}$$

Notice that the OGDM with the previous choice of the bound corresponds to the OGD algorithm with a learning rate of $\eta_t = K_\eta/\sqrt{t}$. Indeed, this is consistent with the fact that $R_T(OGD) = \mathcal{O}(\sqrt{T})$ for a generic convex function $f_t(x)$, as shown by (Belmega et al., 2018). Even if this is the choice that minimizes the upper bound on the total regret, it might be suboptimal in practice. In the experimental section we analyze the empirical performance of choices of $K_\lambda \neq 0$.

Finally, knowing the time horizon T in advance from the starting of the investment period we have:

Corollary 5.3. *If Assumption 5.1 holds, the OGDM algorithm with* $\lambda_t = 0$ *and with* $\eta_t = \frac{1}{\sqrt{T}}\left[\frac{M\epsilon_u}{\epsilon_l}\left(\frac{\epsilon_u}{2\epsilon_l} + \gamma\right)\right]^{-1/2}$ *(which results in a constant* η_t*), has a total regret of:*

$$R_T^C(OGDM) \leq 2\sqrt{\frac{M\epsilon_u}{\epsilon_l}\left(\frac{\epsilon_u}{2\epsilon_l} + \gamma\right)T}.$$

This result provides a slightly improved constant in the bound over the *any-time* bound given by Corollary 5.2. In the next sections, we compare the theoretical guarantees of OGDM in terms of computational complexity and total regret with OLU and U$_C$P, the only algorithms that provide upper bounds to total regret.

5.1.4 Comparison with State-of-the-art OPO algorithms

Discussion on the Per-round Computational Complexity In this section we explore the theoretical results of OGDM in comparison to the existing literature with respect to regret bounds and computational complexity, summarized in Table 5.1.[20]

	OGDM	**U$_C$P**	**OLU**	**ONS**
R_T	$\mathcal{O}(\sqrt{T})$	$\mathcal{O}(\log T)$	$\mathcal{O}(\sqrt{T})$	$\mathcal{O}(\log T)$
R_T^C	$\mathcal{O}(\sqrt{T})$	$\mathcal{O}(\log T)$	$\mathcal{O}(T)$	-
Complexity	$\Theta(M)$	$\Theta(T^M)$	$\Theta(M)$	$\Theta(M^2)$

Table 5.1: *Theoretical results, in terms of regret and computational complexity, for the analysed algorithms.*

Regarding the computational complexity of the OGDM algorithm, at each round t, it evaluates a scalar product, a division and two vector subtractions (see Line 4, Algorithm 2), which require a number of operations linearly proportional to the number of assets M. It also performs a projection onto the simplex, which can be computed in linear time with

[20] We report the regret bounds and complexity also for the ONS algorithm for sake of completeness.

the number of assets M (see (Duchi et al., 2008) for details). Therefore, the total expected computational cost per round is $\Theta(M)$. Note that, since the learning rate η_t is decreasing over time, the projection operation is less likely to be required as we proceed with the investment process, decreasing the per-step computational effort. Conversely, the technique used in literature to implement the $\mathrm{U}_C\mathrm{P}$ strategy requires a number of operations per round of $\Theta(T^M)$ (Kalai and Vempala, 2002), which does not scale well for large horizons T, or settings where the number of assets M is large.

Das et al. (2013) propose to use the Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011) to implement the update rule of OLU. Although in terms of computational complexity it has the same properties of OGDM, the OLU algorithm is more computationally costly (in terms of constants) than the OGDM update, since it consists of solving a problem with linear complexity in M multiple times until ADMM converges, but still provides a feasible solution in terms of computational effort. To conclude, $\mathrm{U}_C\mathrm{P}$ is a solution suitable only for problems with a small number of assets M and a short investment horizon T. Conversely, OGDM and OLU can handle data streams that come at higher frequencies, *e.g.*,, the ones required by some specific financial applications (Abernethy and Kale, 2013).

Discussion on the Total Regret Bounds As discussed above, the OLU algorithm is the only algorithm competing with OGDM, in terms of per-round computational complexity. OLU can be interpreted as an instance of Composite Objective Mirror Descent (COMID) (Duchi et al., 2010), whose update is the following:

$$\mathbf{a}_{t+1} = \underset{\mathbf{a}\in\Delta_{M-1}}{\arg\min}\{\eta\langle\nabla f_t(\mathbf{a}_t), \mathbf{a}\rangle + \eta\, r(\mathbf{a}) + d_\psi(\mathbf{a}, \mathbf{a}_t)\}\,,$$

where $r(\mathbf{a})$ is a regularization term of the loss function $f_t(\mathbf{a})$, and $d_\psi(\mathbf{a}, \mathbf{y})$ is a Bregman divergence (Banerjee et al., 2005) generated by the convex function $\psi(\mathbf{a})$. Specifically, the OLU algorithm uses as regularizer $r(\mathbf{a}) := ||\mathbf{a} - \mathbf{a}_t||_1$ and divergence $d_\psi(\mathbf{a}, \mathbf{y}) := \frac{1}{2}||\mathbf{a} - \mathbf{y}||_2^2$.

Assuming to know *a priori* the time horizon T and under Assumption 5.1, the authors of OLU provide the following guarantee (Das, 2014):

Theorem 5.2. *If Assumption 5.1 holds, the OLU algorithm with* $\eta = \frac{K}{\sqrt{T}}$, $\forall K \in \mathbb{R}^+$ *has a total regret of:*

$$R_T^C(OLU) \leq \left(\frac{1}{K} + \frac{MK\epsilon_u^2}{2\epsilon_l^2}\right)\sqrt{T} + 2\gamma T. \tag{5.10}$$

Notice that the OLU algorithm achieves a regret of $\mathcal{O}(\sqrt{T})$ only if the transaction rate $\gamma \propto \frac{1}{\sqrt{T}}$, *i.e.*, if the transaction rate decreases over time. We can observe that the first term of the r.h.s. of Equation (5.10) corresponds to the regret on the wealth. Instead, if we focus on the second term of the r.h.s. of Equation (5.10) and we assume that γ is constant over the investment horizon T, we would have a total regret of the order of $\mathcal{O}(T)$ for the OLU algorithm. This does not happen to OGDM, which, under these assumptions, provides a total regret of the order of $\mathcal{O}(\sqrt{T})$. Conversely, if we assume $\gamma \propto \frac{1}{\sqrt{T}}$ as in (Das et al., 2013), the last term in Equation (5.9) would have constant regret on the costs, *i.e.*, $C_T(OGDM) \leq \frac{2\epsilon_u M}{\epsilon_l} = \mathcal{O}(1)$, compared to an order of $\mathcal{O}(\sqrt{T})$ obtained by OLU, which makes OGDM strictly better than OLU in terms of total regret bound.

5.1.5 Experimental Results

In this section we analyze the empirical performance of the OGDM algorithm and the two algorithms from the OPO literature that provide guarantees on total regret: U_CP (Blum and Kalai, 1999), and OLU (Das et al., 2013).[21] Furthermore, we compare OGDM with OGD, to evaluate the empirical improvement provided by the momentum, and with ONS (Agarwal et al., 2006), which has theoretical guarantees and is known to provide the best empirical results for the regret on the wealth $R_T(\mathfrak{U})$.

To compare the algorithms, we used four different datasets, summarized in Section 4.2, specifically the Corona dataset and the Benchmark data (the latter represented in Table 4.2). The experiments on each of the above datasets consist in the execution of the analyzed algorithms on portfolios selected by randomly drawing 5 different assets from the datasets.

For comparison purposes, we set the same η_t for OGD and OGDM as prescribed by Corollary 5.2, with $\epsilon_l = 0.8$ and $\epsilon_u = 1.2$, for which Assumption 5.1 holds for all the datasets. Instead, to tune the K_λ for the sequence λ_t in Corollary 5.1 for OGDM and the parameters required by the other algorithms (OLU, ONS, and U_CP), we divided the datasets into a validation and testing set of equal size, we optimized the parameters on the former one and evaluated the performance of the algorithms on the latter one. All algorithms have been initialized with $\mathbf{a}_1 = \frac{1}{M}\mathbf{1}$.

As performance indexes to compare the algorithms we used:

- the *wealth with costs*:

$$W_T^C(\mathbf{a}_{1:T}, \mathbf{y}_{1:T}) = W_T(\mathbf{a}_{1:T}, \mathbf{y}_{1:T}) - \gamma \sum_{t=1}^{T} ||\mathbf{a}_t - \mathbf{a}_{t-1}||_1;$$

- the *annualized* wealth with costs W_A^C as defined in Equation (4.4);

- the *average variation of the portfolio per-round*:

$$V_t(\mathfrak{U}) := \frac{C_t(\mathfrak{U})}{\gamma t},$$

  where $V_t(\mathfrak{U})$ is defined so that it is independent from the parameter γ.

Experiments without transaction costs

In this first experimental section, we test the performance of the OGDM algorithm in a situation where transaction costs are absent ($\gamma = 0$). Specifically, we present two different experiments that allow us to draw conclusions on the behavior of the different algorithms in two radically different market scenarios.

Figure 5.1 (a) and 5.1 (b) show the evolution of the total wealth $W_t^C(\mathfrak{U})$ of the different algorithms over the investment horizon in a specific run, on the Corona dataset and on the NYSE dataset, respectively. In these experiments OGDM obtains a cumulative wealth and Sharpe ratio larger than any other algorithm analyzed, suggesting that it can obtain

[21] We used a naïve version of U_CP since the classic implementation would have taken an unfeasible amount of time for the experiments. More specifically, we discretized the simplex with 10^4 points and used the corresponding CRPs to approximate the integrals used by U_CP.

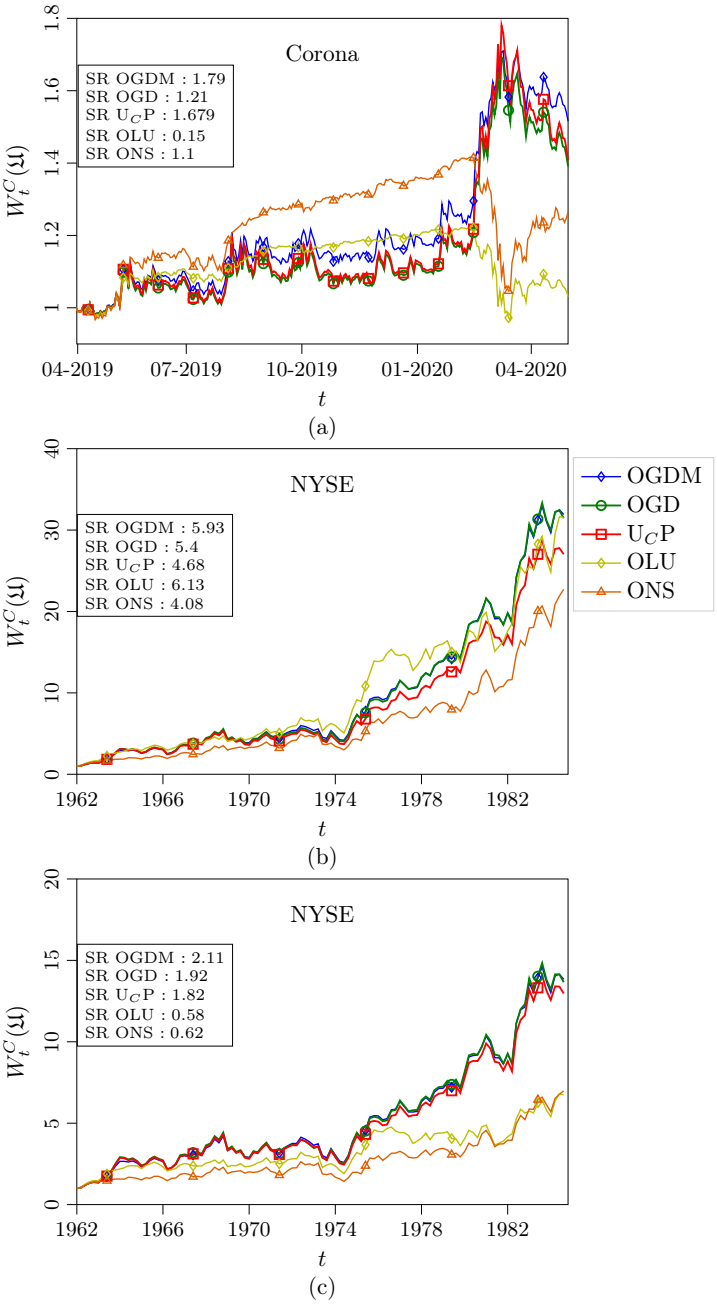

Figure 5.1: *Wealth $W_T^C(\mathfrak{U})$ of a specific run, on the Corona dataset (a), on 5 stocks of the NYSE (see Figure A.1) for $\gamma = 0$ (b), and $\gamma = 0.01$ (c). Sharpe Ratio (SR) is calculated with a risk-free rate of 0% for (a) and 14% for (b) and (c).*

the best performance even in the absence of costs. Comparing the two figures we notice that, while in Figure 5.1 (b) all tested algorithms have a generally positive trend in terms of wealth $W_t^C(\mathfrak{U})$ over time, in Figure 5.1 (a) OGDM, OGD, and U$_C$P are able to provide significantly better performance in the last part of the time horizon ($220 \leq t \leq 250$). The superior behavior is also seen by observing the Sharpe ratio. This performance is due to the presence of the VIX in the investment universe, which had an impressive gain during the initial phased of the Covid-19 spread. Instead, if we look at the periods of general market stability (the entire time horizon of Figure 5.1 (b) and the period in $1 \leq t \leq 220$ of Figure 5.1 (a)), there is no clear outstanding algorithm among the ones we analyzed. It is curious to notice how OGDM, OGD, and U$_C$P show similar behaviors, while OLU and ONS are different from the first three, but similar to each other. Overall, the OGDM algorithm is capable of obtaining a performance comparable to the ones present in the literature in both settings.

Experiments with transaction costs

In the second set of experiments, we ran the algorithms on the NYSE, SP500, and TSE datasets and evaluated the performance of the algorithms in settings with different values of the transaction cost rate $\gamma \in \{0, 0.0005, 0.001,\ 0.003, 0.006, 0.01, 0.02, 0.04\}$. We evaluated the different algorithms in terms of annualized percentage wealth (considering costs) W_A^C (see Equation (4.3)) and average variation of the portfolio per round $V_t(\mathfrak{U})$.[22] The 95% confidence intervals for the analyzed quantities have been computed with statistical bootstrapping and appear as semi-transparent areas.

Figure 5.1 (c) provides the results of a setting with $\gamma = 0.01$ for the same set of assets shown in Figure 5.1 (b). This example shows that OGDM is essentially overlapping with OGD, but achieves a slightly better performance which results in a marginally better Sharpe ratio.

In Figure 5.2, we present the results for the average W_A^C on three datasets: NYSE, SP500, and TSE. Without transaction costs ($\gamma = 0$), all the analyzed algorithms obtain consistently a W_A^C between around 10% and 15%. In this setting, ONS is the algorithm with the largest average W_A^C, but as we increase the transaction costs rate, it is the algorithm that suffers the most, along with OLU. The performance of U$_C$P deteriorates as the transaction cost rate increases, but its loss is limited compared to the ones of ONS and OLU, always providing a wealth $W_A^C(U_CP) > 0$. OGDM and OGD outperform the other algorithms when the transaction rate is $\gamma \geq 0.01$. While showing similar behavior for the SP500 dataset, in the other two experiments the OGDM is able to outperform OGD obtaining almost the same W_A^C as in the setting with no costs, even when $\gamma = 0.04$. Conversely, OGD, not using an explicit term for costs in its optimization procedure, has a slight decrease in terms of wealth as costs increase.

In Figure 5.3, we present the results in terms of average variation of the portfolio per round $V_t(\mathfrak{U})$ for three datasets: NYSE, SP500, and TSE. The worst performing algorithm is OLU since, as expected from the theory (see Section 5.1.4), its variation of the portfolio per round, which is proportional to the transaction costs, is approximately constant. The second worst performer is ONS, which, even though starting with a larger variation than OLU and

[22] We also run experiments with transaction costs on the Corona dataset, and the results are not presented here as they are in line with the ones presented for the other datasets.

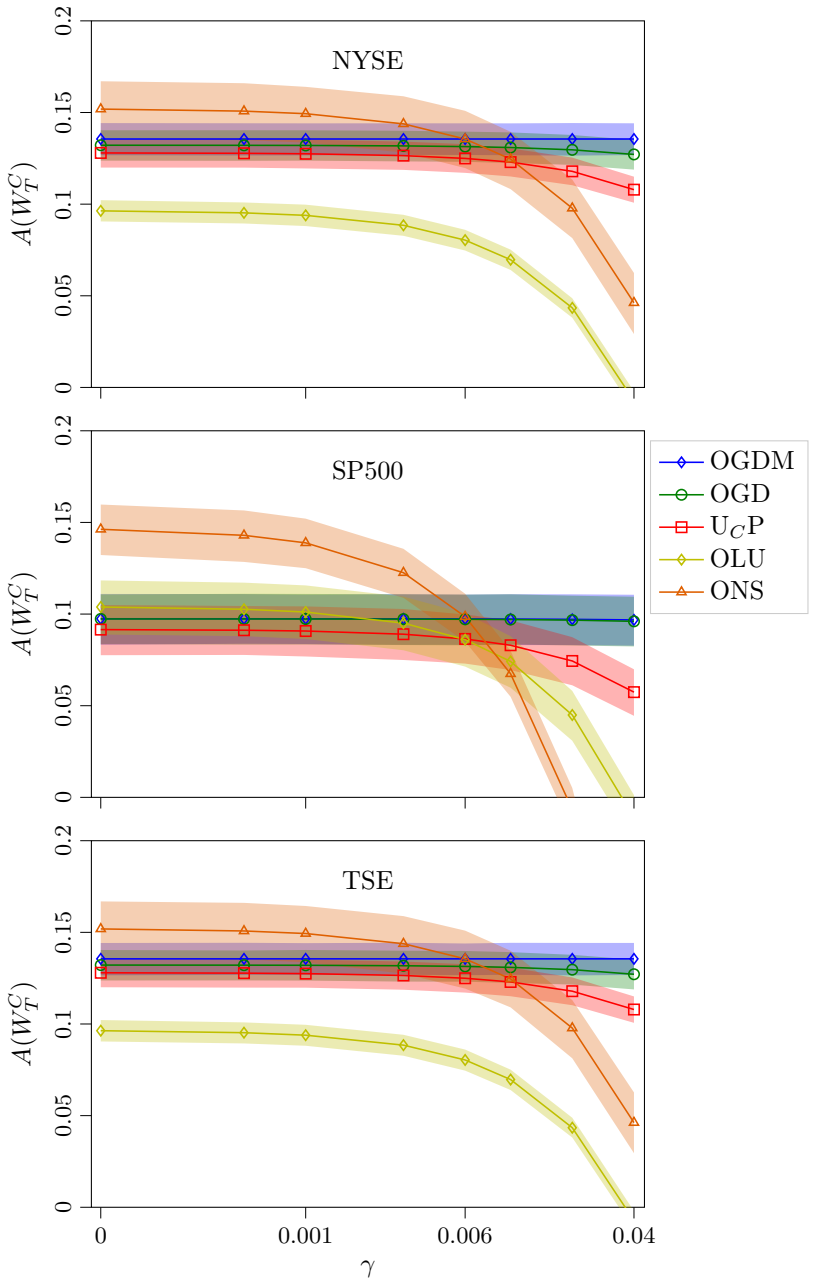

Figure 5.2: *Average annualized percentage wealth with 95% confidence intervals computed on the wealth* $W_T^C(\mathbf{a}_{1:T}, \mathbf{y}_{1:T})$.

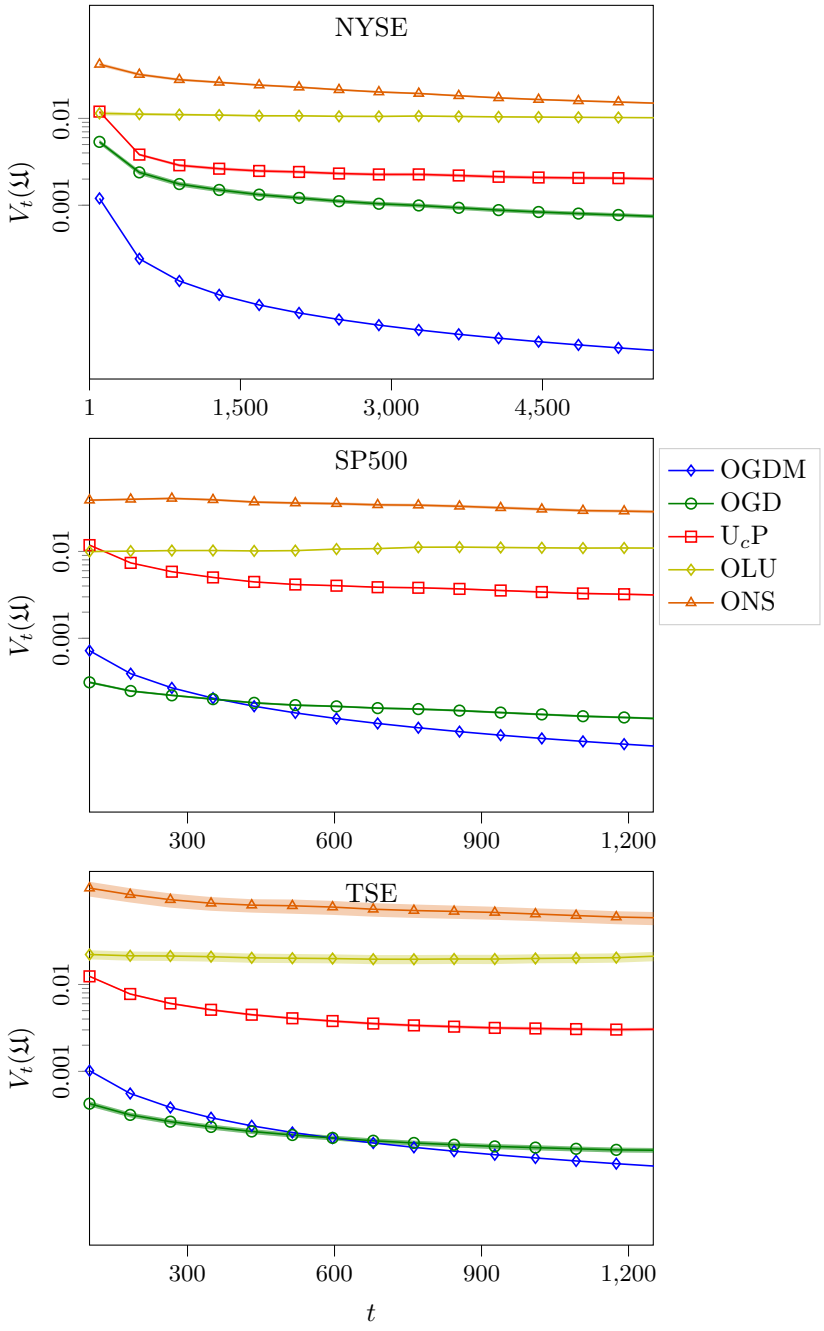

Figure 5.3: *Average variation of the portfolio $V_t(\mathfrak{U})$ incurred on a varying time horizon t.*

not having any theoretical guarantee, is likely to have a sublinear variation per round that decreases over time, but at a slower pace than the remaining three algorithms. Comparing OGDM, OGD, and U$_C$P, we can see that they all have a variation of the portfolio per round that decreases over time. Out of these three, U$_C$P is the one with the worst performance. Finally, comparing OGDM and OGD, the respective behavior varies between the three datasets, and it seems that OGDM is slightly better at minimizing the costs as time increases. This is expected given the previous results, in which OGDM seems to provide a larger W_A^C by keeping costs low. Thus, this result suggests that OGDM performs well not just because it is good at handling transaction costs, but also because it has a superior investment strategy. To conclude, the experiments confirm the theoretical properties discussed in Section 5.1.3 and suggest that OGDM is the best algorithm to use in the presence of large ($\gamma > 0.01$) transaction costs.

Interpreting the Results It is not straightforward to relate the obtained strategies with traditional strategies used in finance. OGDM has somewhat trend following behavior as it increases the weights of the assets which have better performance. It is also important to notice that since the control quantity are the weights of the portfolio, keeping constant weights, for example in the case of a CRP, means selling assets which increase in value and buying assets which increase, thus relating to mean-reverting strategies.

5.2 Online Portfolio Optimization with a Benchmark

While many asset managers aim at obtaining the best possible performance, the majority compare their gains with a benchmark. A benchmark is usually a market index (see Definition 2.4), for example the S&P 500 index, and the objective of the asset manager is to invest in a subset of the components of the index or to use a different weighting than the index, to outperform the index itself. To study this problem from the OPO perspective, we introduce Conservative Online Convex Optimization (COCO) framework.

COCO is a novel approach, deriving from the Online Convex Optimization (OCO) field, in which the learner has to perform online asymptotically as well as the best-fixed decision in hindsight while satisfying a conservativeness constraint, *i.e.,* during the operational life of the system it has to perform no worse than a given fixed strategy. Learning an optimal strategy while satisfying a conservativeness constraint during the exploration phase is of paramount importance in multiple domains, here we concentrate on the asset management problem.

The idea of learning while guaranteeing the performance of a fixed and known policy has also been studied in RL (Garcelon et al., 2020a) and MAB (Wu et al., 2016) (see also Section 3.3.3 for an intro to MAB). To solve this problem, we extend the techniques from the OCO literature, and propose a meta-algorithm, namely Conservative Projection (CP), which extends *any* online learning algorithm to satisfy the requirements of the COCO framework. Thanks to the use of a pseudo-loss and a projection in a so-called *conservative ball*, the proposed CP algorithm provides anytime guarantees with respect to a fixed default strategy.

5.2.1 Background on Conservative OCO

Problems closely related to those of conservativeness have been commonly addressed by *safe* RL techniques. In García and Fernández (2015), the authors provide a comprehensive overview of the different definitions of *safety* in RL. The most common assumption is to have access to a safe policy, and the goal is to improve that policy monotonically throughout the learning process. The seminal paper for this setting is Kakade and Langford (2002), which proposes a conservative policy iteration algorithm with monotonic improvement guarantees for mixtures of greedy policies. This approach is generalized to stationary and stochastic policies in Pirotta et al. (2013); Schulman et al. (2015). Building on the former, in Papini et al. (2017); Pirotta et al. (2015); Papini et al. (2019) the authors have designed monotonically improving policy gradient algorithms for Gaussian, Lipschitz, and, recently, smoothing policies. This setting differs substantially from ours as the underlying environment is assumed to be stochastic. We analyze this setting more accurately in Chapter 8, where we propose a risk-averse and safe RL algorithm to address the hedging problem.

In the bandit setting, the authors of Lattimore (2015) analyzed the same concept of conservativeness, characterizing the Pareto regret frontier in the stochastic case, *i.e.,* a surface determined by the admissible regret bounds for each arm. Following these seminal works, the interest of the MAB community in conservative exploration has grown in recent years, starting with the work presented in Wu et al. (2016), where the authors modified the UCB1 algorithm (Auer et al., 2002; Auer and Ortner, 2010) to guarantee the safety constraint. Later, the idea was applied to contextual linear bandits in Kazerouni et al. (2016) and improved in Garcelon et al. (2020b), as well as to GPUCB, as presented in Sui et al. (2015, 2018). We inherit the concept of *safety as conservatism* from these works on stochastic bandit feedback and apply it to the context of adversarial full-information feedback.

In the expert learning literature, a work similar to ours is Sani et al. (2014). In this article the authors design a strategy, named (A, B)-prod, that provides regret guarantees with respect to the regret of two generic strategies A and B. However, their conservativeness definition is not comparable to ours, since it does not hold all the time. The question of bounding the regret not only to the best action but also to other strategies is addressed in Hutter and Poland (2005); Koolen (2013), in which the authors proved, for the full information setting, that there exists an algorithm that guarantees a regret of $\mathcal{O}(\sqrt{T})$, with a specific constant for each expert. In particular, the main focus of the paper is to characterise the admissible vectors $\{p_k\}_{k\in K}$ guaranteeing a regret $R_T^k \leq p_k$ with respect to each expert k. Even if these works cover a more general theoretical framework than ours, *i.e.,* multi-objective regret minimization, the algorithms therein do not guarantee that their loss is strictly smaller than that of a given expert, and, therefore, their results cannot be compared with ours.

5.2.2 Formulating Conservativeness in OCO

The problem is formulated starting from a generic OCO framework (Shalev-Shwartz et al., 2011) in which a learning agent, at each round t, has to select a parameter $\theta_t \in \Theta$, representing a strategy, where $\Theta \subset \mathbb{R}^d$ is a closed and convex set of a finite d dimensional

Euclidean space.[23] At each round t, the agent receives a loss $f_t(\theta_t)$ where $f_t : \Theta \to [\varepsilon_l, \varepsilon_u]$ is a convex and differentiable function, and $\varepsilon_l, \varepsilon_u$ are the minimum and maximum value of the function $f_t(\cdot)$, respectively, and $0 \leq \varepsilon_l < \varepsilon_u$.[24] The objective of the learning agent $\mathfrak{U}$ is to minimize the regret $R_T(\mathfrak{U})$ over a given time horizon $T \in \mathbb{N}$.

In the COCO setting, we are interested in those algorithms $\mathfrak{U}$ for which the regret $R_T(\mathfrak{U})$ is bounded by a sublinear function of the time horizon T, and, at the same time, perform throughout the optimization at least as well as an established default parameter $\tilde{\theta} \in \Theta$, selected at the beginning of the learning process. While the former requirement represents the so-called no-regret property of an algorithm (Cesa-Bianchi and Lugosi, 2006), the latter one is formally defined as follows:

Definition 5.1 (Conservativeness in Online Learning)**.** *An online algorithm $\mathfrak{U}$ is said to be* conservative *if it satisfies the following conservativeness constraint for each $t \in [T]$:*

$$L_t \leq (1 + \alpha)\tilde{L}_t, \tag{5.11}$$

where $\alpha > 0$ is the conservativeness level required by the problem, and $\tilde{L}_t := \sum_{k=1}^{t} f_k(\tilde{\theta})$ is the cumulative loss of the default parameter $\tilde{\theta}$ over t rounds.[25,26]

From now on, we refer to the quantity $Z_t(\mathfrak{U}) := (1 + \alpha)\tilde{L}_t - L_t$ as the *budget* of the algorithm $\mathfrak{U}$, *i.e.,* the advantage in terms of loss accumulated by $\mathfrak{U}$ over time with respect to the one provided by a constant choice of the default parameter $\tilde{\theta}$. We also assume that there exists $\mu > \varepsilon_l$ s.t. $\tilde{L}_t \geq \mu t$, which imply that the fixed strategy $\tilde{\theta}$ is sub-optimal.

We remark that, in this work, we require the constraint in Equation (5.11) to be satisfied at each round $t \in [T]$. Indeed, any online learning algorithm $\mathfrak{U}$ providing a regret of $R_t(\mathfrak{U}) \leq \xi\sqrt{t}$ is guaranteed to satisfy the above constraint for $t > \left(\frac{\xi}{\alpha\mu}\right)^2$, instead we require that it holds for each $t \in [T]$.[27] Therefore, satisfying the condition imposed by our constraint requires the design of ad-hoc algorithms. Conversely, the design of algorithms with a higher grade of conservativeness, *i.e.,* $\alpha \leq 0$, is not a viable option due to the following:

Theorem 5.3. *In the OCO setting, there is no algorithm $\mathfrak{U}$ which obtains $L_t \leq \tilde{L}_t$, unless $\theta_t = \tilde{\theta}$ for all $t \in [T]$.*

In other words, it is impossible to guarantee that an algorithm does strictly better than or equal to a given default parameter $\tilde{\theta}$ over the entire time horizon T, unless one always plays the default parameter.

5.2.3 The Conservative Projection Algorithm

We begin this section by characterizing a set of parameters in the parameter space Θ that guarantees that their choice implies the conservativeness of an algorithm at round t.

[23] We will consider a generic OCO setting and then focus on OPO in Section 5.2.4.

[24] As a remark on the notation, in the OCO literature, loss functions are written as $f_t(\theta_t) \equiv f(\theta_t, y_t)$, dropping the explicit dependence on the outcome y_t.

[25] The conservativeness constraint in Equation (5.11) is expressed in terms of losses, as commonly done in the OCO framework.

[26] We denote with $[T]$ the set $\{1, \ldots, T\}$.

[27] This comes from the fact that $L_t - \tilde{L}_t \leq R_t(\mathfrak{U})$ and $\xi\sqrt{t} \leq \alpha\mu t$ holds for $t > \left(\frac{\xi}{\alpha\mu}\right)^2$.

Then, we select a specific parameter from this set, thus defining the CP algorithm, and, subsequently, we show it is conservative and has sublinear bounds for the regret.

The Conservative Ball

Let us define the following:

Definition 5.2 (Conservative Ball)**.** *A conservative ball $B(\tilde{\theta}, \omega_t) \in \mathbb{R}^d$ is a d-dimensional ball centered in $\tilde{\theta}$ with radius:*

$$\omega_t := \left[1 - \left(\frac{L_{t-1} - (1+\alpha)\tilde{L}_{t-1} - \alpha\varepsilon_l}{DG} + 1 \right)^+ \right] D, \tag{5.12}$$

where $D := \sup_{x,y \in \Theta} ||x - y||_2$ is a bound on the diameter of the parameter space Θ, $G := \sup_{x \in \Theta} ||\nabla f_t(x)||_2$ is the upper bound on the norm of the gradient of the loss $f_t(\cdot)$, $||\cdot||_2$ denotes the L2 norm of a vector, and $(a)^+$ denotes the maximum between the quantity a and zero.

From now on, we refer to this ball as the *conservative ball* (also depicted in Figure 5.4) since this choice of ω_t implies that playing any of the parameters $\theta \in B(\tilde{\theta}, \omega_t)$ at round t guarantees that the accrued budget $Z_t(\mathfrak{U})$ does not become negative. Formally:

Theorem 5.4. *Let $B(\tilde{\theta}, \omega_t)$ be the conservative ball defined in Equation* (5.12) *and assume that Equation* (5.11) *is satisfied at round $t-1$. Then, each parameter $\theta \in B(\tilde{\theta}, \omega_t) \cap \Theta$ satisfies Equation* (5.11) *at round t.*

Notice that the projection of a generic parameter z_t on the ball $B(\tilde{\theta}, \omega_t)$ can be computed analytically and efficiently. Indeed, the projection operation on the conservative ball satisfies the following:

$$\theta_t = \Pi_{B(\tilde{\theta}, \omega_t)}(z_t) = \beta_t \tilde{\theta} + (1 - \beta_t) z_t, \tag{5.13}$$

where

$$\beta_t = \begin{cases} 1 - \frac{\omega_t}{||z_t - \tilde{\theta}||_2} & z_t \notin B(\tilde{\theta}, \omega_t) \\ 0 & z_t \in B(\tilde{\theta}, \omega_t) \end{cases}. \tag{5.14}$$

In what follows, we choose z_t as the parameter provided by a generic OCO algorithm at round t.

Description of the CP Algorithm

Theorem 5.4 provides a way to choose a sequence of parameters over time, for which the conservativeness constraint is satisfied. The CP algorithm uses this result by choosing, at each round t, the parameter θ_t in the ball $B(\tilde{\theta}, \omega_t)$ as close as possible to the prediction z_t provided by the OCO algorithm fed using the pseudo-loss function $g_{t-1}(z_{t-1}) := (1 - \beta_{t-1}) f_{t-1}(z_{t-1})$, *i.e.,* it selects a convex combination of the default parameter $\tilde{\theta}$ and z_t. The intuition behind this choice is that we want to choose θ_t as close as possible to the no-regret prediction z_t of the OCO algorithm that is guaranteed to have sublinear regret. Furthermore, we show that this algorithm increases the radius ω_t over time, and therefore, in

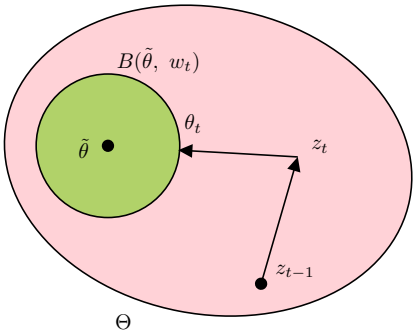

Figure 5.4: *Graphical representation of CP algorithm. In green the conservative ball $B(\tilde{\theta}, \omega_t)$, and in red the parameter set Θ. The CP algorithm selects the parameter θ_t for round t by projecting the parameter z_t, selected by $\mathcal{A}$, on the conservative ball.*

Algorithm 3: Conservative Projection Algorithm

1 **Initialize:** online learning algorithm $\mathcal{A}$, conservativeness level $\alpha > 0$, default parameter $\tilde{\theta} \in \Theta$
2 Set $\tilde{L}_0 \leftarrow 0$, $L_0 \leftarrow 0$, and $\beta_0 \leftarrow 1$
3 **for** $t \in [T]$ **do**
4 Get point z_t from $\mathcal{A}$ applied to loss $g_{t-1}(z_{t-1})$
5 Compute ω_t as in Equation (5.12)
6 Select $\theta_t = \Pi_{B(\tilde{\theta},\omega_t)}(z_t)$
7 Suffer loss $f_t(\theta_t)$
8 Observe $f_t(z_t)$ and $f_t(\tilde{\theta})$
9 Set $g_t(z_t) \leftarrow (1 - \beta_t) f_t(z_t)$

finite-time, the conservative ball includes the parameter z_t, allowing CP to have a sublinear regret. Finally, we remark that the CP algorithm is designed so that as the distance of the default parameter $\tilde{\theta}$ to the optimal one increases, the value of the radius ω_t increases, which, in turn, decreases the cost of guaranteeing conservativeness.

The pseudo-code of the CP algorithm is presented in Algorithm 3, and its visual representation is depicted in Figure 5.4. The algorithm requires as input a generic online learning algorithm $\mathcal{A}$, which selects the parameter z_t to play at each round t, a conservativeness level $\alpha > 0$, and the default parameter $\tilde{\theta} \in \Theta$. At first, we set the initial value of the cumulative losses $L_0 = 0$, that of the default parameter $\tilde{L}_0 = 0$ (Line 2), and we set the parameter $\beta_0 = 1$. Afterwards, at each round t, z_t is chosen by the algorithm $\mathcal{A}$ by considering the pseudo-loss $g_t(x)$ (Line 4). Thanks to a projection operation (Line 6), which projects z_t into the conservative ball $B(\tilde{\theta}, \omega_t)$, the resulting parameter θ_t satisfies the conservativeness constraint in Equation (5.11). Finally, the algorithm suffers the loss $f_t(\theta_t)$, and observes $f_t(z_t)$ and $f_t(\tilde{\theta})$, *i.e.,* the loss of the algorithm $\mathcal{A}$ and the default parameter $\tilde{\theta}$, respectively (Lines 8-9).

Notice that, from a computational point of view, the CP algorithm has a small computational overhead with respect to the original online learning algorithm $\mathcal{A}$, *i.e.,* an overhead

proportional to d, due to the additional projection on the conservative ball and the evaluation of the losses $f_t(\theta_t)$, and $f_t(\tilde{\theta})$.

Analysis of the CP Algorithm

In this section, we prove that the CP algorithm has the desired conservativeness property and maintains the sublinear regret of the subroutine algorithm $\mathcal{A}$. Since the CP algorithm selects a parameter θ_t inside the conservative ball $B(\tilde{\theta}, \omega_t)$, a straightforward corollary of Theorem 5.4 guarantees that the conservativeness constraint is satisfied. Formally:

Corollary 5.4. *The CP algorithm applied to a generic online learning algorithm $\mathcal{A}$ is conservative.*

Once we established the conservativeness of our approach, we need to prove that the CP algorithm has sublinear regret. Intuitively, we need to show that the radius ω_t grows over time, and eventually includes the entire space Θ, so that from a specific round we are allowed to follow the no-regret choice z_t. Formally, we show the following:

Theorem 5.5. *Consider any OCO algorithm $\mathcal{A}$ that guarantees a regret of $R_T(\mathcal{A}) \leq \xi\sqrt{T}$. The CP algorithm using $\mathcal{A}$ as subroutine has the following regret bound:*

$$R_T(CP) \leq \xi\sqrt{T} + \tau DG, \tag{5.15}$$

for any $T > \tau$, where:

$$\tau = \frac{2\alpha\mu(DG + \alpha\mu) + \xi\left(\sqrt{\xi^2 + 4\alpha\mu(DG + \alpha\mu)} + \xi\right)}{2\alpha^2\mu^2}. \tag{5.16}$$

A regret of order $\mathcal{O}(\sqrt{T})$ is tight in general OCO problems (Abernethy et al., 2008), but there exists specific settings in which a $\mathcal{O}(\log T)$ regret can be achieved, *e.g.,*, in the case of H-strongly convex losses or in the case of exp-concave losses (Hazan et al., 2007). In such settings, the CP algorithm guarantees $\mathcal{O}(\log T)$ regret together with the conservative constraint, formally:

Theorem 5.6. *Consider any OCO algorithm $\mathcal{A}$ that guarantees a regret of $R_T(\mathcal{A}) \leq \rho\log(T)$. The CP algorithm using $\mathcal{A}$ as subroutine has the following regret bound:*

$$R_T(CP) \leq \rho\log(T) + \tau DG, \tag{5.17}$$

for any $T > \tau$, where:

$$\tau := \frac{\alpha e^2\mu(DG + \alpha\mu) + 2\rho\left(\sqrt{\alpha e^2\mu(DG + \alpha\mu) + \rho^2} + \rho\right)}{e^2\alpha^2\mu^2}. \tag{5.18}$$

Notice that for Theorem 5.5 and 5.6 we have that $\tau \propto 1/\mu$, meaning that for default parameters $\tilde{\theta}$ with smaller accrued losses with respect to the optimum $\bar{\theta}$ (and hence smaller μ), the CP algorithm is required to wait longer to play the action prescribed by the no-regret strategy $\mathcal{A}$. Moreover, the bound shows a dependence $\tau \propto 1/\alpha$, meaning that a tighter conservative constraint makes the problem more challenging for the CP algorithm.

5.2.4 Experimental Results

In this section we present the experimental results of the CP algorithm applied to the OPO framework.[28] We compare our performances to OGD (Zinkevich, 2003), the non-conservative version of the proposed algorithm, the Conservative Switching (CS) algorithm, a naive conservative baseline, and the Constrained Reward Doubling Guess (CRDG). CS is a budget-first algorithm we designed. This algorithm plays the fixed default action until enough budget has been accrued, then it plays the no regret strategy. We describe CS and provide its theoretical properties in Appendix A.3.1. As for CP, in CS we consider OGD as subroutine and, thus, refer to it as CS-OGD. CRDG is a conservative baseline obtained by combining the Reward Doubling Guess algorithm (Streeter and McMahan, 2012), originally designed for unconstrained online optimization setting, with the *Constraint Set Reduction* procedure presented in Cutkosky and Orabona (2018). We provide also its detailed pseudo-code and a discussion on its theoretical properties in Appendix A.3.1.

In the experimental setting, we consider the portfolio allocation $\mathbf{a}$ instead of the parameter θ, recalling the definition of wealth $W_t(\mathfrak{U})$ of Equation (2.14) and the negative log loss function of Equation (5.1). We consider Assumption 5.1 from Section 5.1.2, which states that $y_{j,t} \in [\epsilon_l, \epsilon_u]^d \; \forall j \in \{1, \dots, M\}$, where $\epsilon_l = e^{-\varepsilon_l}$ and $\epsilon_u = e^{-\varepsilon_u}$.

In this setting, the default parameter $\tilde{\mathbf{a}} \in \Delta_{M-1}$ represents the index that an investor wants to outperform over the entire time horizon T. The conservativeness constraint formulation is defined as follows:

$$W_t(\mathfrak{U}) \geq (1-\kappa)\tilde{W}_t \; \forall t \in [T], \tag{5.19}$$

where $\tilde{W}_t$ the wealth gained by playing $\tilde{\mathbf{a}}$ over t steps, and $\kappa \in (0, 1)$ represents the conservativeness level in this context.

Notice that the log-loss used in the OPO case is not positive, therefore, the use of CP-OGD algorithm requires to shift the loss to positive values, *i.e.,* using the following loss function:

$$f_t(\mathbf{a}_t) = -\log(\langle \mathbf{a}_t, \mathbf{y}_t \rangle) + \varepsilon_l. \tag{5.20}$$

Results

We used a public dataset containing the components of the S&P described in Section 4.2. We selected 100 random stocks among the stocks that had a one-time-step return of maximum $\pm 4\%$ and chose uniformly from them the default strategy $\tilde{\theta}$. The conservative level κ has been set to $\kappa = 0.2$, the diameter in this setting is $D = \sqrt{2}$ and the gradient is bounded by $G = \frac{\varepsilon_u}{\varepsilon_l}$, where $\varepsilon_l = 0$ and $\varepsilon_u = \log(\epsilon_u) - \log(\epsilon_l)$ since we used the shifted loss defined in Equation (5.20). We used $\eta_t = \frac{K}{\sqrt{t}}$, with $K = \frac{D}{\sqrt{2}G}$, as learning rate for all the analysed algorithms. This choice for K minimizes the theoretical bound on the regret of the OGD algorithm.

We evaluated the algorithms in terms of wealth $W(\mathfrak{U})$ and in terms of wealth budget, defined as:

$$P_t(\mathfrak{U}) = W_t(\mathfrak{U}) - (1-\kappa)\tilde{W}_t.$$

[28] While an extensive experimental campaign was carried out in Bernasconi de Luca et al. (2021), in this thesis we concentrate on the results obtained in the OPO setting.

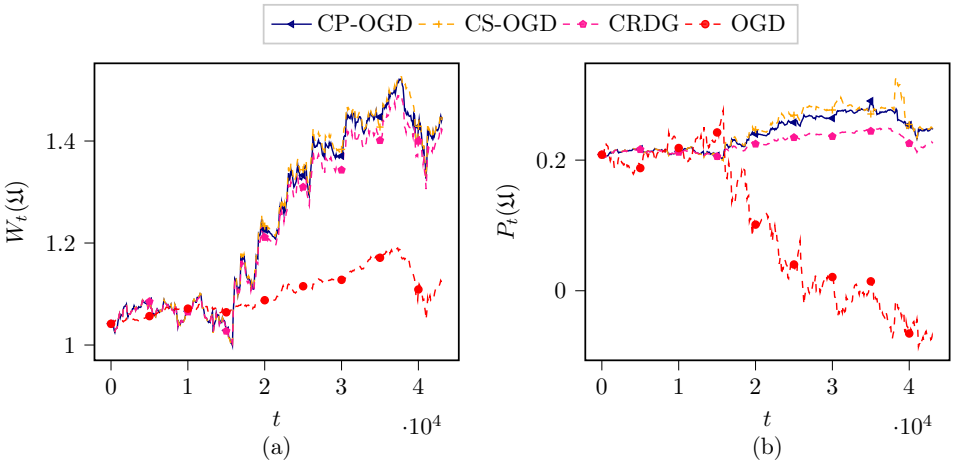

Figure 5.5: *Results on a specific run: (a) wealth $W_t(\mathfrak{U})$, (b) wealth budget $P_t(\mathfrak{U})$.*

The results of the experiment are presented in Figure 5.5. In Figure 5.5 (a) we can see that wealth of the investment strategy proposed by the CP-OGD and CS-OGD algorithms outperforms the OGD algorithm. This suggests that in some cases the information given by the default strategy can greatly help the performance. Moreover, the budget of the OGD algorithm presented in Figure 5.5 (b), does not satisfy the budget constraint of Equation (5.19), while the CP-OGD and CS-OGD satisfy the constraint $\forall t \in [T]$. This confirms the theoretical results provided. CRDG also performs well, but slightly worse than CP-OGD and CS-OGD.

Interpreting the Results It is not straightforward to interpret the resulting investment strategy in financial terms. In fact, the strategy has a mixed behavior caused by the combination of the subroutine algorithm and the CP algorithm. In this experimental section, OGD is the subroutine algorithm, which generates a trading strategy which can be related to trend following strategies, as we saw also in Section 5.1.5. The CP algorithm instead forces the portfolio weights to be close to the index weights, adjusting the constraint depending on how well the OGD strategy is behaving compared to the index. This means that the CP algorithm also has a trend following behavior, where the trend is not set by the individual assets but by the two strategies: the default index and the subroutine algorithm.

5.3 Chapter Summary

In this chapter we analyzed the Online Portfolio Optimization framework. Section 5.1 focused on controlling transaction costs and Section 5.2 focused on the problem of conservative optimization.

Transaction Costs in OPO In Section 5.1, we started by defining the framework and introducing total regret, a regret formulation that includes transaction costs. We then introduced a novel algorithm, namely OGDM, to control, theoretically and empirically, the

costs. We proved that OGDM is capable of achieving a total regret of $\mathcal{O}(\sqrt{T})$. Moreover, we verified the analytical results through an extensive experimental campaign. The experiments demonstrated the superior behavior of OGDM in the presence of costs with respect to state-of-the-art OPO algorithms, and OGDM's adherence with the proved theoretical guarantees.

Future developments could be to extend the bound on the transaction costs to a wider class of algorithms, *e.g.,*, the ones derived from Online Mirror Descent (OMD). Furthermore, it would be interesting to extend the transaction cost model to include liquidity constraints and market impact, combining with the optimal execution approach of Chapter 9.

Beating a Known Benchmark In Section 5.2, we started by formulating the conservativeness constraint, which denotes the maximum "distance" with respect to the benchmark in consideration. In order to comply with this constraint, we proposed the CP algorithm, a "wrapper" that can be applied to any existing OCO algorithm. We proved that CP maintains the same regret order of the OCO algorithm it uses as subroutine while satisfying the conservativeness property. Finally, we confirmed the theoretical results through the experiments on the OPO framework, comparing with state-of-the-art algorithms.

An interesting direction is whether the assumption that the default strategy $\tilde{\theta}$ is fixed can be relaxed to include specific classes of time-varying strategies.

CHAPTER 6

Quantitative Trading with FQI and MCTS

The quantitative traders introduced in Section 2.3.2 try to find patterns, trends, and inefficiencies in the price process of a large number of instruments by using statistical tools. These potential arbitrages are usually very small and last for a limited amount of time, thus it is necessary to have a systematic approach and use leverage. The resulting gain process has no correlation with the general market movements. This is different from the portfolio optimization techniques seen in Chapter 5, where the objective is to have exposure to the general market, but to perform better than the latter by changing the portfolio weights. While having a clear distinction in this thesis, practitioners often mix the two approaches to diversify their investing strategies.

Given the aggressive and frequent actions (long, short, flat) we consider in the quantitative trading approach of this chapter, carefully modeling transaction costs becomes of fundamental importance. Thus, recalling Lemma 3.1, RL or online planning techniques are more appropriate in this scenario compared to the online learning approaches of the previous chapter. This chapter addresses the negative impact of transaction costs on the trading strategy. The approach adopted could be used together with an optimal execution approach, such as that presented in Chapter 9 to reduce costs.

The use of RL in trading has received increasingly more attention for its goal being well aligned with trading objectives (Fischer, 2018; Meng and Khushi, 2019; Bacoyannis et al., 2018). The first applications to trading using recurrent RL have shown promising results (Moody and Saffell, 2001; Gold, 2003). Later works have confirmed this encouraging direction in a variety of contexts, including high frequency trading using order book information (Briola et al., 2021) with Proximal Policy Optimization (Schulman et al., 2017)

or stock trading using OHLCV (open, high, low, close, and volume) data (Théate and Ernst, 2021) with Deep Q Network (DQN) (Mnih et al., 2013).

In this chapter, we focus initially on the use of FQI to learn a quantitative trading strategy considering the FX pairs EURUSD and GBPUSD. Then, in Section 6.3, we propose the use of online planning, specifically MCTS, to trade the EURUSD pair.

Chapter outline The chapter begins with an overview of the state-of-the-art regarding trading using RL. It is then divided in two parts, Section 6.2 starts by describing FQI, focusing on its use to learn a quantitative trading strategy with two correlated Foreign eXchange (FX) pairs in Section 6.2.3. The experimental campaign is described in Section 6.2.4, where we focus on hyper-parameter tuning and changing the persistence of the actions to optimize performance. Section 6.3 starts by defining the Open Loop extension that allows MCTS to work in problems with stochastic state transitions. In Section 6.3.1 we describe the algorithm used to tackle the trading problem, including the novel generative model, followed by the experimental results.

6.1 Background on RL for Trading

There is growing literature that experiments the use of RL for trading, while, to our knowledge this is the first attempt in using MCTS. We concentrate on the papers that address topics similar to what is presented in this work, starting with articles on FX trading, then on multi asset-trading.

RL for FX Trading Popular approaches to FX trading include recurrent RL considering several currency pairs (Gold, 2003), Q-learning for GBPUSD (Dempster et al., 2001), DQN on EURUSD and USDJPY (Sornmayura, 2019), DQN on EURUSD (Carapuço et al., 2018), FQI on EURUSD (Bisi et al., 2020a), and DQN on 12 currency pairs (individually) (Huang, 2018). RL for FX trading is extremely promising, as highlighted by Sornmayura (2019) whose agent outperforms an experienced trader when considering the EURUSD pair.

RL for Multi-asset Trading. To the best of our knowledge, using RL to trade simultaneously more than one currency pair has not been evaluated on FX. Considering other asset classes, Jiang et al. (2017) applied Deep Deterministic Policy Gradient (Lillicrap et al., 2015) using past price information to manage a portfolio of 12 crypto-currencies with a trading frequency of 30 minutes. Adopting an approach similar to Jiang et al. (2017), Alonso and Srivastava (2020) considered the daily re-balancing of 24 US listed stocks. Other works on multi-asset investment include Jangmin et al. (2006), whose algorithm learns a meta-policy with Q-learning to select, among a set of traders, which trader's allocation proposition to follow. Hongyang et al. (2020) examined the daily trading of 30 stocks included in the Dow Jones. Interestingly, the authors consider an ensemble approach and select for the next testing period the algorithm that obtained the best Sharpe ratio in the previous periods.

6.2 Learning to Trade with FQI

FQI is a value based, batch and off-policy RL algorithm as mentioned in Section 3.3.1. Starting from the notations and definitions of Chapter 3, we describe the Fitted Q Iteration algorithm defined by Ernst et al. (2005) and then explain how it can be used to learn a quantitative trading policy in Section 6.2.3.

6.2.1 Fitted Q Iteration

Recalling the Bellman equation for the Q-function (Equation 3.6), we can define the Bellman operator T^π associated to a policy π as:

$$(\mathcal{T}^\pi Q)(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{\substack{s'\sim\mathcal{P}(\cdot|s,a)\\ a'\sim\pi(\cdot|s')}} \big[Q(s',a')\big].$$

The Q-function and the Bellman operator are linked in that Q_π is a fixed point of $\mathcal{T}^\pi$. Notably, the result holds also for the optimal policy π^* that gives the fixed point of the optimal Bellman operator $\mathcal{T}^*$:

$$(\mathcal{T}^* Q)(s,a) = r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)} \left[\max_{a'\in\mathcal{A}} Q(s',a')\right]. \tag{6.1}$$

By the Banach-Caccioppoli fixed point Theorem (Banach, 1922), with $\gamma \in [0,1)$ and a finite or infinite horizon, we can thus obtain $Q^* = Q_\pi$ (where $Q^* = Q_{\pi^*}$) starting from any Q-function by recursively applying the optimal Bellman operator. In this chapter, we are in a receding horizon case with a finite number of FQI iterations, thus the undiscounted setting also holds. (Chang and Marcus, 2003).

FQI (Ernst et al., 2005) uses the optimal Bellman operator and Supervised Learning techniques to generalize the knowledge obtained from the training samples, extending the information to unseen samples in the state and action space. To train using FQI, it is necessary to create a dataset $\mathcal{D}$ of 4-tuples (s,a,r,s') containing a state s, an action a, the resulting reward $r(s,a,s')$, and next state s'. It is preferable to span as much of the state-action space as possible to minimize the interpolation task. This dataset can be created before beginning the training and can be used without updates, since FQI is an offline algorithm.

Algorithm 4: Fitted Q Iteration Algorithm

1 **Initialize:** $\hat{Q}_0(s,a) \leftarrow 0\ \forall s \in \mathcal{S}, a \in \mathcal{A}$, number of iterations J, inverse temperature parameter τ and load dataset $\mathcal{D}$

2 **for** $j \in [J]$ **do**

3 $\quad \hat{Q}_{j+1} = \arg\min_{f\in\mathcal{F}} \sum_{s,a,r,s'\in D} \left(f(s,a) - r - \gamma \max_{a\in\mathcal{A}} \hat{Q}_j(s',a)\right)^2$

4 Extract $\pi(s) = \frac{\exp(\tau \hat{Q}_J(s,\cdot))}{\sum_{a\in\mathcal{A}} \exp(\tau \hat{Q}_J(s,a))}\ \forall s \in \mathcal{S}$

5 Return π

Indeed, the goal of the FQI algorithm is to approximate the Q-function $Q(s, a)$ by means of a regressor $\hat{Q}(s, a)$ that exploits the dataset $\mathcal{D}$.[29] FQI is described in Algorithm 4 where, starting in Line 1, we initialize for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ the approximated state-action function $\hat{Q}_0(s, a)$ to zero, we initialize τ the inverse temperature parameter to be used in a Boltzmann distribution and load the dataset. Subsequently, the algorithm solves a regression problem having as target $r + \gamma \max_{a \in \mathcal{A}} \hat{Q}_j(s', a)$ (Line 3), *i.e.*, the currently available estimate of the value of the action a in state s. Depending on the computational complexity required by this step and the characteristics of the environment, one might select a different regressor, e.g., Support Vector Machines or Neural Networks - see Antos et al. (2007) for details. Finally, we extract the policy from the approximate Q-function in Line 4.

Two conflicting phenomena appear when training FQI. On the one hand, the higher the number of iterations, the more future outcomes are taken into account in the computation of the approximate Q-function, as at each iteration,the horizon considered increases by one step. On the other hand, the regression of the Q-function introduces errors, which are increased through the iterations, preventing the Q-function from converging. Thus, a trade-off is usually observed in the number of iterations.

6.2.2 Persistent Actions

Due to the diversity of market participants and investment strategies (Mantegna and Stanley, 1999; Bouchaud and Potters, 2003), it is reasonable to consider that the market is built upon different time scales. This property is comprehended inside the Adaptive Market Hypothesis (AMH) (Lo, 2019; Di Matteo et al., 2003), which extends the Efficient Market Hypothesis (EMH) to add the possibility that investors adapt their investment decisions based on new information. The EMH states that asset prices reflect all the currently available information, and thus investors and so the markets are rational. Instead the AMH states that the EMH may not always be true, as investors can become irrational in response to heightened market volatility. The AMH is often tested through the lens of multifractal analysis (Lopes and Betrouni, 2009) to study the scaling laws of financial time series. If a timeseries exhibits a multifractal behavior, then it does not have a characteristic scale. This implies that whatever the trading horizon, the scaled opportunities will be the same. In particular, experiments suggest that such is the case for the FX markets (Corazza and Malliaris, 2002; Garcin, 2019).

In the field of RL for trading, the study of the impact of the trading time scale has not been a primary focus. Most works do not consider changing the frequency of interaction with the environment even though the variety of time scales across papers ranges from high-frequency to daily to even longer periods. We note, however, that Pendharkar and Cusatis (2018) compares quarterly, semi-annual, and annual frequencies and finds the latter to offer the best performance. This can be partly explained because the hyper-parameters have been tuned for the annual frequency, yet the author also suggests that it could be due to the different probability distribution of the returns, which would favor riskier but more profitable assets. Some previous works highlight the effect of assumptions on trading frequency, such as transaction costs (Elder, 2008), or the agent's risk aversion (Bisi et al., 2020a). In these works, the authors do not change the trading frequency, but the agent learns by itself to act with a lower frequency.

[29] The typical regressor is extra trees, see Appendix B.4.2 for details.

In RL, continuous time control problems are typically addressed by means of time discretization inducing a certain control frequency. On the one hand, a higher control frequency, especially in trading and finance, gives the agent more control opportunities; on the other hand, it shows several drawbacks, such as an increase in sample complexity due to the reduced effects of the single actions. Moreover, as specified in Section 6.2.1, an increasing number of FQI iterations leads to a larger planning horizon and a propagation of regression errors. Hence, there is a trade-off between the possibility to detect immediate opportunities and the learning capabilities. Metelli et al. (2020) introduces the idea of action persistence, which consists in the repetition of each individual action for a number of consecutive steps. Given a discrete-time MDP with initial state distribution μ, $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu \rangle$ modeled at the highest possible control frequency, the persistence can be seen as an environmental parameter k that can be configured to generate a family of related decision processes $\mathcal{M}_k = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}_k, \mathcal{R}_k, \gamma^k, \mu \rangle$ in which, whenever an action is issued, the resulting transition lasts for k time steps, with all the one-step rewards collected (with discount) in the new distribution $\mathcal{R}_k$. In the following sections we analyze, experimentally, the effect of changing the action persistence on the performance of the trading strategy.

6.2.3 Using FQI for FX Trading

In this section, we focus on defining the task in the case of a single FX currency pair, adding details to what is described in Definition 2.6, and afterwards extend to trading with two currency pairs.

One Currency Pair FX Trading In the FX market, it is important to define a *domestic* or *base* currency and a *foreign* currency, keeping in mind that we aim at maximizing the profits obtained in the domestic currency. Two options are then viable: trading a fixed quantity of the foreign currency for a variable amount of the domestic, or, vice versa, trading a variable amount of foreign currency for some fixed amount of the base one. We are interested in the second case since, from a financial point of view, this allows an easier characterization of the risk. We assume our base currency is USD. With this formulation, it is possible to see that this corresponds to treating the base currency as an asset, where in place of the price we have the instantaneous exchange rate. As a result, rewards are expressed in the foreign currency. However, to have an effective evaluation of the agent's performance, it is desirable to express the returns in the same currency. Thus, we convert on a daily basis the collected rewards in the domestic currency. While considering transaction costs, following Equation (2.22), we assume there is no market impact. The considered episodes are one business day long, with 1-minute time-steps, hence, we use the undiscounted setting (i.e. we set $\gamma = 1$). Thus, to summarize, the MDP is defined as:

- **State** The state contains a window of the last M observed prices. In our experiments, the decisions are taken every minute, so we use a window of 60 prices, *i.e.,* an hour of observations. This window is used to incorporate market trends in the state, as including only the current price would make the state non-Markovian. Since we consider trading in a finite horizon of length H, we include in the state also the current time-step $t \in [0, H]$ and the portfolio position $a_{t-1} \in \{-1, 0, 1\}$ of the previous time-step.

- **Action** We consider a discrete action space where the action at time t, a_t is the portfolio position the agent will hold, so -1 indicates buying 1 USD and selling the equivalent amount of EUR and buying the equivalent amount of USD, 0 indicates not holding any exposure, $+1$ indicates selling 1 USD and buying the equivalent amount of EUR. Each action is characterized by the same size of unitary amount.

- **Reward** Given the current portfolio position a_{t-1}, the action taken a_t, and the prices, the reward is defined as Equation (2.17):

$$r_{t+1} = \underbrace{a_t(P_{t+1} - P_t)}_{\text{market movement}} - \underbrace{\frac{\text{bid-ask}}{2}|a_t - a_{t-1}|}_{\text{transaction costs}}.$$

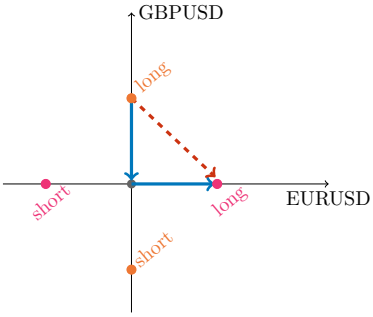

Figure 6.1: *Graphical representation of two FX pairs model. USD is considered as domestic currency. The dots (including the origin) represent the possible portfolio positions. To switch from a long position in the GBPUSD pair to a long position in the EURUSD pair, the agent needs to pay twice the transaction fees, both for closing one position and opening the new one (blue path).*

Two Currency Pairs FX Trading We consider two currency pairs EURUSD and GBPUSD (thus three currencies), with the same domestic currency (USD). To simplify the model and reduce the size of the action space, we do not allow positions involving simultaneous allocations on different foreign currencies (see Figure 6.1). This is not restrictive since we are pursuing a risk-neutral objective. In practice, this means that it is only possible to be long/short with respect to to one pair at each time-step. Therefore, we allow the agent to take 5 possible positions that correspond to being long (or short) with respect to each of the foreign currencies, or to being flat with respect to both, as shown in Figure 6.1. The agent can switch from a currency pair to the other one in just one step. However, this transition is considered as the composition of two operations: the transition to the flat allocation, and from flat to the final position. Consequently, such operations involve twice the transaction costs. The state has the same features as described above but includes also the prices of the additional currency pair. We expect the three-currency scenario to be more profitable since the agent, at each time-step, has more instruments to choose amongst, and hence, it may exploit additional trading opportunities.

In this section, we describe the results achieved both in a multi-currency and single-currency framework. We consider real data from EURUSD and USDGBP pairs from 2017

to 2020 downloaded from HistData.com (as described in Section 4.2). A fixed \$100K allocation was considered and a fixed bid-ask spread of $\$2 \cdot 10^{-5}$. Performances are shown as percentages of the invested amount and annualized.

Dataset Generation

As explained in Section 6.2.1, the FQI training set is composed of a series of tuples, each of which contains the current state, the action of the agent, the resulting next state, and reward. To build the dataset, starting from the collected market prices spanning over 24 hours, we filtered the data to focus on the European trading time window (from 8:00am to 6:00pm CET).[30] We then added the 60 consecutive price variations and time of the day to each state. Finally, we associated to each pair (s, s') all the possible portfolio-action configurations and the correspondent rewards, computed using Equation 2.17.

Model Selection

To select the best FQI model, it is necessary to tune both the hyper-parameters related to the extra tree regressors (see Appendix B.4.2 for details on extra trees) and the ones that characterize the general training algorithm.

Based on our experience and following what is suggested in Geurts et al. (2006), given a reasonable number of trees that guarantee a good trade-off between high computational time and low variance of the estimation, only the *min-split* has to be tuned to regulate the model complexity. Typically, the higher the min-split threshold is, the simpler is the trained model, because trees are forced to use a greater number of samples to perform a split, and hence complicated patterns are excluded. However, a low min-split threshold allows for more complex models, although it also increases the risk of overfitting the training data.

The training algorithm, instead, is characterized only by the number of iterations. As the number of iterations grows, the optimized horizon increases, allowing the model to learn longer-term patterns. Nevertheless, as mentioned in Section 6.2.1, iterating the Q-function approximation procedure leads to the propagation of the noise contained in the data. Therefore, we have to deal with the trade-off between extending the optimization horizon and propagating approximation errors through iterations.

6.2.4 Experimental Results

We analyze the impact of the persistence by testing three different sampling frequencies: 1-minute, 5-minute, and 10-minute. For each frequency, we trained a model using two different min-split thresholds and 10 FQI iterations. Moreover, to take into account the randomness of extra trees regressors, we performed 2 different training and validation runs for each set of hyperparameters. We trained on data from the years 2017 and 2018, validated using 2019, to select the best min-split and iteration number, and tested on 2020. An example of this validation procedure is shown in Figure 6.2 for the EURUSD case for each considered persistence. The procedure gives 7 FQI iterations as the optimal choice for the 1-minute persistence, consisting thus in an optimization horizon of 7 minutes. In

[30] This choice is motivated by the fact that two out of the three considered currencies are European, and that this is the time with the highest traded volumes; the remaining time has been excluded for the lower trading volumes, to make the approach more consistent and robust.

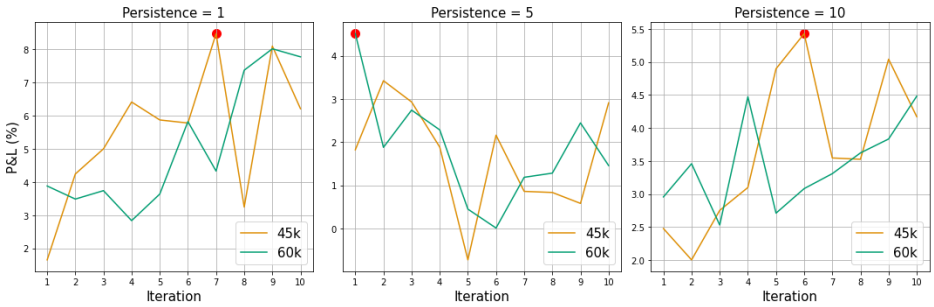

Figure 6.2: *Annualized percentage P&L on single currency pair EURUSD on validation set. Each figure has a different persistence (1-minute, 5-minute, and 10-minute). The performance, averaged over two seeds, is reported for each FQI iteration, and for two different values of the min-split parameter (45k and 60k). The selected models are highlighted with a red dot. Performances are reported as annualized percentages with respect to the invested amount.*

	Pers	MS	Ite	P&L (%)	Sharpe ratio	MDD (%)
EUR	1	45k	7	-1.45 ± 1.16	-0.22	9.28
	5	**60k**	**1**	$\mathbf{6.91 \pm 2.63}$	**1.34**	**4.83**
	10	45k	6	1.65 ± 1.99	0.27	6.16
GBP	1	45k	1	-11.30 ± 3.05	-1.37	14.89
	5	**45k**	**2**	$\mathbf{14.29 \pm 4.65}$	**1.93**	**7.66**
	10	45k	9	6.43 ± 1.57	0.63	10.54
Both	1	75k	3	-10.12 ± 3.64	-1.45	15.63
	5	**60k**	**1**	$\mathbf{14.83 \pm 7.34}$	**2.02**	**7.96**
	10	75k	7	3.00 ± 3.40	0.33	11.07

Table 6.1: *Annualized percentage P&L on testing set of the selected models averaged over 5 runs with the corresponding standard deviation. The measures are P&L, Sharpe ratio, and MDD (defined in Section 4.5) as a percentage of the allocation. min-split=MS and iteration=Ite.*

the 5-minute persistence case the optimal choice is 1 iteration, signifying a horizon of 5 minutes. Finally, in the 10-minute persistence, the optimal iteration gives an optimization horizon of 60 minutes.

Results

Table 6.1 represents the performance on the testing set (year 2020) of the models that obtained the best performance in the validation set, where Figure 6.2 exemplifies validation in the EURUSD case. To interpret the results of Table 6.1, it is useful to look at the Sharpe ratio: in general, a Sharpe ratio greater than 1 is acceptable to investors, thus, in this case, it corresponds to the strategies with a persistence of 5 minutes. Models trained with a persistence of 1-minute are characterized by worse performance both in terms of

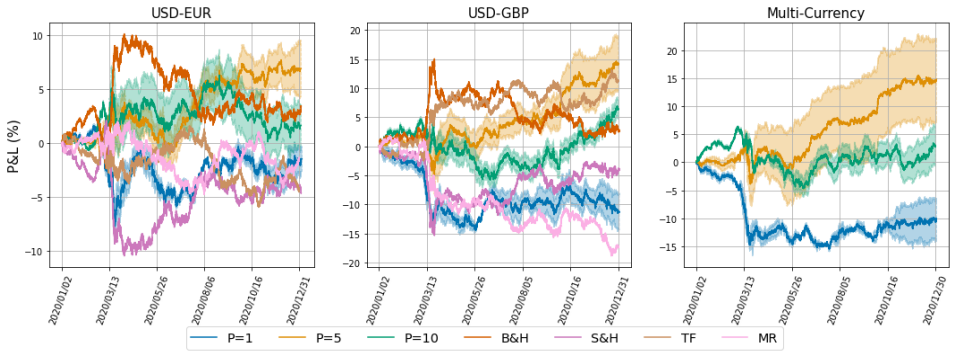

Figure 6.3: *Percentage P&L on testing set of the selected models averaged over 5 runs with confidence intervals. Buy and Hold (B&H), Sell and Hold (S&H), Trend Following (TF) and Mean Reverting (MR) baselines are included in the single asset case. Performances are reported as percentages of the invested amount.*

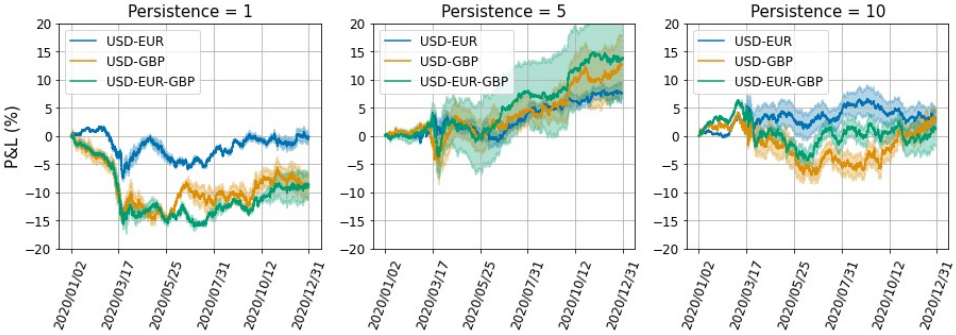

Figure 6.4: *Percentage P&L on testing set of the selected models averaged over 5 runs with confidence intervals. Divided in terms of persistence. Performances are reported as percentages of the invested amount.*

cumulated yearly P&L and Sharpe ratio. The same models of Table 6.1 are shown in Figure 6.3 where we see the cumulated P&L through time. This figure also includes four benchmark strategies: Buy and Hold (B&H), Sell and Hold (S&H), trend following, and mean reverting. B&H and S&H are passive strategies that consist in keeping a constant position, respectively, long or short. The trend following strategy goes long ($a_t = 1$) when the 50 minute moving average crosses above the 200 minute moving average and goes short ($a_t = -1$) when it crosses below. The mean reverting strategy does the opposite, going short when the 50 minute moving average crosses above the 200 minute moving average and going long when it crosses below. Both the FQI strategies and the benchmark strategies trade between 8:00am and 6:00pm CET, closing any position at the end of the trading day.

Figure 6.4 also contains the same models of Table 6.1, but subdivided for the three persistence values when trading multi-currency with respect to trading only one FX pair. We can see that the 5-minute persistence three-currency model guarantees the highest

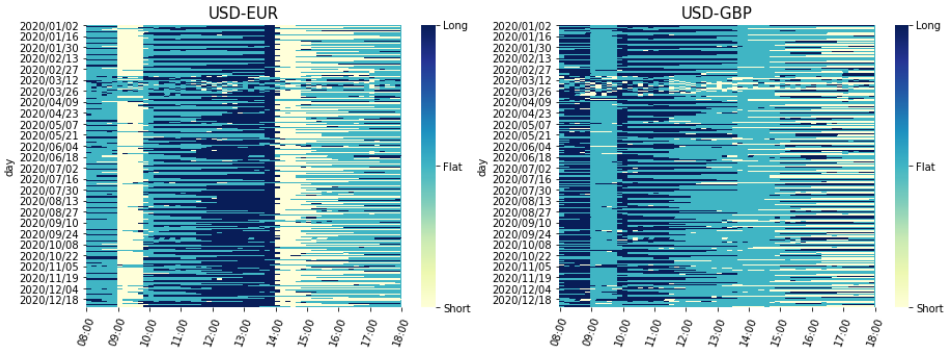

Figure 6.5: *Visualization of trades on test set in the multi-currency scenario, 10-minute persistence, FQI iteration 5, and min-split 75k. Each row corresponds to a different business day, with the time of day on the columns.*

cumulative returns at the end of the 2020, even if it does not consistently outperform the single FX pair models through the whole year. Again, the poor performances of the models with persistence equal to 1-minute can be explained by the worse signal-to-noise ratio that is present using the base interaction frequency. A higher persistence gives computational advantages, in fact, for the same number of iterations, the optimization horizon becomes longer as the persistence increases. Moreover, we also noticed that the mean time per iteration decreases with the persistence, even if the sample size of the FQI training set is the same. By inspecting the resulting models, we found out that the regression trees obtained with persistence 1-minute are characterized by almost double the number of nodes and leaves with respect to the ones with higher persistence. This may be due to the impact of the noise embedded in the data, which increases with higher frequencies.

We can analyze the learnt policy by considering Figure 6.5, which shows the policy of the best agent with persistence 10-minutes on the test set in the multi-currency scenario. Observing the patterns (vertical stripes of the same color in the allocation heatmaps), it is possible to see that the agent tends to go long with respect to USDGBP during the first hour of most days, then it changes the portfolio allocation moving to a short position with respect to EURUSD and keeps it until 10:00am. Some of these patterns may be associated with particular events and/or actions of financial institutions that are repeated each trading day e.g., when American traders enter the FX market around 2:00pm the agent usually changes its position with respect to EURUSD from long to short.

Finally, it is worth noting that the performances of all the models, both in the two-currencies setting and in the three-currencies one, are strongly affected by multiple drawdowns registered between March and May 2020, which might be related to the high volatility and unpredictability of the FX market due to the spread of the Covid-19 pandemic. The impact of the pandemic can also be observed by looking at the portfolio allocations displayed in Figure 6.5, where one can easily notice how the solid temporal patterns learned by agent do not hold during the whole month of March, when the pandemic exploded at global level. Nevertheless, higher persistence models were able to recover from the drawdown, ending up with a positive cumulated return.

Interpreting the Results To better understand the trading strategy obtained, it is interesting to focus on Figure 6.5. From this figure, it is clear that there is a daily recurrence, which hints that there is a strong dependence on the time feature in the state. Nevertheless, the change in behavior seen in March and May 2020 suggests that time is not the only feature of interest, but market behavior influences the policy as well. This observation is enforced from the fact that the patterns are not so well defined even during the other months of the year. The trends observed are probably given by recurrent behaviors of large financial institutions which repeat the same action each day at the same time, thus causing a small but recurring price change.

It is also interesting to see that the same action is generally held for a longer period of time than the 10 minutes time-step, this is the effect of the transaction costs, in fact from other experiments, with lower cost the agent tends to trade very frequently, while as we increase transaction costs the frequency decreases until it completely stops trading. This means that the algorithm learns to estimate if the trade is convenient given the transaction cost.

6.3 Trading with MCTS

In this section, we present an alternative approach to learning a trading strategy, specifically by using MCTS instead of FQI. As we have seen in Section 6.2, batch RL algorithms such as FQI optimize the policy by creating a fixed dataset of experience. This means that, in the continuously changing world of financial markets, batch RL algorithms suffer from the non-stationarity of the price processes, requiring to update the policy. Instead, as specified in Section 3.3.2, online planning algorithms use a 1-step model of the environment to construct a search-tree that evaluates the different available actions. This makes online planning more robust to non-stationary environments, since when the environment changes, it suffices to update the generative models used during the search – in general it is less expensive than updating the policy maintained by RL algorithms. For these reasons, we believe online planning offers greater flexibility when dealing with the continuously changing market environment. Indeed, it allows to easily handle changes in the volatility, in the bid ask spread, and in market impact.

The MCTS algorithm proposed is a variant of UCT (Lecarpentier et al., 2018), the latter described in Section 3.3.2. UCT has been originally designed for application in finite sequential decision-making models: the state transition model is deterministic and the action space is discrete and finite. As we have seen, in the trading context, the state space is continuous with stochastic state transitions *i.e.* given a state-action pair, there is high uncertainty about the possible next state. To deal with the stochastic states, we propose an *open-loop* variant of UCT. Furthermore, we propose a novel backup procedure for the MCTS algorithms, which uses Q-Learning Temporal Difference (TD) (Sutton and Barto, 2018) updates to address the high variance of the returns observed in the nodes of the search-tree. We propose a novel generative model that uses past observations of the assets of interest to generate possible future realizations of the market to be employed during planning to search for the optimal trading strategy. Finally, we perform an evaluation of the proposed algorithm and generative model on real financial data.

6.3.1 The Open Loop Q-Learning UCT Algorithm

In this section, we present the planning algorithm used in this work. To tackle the continuous state space with stochastic transitions, we resort to an open-loop approach that looks for the optimal sequence of actions to apply to the environment. Even though the planning phase is conducted in an open-loop fashion, only the first action identified by the planning procedure is applied, since, in our online framework, planning is interleaved with acting in the environment. The alternative to an open-loop setting is the application of Progressive Widening (PW) (Couetoux, 2013) to the state space, but this comes with higher memory costs as well as a higher planning budget needed to represent the (approximate) full search tree.

Open Loop Planning The open-loop planning approach is adopted in problems with continuous state spaces and stochastic transition models since the true search tree is infinite. In this setting, the problem consists in finding the optimal sequence of actions to be employed at the root state of the tree, without considering the states visited during the search, transforming the infinite search tree of the original problem into a finite tree with branch factor equal to the number of actions. Thus, it is necessary to adapt the definitions of Section 3.2 to the open loop setting. Formally, given a starting state $s \in \mathcal{S}$ and a sequence of actions of length m: $\tau = (a_1, \dots, a_m)$, $a_i \in \mathcal{A}, \quad i = 1, \dots, m$, we define the open-loop value of the sequence τ starting from state s as the discounted sum of the rewards collected executing the complete sequence τ starting from state s:

$$V_{OL}(s, \tau) = \mathbb{E}_{\pi}\left[\sum_{t=1}^{m} \gamma^t r_{t+1} \Big| s_0 = s, a_t \in \tau\right].$$

The sequence length m can be infinite when $\gamma < 1$. The optimal open-loop value function is the maximizer over the possible sequences $\tau \in \mathcal{A}^m$: $V^*_{OL}(s) = \max_{\tau \in \mathcal{A}^m} V_{OL}(s, \tau)$. Similarly, we define the optimal open-loop state-action value function for each state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ as the maximizer of the open-loop value over all the possible sequences of actions starting with a, $\tau_a \in \left\{(a, \tau) \Big| \tau = a_1, \dots, a_{m-1}\right\}$: $Q^*_{OL}(s, a) = \max_{\tau_a} V_{OL}(s, \tau_a)$.

Defining Nodes We denote by $\mathcal{N}_{d,i}$ the i-th node at depth d for $i \in \mathbb{N}$ and $d = 1, \dots, D$. The root node $\mathcal{N}_{0,0}$ contains a single state, $s_0 \in \mathcal{S}$, from which we want to perform planning. Each node $\mathcal{N}_{d,i}$, with $d > 0$, represents the collection of states which can be reached by a defined sequence of actions that starts from the root of the tree and is of length d. Specifically, given a node $\mathcal{N}_{d,i}$ and the sequence of actions that identifies it $\tau_{d,i} = (a_{1,i}, \dots, a_{d,i})$, the possible states observed in the node $\mathcal{N}_{d,i}$ represent the state distributions induced by executing $\tau_{d,i}$ starting from the root state s_0.

We define the open-loop state value function of a node $\mathcal{N}_{d,i}$ as:

$$\mathcal{V}^*(\mathcal{N}_{d,i}) = \mathop{\mathbb{E}}_{s \sim \mathcal{P}(\cdot | s_0, \tau_{d,i})}\left[V^*_{OL}(s)\right],$$

and the state-action value function as:

$$\mathcal{Q}^*(\mathcal{N}_{d,i}, a) = \mathop{\mathbb{E}}_{s \sim \mathcal{P}(\cdot | s_0, \tau_{d,i})}\left[Q^*_{OL}(s, a)\right],$$

$$= \mathop{\mathbb{E}}_{s \sim \mathcal{P}(\cdot|s_0, \tau_{d,i})} [r(s,a)] + \gamma \mathcal{V}^* \left(\mathcal{N}_{d+1,j} \right),$$

The goal of our proposed planner is to estimate the optimal open-loop action values at the root node by applying a UCT-like selection policy that selects, in each node, the action that maximizes the upper bound of the $\mathcal{Q}$ values according to UCB1 of Equation (3.10).

Q-Learning Backup Operator Our second change of the base UCT algorithm is the backup operator employed in the back-propagation phase. During the tree expansion, exploitation, and exploration is interleaved thanks to the selection rule of UCB1 (see Equation (3.10)), meaning that the values observed in each node come from very different policies. Also, in the simulation phase, a suboptimal rollout policy is employed to give an initial evaluation of each node. This rollout policy is clearly suboptimal (if we had an optimal policy for the rollout we would not need to perform planning) adding further noisy samples being backed-up. Both of these factors make the backup values observed extremely noisy, which is a further problem in our financial settings. For these reasons, instead of the plain Monte Carlo updates, that average the return values observed in each node in the tree, we employ a Temporal Difference update, based on the Q-Learning update rule, which, similarly to Equation (3.8), is defined as follows:

$$\mathcal{Q}_t \left(\mathcal{N}_{d,i}, a \right) = (1 - \alpha_t) \mathcal{Q}_t \left(\mathcal{N}_{d,i}, a \right) + \alpha_t \left(r_t + \gamma \max_{a'} \mathcal{Q}_t \left(\mathcal{N}_{d+1,j}, a' \right) \right),$$

where r_t is the reward observed in the current search pass at node $\mathcal{N}_{d,i}$ and α_t is the learning rate employed, which constitutes an added hyperparameter of our planner.

Q-Learning Open Loop Planning We present the pseudocode of our planner in Algorithm 5. This planner is devised to be employed in each decision interval with a given planning budget, specified as the environment transition samples from the model. At each search iteration, we perform the selection phase, plain UCB1, until a leaf of the tree is reached (described in Lines 7 through 13). We then perform a rollout from the leaf in Line 4. The specific rollout policies employed in both scenarios are described in the following sections. The rollout gives us an initial estimate of the node value. Next, we recursively employ Q-learning updates up the tree, updating the node action values and node counts in the BACKUP procedure in Lines 14 through 26. This means that the initial noisy back-up value given by the rollout, even though it is stored in the leaf node, might not make its way up to the root, since at each node we employ the $\max$ operator to define the target value, as shown in the BACKUP procedure. If all the children of a node have not been explored yet, for the Q-learning update of Equation 6.2, we apply the max operator only to the visited nodes, disregarding the unexplored actions. The BESTCHILD procedure (Line 6) has not been described since it depends on the action space of the specific problem. The selection is based on UCB1.

Remark 6.1. It is worth noting that, while in general the open-loop setting comes with a loss of performance compared to the closed-loop setting, this is not issue in the trading task. This is due to the fact that the market is not influenced by our actions. The state space is composed of features relative to the market (the price history) and features related to the agent (the agent's current position). While the market features follow stochastic transitions, they are not influenced by the agent's actions (see Lemma 3.1). The only features influenced

Algorithm 5: Q-Learning Open Loop Planning

Initialize: root node $\mathcal{N}_{0,0}$ from state s_0, budget B
while *within computational budget* B **do**
  $\mathcal{N}_{d,i}, s \leftarrow$ TREEPOLICY$(\mathcal{N}_{0,0})$
  $\mathcal{V}(\mathcal{N}_{d,i}) \leftarrow$ ROLLOUT$(\mathcal{N}_{d,i}, s)$
  BACKUP$(\mathcal{N}_{d,i})$
return BESTCHILD$(\mathcal{N}_{0,0})$

Procedure TREEPOLICY$(\mathcal{N})$
while $\mathcal{N}$ *not terminal* **do**
  if $\mathcal{N}$ *not fully expanded* **then**
    Return EXPAND$(\mathcal{N})$
  else
    $\mathcal{N} \leftarrow$ BESTCHILD$(\mathcal{N}, C_p)$
return $\mathcal{N}$

Procedure BACKUP$(\mathcal{N}, V)$
$C'(\mathcal{N})$ denotes explored children nodes of $\mathcal{N}$
$\mathcal{N}' \leftarrow$ parent of $\mathcal{N}$
$\mathcal{N}.n \leftarrow \mathcal{N}.n + 1$
while $\mathcal{N}'$ *is not null* **do**
  if $\mathcal{N}$ *is leaf* **then**
    $\Delta \leftarrow V$
  else
    $\Delta \leftarrow \max_{a' \in C'(\mathcal{N})} Q(\mathcal{N}, a')$
  $Q(\mathcal{N}', a) \leftarrow Q(\mathcal{N}', a) + \alpha(\mathcal{N}'.r + \gamma\Delta - Q(\mathcal{N}', a))$
  $\mathcal{N}'.n \leftarrow \mathcal{N}'.n + 1$
  $\mathcal{N} \leftarrow \mathcal{N}'$
  $\mathcal{N}' \leftarrow$ parent of $\mathcal{N}$

by the agent are its own positions regarding the underlying assets that follow deterministic transitions. Trading forms a special case of Factored MDPs (Degris and Sigaud, 2013), where the state transitions consist in two independent clusters of features. This means that the agent can effectively react to the differences in the features that it can control, also in the open-loop setting.

6.3.2 Nearest Neighbor Generative Model

A key element when applying MCTS, apart from the specific planning algorithm is also the generative model used to generate the simulations during the planning phase. A first alternative is to use classical models such as a GBM or a Vasicek (see Section 4.3), or econometric models such as ARIMA (see Section B.3.1), with parameters calibrated to fit the real data. In this section, we propose a novel technique to generate Monte Carlo simulations during the planning phase, based on a Nearest Neighbors (Bishop, 2006) framework, to retrieve, from the historical data available, price sequences that are "similar" to the current price window in the state.

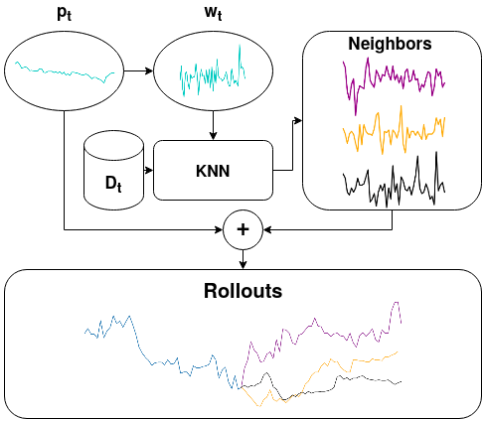

Figure 6.6: *Visual representation of the nearest neighbor generative model.*

Formally, we consider the time series of historical prices of the asset, $(p_1, p_2, \ldots, p_T)$ where T is the length of the time series. At time t, we observe the window of the last M prices of the asset, $(p_{t-M}, \ldots, p_{t-1})$. Since the length of the dataset T might cover multiple years, and the price of the asset might have changed substantially in these years we consider, instead of the series of prices, the series of price variations, $D = (\delta_1, \delta_2, \ldots, \delta_T)$ where $\delta_j = \frac{p_j - p_{j-1}}{p_{j-1}}$. Our goal is to find the "closest neighbors" of the window $w_t = (\delta_{t-M}, \delta_{t-M+1}, \ldots, \delta_{t-1})$ in the partial dataset $D_t = (\delta_1, \delta_2, \ldots, \delta_{t-1})$, that is the historical data before time t. By finding the nearest neighbors of w_t, we can use the continuation of the windows as simulations during the rollout.

Specifically, given a rollout length N, we aim to retrieve the K nearest neighbors of w_t (relative to a distance measure d), $\{w_{t_i}\}_1^K$, where $M < t_i < t - N$ is the time index of the i^{th} neighbor. We split the time series D_t in overlapping windows of length M, generating the dataset X, where each row of the dataset is a window of length M (same as the state window), where $X_0 = (\delta_1, \ldots, \delta_M)$, $X_1 = (\delta_2, \ldots, \delta_{M+1})$ and the last row $X_{t-N} = (\delta_{t-N-M}, \ldots, \delta_{t-N})$. Note that, the last price in the dataset X, is the price at time $t - N$, meaning that the last price of the corresponding rollout simulation is the last time-step. Before starting the planning phase, we search in the dataset X, for the K nearest neighbors of the current window w_t. The K neighbors give K possible future continuations of the price variations, which are used during planning. Figure 6.6 shows a visual representation of the nearest neighbor model. This generative model is quite flexible and can be rapidly updated with new data, simply by including new data points in the dataset.

6.3.3 Experimental Results

In this section, we present an experimental campaign evaluating the MCTS approach in the trading problem. We use as planner QL-OL UCT described in Section 6.3.1. As a generative model, we use the nearest neighbor approach illustrated in the previous section, thus the number of neighbors K becomes a hyper-parameter of our approach. At the beginning of each planning iteration, we sample one of these neighbors to use as a trajectory for the next

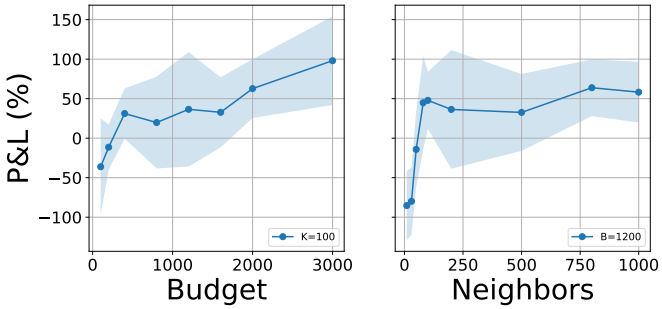

Figure 6.7: *Annualized average P&L with no transaction costs, as a function of the search budget and the numbers of neighbors. Average over 50 runs, 95% confidence intervals.*

rollout.

We concentrated our experiments on the EURUSD FX pair. We used the same dataset as Section 6.2 with the exchange rate from 2017 to 2019. In each episode, we sample a random date from the year 2019 and begin a trading episode for the next H minutes, where the horizon H is set to 200 in our episodes. We repeat this 50 times and present as result average return together with the 95 % confidence intervals.

For each episode, we use as dataset for the nearest neighbors model the series of prices from 1st of January 2017 to the current date. For the generative model, we considered as neighbors only the prices in a window of 3 hours centered at the current hour of the day, meaning that if we are currently trading at 10:00, we consider only windows from 9:00 to 11:59. This yields better results compared to considering all the possible windows, since there appears to be some correlation in the return windows depending on the hour of the day. Furthermore, this decreases the computational costs of retrieving the neighbors.

At each time-step, we perform a tree search, using QL-OL UCT with planning budget B and by sampling K neighbors. Both B and K represent hyper-parameters.

During planning, we decrease the decision frequency, meaning that every action chosen during the simulation phase (tree search) is repeated C times in the environment, before allowing to chose another action. This is similar to the idea of *persistence* introduced in Section 6.2.2. The increased persistence during planning has the effect of increasing the planning horizon, without increasing the planning cost, as frequent changes of the position are often non-optimal, especially under the presence of transaction costs. In all our experiments we use $C = 5$. Moreover, the maximum tree-depth has been set to 5, to allow for fast tree-search construction with a low budget because of the quasi-real time requirements of the application. This essentially means that the tree-search algorithm optimizes the return over the next 25 minutes given a tree depth of 5, where each level represents 5 minutes of transactions.

Results In Figure 6.7 we evaluate our approach in a setting without transaction costs. This enables to assess whether the nearest neighbour model is able to accurately predict future trends in the market. A fixed $100K allocation was considered and, similarly to Section 6.2.4, the performances are shown as percentages of the invested amount: P&L (%). The difference with respect to the FQI case, is that we averaged over 50, 200-minute

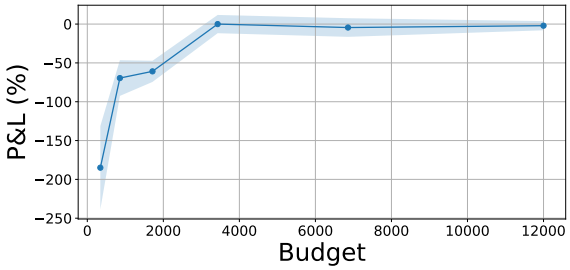

Figure 6.8: *Annualized average P&L with transaction costs as a function of the search budget, $K = 100$. Average over 50 runs, 95% confidence intervals.*

episodes and annualized, instead of testing for an entire year (given the computational difficulties of running the algorithm for such an extended amount of data).

On the left of Figure 6.7 we present the annualized percentage P&L as a function of the planning budget B, measured as transitions sampled from the forward model. In this experiment, we fix the number of neighbors $K = 100$. For very low budgets, of 100 and 200 transitions, corresponding to 20 and 40 future observed trajectories, the agent comes at a loss. Starting from the still low budget of 400, the agent achieves a profit. As we increase the budget, the returns improve (as we would expect). Moreover, on the right of Figure 6.7 we show the dependence of the returns on the number of neighbors K, while fixing the planning budget $B = 1200$ samples. Similarly to the previous experiments, a low number of neighbors yields a really low performance, as the trajectories seen during the simulation do not accurately reflect the true future trajectories, and so the planning agent "overfits" these simulations. As the number of neighbors surpasses 100, the agent becomes profitable, and the return improves slightly with the increase in neighbors.

Next, we evaluate our approach in the more realistic scenario of trading with transaction costs. We set the constant bid-ask spread of Equation 2.17 to $\$2 \cdot 10^{-5}$ as in Section 6.2. Figure 6.8 shows the results of this experiment, where we vary the search budget and fix the number of neighbors to 100.

While, similarly to the previous case, for low budget values, the agent performs at a loss, when increasing the budget, even further than the previous case, the agents converges to a policy of not trading and ensuring 0 profit (and loss). This probably happens because the search horizon is shorter than the true horizon of the interaction, making it inconvenient to trade with transaction costs because there is not enough time to observe the benefits of paying these costs. However, increasing the horizon search comes with increased computational demand for taking each decision. Nonetheless, even for budgets of 3000 samples during the search (that allow for near real-time response), the agent does not suffer losses.

In future works, we aim to tackle the problem of mismatching horizons during search and interaction, with the aim of increasing the search horizon without suffering costs in the response time. Potentially, to do so, we would like to explore more intelligent rollout policies that focus the tree construction in the relevant parts.

6.4 Chapter Summary

In this chapter we proposed two methodologies to create a quantitative trading strategy: using FQI and using MCTS.

FX Trading with FQI Section 6.2 is mostly experimental, and after initially defining FQI, the focus is on defining the MDP to model the environment realistically. We considered and compared two different scenarios, a multi-currency framework with EURUSD and USDGBP, and a single currency framework considering the FX pairs individually. On top of the FQI parameters like min-split and number of training iterations, we also considered persistence, namely how long an action lasts. We focused on three persistence values: 1-minute, 5-minute, and 10-minute. To summarize the experimental results, we can conclude that a 1-minute persistence is probably too short and thus the high signal to noise ratio makes it challenging for the agent to learn a profitable strategy. Out of the remaining persistences, the 5-minute achieves a superior performance on our tests both in terms of performance and Sharpe ratio. Finally, the multi-currency setting surpasses the single currency cases, this is expected as it may exploit additional trading opportunities.

This can be the starting point for several possible future research directions. First of all, in this chapter, the three-currency framework is modelled as a portfolio with two assets: when the agent chooses to change asset, it is forced to pay twice the transaction costs. We could take into account the missing pair of the triplet to reduce the costs. Secondly, we could consider real transaction costs, that is a bid-ask spread that depends on the market conditions, taking also into account market impact and thus optimizing execution as in Chapter 9.

FX trading with MCTS Section 6.3 proposed, for the first time, the use of MCTS for trading. Initially, we focused on modifying the standard UCT algorithm, considering open loop planning to handle the continuous state space and stochastic transition model. We also reduced the noise of the backups by introducing a Q-learning backup operator. The resulting algorithm was coined QL-OL UCT. We then applied QL-OL UCT to the trading setting, where we adopted a novel generative model using historical data and a clustering approach. In the experimental section, we tested the algorithm on the EURUSD FX pair, concentrating on optimizing parameters such as the search budget and the number of neighbors to use in the generative model. While we managed to consistently achieve profit, even for small planning budgets without considering transactions costs, adding these costs causes the agents to decide not to trade. Further work is necessary to improve the performance also compared to the FQI approach.

In future works, we plan to examine alternate generative models and tree-search procedures to allow the agent to achieve a profit also with the addition of transaction costs. Furthermore, we aspire to extend Alphazero (Silver et al., 2017), which has achieved astonishing experimental results, to be compatible with stochastic states and thus also the trading environment.

CHAPTER 7

Dealer Markets: a Mean-Field RL Approach

In this chapter we consider the dynamic pricing and RFQ response tasks of bond market makers, described in Section 2.3.4. The goal of this chapter is to explore a method to design autonomous market makers that exploit RL techniques to learn effective market making strategies even in the presence of strategic dealers.

A common market making strategy easily executed requires looking at the average market spread, e.g., the CBBT bid-ask spread, and offering a lower ask in case of a positive inventory, or, vice versa, offering a higher bid in case of a negative inventory.[31] The arrival time and characteristics of each RFQ depend on a variety of factors; among others, the general market sentiment, the specific asset in each asset class considered, and the time of day. Nonetheless, even with complete knowledge of the future RFQs, the definition of a strategy to perform the market maker's task in an optimal way is a rather complex problem. Indeed, the rewards obtained by each market maker depend not only on her actions, but also on the actions of all the other market makers involved in the exchange. This mutual dependence among market makers requires to model this environment in a *game theory* setting, where multiple rational players are competing to maximize their gain over time.

However, the naïve formulation of such a framework using a classical game-theoretical fashion results in a computationally intractable N-player stochastic game. In this work, we propose the use of the Mean Field Games (MFGs) framework, defined by Huang et al. (2006) to model the dealer market framework and find an approximate solution to

[31] The Composite Bloomberg Bond Trader (CBBT) is a weighted average of dealer-contributed prices that indicates where you can reasonably expect to find transaction opportunities on Bloomberg's Fixed Income Trading platform.

the original problem. Indeed, this approach is capable of approximating an N-player stochastic game in an efficient way, and offers theoretical guarantees on the convergence to an equilibrium. The idea of using MFGs to solve such a problem has been inspired by the works of Iyer et al. (2014); Balseiro et al. (2015), which have shown promising results in modeling and solving other auction type problems. In fact, the proposed formulation is of crucial importance for practical applications, since the existing approaches in literature assume the availability of data on the behaviour of the other dealers, which is commonly private information and not disclosed to competitors.

Motivated by the fact that finding an analytical solution to MFGs in general cases is an open problem, we propose the application of the numerical techniques of multi-agent RL to find a suitable approximate solution to this game. Specifically, for the first time, we model the dealer market environment as a multi-agent game, and solve it using model-free discrete-time Mean-Field RL (Gomes et al., 2010).

In this chapter, we focus on the role of dealers as liquidity providers, although market making is a broader concept and can be combined with other goals.[32] For instance, specialized trading firms may combine market making with quantitative trading aiming to increase the deriving profits, while at the same time using hedging strategies to manage unwanted risks and optimizing execution to reduce transaction fees.

Chapter outline The chapter is structured in four main sections. In Section 7.1, we describe and comment on the existing approaches that model the market maker paradigm, focusing on the ones using an RL approach. Then, we outline the works that define the theory of MFGs. In Section 7.2, we specify how the market making problem can be modeled as a stochastic game, highlighting the most relevant modeling assumptions required. In Section 7.3, we describe the approach we develop to solve the market making game and, finally, in Section 7.4, we provide an experimental evaluation both in the case of a market with a strategic adversarial opponent, and in that of a market populated by different configurations of dealers, commonly used in literature.

7.1 Background on Dealer Markets and MFGs

The proposed framework unifies two different streams of literature: the former focusing on modeling dealer markets, and the latter on model-free MFGs. To the best of our knowledge, our work is the first to use MFGs to model the dealer market scenario. In the subsequent paragraphs, we provide a description of the existing literature on these two topics starting with the existing market making approaches, and focusing on those that either model MD2C platforms (see Section 2.1.1) or use RL approaches to solve them. In the following section, we describe the current literature on MFGs, focusing on works regarding a model-free approach.

Dealer Markets

The most common market making approach is that by Avellaneda and Stoikov (2008), which revived old papers by Ho and Stoll (1981, 1983). After this seminal paper, several other works have been proposed, to cite a few: Cartea et al. (2014); Guéant et al. (2013);

[32] Even if dealers are specific for OTC trading, in this chapter we refer to them also as market makers.

Guéant (2016, 2017); Guéant and Manziuk (2019). The most important assumptions are that market prices are given by stochastic processes exogenous to the market maker's behavior, commonly a GBM (see Section 4.3.1), and that the outcome of an auction depends on a stochastic process that is assumed to model the behaviour of other market participants. Out of the mentioned works, the most similar to our approach is by Guéant and Manziuk (2019), who use a deep RL approach, i.e., a model-based actor-critic-like algorithm to find the optimal solution in discrete time. Nonetheless, even in this work, the probability of winning the RFQ, and thus the other player's behaviour, is modeled as a stochastic process. Indeed, all the above-mentioned works are making the rather strong assumption that the market makers are behaving stochastically, even if in reality dealers are capable of adapting over time to the strategy of the other market makers. Conversely, our approach models multiple competing market makers as opponents in a game, therefore not requiring any assumption on the behaviour of the other market makers.

Modeling multiple competing market makers can be naturally cast in a multi-agent formulation, which provides a model closer to reality. Indeed, multi-agent modeling of the markets has been already used to describe the behavior of the CDA process, as, for example, in Darley et al. (2000); Das (2005, 2008), to simulate data in Byrd et al. (2019), and to analyze the liquidity in the dealer market framework in Bank et al. (2021). However, the only work that uses a multi-agent approach to model market makers in a MD2C platform is by Ganesh et al. (2019), which considers M competing market maker agents. Out of all the mentioned papers, this last one is most closely related to our work, as it also applies RL techniques to optimize agent's payoff in a multi-agent framework. However, differently from our approach, the other market makers are hard-coded and fixed, and thus cannot behave in an adversarial way. This approach entails the risk that the proposed solution might suffer from large losses when deployed in a real-world setting, competing against real market makers that differ in behaviour compared to what was hypothesized during training. As a matter of fact, one would prefer to have an agent that learns a policy that works against *all* possible behaviours of the other players.

Finally, there are other approaches that consider RL in a market making scenario, but focusing on the CDA auction process, which is inherently different from the MD2C markets. These include Chan and Shelton (2001); Lim and Gorse (2018); Spooner et al. (2018); Spooner and Savani (2020)

Mean Field Games

The notion of MFGs was inspired by economic literature, studying topics related to the financial markets (Aumann, 1964). MFGs are designed to learn an approximate *Nash equilibrium* in stochastic games with a large number of identical agents, where the equilibrium is expressed as a policy/state measure pair. Specifically, under the given state measure, the policy should be optimal and when the agent applies this policy, the resulting agent state distribution is the same as the state measure. In such a framework, all players are identical, anonymous, and have symmetric interest. Thus, the learning problem can be reduced to characterizing the optimal interactions between a reference player and the population, which in turn behaves as the reference player. In this context, with continuous-time interactions, the Nash equilibrium is commonly computed providing the solution of a coupled system of dynamical equations where the first equation models the forward dynamics of the population distribution, and the second one consists in the dynamic programming equation

of a reference player.

This framework has been introduced by Huang et al. (2006); Lasry and Lions (2007), who showed the existence of approximate Nash equilibrium in continuous-time MFGs. Following, several different models of continuous time MFGs have been proposed, to cite a few: Huang et al. (2007); Huang (2010) consider linear-quadratic games, Saldi et al. (2019, 2020); Tembine et al. (2013) consider risk sensitive MFGs, while Moon and Başar (2016) study games with Markov jumps. Other works focus on finding numerical approximations to efficiently solve the system of dynamical equations using finite difference methods (Achdou and Capuzzo-Dolcetta, 2010; Achdou et al., 2012), semi-lagrangian schemes (Carlini and Silva, 2014, 2015), primal-dual methods (Briceño-Arias et al., 2018, 2019), neural network approximations (Carmona and Laurière, 2019; Fouque and Zhang, 2020; Ruthotto et al., 2020), generative adversarial networks (Cao et al., 2020; Lin et al., 2020), and fictitious play schemes (Robinson, 1951; Cardaliaguet and Hadikhanloo, 2017; Perrin et al., 2020). An overview of MFGs can be found in the survey by Gomes et al. (2014), or in the books by Fudenberg et al. (1998); Bensoussan et al. (2013).

Conversely, discrete-time MFGs, as required by the dealer market scenario, have not been studied as much. Gomes et al. (2010); Adlakha et al. (2015); Saldi et al. (2019) establish the existence of mean-field equilibrium without proposing algorithms capable of converging to the equilibrium. The problem of finding a solution, i.e., an equilibrium, in the discrete-time MFGs setting, has been successfully tackled by using RL algorithms. For instance, Fu et al. (2019) develop an actor-critic algorithm learning the equilibrium for mean field control, Carmona et al. (2019a) establish the convergence of a policy gradient algorithm, Carmona et al. (2019b); Anahtarcı et al. (2019) elaborate the convergence of Q-learning for deterministic systems. Finally, Yang et al. (2018a,c) apply classical RL algorithms to compute the mean field equilibrium. Among the aforementioned works, of particular relevance for this paper is that by Anahtarcı et al. (2019), which proposes the use of Q-learning and Fitted Q iteration (FQI). They give convergence guarantees with Q-learning, approximation error bounds with FQI, and analyze also the approximate case of a finite number of players.

In the classical MFG framework used in the previously cited works, the rewards and the dynamics for each player are known. Specifically, they depend only on the state/action pair of the player, and the population state distribution. Conversely, in the market making problem, the reward and the dynamic for each player depend also on the actions of all the other players. Therefore, the more inclusive framework of General Mean Field Games (GMFGs) by Guo et al. (2019, 2020) is required to handle this additional complexity faced by the market making scenario. Notice that, the analyses provided in the mentioned works are merely theoretical, and do not provide any constructive solution to finding an equilibrium. Moreover, they are general formulations of GMFG, while the application of this framework to the problem of dealer markets, to the best of our knowledge, has not been explored before.

7.2 Modelling Dealer Markets as a Stochastic Game

Let $\{\mathcal{F}_t\}_t$ be the natural filtration generated by the market information available up to time t. The market maker is in charge of an asset whose price P_t is an $\mathcal{F}_t$-adapted process. At each time $t \in \mathbb{N}$, a client can submit an RFQ to the MD2C platform, specifying the size

v_t (that is $\mathcal{F}_t$-adapted), where its sign specifies the side of the trade (positive or negative to manifest the intention to acquire or sell an asset, respectively). The MD2C platform is available to a large number N of market makers, even though at each time t only a number $M_t < N$ (possibly stochastic and $\mathcal{F}_t$-adapted), a.k.a. *market thickness* (Iyer et al., 2014), will respond to the RFQ. The objective of the market makers is illustrated in Definition 2.8. Each market maker i responds to the RFQ by showing the client the prices $(P^i_{t,\text{buy}}, P^i_{t,\text{sell}})$, that characterize the firm price at which the client can buy or sell the asset when market maker i is selected for the trade. We assume that the spreads published by the market maker are a function of the size of the trade v_t communicated by the client. The selected market maker i will buy from the client at price $P^i_{t,\text{buy}}(v_t)$ and will sell to the client at price $P^i_{t,\text{sell}}(v_t)$. Given the quotes from all the M_t market makers that answered the RFQ at time t, the client will select the market maker i^* giving the best quote for the placed RFQ, i.e., the smallest spread in the direction indicated by the RFQ. We denote with z^i_t the inventory of dealer i at time t.

N-player Stochastic Games Let us define a discrete-time N-player Markovian stochastic game. This model is an extension of MDPs defined in Section 3.1, in which the transition probabilities depend on the aggregated action of all players, and the reward depends on the state of all the players. At every time $t \in [N]$, each agent is in state $s^i_t \in \mathcal{S}_i$, and chooses an action $a^i_t \in \mathcal{A}_i$.[33] Define $\mathcal{S}^N := \mathcal{S}_1 \times \ldots \times \mathcal{S}_N$ as the set of possible joint states and $\mathcal{A}^N := \mathcal{A}_1 \times \ldots \times \mathcal{A}_N$ as the set of possible joint actions performed by players. Given the current joint state $\mathbf{s}_t := (s^1_t, \ldots, s^N_t) \in \mathcal{S}^N$, and current joint action $\mathbf{a}_t = (a^1_t, \ldots, a^N_t) \in \mathcal{A}^N$, each player $i \in [N]$ receives a reward specified by the reward function $r^i : \mathcal{S}^N \times \mathcal{A}_i \to \mathbb{R}$. Moreover, the current state s^i_t of each player evolves in the next state s^i_{t+1} according to the transition function:

$$s^i_{t+1} \sim \mathcal{P}(\cdot|s^i_t, \mathbf{a}_t) \; \forall i \in \mathcal{N}. \tag{7.1}$$

Similarly to the case of MDPs, the policy $\pi^i(\mathbf{s}_t)$ of an agent i is a function that maps a state $\mathbf{s}_t$ to a distribution over the action space $\Delta(\mathcal{A}_i)$, formally $\pi^i : \mathcal{S}^N \to \Delta(\mathcal{A}_i)$. We denote with π^{-i} the policies of all players but the i-th one, so that $\pi = (\pi^i, \pi^{-i})$. The expected cumulative reward of agent i following a generic policy sequence π is:

$$J^i_\pi := \mathop{\mathbb{E}}_{\substack{\boldsymbol{a}_t \sim \pi(\boldsymbol{s}_t), \\ s^i_{t+1} \sim \mathcal{P}(\cdot|s^i_t, \mathbf{a}_t)}} \left[\sum_{t=1}^{+\infty} \gamma^t r^i(\boldsymbol{s}_t, a^i_t) \Big| s_0 = s \right]. \tag{7.2}$$

When the policy π^{-i} of the other agents is fixed, the objective of each agent $i \in [N]$ is to solve the following stochastic optimization problem:

$$\pi^i = \arg\max_\pi V_{(\pi, \pi^{-i})}, \tag{7.3}$$

where $V^i_\pi := \mathbb{E}_{s\sim\xi}[J^i_\pi(s)]$, for an initial state distribution ξ.

Notice that, the modeling provided by N-player stochastic games is able to describe the dealer markets problem. In what follows we show the correspondence between the two.

[33] Given $W \in \mathbb{N}$, with $[W]$ we denote the set $\{1, \ldots, W\}$.

Reward The reward for each market maker i is defined as in Equation 2.20:

$$r_t^i = \underbrace{v_t(P_{t,h}^i(v_t) - P_t)\mathbb{I}_{i=i^*}}_{\text{spread P\&L}} + \underbrace{z_{t-1}^i(P_t - P_{t-1})}_{\text{inventory P\&L}} - \underbrace{\phi(z_t^i)}_{\text{inventory penalty}}, \tag{7.4}$$

where we are also considering the dependence of $P_{t,h}^i(v_t)$ on the trade size v_t. This definition of the reward is common in literature (Guéant and Manziuk, 2019; Avellaneda and Stoikov, 2008; Cartea et al., 2015; Guéant, 2017), where the risk-aversion of a market maker is encoded implicitly in the function $\phi(\cdot)$. Notice that it is possible to add an explicit penalization for the variation of the rewards over time, by making the reward r_t^i dependent on the variability of the past inventory.

Actions The action space $\mathcal{A}_i$ for dealer i is given by the prices $(P_{t,\text{buy}}^i(v), P_{t,\text{sell}}^i(v))$. Here, we choose to follow a modeling approach similar to the one proposed by Ganesh et al. (2019). Defining the action space $\mathcal{A}_i = [-\epsilon, \epsilon]^2$ for all $i \in \mathcal{N}$, with $\epsilon > 0$, the pricing function is defined as follows:

$$P_{t,h}^i(v) = \tilde{P}_{t,h}(v)(1 + \epsilon_h^i), \tag{7.5}$$

where $h \in \{\text{buy}, \text{sell}\}$, $\epsilon_h^i \in [-\epsilon, \epsilon]$ is the action chosen by the market maker and $\tilde{P}_{t,h}(v)$ is a reference price calibrated on the market.

State The state space is $\mathcal{S} = [-Z_m, Z_m] \times [P_l, P_u]$, where $Z_m > 0$ is the maximum inventory allowed to be held, and $0 < P_l < P_u$ are the minimum and maximum values for the asset price in consideration, respectively.

Transition Function The transition function $\mathcal{P}(\cdot|s_t, \mathbf{a}_t)$ of the MDP specifies the evolution of the state $s_t = (z_t, P_t)$ to the next state $s_{t+1} = (z_{t+1}, P_{t+1})$ for each player. Specifically, the asset price P_t evolves according to a transition function common to all the players, while the inventory z_t^i evolves differently for the winner of the auction. Formally, the z_t^i dynamic is described by the following equation:

$$z_t^i = z_{t-1}^i + v_t \, \mathbb{I}_{i=i^*}, \tag{7.6}$$

where i^* is the index of dealer that provided the best quotes.

The objective of each dealer i is to find a policy π^i specifying, for each combination inventory and price in $\mathcal{S}$, the optimal bid-ask spreads in $\mathcal{A}_i$ that maximize their objective function in Equation (7.3).

Remark 7.1. The main appeal of the proposed framework is that it allows to model various features of interest to the dealer. For instance, one could be interested in including in the model the relationship between the distribution of the RFQs and the price process of the underlying asset, i.e., simulating a market sell-off with many bid RFQs but no ask RFQs.

Remark 7.2. Notice that the information required to run this scheme is easily available to the dealers. Indeed, to fit the parameters of the RFQ arrival process, one needs only the information collected by interacting with the MD2C platform (such as the distribution of

the RFQ sizes v_t, the price P_t, and the reference market price $\tilde{P}_{t,h}(v)$). Conversely, other existing approaches make use of information on the behaviour of the other dealers, e.g., the work by Fermanian et al. (2016), which is in general hard to obtain since it constitutes private information of the competitors.

7.3 Learning Equilibrium via General MFGs

The model described in the previous section provides an accurate description of a real dealer market. However, modeling this environment as an N-player stochastic game suffers from the combinatorial complexity with respect to the number N of players (Daskalakis et al., 2009). In the subsequent sections, we describe the Mean Field Game approach, which allow us to obtain an approximate equilibrium strategy of the game.

General Mean Field Game Approximation of the Dealer Market Problem For the mean-field approximation, we assume homogeneous market participants and a number of players N that goes to infinity, while maintaining a finite market thickness M_t. In practice, under this approximation, the players' population is compactly represented by a distribution on the possible states-actions space $\mathcal{L} \in \Delta(\mathcal{S} \times \mathcal{A})$ and the interaction between each single player and the rest of the distribution happens only trough the mean-field $\mathcal{L}$. At each time t and for each RFQ, the market participants answering the RFQ are generated sampling $M_t - 1$ players (market makers) from the mean-field $\mathcal{L}$, thus specifying their state s^k and action a^k for $k \in [M_t - 1]$. In the next paragraph, we describe formally the GMFG framework, which expands and generalizes MFGs.

Notice that assuming a homogeneous policy between the dealers is only an assumption needed to find an approximate equilibrium, while it does not represent a simplifying assumption with respect to a real market making framework. Indeed, we show in the experimental campaign that the agents trained with the MFG approach provided here perform exceptionally even when tested against agents with different policies, i.e., in settings in which the homogeneous assumption does not hold.

General mean-field Games GMFG is an extension of mean-field Games introduced by Guo et al. (2019), which enables the next state transition and reward to depend also on all the actions played by the population. Let the *mean-field* $\mathcal{L}_t$ be a distribution of the population over the state-action space, or, formally, $\mathcal{L}_t \in \Delta(\mathcal{S} \times \mathcal{A})$. At each time t, a player in state $s_t \in \mathcal{S}$ plays the action $a_t \in \mathcal{A}$ and receives the reward $r(s_t, a_t, \mathcal{L}_t)$. Consequently, the player's current state evolves to s_{t+1}, according to the distribution $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t, \mathcal{L}_t)$. In such a modeling approach, the definition of the value function, given a mean-field $\mathcal{L}$ is:

$$V(\pi, \mathcal{L}) := \mathop{\mathbb{E}}_{\substack{a_t \sim \pi_t(\cdot|s_t, \mu_t), \\ s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t, \mathcal{L}_t)}} \left[\sum_{t=1}^{+\infty} \gamma^t r(s_t, a_t, \mu_t) \right], \tag{7.7}$$

where $\mu_t := \int_{\mathcal{A}} \mathcal{L}(\cdot, a)\, da$ is the marginal distribution of the population $\mathcal{L}$ over the state space $\mathcal{S}$.

The definition of a Nash Equilibrium profile for a GMFG is defined as follows:

Algorithm 6: Model Free GMFG

1 **Initialize:** mean-field $\mathcal{L}_0$, environment simulator $\mathcal{E}(\cdot,\cdot;\mathcal{L})$, number of iterations W
2 **for** $w \in [W]$ **do**
3 Find the single-agent optimal policy π_w with fixed mean-field $\mathcal{L}_w$
4 Update $\mathcal{L}_{w+1}$ using $\mathcal{E}(\cdot,\cdot;\mathcal{L}_w)$
5 Return $(\pi_W, \mathcal{L}_W)$

Definition 7.1. *The tuple $(\pi^*, \mathcal{L}^*)$ is a Nash Equilibrium for the GMFG if, for any policy π, it holds that:*

$$V(\pi^*, \mathcal{L}^*) \geq V(\pi, \mathcal{L}^*), \tag{7.8}$$

and

$$a_t \sim \pi_t^*(s_t, \mu^*), s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t, \mathcal{L}^*), \tag{7.9}$$

where μ^* is the marginal distribution of the mean state-action population $\mathcal{L}^*$.

To find the equilibrium strategy π^*, we employ the algorithm proposed by Guo et al. (2019). The high-level structure of the algorithm is presented in Algorithm 6. The main idea is to alternatively evolve the policy π_w and the mean-field $\mathcal{L}_w$ for $w \in [W]$. Specifically, the algorithm requires an initial mean-field $\mathcal{L}_0$, an environment simulator $\mathcal{E}(\cdot,\cdot;\mathcal{L})$, and the number of iterations W. The simulator $\mathcal{E}(\cdot, \mathcal{L}_w)$ is a function that simulates the environment step, or, formally, for a mean-field $\mathcal{L}$, we have that $(s_{t+1}, r_{t+1}) = \mathcal{E}(s_t, a_t; \mathcal{L})$. For W steps, the algorithm learns an (approximate) optimal policy π_w, by fixing the mean-field distribution $\mathcal{L}_w$ (Line 3).[34]

After that, during the update of the mean-field $\mathcal{L}_w$, the optimal policy π_w is used to generate a new mean-field distribution $\mathcal{L}_{w+1}$ (Line 4). At the end of the process, the algorithm returns the tuple $(\pi_W, \mathcal{L}_W)$, which is guaranteed to converge to an approximate Nash equilibrium for the GMFG as proved in Guo et al. (2020).

GMFG with FQI In the subsequent paragraphs, we present the FQI for GMFG algorithm, whose pseudo-code is provided in Algorithm 7, an implementation of Algorithm 6 in which the FQI algorithm (see Section 6.2.1) is used to find the policy π_w. Specifically, the procedure described in Line 3 of Algorithm 6 corresponds to Lines 3 through 7 of Algorithm 7, while the update of the mean-field $\mathcal{L}_w$ of Line 4 of Algorithm 6 corresponds to Lines 8 through 13 of Algorithm 7.

More precisely, the algorithm requires as input an initial mean-field $\mathcal{L}_0$, a simulator $\mathcal{E}(\cdot,\cdot;\mathcal{L})$, and the parameters W, J, I, and τ. Specifically, W is the number of times the FQI algorithm and the evolution of the mean-field $\mathcal{L}_w$ are performed, J represents the number of iterations to be set in the FQI algorithm, I is the number of samples to generate the empirical distribution for $\mathcal{L}_w$ and τ is the inverse temperature parameter to be used in a Boltzmann distribution.

For each of the W iterations, the FQI algorithm is employed to find the policy π_w. Indeed, the goal of the FQI algorithm is to approximate the state-action function $Q(s, a)$,

[34] The number of iterations W should be selected so that the error on the Weierstrass distance between $\mathcal{L}_{W-1}$ and $\mathcal{L}_W$ is smaller than a required threshold. See the work by (Guo et al., 2020) for details.

Algorithm 7: FQI for GMFG

1 **Initialize:** state-action distribution $\mathcal{L}_0$, environment simulator $\mathcal{E}(\cdot,\cdot;\mathcal{L})$, number of iterations I, W, and J, inverse temperature parameter τ
2 **for** $w \in [W]$ **do**
3 Initialize $\hat{Q}_{w,0}(s,a) \leftarrow 0\ \forall s \in \mathcal{S}, a \in \mathcal{A}$
4 Generate dataset $\mathcal{D}_w = \{(s_i, a_i, r_i, s_i')\}_{i\in[D]}$ using simulator $\mathcal{E}(\cdot,\cdot;\mathcal{L}_w)$
5 **for** $j \in [J]$ **do**
6 $\hat{Q}_{w,j+1} = \arg\min_{f\in\mathcal{F}} \sum_{i\in D} \left(f(s_i,a_i) - r_i - \gamma \max_{a\in\mathcal{A}} \hat{Q}_{w,j}(s',a) \right)^2$
7 Extract $\pi_w(s) = \frac{\exp(\tau \hat{Q}_{w,J}(s,\cdot))}{\sum_{a\in\mathcal{A}} \exp(\tau \hat{Q}_{w,J}(s,a))}\ \forall s \in \mathcal{S}$
8 $\mu_w \leftarrow \int_{\mathcal{A}} \mathcal{L}_{w-1}(s,a)\, da$
9 Initialize $\mathcal{L}_w(s,a) \leftarrow 0\ \forall s \in \mathcal{S}, a \in \mathcal{A}$
10 **for** $i \in [I]$ **do**
11 $s_i \sim \mu_w$, $a_i \sim \pi_w(s_i)$
12 $(s_i', r_i') \leftarrow \mathcal{E}(s_i, a_i; \mathcal{L}_{w-1})$
13 $\mathcal{L}_w(s_i', a_i) \leftarrow \mathcal{L}_w(s_i', a_i) + 1/I$
14 Return π_W and $\mathcal{L}_W$

i.e., the cumulative value of a specific action a in a state s, by means of a regressor $\hat{Q}_w(s,a)$ that exploits the dataset $\mathcal{D}_w$. At first, for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ the algorithm initializes the approximated state-action function $\hat{Q}_{w,0}(s,a)$ to zero (Line 3) and generates, using the current environment simulator $\mathcal{E}_{\mathcal{L}}$, a dataset $\mathcal{D}_w$. With the dataset, we apply FQI as specified also in Algorithm 4.[35] The regression step is repeated J times (Line 5), as prescribed by the FQI algorithm, and results in an approximation $\hat{Q}_{w,J}(s,a)$ of the true state-action function. From the state-action function, we extract the approximate optimal policy $\pi_w(\cdot)$ for each state $s \in \mathcal{S}$ by applying the soft-max function to $\hat{Q}_{w,J}(s,a)$ (Line 7). This ensures that $\pi_w(s)$ defines a probability distribution over the actions $\mathcal{A}$ for a specific state s.

In the pseudo-code following Line 7, the algorithm performs an update of the population $\mathcal{L}_w$ due to the change in the players' policy $\pi_w(\cdot)$. First, it computes the marginal distribution of the population over the state space μ_w (Line 8). After that, it samples I states and actions pairs from the distributions μ_w and π_w, respectively (Line 11), that are used in the simulator $\mathcal{E}(\cdot,\cdot;\mathcal{L}_w)$ to generate the next state s_i' (Line 12). Finally, it updates the empirical distribution of the population $\mathcal{L}_w(s_i', a_i)$.

The above two steps are repeated W times, after which the algorithm returns an estimate of the approximate optimal policy π_W and the mean-field $\mathcal{L}_W$.

Remark 7.3. It is important to note that the approximation of infinite players still provides theoretical guarantees to the original stochastic game, where the number of players N to whom the platform is available is finite. In particular, if we have an equilibrium for the

[35] In what follows we used as hypothesis space for the regression problem the class $\mathcal{F}$ of extra trees (Ho, 1995) (see also Appendix B.4.2), due to their lightweight computational expenses during training. Depending on the computational complexity required by this step and the characteristics of the environment one might select a different regressor, e.g., Support Vector Machines or Neural Networks. See Antos et al. (2007) for details.

GMFG, then it is an ε-Nash equilibrium for the finite player game with $N = N(\varepsilon)$ players. An ε-Nash equilibrium for a N-player stochastic game, is a collection of N policies $(\pi_1^*, \pi_2^*, \ldots, \pi_N^*)$ such that for all players $i \in [N]$:

$$V(\pi_i{}^*, \pi_{-i}^*) \geq V(\pi, \pi_{-i}^*) - \varepsilon \; \forall \pi, \tag{7.10}$$

with $\varepsilon > 0$. For reference, see Theorem 2.1 in (Anahtarcı et al., 2019).

Remark 7.4. Note that Guo et al. (2019) proposed a solution, which, instead of using FQI for the optimal policy estimation, uses Q-learning (see Equation 3.8).[36] However, this approach does not perform well on the problem of dealer markets, due to the fact that it models the state-action function $Q(\cdot, \cdot)$ in a tabular fashion, therefore not exploiting its regularity structure. Indeed, this is crucial in the dealer market framework, as we expect the optimal Q-function $Q_w^*(s, a)$ to show continuity with respect to state s and action a.

7.4 Experimental Results

In what follows, we describe the experimental results achieved with the proposed framework. At first, we introduce the model of the environment, i.e., the evolution of the price, the reference spread, and the RFQ process used in the experiments. Then, we present the different dealer agents that interact in the market making simulations. Finally, we cover the relevant metrics employed in the evaluation of our method.

After explaining the setting, we provide empirical simulations for two different dealer market environments. In the former, we explore the robustness of the equilibrium policy by testing it in a market in which one player has knowledge over the strategy of the other player. In the latter, we provide a wide experimental campaign in which we populate the market with many players with different strategies, strategic and/or stochastic, and verify how they interact over time.

7.4.1 Experimental Setting

Synthetic Environment The price process P_t is modeled as a GBM as in Section 4.3.1. These values have been annualized and we consider one RFQ per day over a year, $\mu = 0$ to ensure the price process is a martingale, and $\sigma = 20\%$. Such a volatility may be found for example in high yield bonds in periods of market distress. The volume v_t of the RFQ at time t is extracted from $\{-1, 1\}$ with equal probability. The possible actions for buying and selling for each market maker are in $\mathcal{A} = \{-0.03, -0.02, -0.01, 0, 0.01, 0.02, 0.03\}$. The inventory penalization of the reward defined in Equation (7.4) has been set to $\phi(z) = \lambda z^2$, as commonly done in literature (Ganesh et al., 2019; Guéant and Manziuk, 2019), where λ can be seen as a risk aversion parameter, which controls the amount of inventory. We consider $\lambda = \frac{1}{2}$ unless stated otherwise. Finally, following what has been done in Kolm and Ritter (2019), we defined the market price as:

$$\tilde{P}_{t,h}(v) = P_t + \delta(|v| + 0.01v^2), \tag{7.11}$$

where $h \in \{\text{buy}, \text{sell}\}$ and $\delta = 0.1$.

[36] See Algorithm 16 in Appendix A.4.1 for details.

MFG Agents We trained two FQI-GMFG agents: FQI_2 and FQI_4. FQI_2 was trained considering a market thickness of $M_t = 2$, for each $t \in [T]$, while FQI_4 considered a market thickness of $M_t = 4$, for each $t \in [T]$.

In both cases, we fixed $W = 5$, $J = 5$, $I = 2 \cdot 10^6$, $\tau = 4$, and $\gamma = 1$. As a regressor we used extra trees (see Appendix B.4.2) with 500 trees and a min-split parameter of 0.1% of the total samples.

Benchmark Agents We compared our solution with agents following stochastic strategies and with solutions from the literature, e.g., in Ganesh et al. (2019). Specifically, we analysed:

- Persistent ($\mathfrak{P}$) agents, which select the action $(\epsilon_b, \epsilon_s) = (0, 0)$ for each state $s \in \mathcal{S}$;
- Uniform ($\mathfrak{U}$) agents, which randomly select an action $(\epsilon_b, \epsilon_s) \in [-\epsilon, \epsilon]^2$ for each state $s \in \mathcal{S}$;
- Normal ($\mathfrak{N}$) agents, which randomly select an action $\epsilon_b \sim \mathcal{N}(0, 0.04)$, and $\epsilon_s \sim \mathcal{N}(0, 0.04)$, i.e., from Gaussian distributions;[37]
- Q-learning agents (Q_2, Q_4), generated using the Q-GMFG algorithm described in Appendix A.4.1. We trained two agents Q_2, and Q_4, for market thickness $M_t = 2$, for each $t \in [T]$, and $M_t = 4$, for each $t \in [T]$, respectively.

Metrics We evaluate the strategies in terms of:

- mean dollar reward: $L := \sum_{t \leq T} \frac{l_t}{T}$, where the loss l_t for each agent i is defined as $l_t := \mathbb{I}_{i=i^*} |v_t(P_{t,\cdot}(v_t) - P_t)| + z_{t-1}(P_t - P_{t-1})$;
- mean Sharpe ratio on the dollar reward: $S := \frac{L}{\sqrt{\sum_{t \leq T} \frac{(l_t - L)^2}{T}}}$;
- the average reward: $R := \sum_{t \in [T]} \frac{r_t}{T}$, where the reward r_t is defined in Equation (7.4);

where the length of each run is $T = 500$ days.

Note that we study two different metrics to evaluate different aspects of each strategy. Specifically, we want to evaluate the average dollar gain provided by a strategy and its reliability.

7.4.2 Equilibrium Policy

As a preliminary analysis, we investigate the behaviour of the policy π_W learnt by the FQI-GMFG agents in the environment described in the previous section. In particular, we explore the policy π_W as a function of the inventory z_t. For each inventory $z \in [-Z_m, Z_m]$, we generated 100 states by sampling uniformly a price $P_k \in [P_l, P_u]$, registered $(\epsilon_{\text{buy}}, \epsilon_{\text{sell}}) = \pi_W(z, P_k)$ for each state (z, P_k), and averaged over the 100 samples. Figure 7.1 shows the average strategy for the two actions (buy and sell) with respect to the inventory z_t for the

[37] The distribution are truncated so that they have probability 1 of being in the action space $\mathcal{A}$.

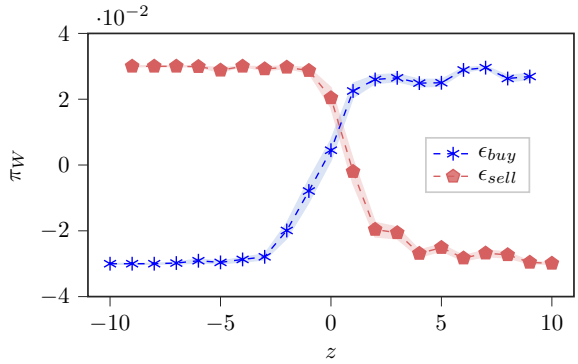

Figure 7.1: *Equilibrium policy π_W for the FQI_2 agents.*

FQI_2 agent with dashed lines, and the 95% confidence interval, collected with statistical bootstrap, with semi-transparent areas. We observe that the learnt policies bid aggressively for large negative and positive inventory. This behaviour allows the market maker to keep on average a zero net inventory. This has an important economical interpretation: it shows that the equilibrium policy for the dealers tends to a zero inventory. Note that reaching the equilibrium requires receiving negative rewards to adjust the inventory. Specifically, when ϵ_{sell} and ϵ_{buy} are negative, the agent receives an immediate negative reward with probability 1, due to the fact that it will buy (sell) at price a higher (lower) price than the reference market. FQI_4 gives results that are in line with the ones just discussed. Note that, this behaviour, a.k.a. *skewing*, is also a common trait in most of the market making strategies designed in the economic literature, e.g. Guéant (2017).

7.4.3 Exploitability Study

In this section, we study the setting of a market with a strategic dealer that has information allowing her to exploit the other dealers. This investigation is of paramount relevance to evaluate the performance of strategies in case opponents are aware of the currently adopted strategy. Indeed, in a multi-agent setting, there is no concept of *best policy* as its optimality also depends on the aggregate policy of the other players. For instance, when playing a *rock/paper/scissors* game against a player that plays rock more often than scissors, then playing paper is favorable in this situation. However, this is ruinous in the case the other player switches to playing scissors. This is why, in a multi-agent framework, the correct concept to analyze is *exploitability*, which is commonly used in the game theory literature to evaluate multiple-player settings. Formally, in a N-player stochastic game, exploitability for the i-th player with policy π^i is defined as:

$$J_{exp}(\pi^i) := \min_{\pi^{-i}} J^i_{(\pi^i, \pi^{-i})}. \tag{7.12}$$

Intuitively, the above quantity measures how much value can be gained from an opponent by the policy π^i. For instance, in the example above (rock/paper/scissors game), assuming each turn we bet $\$1$, the exploitability of playing rock with probability 100% is $\$1$, i.e., the opponent can make us lose at most $\$1$ per round if she knows our strategy, while the

	FQI_4	FQI_2	Q_4	Q_2	$\mathfrak{P}$	$\mathfrak{U}$	$\mathfrak{N}$
L	**0.049**	0.048	0.0021	-0.02	0.009	0.018	0.026
S	**0.008**	**0.008**	0.002	-0.0	0.001	0.002	0.002

Table 7.1: *Performance of the different agents $\mathfrak{E}$ in the setting where the other dealer is the corresponding exploitative agent $E(\mathfrak{E})$. Best results, or tied for the best, are highlighted in boldface.*

exploitability of playing rock/paper/scissors each with probability $1/3$ is \$0. To test the exploitability of an agent $\mathfrak{E}$ in our setting, we generate an exploitative agent $E(\mathfrak{E})$ and train it in the following way. We build an environment only composed by the agent $\mathfrak{E}$ and the agent $E(\mathfrak{E})$. Then, we train the agent $E(\mathfrak{E})$ so that it maximizes its own value function. To find an approximate optimal policy, we employ a RL algorithm on the MDP that is generated by fixing the behaviour of agent $\mathfrak{E}$.[38] In particular, we used the Proximal Policy Optimization (PPO) policy by Schulman et al. (2017) with a policy network of 2 layers and 64 neurons each. We trained it against the $\mathfrak{E}$ agent described above, creating an exploitative agent E($\mathfrak{E}$) for each policy described in Section 7.4.1. We used the implementation of Stable Baselines (Hill et al., 2018) with default parameters for PPO. The provided results have been averaged over $1,000$ independent runs.

Exploitability Results

Table 7.1 reports the performances of the different agents $\mathfrak{E}$ when employed against their corresponding exploitative agent $E(\mathfrak{E})$. We see that the FQI_2 and FQI_4 agents, which are considering the strategic nature of the dealer market setting, perform better than the others when employed against exploitative agents. Indeed, the dollar reward L for FQI_4 and FQI_2 is 0.049 and 0.048, respectively, which is almost twice the second best agent (Normal $\mathfrak{N}$ with $L = 0.026$). Moreover, for the same agents, we have an improvement of a factor 4 of the Sharpe ratio S (0.008 for FQI_4 and FQI_2 vs. 0.002 for $\mathfrak{N}$). The improved performance is due to the fact that the FQI_2 and FQI_4 cannot be strategically manipulated as that they have been trained to cope with these kind of adversarial environments. The same is not true for the Q_2 and Q_4 agents, which still consider the strategic aspect of the market making setting, but are not learning appropriate equilibrium policies. This suggests that the tabular nature of the Q-GMFG algorithm fails in incorporating the intrinsic regularity of the reward with respect to the state and action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, which, instead, is crucial in training the FQI-GMFG agents.

7.4.4 Market Simulation Study

In this section, we analyse several market simulations. Specifically, we used the setting of Section 7.4.1 and for each of the agents described, we created a market simulation with one or more other dealer agents. We studied two different scenarios: $M_t = 2$, for all $t \in T$, and $M_t = 4$, for all $t \in T$. For each of the defined metrics, i.e., L, S, and Z, we provided

[38] This is a standard approach to approximate the exploitability of learnt policies used by Greenwald et al. (2013).

	FQI_2	Q_2	$\mathfrak{P}$	$\mathfrak{U}$	$\mathfrak{N}$
FQI_2	0.049	0.059	**0.075**	0.065	0.061
Q_2	0.03	0.048	**0.053**	0.044	0.045
$\mathfrak{P}$	0.019	**0.052**	0.044	0.043	0.048
$\mathfrak{U}$	0.028	0.041	0.042	**0.064**	0.047
$\mathfrak{N}$	0.04	**0.057**	0.046	0.048	0.06

Table 7.2: *Mean dollar reward L for $M_t = 2$. Larger is better.*

	FQI_4	Q_4	$\mathfrak{P}$	$\mathfrak{U}$	$\mathfrak{N}$
FQI_4,FQI_4,FQI_4	0.03	0.029	0.031	0.031	**0.037**
Q_4,Q_4,Q_4	0.013	0.016	0.009	0.023	**0.033**
$\mathfrak{P},\mathfrak{P},\mathfrak{P}$	0.019	0.039	0.026	0.035	**0.066**
$\mathfrak{U},\mathfrak{U},\mathfrak{U}$	0.01	0.023	0.015	0.019	**0.037**
$\mathfrak{N},\mathfrak{N},\mathfrak{N}$	0.015	0.011	0.008	0.017	**0.021**

Table 7.3: *Mean dollar reward L for $M_t = 4$. Larger is better.*

the result obtained by the reference agent (reported on the columns) in a setting where the other market makers are configured as reported on the rows. The provided results have been averaged over $1,000$ independent runs. We highlight the best agent over each row in bold.

Market Simulation Results

Table 7.2 and Table 7.3 present the average dollar reward L of the agents for $M_t = 2$ and $M_t = 4$, respectively. In particular, when $M_t = 2$ there is always at least one agent that outperforms the FQI_2 one, but there is no clear agent able of overcoming the others in all the settings. Indeed, the Persistent agent $\mathfrak{P}$ provides the largest dollar reward against FQI_2 and Q_2, the Q_2 agent is the best option against $\mathfrak{P}$ and $\mathfrak{N}$, and the Uniform agent $\mathfrak{U}$ outperforms the others only when tested against an agent of the same kind. Overall, in terms of mean dollar reward, it seems that there is no algorithm able to outperform all the others. Instead, when $M_t = 4$, the Normal agent $\mathfrak{N}$ outperforms all the other agents, with a mean dollar reward L in the range $[0.021, 0.066]$ for all the market configurations considered. However, one cannot base the decision for the best strategy only looking at the average dollar reward, since large values of this metric might be achieved at the cost of a large risk (in terms of variance of the gain over time).

Conversely, the results are significantly different if we analyze the Sharpe ratio S, for $M_t = 2$ and $M_t = 4$, as presented in Table 7.4 and Table 7.5, respectively. Differently to what has been observed with the mean dollar reward L, the FQI_2 and FQI_4 agents achieve, in most scenarios, the best results for the metric S, where FQI_2 is tied in only one setting by the Uniform agent $\mathfrak{U}$. In particular, this happens in the case of $M_t = 2$ as the FQI_2 agent achieves a larger mean Sharpe ratio S than the other agents in all the market configurations. Indeed, S is in the range $[0.009, 0.024]$ for FQI_2 while all the other agents have a mean Sharpe ratio S in the range $[0.006, 0.01]$.

However, when $M_t = 4$, we have that in all the market configurations considered,

	FQI_2	Q_2	$\mathfrak{P}$	$\mathfrak{U}$	$\mathfrak{N}$
FQI_2	**0.009**	0.008	0.01	**0.009**	0.008
Q_2	**0.024**	0.007	0.008	0.006	0.007
$\mathfrak{P}$	**0.021**	0.008	0.006	0.006	0.006
$\mathfrak{U}$	**0.023**	0.006	0.006	0.009	0.007
$\mathfrak{N}$	**0.019**	0.008	0.007	0.007	0.009

Table 7.4: *Mean Sharpe ratio S for $M_t = 2$. Larger is better.*

	FQI_4	Q_4	$\mathfrak{P}$	$\mathfrak{U}$	$\mathfrak{N}$
FQI_4, FQI_4, FQI_4	**0.008**	0.004	0.004	0.004	0.004
Q_4, Q_4, Q_4	**0.014**	0.003	0.002	0.003	0.005
$\mathfrak{P}, \mathfrak{P}, \mathfrak{P}$	**0.021**	0.007	0.004	0.005	0.009
$\mathfrak{U}, \mathfrak{U}, \mathfrak{U}$	**0.011**	0.002	0.003	0.002	0.006
$\mathfrak{N}, \mathfrak{N}, \mathfrak{N}$	**0.011**	0.005	0.001	0.003	0.004

Table 7.5: *Mean Sharpe ratio S for $M_t = 4$. Larger is better.*

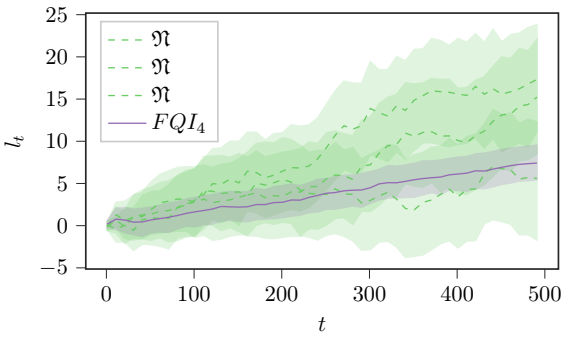

Figure 7.2: *Average dollar reward l_t of the FQI_4 agent (solid purple line) and those of three identical Normal agents $\mathfrak{N}$ (dashed green lines), when dealing in the same market simulation.*

the FQI_4 agent achieves a larger Sharpe ratio, specifically in the range $[0.008, 0.021]$. Comparing these results with the ones we provided in Table 7.2 and Table 7.3, we conclude that the advantage of the Persistent $\mathfrak{P}$, Uniform $\mathfrak{U}$, and Normal $\mathfrak{N}$ agents we saw in terms of mean dollar reward L comes at the cost of high risk, increasing the Sharpe ratio S.

To better understand this phenomenon, we present a detailed example of a specific market configuration for one of the market simulation settings described above.[39] Figure 7.2 depicts the dollar reward $l_t, t \in [T]$ in a setting where the FQI_4 agent is competing with three Normal agents $\mathfrak{N}$ (i.e., corresponding to the 1-st column and 5-th row of Table 7.3). The shaded areas are the 95% confidence intervals for the values computed by statisti-

[39] We opt to provide this level of detail for a single configuration since the behaviour we show is common to all of them, therefore the analysis and comments would be the same.

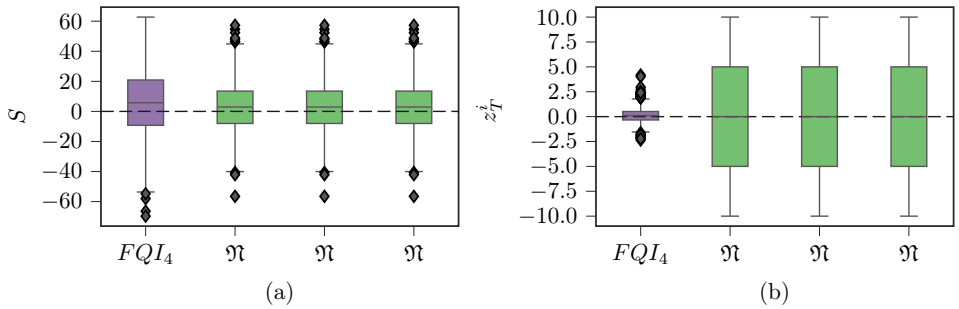

Figure 7.3: *Box-plot of the distribution over* $1,000$ *runs for (a) Sharpe ratio* S*; (b) final inventory* z_T^i*.*

	FQI_2	Q_2	$\mathfrak{P}$	$\mathfrak{U}$	$\mathfrak{N}$
FQI_2	**-11.203**	-19.188	-19.18	-21.129	-20.982
Q_2	**-0.287**	-16.695	-18.585	-20.271	-19.328
$\mathfrak{P}$	**-0.132**	-17.526	-17.998	-19.057	-17.809
$\mathfrak{U}$	**-0.322**	-16.408	-18.697	-17.849	-17.748
$\mathfrak{N}$	**-1.245**	-17.524	-18.375	-19.153	-18.848

Table 7.6: *Average reward* R *for* $M_t = 2$*. Larger is better.*

cal bootstrap over the $1,000$ independent runs of the setting. The results show that the FQI_4 agent achieves positive returns in the market simulation since its trend is increasing approximately linearly over time. Note that, also the Normal agents $\mathfrak{N}$ are generating profit, as the dashed lines corresponding to the average dollar reward are progressively increasing. However, differently from the FQI_4 agent, the Normal agents $\mathfrak{N}$ have a large confidence interval, indicating that they are not reliably achieving profits in each simulation. Conversely, looking at the confidence areas of FQI_4, we conclude that it consistently provides profit over time. This is due to the specifically crafted reward function used during the training phase, which allows it to maintain a smaller net inventory, thus reducing the standard deviation associated with large net inventory.

We now further analyse the scenario in the first column and fifth row of Table 7.3. We report in Figure 7.3 the box-plots associated with the distribution over the $1,000$ runs of the Sharpe ratio S (Figure 7.3 (a)) and the Inventory z_T^i at the final time T (Figure 7.3 (b)). In Figure 7.3 (a) the FQI_4 agent has a slightly larger distribution of the Sharpe ratio S compared to the three Normal agents $\mathfrak{N}$. However, it also has a larger expected value, which is consistent with the results in Table 7.2 and Table 7.3. In Figure 7.3 (b), the distribution of the final inventory z_T^i is smaller than the ones corresponding to the three Normal agents $\mathfrak{N}$. This is also consistent with the behaviour observed in Figure 7.2, and shows that for all the runs, the final inventory z_T^i was small. We recall that having small inventory constitutes a favorable characteristic of the learned strategy since it prevents the market makers from having large capital requirements as stated by the current legislation in most countries.

Table 7.6 and Table 7.7 present the average reward $R3$. We can see that in almost all

	FQI_4	Q_4	$\mathfrak{P}$	$\mathfrak{U}$	$\mathfrak{N}$
FQI_4, FQI_4, FQI_4	**-8.617**	-21.076	-20.473	-19.996	-21.521
Q_4, Q_4, Q_4	**-0.127**	-18.001	-19.951	-18.028	-17.909
$\mathfrak{P}, \mathfrak{P}, \mathfrak{P}$	**-0.162**	-17.355	-19.137	-17.999	-17.998
$\mathfrak{U}, \mathfrak{U}, \mathfrak{U}$	**-0.141**	-18.438	-17.66	-17.611	-18.085
$\mathfrak{N}, \mathfrak{N}, \mathfrak{N}$	**-0.333**	-17.562	-18.776	-17.62	-18.638

Table 7.7: *Average reward R for $M_t = 4$. Larger is better.*

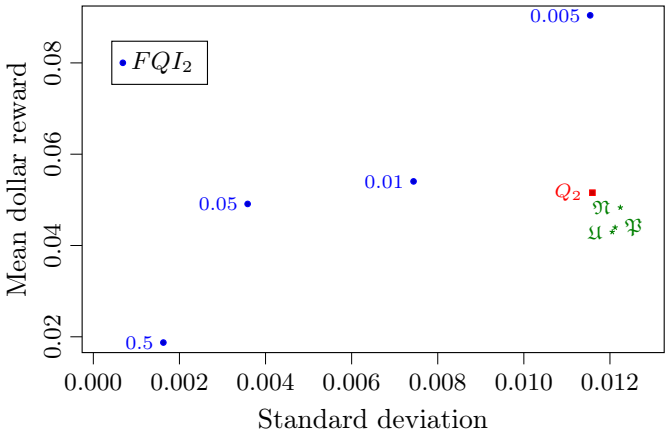

Figure 7.4: *Each dot represents the performance of the agent against $\mathfrak{P}$, with the mean dollar reward l_t on the y-axis and the standard deviation of l_t on the x-axis. For the case of FQI and Q, the label next to each dot represents the value of λ in the inventory penalization term $\phi(z) = \lambda z^2$.*

the instances, the FQI_2 and FQI_4 agents perform better than the other agents. Focusing on the case $M_t = 2$ (Table 7.6), over all the configurations, FQI_2 has a mean reward R of ≈ -11 and ≈ -0.1 against the FQI_2 and Q_2 agents, while all the other agents have a mean reward of ≈ -19.

Similarly, in the case of $M_t = 4$ (Table 7.7), we observe that the FQI_4 agent has a mean reward of in the range $[-8.6, -0.3]$, in all the configurations, while the other agents have a mean reward of ≈ -20 in all configurations. We remark that this table is presented only for completeness, as the agents $\mathfrak{P}$, $\mathfrak{U}$, $\mathfrak{N}$ are not explicitly optimizing the metric R, while the strategic agents are optimizing this value in their learning procedure. However, such table shows that the Algorithm 16 fails to learn optimal policies due to the large sizes of the Q-tables involved. This justifies the introduction of Algorithm 7, which solves this problem by using function approximation tool on the Q-tables, namely regression trees.

Finally, in Figure 7.4, we analyse the behavior of FQI when changing the risk aversion coefficient λ of the inventory penalization $\phi(z)$. In the figure, we are plotting the mean dollar reward l_t and the standard deviation of l_t for each agent against $\mathfrak{P}$ (row 3 of Table 7.2), in the setting $M_t = 2$. We can see that when decreasing λ the performance improves but

with increased risk. Furthermore, we can see that with $\lambda = 0.05$ we are already achieving a dollar reward similar to the benchmark agents but with a lower standard deviation.

7.5 Chapter Summary

In this chapter, we presented a novel solution to the problem of market making in dealer markets in the presence of strategic players. After defining the market making framework as a N-player stochastic game, we proposed a solution based on the framework of learning in MFGs, which assumes homogeneous market participants. To find the equilibrium, the policy and the mean-field are evolved in an iterative manner. We used FQI to optimize the policy at each iteration. Thanks to this training phase, the strategy learned this way can be executed without requiring further learning processes during the daily operations of the market maker. This approach is capable of handling the competitive nature of the problem, due to the presence of other market makers, by learning an equilibrium strategy in a multi-agent framework. After presenting the approach, we empirically tested the robustness of the equilibrium strategy in terms of multiple metrics. The experimental evaluation showed that, in the presence of strategic opponents, the method outperforms the other benchmark agents. Instead, in the presence of a generic market configuration, other agents might perform better when considering P&L. However, since proposed methods guarantee a *safe* behaviour, they are capable of achieving a lower risk in terms of a higher Sharpe ratio and a smaller inventory. This shows that the *learning in games* framework can lead to promising results when employed in the market making context.

An interesting future direction to explore is to add elements of realism to the problem, *e.g.* by considering a portfolio of correlated assets, or by training on real data. Moreover, one might consider other RL techniques, such as policy search methods, which might perform better from an empirical point of view, but at the expense of the theoretical guarantees and/or computational costs. Finally, our approach can be tested/compared to other strategies currently used in a production setting, to check the rationality level, from a game-theoretic point of view, of the players.

CHAPTER 8

Hedging Options with Risk-Averse RL

As described in Section 2.3.3, hedging the delta risk of options is a task common to options market makers. Delta hedging is executed in an automatic fashion in some of the more advanced trading companies by means of hedging algorithms. This framework, as we have seen in Sections 2.2.3 and 2.2.4, makes unrealistic assumptions such as continuous time hedging and no transaction costs when trading the underlying. Thus, blindly following the B&S delta hedge (afterwards we refer to it as simply delta hedge) can generate relevant costs. This chapter analyzes the delta hedging problem proposing an approach to manage the trade-off between risk and return by using RL.

In this chapter, we analyze two different delta hedging problems, starting with equity options (see Section 2.2.3), and then turning to credit index options (refer to Section 2.2.4). Hedging is a form of risk management that requires being risk-averse, thus it is necessary to optimize for a risk-averse objective function. Risk-averse RL has been the object of extensive literature, which was analyzed to select the appropriate algorithm. The chosen algorithm, Trust Region Volatility Optimization (TRVO) (Bisi et al., 2020b), a risk-averse variant of TRPO (Schulman et al., 2015), is presented in Section 8.2. TRVO not only optimizes a risk-averse objective but, being a policy search algorithm, is also natively compatible with continuous states and actions, a necessary property to handle the hedging environments. We solve both hedging problems using TRVO. With the same technology, we also address DVA hedging. DVA is a hybrid risk with no possibility of trade the underlying generating one of the risks. Given the complexity of the problem, there are very preliminary results, thus we only explain the methodology in Appendix C.

In Chapter 7 we covered the pricing task in the case of bonds. A very similar task exists

in the case of options. Indeed, options market makers often find themselves with a net positive or negative inventory of options for which the delta needs to be hedged. Hedging the delta in a realsitic setting is the topic explored in this chapter. This topic has a long history and following Black and Scholes (1973), several approaches have been proposed to extend the B&S model to account for realism, starting with Leland (1985) and more recently Guéant and Pu (2017). In this chapter we take a different perspective, proposing a model free risk-averse RL approach to solve the problem of hedging the delta. Being model free, it is independent from the model generating the prices of the underlying, we use a GBM model as a proof of concept, but also verify the robustness of the approach by testing the behavior on a Heston simulated market and, finally, on real data.

Chapter outline The chapter begins with an overview of the state-of-the-art on option hedging using RL, as the background is common for both topics. Then, in Section 8.2, we introduce the Trust Region Volatility Optimization (TRVO) algorithm. Once the common elements have been described, we elaborate on each of the hedging problems, starting with equity option hedging in Section 8.3 and credit index option hedging in Section 8.4. In both cases, we focus our attention on the promising experimental results.

8.1 Background on RL for Hedging

The issue of delta hedging using RL has been analyzed by various authors. Among the most recent approaches we mention Du et al. (2020); Kolm and Ritter (2019); Buehler et al. (2019); Halperin (2017, 2019); Cao et al. (2019). These papers can be subdivided into two categories, one addresses the problem from a practitioner's perspective and is focused on the details of the hedging strategies chosen by the agent; the other builds on the formal mathematical structure of option pricing and uses machine learning techniques to overcome the problems posed by realistic features such as transaction costs. The distinction is faint as a hedging strategy implies a price, and vice versa.

The first category includes Kolm and Ritter (2019); Cao et al. (2019) and is also pertinent for this chapter. The most comparable, regarding the financial environment, are Du et al. (2020); Kolm and Ritter (2019), which use the same MDP formulation considered in this dissertation. The main difference consists in the use of an approximate variance formulation in the RL objective, compared to the full variance used in this chapter. Furthermore, Kolm and Ritter (2019), uses a one-step SARSA update, a value based approach (see Section 3.3.1), instead of a policy search method, while Du et al. (2020) considers both DQN (Mnih et al., 2013) and PPO (Schulman et al., 2017). Cao et al. (2019) also consider an environment very close to ours, but with a transaction costs size that is ~ 20 times more than what we considered. Regarding the RL algorithm, they use value-function methods and, in particular, risk-averse deep Q-learning. It is an advanced approach taken from the risk-averse reinforcement learning literature (Tamar et al., 2016). They consider two Q functions, one for the first moment and another for the second moment. The paper then focuses on the agent's efficiency as a function of the rebalancing frequency. Differently to this chapter where we also analyze what happens when changing the risk-aversion parameter, in Kolm and Ritter (2019); Cao et al. (2019), only a single value of risk aversion is tested.

The second category includes Halperin (2017, 2019); Buehler et al. (2019). In Halperin (2017, 2019), the problem of option pricing in discrete time has been addressed from a machine learning perspective, neglecting hedging costs. In Buehler et al. (2019), the option pricing problem is undertaken by considering a class of convex risk measures and embedding them in a deep neural network environment. Initially, the dependence of the option price and hedge on the risk aversion parameter is studied in the absence of transaction costs. Then, a study of the option price dependence on transaction costs is discussed and the functional dependence of the price on the cost parameter is reconstructed.

What distinguishes our approach is the algorithm we considered: the risk-averse policy search algorithm TRVO. One of the advantages of TRVO compared to value based algorithms like the ones used by Kolm and Ritter (2019); Cao et al. (2019); Halperin (2017) is the fact that being policy search, TRVO is natively compatible with continuous states and actions and thus does not suffer from the problems of using a function approximator. Furthermore, being risk-averse, it is not necessary to apply any transformation to the reward differently from what is done for example in Kolm and Ritter (2019) and it is able to create a policy specific on the risk aversion of the user. Moreover, an advantage of model free RL algorithms, is that the policy learned is independent from the model used to generate the data. Thus TRVO can be used as is in an option hedging framework, and only requires the standard hyperparameter tuning typical of RL algorithms.

8.2 Risk Aversion in RL with TRVO

In Section 4.5, we introduced the concept of reward volatility (Equation 4.11) as a performance metric. In this section, we analyze how to insert this metric in the objective function, thus optimizing for a risk-averse objective.

In the RL framework, there are two main sources of risk: the first is the inherent risk, which is generated by the stochastic nature of the environment, while the second is the model risk, which is related to the imperfect knowledge of the model parameters. In a certain way, the inherent risk is related to the distribution of the results once a policy is selected, while the model risk is related to the safety in the learning process, which is desired to monotonically improve the results. Several ways of minimizing inherent risk have been taken into consideration in the RL literature with many different approaches (García and Fernández, 2015): employing a utility function for the return (Shen et al., 2014), changing the objective function, or adding a constraint (Tamar et al., 2015). A number of modified objectives have been studied, for example the minimization of variance of the returns in a mean-variance (Tamar and Mannor, 2013; Prashanth and Ghavamzadeh, 2014) or Sharpe ratio (Moody and Saffell, 2001) fashion. Another example is a family of well-behaved risk measures, which includes CVaR, called coherent risk measures (Tamar et al., 2017). Nevertheless, all these approaches consider only the minimization of the long-term risk, while in financial trading interim results are also fundamental, and keeping a low-varying intermediate P&L becomes crucial. Moreover, the analytical intractability of all these formulation does not allow the related algorithms to perform (in terms of learning improvements) as the state-of-the-art algorithms in the standard RL framework, such as TRPO (Schulman et al., 2015). For these reasons, we introduced in Bisi et al. (2020b) a new risk measure, which takes into account the variance of the reward at each time-step with respect to state visitation probabilities and called it *reward volatility*.

In most trading and even hedging applications, achieving a profit is at least as relevant as being risk-averse thus, we decide to consider an objective that handles the risk-return trade-off through a risk aversion coefficient, the parameter λ. Recalling the J_π function of Equation (3.7): $J_\pi \coloneqq (1-\gamma)\,\mathbb{E}_{\pi\, s_0\sim\mu}\left[\sum_{t=0}^{T}\gamma^t r_t\right]$, and reward volatility of Equation (4.11): $\nu_\pi^2 \coloneqq \mathbb{E}_{\substack{s\sim d_{\mu,\pi}\\ a\sim\pi(\cdot|s)}}\left[\left(r(s,a)-J_\pi\right)^2\right]$ the objective related to the policy π can be defined as:

$$\eta_\pi \coloneqq J_\pi - \lambda\nu_\pi^2, \tag{8.1}$$

called *mean-volatility* hereafter, where $\lambda \geq 0$ allows to trade-off expected return maximization with risk minimization. Similarly, the mean-variance objective is $J_\pi/(1-\gamma) - \lambda\sigma_\pi^2$, where σ_π is the return variance defined in Equation (4.9): $\sigma_\pi^2 \coloneqq \mathbb{E}_{\pi\, s_0\sim\mu}\left[\left(G_\tau - \frac{J_\pi}{1-\gamma}\right)^2\right]$ An important result on the relationship between the two variance measures is the following:

Lemma 8.1. *Consider the return variance σ_π^2 (Equation* (4.9)*) and the reward volatility ν_π^2 defined in (Equation* (4.11)*). The following inequality holds:*

$$\sigma_\pi^2 \leq \frac{\nu_\pi^2}{(1-\gamma)^2},$$

It is important to notice that the factor $(1-\gamma)^2$ comes from the fact that the return variance is not normalized, unlike the reward volatility. What is lost in the reward volatility compared to the return variance are the inter-temporal correlations between the rewards. However, Lemma 8.1 shows that the minimization of the reward volatility yields a low return variance. The opposite is clearly not true: as counterexample, it is possible to consider a stock price with the same value at the beginning and at the end of the investment period, but making complex movements in-between.

The main advantage in considering this measure consists in its analytical tractability, thanks to which it is possible to derive linear Bellman equations, in a similar form as in Equation (3.6), and a policy gradient theorem, analogous to the standard RL framework. Indeed, we can introduce a volatility equivalent of the action-value function Q_π (Equation 3.5), called *action-volatility* function, which is the volatility observed by starting from state s, taking action a, and following policy π thereafter:

$$X_\pi(s,a) \coloneqq \mathop{\mathbb{E}}_{\substack{s_{t+1}\sim P(\cdot|s_t,a_t)\\ a_{t+1}\sim\pi(\cdot|s_{t+1})}}\left[\sum_{t=0}^{\infty}\gamma^t(\mathcal{R}(s_t,a_t)-J_\pi)^2|s,a\right].$$

Like the Q_π function, this can be written recursively by means of a Bellman equation:

$$X_\pi(s,a) = \left(R(s,a)-J_\pi\right)^2 + \gamma \mathop{\mathbb{E}}_{\substack{s'\sim P(\cdot|s,a)\\ a'\sim\pi(\cdot|s')}}\left[X_\pi(s',a')\right].$$

We can define also the *state-volatility* function W_π as the expected value of X_π under the policy π, *i.e.,* the equivalent of the V function (Equation 3.4) for volatility.

The linearity of this Bellman equation allows an alternative interpretation of the mean-volatility objective. In fact, by applying a reward transformation $R_\pi^\lambda(s_t,a_t) = R(s_t,a_t) -$

Algorithm 8: Trust Region Volatility Optimization

Initialize: policy parametrization $\boldsymbol{\theta}_0$, batch size N, number of iterations K, discount factor γ.
for $k = 0, \dots, K-1$ **do**
  Collect N trajectories with $\boldsymbol{\theta}_k$
  Compute estimates of the risk neutral objective $\widehat{J}$
  Estimate advantage values $A^{\lambda}_{\boldsymbol{\theta}_k}(s,a)$
  Solve the constrained optimization problem

$$\boldsymbol{\theta}_{k+1} = \arg\max_{\boldsymbol{\theta}\in\Theta} \left[L^{\lambda}_{k}(\boldsymbol{\theta}) - \frac{2\epsilon\gamma}{1-\gamma} D^{max}_{KL}(\pi_{\boldsymbol{\theta}_k}, \pi_{\boldsymbol{\theta}}) \right]$$
$$\text{where } \epsilon = \max_{s} \max_{a} |A^{\lambda}_{\boldsymbol{\theta}_k}(s,a)|$$
$$L^{\lambda}_{k}(\boldsymbol{\theta}) = \eta_{\boldsymbol{\theta}_k} + \mathop{\mathbb{E}}_{\substack{s\sim d_{\mu,\pi_k} \\ a\sim\pi_{\boldsymbol{\theta}}(\cdot|s)}} A^{\lambda}_{\boldsymbol{\theta}_k}(s,a)$$

$\lambda(R(s_t,a_t) - J_\pi)^2$, it is possible to formulate the problem as a standard RL problem, where X and W functions are reduced to Q and V.

Thanks to the similarity to the standard framework, we are allowed to include all the value functions to define the mean-volatility advantage function $A^{\lambda}_{\pi}(s,a)$:

$$A^{\lambda}_{\pi}(s,a) \coloneqq \Big(Q_\pi(s,a) - \lambda X_\pi(s,a) \Big) - \Big(V_\pi(s) - \lambda W_\pi(s) \Big)$$

At this point, it is possible to adopt a risk-averse version of the TRPO algorithm described in the previous section: we can consider a surrogate function $L^{\lambda}_{\pi_{\theta_{old}}}(\pi_{\theta_{new}})$, of the parametrized policy π_θ, that approximates the gain in terms of performance of θ_{new} with respect to θ_{old}:

$$L^{\lambda}_{\pi}(\widetilde{\pi}) \coloneqq \eta_\pi + \int_{\mathcal{S}} d_{\mu,\pi}(s) \int_{\mathcal{A}} \widetilde{\pi}(a|s) A^{\lambda}_{\pi}(s,a)\, \mathrm{d}a\, \mathrm{d}s.$$

Thus, we can define the Trust Region Volatility Optimization (TRVO) algorithm where, as in the classic TRPO, the optimization of the objective function is performed in an iterative manner, and each policy update has the constraint on the KL-divergence of two consecutive policies. The pseudo algorithm is summarized in Algorithm 8.

8.3 Equity Option Hedging with RL

This section presents the results achieved by learning the equity option hedging framework, illustrated in Section 2.3.3, with TRVO. This is a generic framework, where the underlying can be a generic equity instrument such as a stock (see Section 2.2.1) or a future (see Section 2.2.2). Equity options are defined in Section 2.2.3 and can be embedded in an MDP where:

- the action a_t is the current hedge portfolio,
- the state $s_t = (S_t, C_t, \frac{\partial C(t,S_t)}{\partial S}, a_{t-1})$,
- the reward $r(s_t, a_t)$ as defined in Equation (2.18).

Regarding the financial framework, we consider a single underlying generated with GBM as explained in Section 4.3.1. There are 5 prices per day. The option has unitary notional and is At The Money (ATM) with a 60 day maturity.[40] The starting price of the underlying is 100 and the annual volatility is 20%, approximately that of an equity option in normal market conditions. With these characteristics, the option price value at inception is ~ 3.24 (also referred to as premium) and the corresponding delta ~ 0.5. 305 time-steps means 61 days, where day 61 is the day the option expires. On the last day, small movements of the underlying around the strike price can cause the delta to jump between 0 and 1, this is called *pin risk*. Finally, we simplify a zero financing cost. In this section, we refer to P&L := $\sum_t r_t$ and P&L volatility is $\sigma := \sqrt{\mathbb{V}\text{ar}[\text{P\&L}]}$, where r_t is defined as defined in Equation (2.18). We remark that this is a proof of concept, with the objective of showing that with risk-averse RL it is possible to learn a realistic delta hedging strategy. For simplicity, we consider an underlying which behaves as a GBM, but the model used to generate the underlying should not impact on the results, ideally should use real data to train and test our algorithms.

8.3.1 Experimental Results

We start the section by considering option hedging in discretized time without hedging costs, then we introduce transaction costs and finally stress our models by testing them on options with different strikes, different volatility, and even portfolios of options. Training is performed on 10,000 scenarios, with ∼2,000 episodes where each batch has ∼120,000 steps, the discount factor is 0.999, max KL is 0.001. The policy is a neural network (see Appendix B.4.3 for more details on neural networks) with 2 hidden layers with 64 neurons. Testing is an average of 2,000 out-of-sample scenarios. All figures present the results of those 2,000 scenarios unless specified otherwise.

Training without transaction costs

In this section we analyze the performance of TRVO on a cost free environment and compare it with the delta hedge. As we can see in Figure 8.1, time discretization injects some volatility in the delta hedge (red bars), but the generated wealth is very small with respect to the option premium (~ 3.24). In fact, the delta hedge remains optimal and the agent learns to reproduce it (as we can see in Figure 8.2). It is clear that the hedging strategy is able to exploit the change in delta typical of a long option position, to compensate for the time decay of the option premium. The presence of a risk aversion factor, even if slightly weighted, forces the algorithm to reduce the volatility as much as possible, thus there is no frontier and everything condenses to the delta hedge.

[40] We have chosen an ATM option because it is the most traded type and generates the most interesting delta behavior. Nevertheless, in Section 8.3.1 we show that an agent trained on ATM options is able to hedge options with different moneyness.

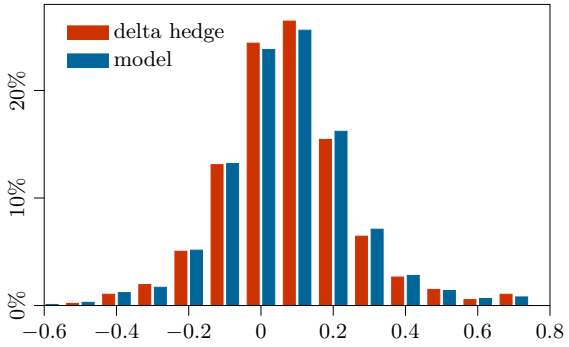

Figure 8.1: *P&L distribution without transaction costs.*

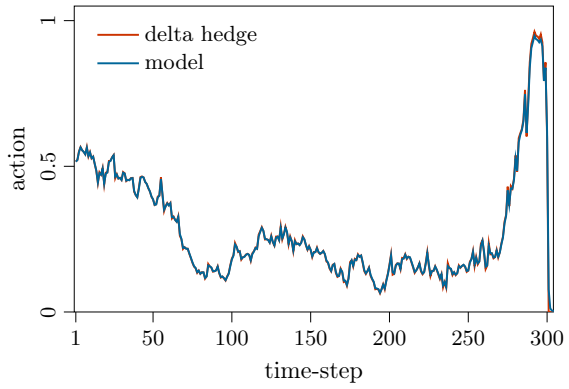

Figure 8.2: *Single scenario, no transaction costs with also an example of pin risk handled appropriately by the agent.*

Training with transaction costs

In this section, we consider costs along the line of (Kolm and Ritter, 2019) as specified in Equation (2.23) with a mid-ask spread of 0.05. The choice of the mid-ask is such that the costs replicate those of listed stocks (the Euro Stoxx Banks futures - see Figure B.1, and FTSE MIB futures, renormalizing the underlying to 100, have a rescaled mid-ask spread of ~ 0.05. A more liquid index such as the S&P 500 mini futures contract, has a typical mid-ask ~ 0.01). We picked risk-averseness parameter of the objective $\eta = J + \lambda\nu^2$ by measuring the typical values assumed by the reward volatility and the average P&L. For the specific environment at hand we found $0.2 \lesssim \lambda \lesssim 20$ as the most interesting range.

The average of the costs generated by the delta hedge on the test set is ~ 0.286, which is $\sim 9\%$ of the option premium.

In Figure 8.3, we can see that lower risk aversions generate lower costs. This can be clearly seen also in Figure 8.4, where the yellow bars show that the distribution of costs generated by an agent trained with $\lambda = 2$ are much lower than those generated by the delta hedge average. This is even more evident with $\lambda = 0.5$ (green bars) and $\lambda = 0.82$ (blue bars), where costs are even lower. Increasing the risk-aversion parameter leads the

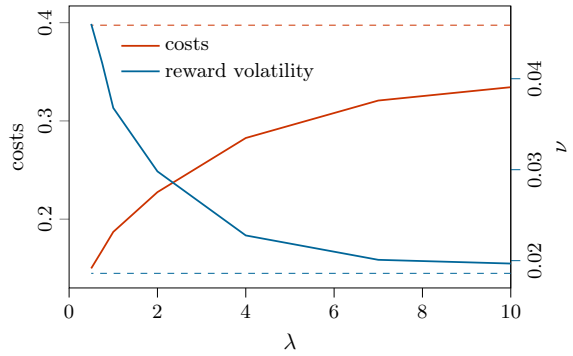

Figure 8.3: *Hedging costs (red) and reward (blue) experienced by the TRVO agent, as functions of the training risk aversion λ. The dotted lines represent the hedging cost (red) and reward volatility (blue) of the delta hedge. Each point is calculated on a single scenario, the same of Figure 8.2.*

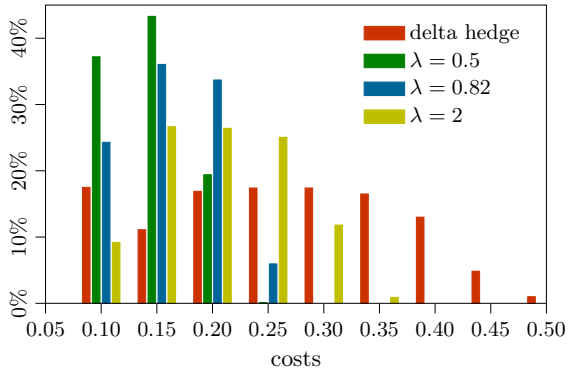

Figure 8.4: *Hedging costs generated by delta hedge (red) and RL agent trained with different risk aversion parameters.*

agent to behaviors more adherent to the delta hedge strategy, which is essentially recovered for $\lambda \sim 20$. This is evident from Figure 8.3, in which the costs due to the actions of the RL agent (red line) approach the costs realized by the delta hedge (dashed red line) as λ increases.

How the agent reduces hedging costs

In light of these results, we tried to understand which strategy was chosen by the agent to reduce costs. The essence is that, even with a very low risk-aversion, the agent tries to control the P&L volatility by mimicking the delta hedge strategy, but by delaying and reducing the action, in line with what is also described in Cao et al. (2019). This means that the agent does not act immediately when the delta spikes up (down), but waits to see if, due to market movements, the delta returns to the previous lower (higher) values. In case the delta spikes up more, surpassing the agent's "comfort zone", the agent covers the

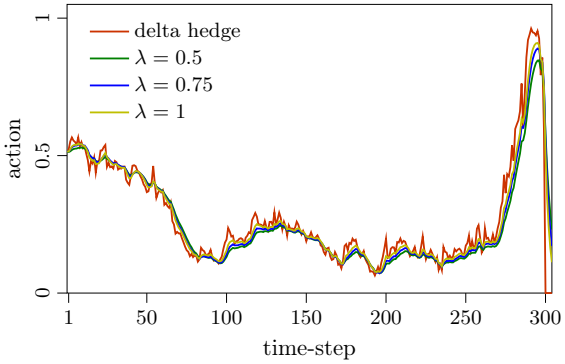

Figure 8.5: *Comparison between the delta hedge and the agent's actions with different risk aversion parameters λ. Same scenario as figure 8.2 but with hedging costs.*

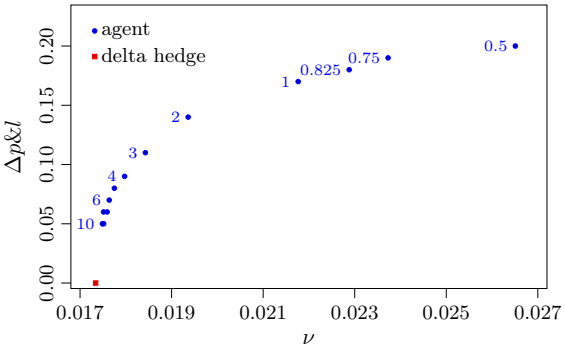

Figure 8.6: *Efficient frontier of the TRVO agent at different risk aversions on the P&L / reward volatility space.*

position. Having studied an agent trained with different risk-aversion levels, we are able to show how this comfort zone depends on the risk aversion parameter: it is very wide for lower values and very tight (or essentially zero) for higher values.

This behavior is well represented in Figure 8.5, where the red line represents the delta hedge, while the other lines represent the action of the agent trained with different risk aversions. For lower risk-aversions, the action is smoother and expresses a significant delay with respect to the delta hedge. For higher values of λ, not shown for the interpretability of the figure, the agent's action is consistently more adherent to the delta hedge, confirming the behavior that could be supposed from Figure 8.2. Comparing Figures 8.2 and 8.5, it is clear how the same risk aversion with different costs gives different hedging strategies.

Efficient frontier

The interplay between reward volatility reduction and cost minimization can be analyzed by observing the efficient frontier in Figure 8.6, where we plotted a point for each value of the risk aversion parameter λ, in the ν-P&L space. The y-axis is not the pure P&L, but

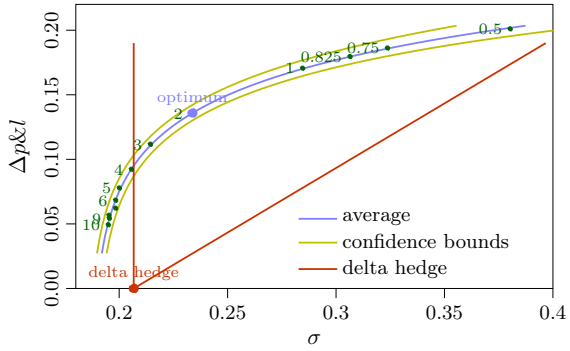

Figure 8.7: *Efficient frontier of the TRVO agent at different risk aversions on the P&L / P&L volatility space.*

the difference of the P&L of the agent with respect to that of the delta hedge. As expected, increasing the risk-aversion coefficient lowers the reward volatility and the P&L. The point in red shows the average wealth and reward volatility experienced by following the delta hedge strategy. Reward volatility is a relevant risk metric to a trading strategy given that, in real life, a portfolio will experience a single scenario: if the loss is too large, the trader may be tempted, or forced, to adopt stop-loss strategies, hampering any chance of profiting from an otherwise properly trained agent. This is the main financial reason why we chose the TRVO agent. Nevertheless, a posteriori, it is important to evaluate the performance of the hedge over the entire life of the option. This can be seen in Figure 8.7 where a point in the σ-P&L space is plotted for each value of λ. As before, the y-axis indicates the P&L of the agent with respect to that of the delta hedge. To measure the uncertainty of the learning model, we performed 20 trainings for each value of λ. The blue line is a logarithmic best fit of the obtained frontier, while the yellow lines provide an indication of the 1-σ (68%) confidence interval. The red dot represents the delta hedge, with performance defined as being zero. The frontier points laying on the left of the vertical red line strictly dominate the delta hedge, since the corresponding agents perform better both in terms of P&L and in terms of volatility. Those on the right, instead, while performing even better in terms of P&L, induce a volatility increase, thus, their relevance depends on the trader's risk aversion. As an example, a trader valuing the volatility increase on the same footing as the P&L gain (a completely arbitrary choice, of course) will consider the whole frontier as an improvement with respect to the delta hedge, since it lays above the diagonal red line, indicating the region of the space where the P&L gain is equal to the volatility increase. That trader will consider the blue point, which has tangent line parallel to the diagonal red line and is very close to the $\lambda = 2$ point, as an optimum. We stress that, whatever risk aversion is chosen, there is a corresponding frontier point performing better than delta hedge.

P&L distribution

One may wonder about the statistical significance of a gain of ~ 0.15 with respect to the delta hedge if the P&L-volatility is of the same order of magnitude. We believe this is

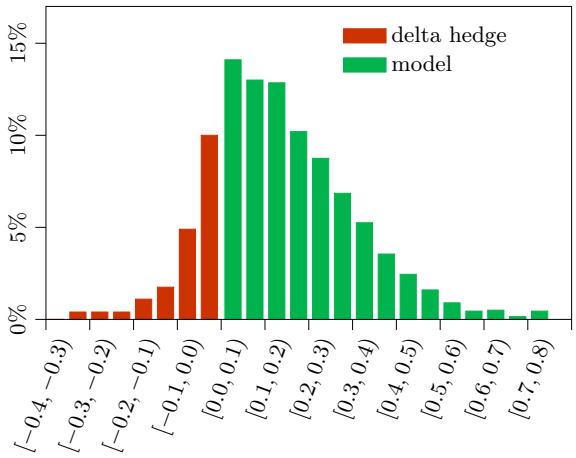

Figure 8.8: *Distribution of the P&L performance of a TRVO agent with $\lambda = 2$ over the delta hedge.*

the case, and support our claim by showing, in Figure 8.8, how the agent performance is distributed in the case $\lambda = 2$, where the green (red) bars show the distribution of scenarios in which the agent performs better (worse) than the delta hedge. Assuming that a performance better than that of the delta hedge *is not always ensured*, one has to observe that in 81% of the scenarios the performance is superior and that the average superior performance, which is 0.15, is more than ten times the average of the worse performance, which is -0.014, and even more of the absolute value of the 5-th percentile of the distribution, which is -0.09. The t-test on the data used for the histogram gives more than a 99.9% confidence that the average of the model is greater than the average of the delta hedge.

Strengths and limits of the approach

In the previous section, we presented the performance of a TRVO agent where the option to be hedged had the same characteristics, as the one on which the agent had been trained. In this section, we consider the same agents, without any type of re-training, and test them on options characterized as follows:

- single in the money (strike 105) option, 60 days maturity,
- single out of the money (strike 95) option, 60 days maturity,
- single at the money option, 60 days maturity, tested on scenarios where the realized volatility is 30% (to be compared with the training set scenarios having volatility 20%),
- portfolio of options with different moneyness (strikes from 90 to 110), 60 days maturity.

The results of the tests are summarized in Table 8.1, by using as performance indicators the difference between the agent and delta hedge P&L (ΔP&L) (as in Figure 8.7) and the

	STD option		ITM option		OTM option		High vol		Portfolio	
λ	ΔP&L	$\Delta\sigma$	ΔP&L	$\Delta\sigma$	ΔP&L	$\Delta\sigma$	ΔP&L	$\Delta\sigma$	ΔP&L	$\Delta\sigma$
0.5	0.2	0.17	0.15	0.17	0.17	0.21	0.2	0.37	0.15	0.22
0.75	0.19	0.12	0.14	0.13	0.15	0.15	0.19	0.23	0.14	0.14
0.82	0.18	0.1	0.13	0.1	0.15	0.12	0.18	0.2	0.13	0.12
1	0.17	0.08	0.13	0.08	0.14	0.1	0.17	0.17	0.13	0.1
2	**0.14**	**0.03**	**0.1**	**0.02**	0.12	0.03	0.13	0.08	**0.1**	**0.05**
3	0.11	0.01	0.08	0	**0.1**	**0**	**0.11**	**0.04**	0.08	0.03
4	0.09	0	0.07	-0.01	0.08	-0.01	0.09	0.03	0.07	0.03
5	0.08	-0.01	0.06	-0.02	0.07	-0.01	0.07	0.02	0.06	0.02
6	0.07	-0.01	0.05	-0.02	0.06	-0.01	0.06	0.02	0.05	0.02
7	0.06	-0.01	0.05	-0.02	0.05	-0.01	0.06	0.02	0.05	0.02
8	0.06	-0.01	0.04	-0.02	0.05	-0.01	0.05	0.01	0.04	0.02
9	0.05	-0.01	0.04	-0.02	0.05	-0.01	0.05	0.01	0.04	0.02
10	0.05	-0.01	0.04	-0.02	0.04	-0.01	0.04	0.01	0.03	0.02

Table 8.1: *Behavior of agent on a test environment where the option characteristics have been modified.*

difference between the agent and delta hedge P&L volatility ($\Delta\sigma$). An optimum (in bold) is identified by looking at the risk aversion-value such that the P&L gain obtained by varying it is equal to the volatility increase (i.e. at the optimum the frontier has a tangent line with unitary steepness).

Table 8.1 proves the robustness of the presented approach, and we believe that a single training may be sufficient to properly hedge any portfolio of options for a given maturity. In fact, given the definition of the state at the beginning of Section 8.3, the agent learns a hedging strategy that is independent of the number of options, their strike, and also on the behavior of the market. Notice also that the λ realizing the optimum (with the same considerations as Section 8.3.1) does not change significantly in the table, indicating that a given λ consistently realizes a certain risk - P&L balance, at least for the hedging cost level we adopted.

One could argue that our training and testing is built on a really simple market generation model, *i.e.* GMB with constant volatility. We believe that a strategy able to deal with a volatility change of around 50% (from 20% to 30%) is already robust and that a real improvement in the strategy would require a massive injection of reality, both on the option and underlying details, and on the market data generation. Such reality boost could take advantage of AI-based approaches, e.g., the use of a generative adversarial network (Goodfellow et al., 2014), as in Kondratyev (2018), or other approaches such as the Restricted Boltzmann Machine described in Kondratyev and Christian (2019) to be more adherent to real-world data. We leave this extension for future work.

We also tried to understand whether this robustness could be extended to the management of portfolios of multiple options with different maturities. Unfortunately, our first attempts seem to indicate that it is extremely difficult to train the agent to learn that, from a certain point on, one or more options expire and so hedging that portion of the portfolio is no longer necessary (and thus the total portfolio delta is not in $[0, 1]$ anymore but in $[0, x]$ where $x < 1$). In our view, the essential point is that the price-value-delta relationship is

broken after expiry, and even the use of expiry signals to inform the agent about the fact that something changed in the game was not able to solve this point completely. However, we must observe that for most of the options traded on the market, the available option maturities are not so many. Thus, it is perfectly viable to split the whole portfolio into maturity sets, each of which is managed by a different instance of the same agent.

Managing increasing hedging costs

Until now, we considered hedging costs given by Equation (5.5) with mid-ask $= 0.05$. Less liquid or less standard listed contracts may have a significantly higher mid-ask , while OTC instruments, such as swaps and CDS, which are perfectly viable option underlyings, have even higher transaction costs (see Section 8.4). For this reason, we verified what happens with costs with mid-ask $= 0.2$. In such a setting, the average cost of the delta hedge is fourfold (from ~ 0.286 to ~ 1.2), as well as the cost distribution width. This enhances the advantage of a parsimonious agent: it is possible to draw an efficient frontier in the P&L/reward volatility space similar to Figure 8.6, just with an increase in the y-axis from 0.2 to 0.8. Nonetheless, a greater width for the cost distribution means a greater P&L volatility induced on the delta hedge *by the hedging costs*, which become the predominant source of volatility. This effect on P&L volatility is such that an agent, when reducing hedging costs, may be able to reduce the P&L volatility as well.

In this sense, in the presence of higher hedging costs, a winning strategy seems to be *decreasing* the risk aversion.[41] In fact, as also mentioned in Cao et al. (2019), an optimal strategy in case of very high costs may require no hedging at all. We also tested the (extreme) case of mid-ask $= 0.5$, where the described behavior is enhanced even further, essentially recovering the results of Cao et al. (2019). As mentioned, the agent is able to outperform delta hedging both in P&L and in P&L volatility, we stress that even in this extreme case the relative outperformance depends on the risk aversion parameter, which in Cao et al. (2019) was chosen as a fixed parameter ($\lambda \sim 1.5$ using our language).

8.4 Credit Index Option Hedging with RL

This section also focuses on option hedging, but it considers a credit index instead of an equity instrument. Credit indexes and credit index options are defined in Section 2.2.4. The main differences with the equity instruments considered in the previous section is that credit indexes are OTC instruments, while stocks and futures are listed on exchanges (see Section 2.1.1). Hence, also transaction costs are handled differently as we saw in Section 2.4. Recalling Section 2.3.3, the credit index option hedging framework can be embedded in an MDP where:

- the action a_t is the current hedge portfolio,
- the state $s_t = (S_t, \text{Pay}_t, N_h(t), a_{t-1})$,
- the reward is defined in Equation (2.19).

Regarding the financial framework, we simulated only the traded credit spread S, by using the GBM described in Section 4.3.1 with σ, the annualized volatility, equal to 60%

[41] This does not necessarily imply decreasing the risk aversion parameter λ, which is not dimensionless.

and neglecting the drift term. We did not consider the possibility of a default of one of the components as no default has been observed in recent times for the instrument in consideration. We trained our agents on generated data, with episodes of 40 working days, with 17 observations per day, beginning at 9.30 and ending at 17.30. In each simulation, the underlying credit spread starts from an initial value of 100 bps; we define the stochastic evolution on the actual time span between the time-steps: 30 minutes during the day, 16 hours between the last step of one trading day and the first step of the next trading day in case of two contiguous trading days, a span of $16 + 24n$ hours in the case of trading days separated by n holidays or weekend days.

We trained our agents to hedge a position long a payer (but any other position would have been equivalent) option with two months maturity, thus maturing at the end of each episode. We considered a strike K of 100 bps, equal to the initial value of the underlying at the beginning of the episode. We assumed an option notional of €100mln, which implies a hedging portfolio containing an underlying notional between €0 and €100mln. Given the market structure, our results are valid even assuming an option notional up to 10 times larger. We also assumed continuous underlying trading, which is reasonable given the option size and the fact that in the market small clips (down to €100k) can be traded. Assuming a risk neutral volatility equal to 60% the option has initial value of €530k.

8.4.1 Experimental Results

In this section we present the experimental results. Once described the data generation and training parameters, we show the results obtained on a GBM simulated market and analyze the robustness of the learnt policy testing also on a market simulated with a Heston model and finally on real market data.

We built a training set of 40,000 episodes, and trained our agents while varying two parameters: the risk aversion parameter λ and the bid-ask. In this section, we also refer to bid-ask as the ba parameter. It is important to notice that in this case, the bid-ask spread refers to the credit spread of the index, and not the price as in the equity case. We considered λ similarly to the previous section, so to span an efficient frontier in the risk-reward space. The values of lambda are $10^{-6} \lesssim \lambda \lesssim 10^{-3}$ but, for better interpretability, in the rest of the paper we rescale λ by 10^5, so to have bounds between 0.1 and 100. The choice of ba as an extra parameter comes from the observation that the bid-ask spread of the instrument considered here shows a higly dynamic pattern. We considered ba ranging from 0.5 to 2 as per Figure 4.2. We also considered the case with low values of ba, even $ba = 0$ to further test our algorithms and to check that the standard delta hedging strategy is smoothly recovered in the limit $ba \to 0$. The relationship between ba and transaction costs is represented by Equation (2.24).

Testing on a GBM-simulated Market We tested our agents on a dataset of 2,000 episodes with the underlying credit spreads generated by a GBM, with the same parameters of the training dataset. We performed different tests varying the ba spread to monitor agents' performances comparing to the delta hedging strategy.

In the $ba = 0$ case, the trained agent perfectly replicates the delta hedge. This can be seen in Figure 8.9, where, for a specific testing scenario, the delta hedging strategy (in red) is compared with the action chosen by the agents trained with different values of the risk

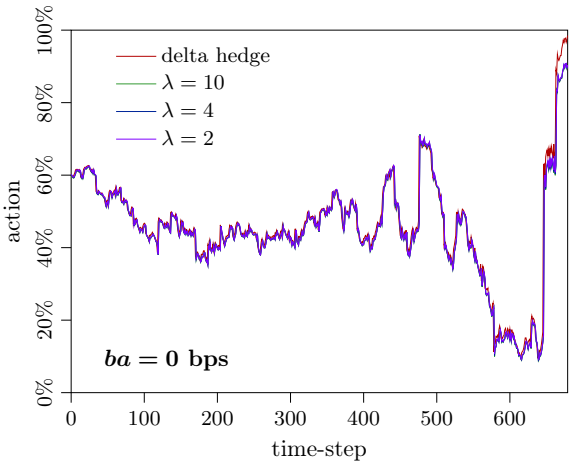

Figure 8.9: *The hedging strategy chosen by agents trained at different values of the risk aversion parameter λ is compared with the delta hedging strategy in a zero-cost environment. The vertical axis represents the hedging notional as a percentage of the option notional.*

aversion parameter (in green, blue, and purple). Given the absence of trading costs, all the agents replicate the same strategy, which is the optimal one, minimizing risks.[42] Under the $ba = 0$ assumption, the strategy has zero cumulated P&L on average.

Introducing hedging costs $ba > 0$, the average cumulated P&L of the delta hedging strategy is shifted to negative values, depending linearly on ba, specifically, considering a ba of 1 bp the cumulated P&L is on average -€136 k. The presence of hedging costs during training induces a smoother strategy for the agent, in terms of underlying allocation changes. Since each action becomes more expensive as ba increases, the agent cuts costs through the reduction of portfolio rebalances. The downside of this approach consists in an increase in the variability of the rewards, since the option is not continuously hedged. The desired balance between cost reduction and low reward volatility, which depends on the trader's preference, can be achieved by changing λ. This relationship is plotted in Figure 8.10, where different degrees of smoothness in the variation of the hedging portfolio can be seen to be dependent on λ. The smoothness degree depends also on the size of the hedging cost: defining a certain risk aversion, a higher ba implies a higher smoothness, as is apparent by comparing the upper and lower plot.

Figure 8.11 summarizes the performance of the agents with respect to the delta hedging strategy in terms of cumulated P&L for different values of λ and the ba parameter. In the figure, each dot represents the performance of an agent with the λ indicated by the nearby annotated number and acting in an environment with ba depending on the color. The position on the vertical axis indicates the average P&L performance of the agent with respect to the delta hedging strategy in an environment having the same ba. The average is taken with respect to the terminal P&L measured on the 2,000 testing scenarios. The position on the horizontal axis, instead, indicates the square root of the variance of the

[42] Indeed, neglecting hedging costs, the B&S paradigm is violated only by the assumed time discretization.

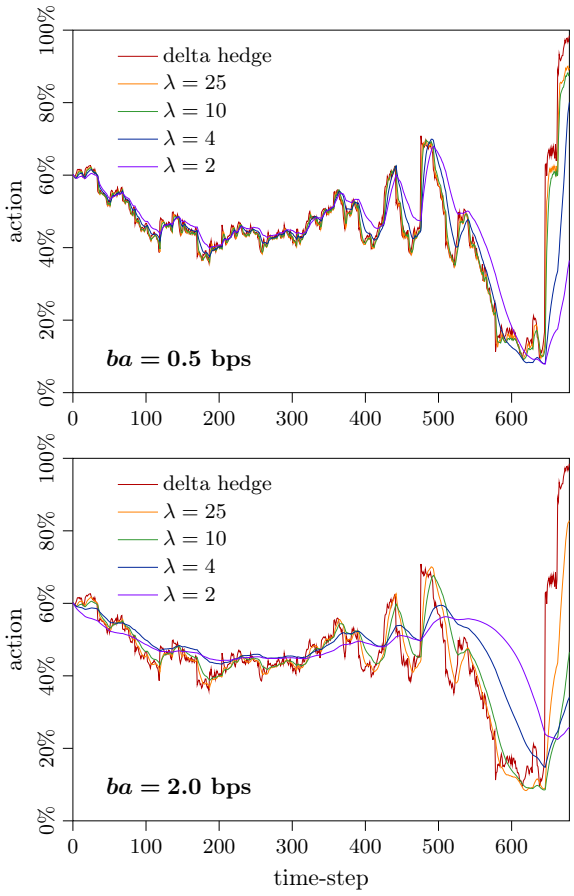

Figure 8.10: *The hedging strategy chosen by agents trained with a different value of the risk aversion parameter λ is compared with the delta hedging strategy in an environment including hedging costs. In the vertical axis the hedging notional as a percentage of the option notional. In the upper plot $ba = 0.5$ bps, in the lower plot $ba = 2$ bps.*

terminal P&L (the *P&L volatility*) on the same testing sample. The colored dots laying on the horizontal axis indicate the performance of the delta hedging strategy in terms of P&L volatility at different values of the ba parameter. It is evident that all the agents perform better than the corresponding delta hedging strategy in terms of P&L, while only a certain number of agents (those lying left of the corresponding colored vertical line) perform better than the delta hedging strategy also in terms of P&L volatility. In this sense, all the frontiers dominate the corresponding delta hedge, and it is striking to notice that the level of dominance depends on the ba parameter: at low costs the dominance is mild (as it was also experienced in the previous section, where the very low hedging costs of listed equity products were considered), at high costs the delta hedging is barely reasonable a strategy. As an example, one can consider the $\lambda = 4$ point of the blue frontier (which assumes very large costs and beats the delta hedge both in terms of P&L and P&L volatility) and observe

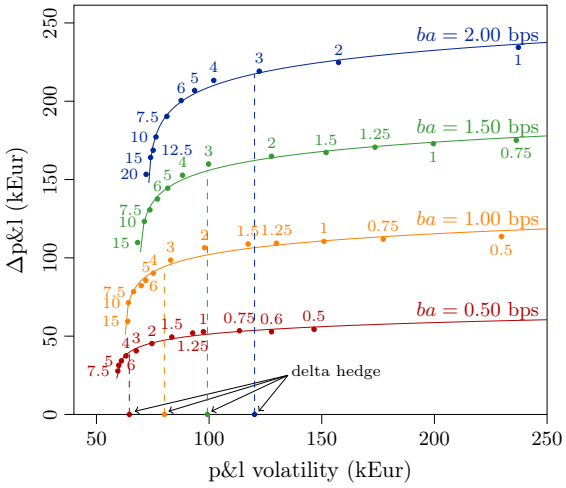

Figure 8.11: *Each dot represents the performance of an agent on a GBM-simulated market in terms of P&L (with respect to delta hedge) and P&L volatility, depending on λ (annotated next to each dot) and the ba parameter.*

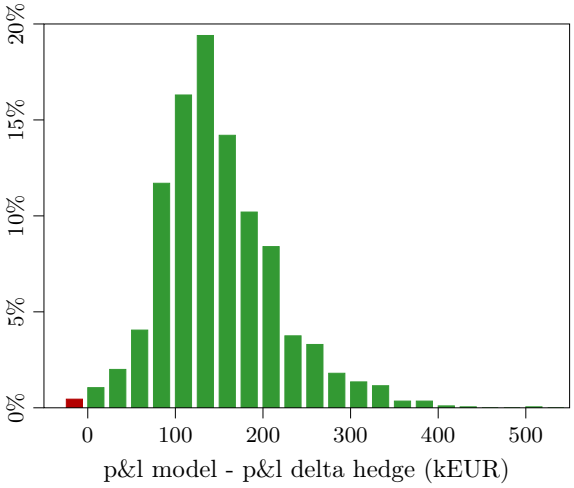

Figure 8.12: *The distribution of the P&L of the $\lambda = 4$ agent relative to the P&L of delta hedge assuming $ba = 1.5$ bps and GBM-simulated market.*

from Figure 8.10 how smooth its action is. Another aspect to notice is the λ parametrization of the different frontiers: there is a shift of λ to the right at the increase of the ba parameter. The benefit of adopting our approach instead of the delta hedging strategy is clear also from Figure 8.12, where we show the distribution of the P&L of the $\lambda = 4$ agent relative to the P&L of the delta hedging strategy in the realistic case of $ba = 1.5$ bps. It is evident that essentially the agent performs always better.

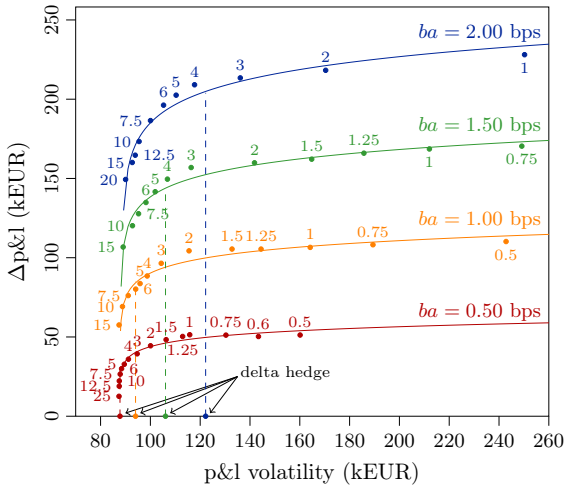

Figure 8.13: *Each dot represents the performance on a Heston simulated market of an agent in terms of P&L (with respect to delta hedge) and P&L volatility, depending on λ (the number close to the dot) and the ba parameter.*

Testing on a Heston simulated Market To make a step towards realism, we challenge the assumption of the GBM constant volatility, as we know it does not hold in the financial markets. We thus construct a new testing set of 2000 episodes with credit spreads generated with the Heston model defined in Equations (4.1) and (4.2), with $\nu(0) = 60\%^2$, so to recover the initial volatility used in training, $\kappa = 2$, $\theta = \nu(0)$, $\xi = 0.9$, and no correlation between the stochastic terms $dW^S(t)$ and $dW^\nu(t)$. With this configuration $\nu(t)$ oscillates significantly reaching values as high as $\sim 120\%$ and as low as $\sim 0\%$. When pricing the option we maintained the B&S formulation with $\sigma = 60\%$.

Even if the agents were trained on a dataset generated with a GBM process, they are able to achieve very good performance over the Heston dataset (see Figure 8.13). The reason behind this behaviour could be that the hedging of an option is a task that implies a deep knowledge of the relationship between the underlying price and the option premium, but the way in which the underlying evolves is probably a secondary aspect.

Testing on Real Market Data To move a further step towards a realistic setup, we consider now real market data for SNRFIN. We use the dataset constructed as in Section 4.2 and processed as in Section 4.4, thus considering real market prices and real transaction parameters ba as seen in Figure 4.2. We simulate the option price with $\sigma = 60\%$. The available data is sufficient for 5 episodes of 40 days, which we used as a test set for agents trained with different values of λ with $ba = 1$ bp.

In Figure 8.14 we show the action of the various agents compared with the delta hedge in one of the episodes. We also show the market data dynamics (in black), on the right vertical axis. We can see as in the previous figures, how lower values of λ generate smoother hedging policies.

Table 8.2 summarizes the performance of the various agents (in €k) and shows the

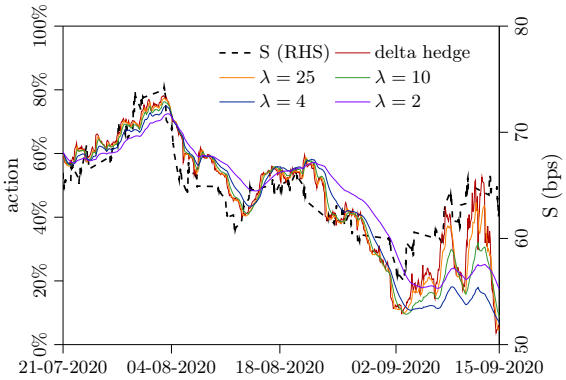

Figure 8.14: *The delta hedging strategy (red line) is compared to the strategy selected by agents trained with $ba = 1$, at various risk aversions (other colored lines), on a real episode. The black line shows the underlying credit spread S observed between July and September 2020.*

	Scenario									
	1		2		3		4		5	
λ	P&L	vol	P&L	vol	P&L	vol	P&L	vol	P&L	vol
1	-177	4.1	183	8.7	-257	3.8	-174	2.6	-282	2.4
1.25	-165	3.6	221	8.9	-224	3.6	-185	2.4	-272	2.4
1.5	-174	3.5	236	8.9	-212	3.6	-200	2.1	-270	2.4
2	-181	3.2	219	7.1	-188	3.5	-223	1.8	-285	2.3
3	-188	3.5	154	5.3	-144	3.6	-215	1.8	-294	2.3
4	-212	3.6	95	3.9	-129	3.7	-234	1.7	-308	2.4
5	-224	3.5	62	3.3	-134	3.4	-242	1.7	-317	2.4
6	-229	3.4	40	3.0	-138	3.3	-247	1.7	-323	2.4
7.5	-231	3.0	18	2.7	-145	3.0	-252	1.6	-329	2.4
10	-237	2.8	-15	2.2	-154	2.8	-259	1.6	-338	2.4
15	-264	2.4	-53	1.9	-191	2.7	-269	1.6	-349	2.3
25	-293	2.2	-80	1.7	-224	2.6	-278	1.6	-356	2.3
δ	-376	2.1	-149	1.6	-310	2.5	-299	1.6	-372	2.3

Table 8.2: *The performance of the agents trained with $ba = 1$ at various risk aversions on the real market data of Figure 4.2. On each 40 day scenario the P&L and path volatility of the delta hedging strategy (last line) is compared with those of the agents (the other lines). The agents always outperform delta hedge in terms of P&L, at the cost of an increase of the path volatility that is null or negligible for high values of λ and it's very moderate in the other cases.*

5 different episodes, all the considered agents overperform the delta hedging strategy in terms of P&L. Considering risk, given the low number of scenarios at hand, the cumulated P&L volatility previously considered is a very noisy estimator, thus, we considered the volatility of the P&L along each scenario, a measure similar to the reward volatility defined

in Equation (4.11) and used in Section 8.2 to define the objective of the TRVO algorithm, as described in Bisi et al. (2020b). Using this measure no agent outperforms delta hedge, but the volatility increase is very small when compared with the cost reduction obtained by adopting our agents.

8.5 Chapter Summary

In this chapter, we learned how to use a risk-averse RL algorithm to optimally hedge the delta risk of options, thus starting the chapter with the definition of TRVO.

Hedging the Delta Risk of Equity Options In Section 8.3, we concentrated on equity options, initially defining the MDP. The focus of this section was on the experimental results, which proved that without transaction costs the agent learns to reproduce the delta hedge. When considering transaction costs, the mean-volatility objective makes it possible to balance risk and return by deciding the agent's level of risk aversion. We noticed that when increasing risk aversion, the policy approaches that of the delta hedge, while when decreasing risk aversion the policy becomes smoother and reduces the transaction costs. Finally, we learned that the policies are robust, as the agents can efficiently hedge options with different characteristics or markets that behave differently than those used in training.

Hedging the Delta Risk of Credit Options In Section 8.4 we are in a dealer market scenario, with higher transaction costs and higher volatility than in the equity options case. We followed similar experimental steps to Section 8.3, showing that in the no costs case the TRVO policy learns to reproduce the delta hedge. With costs, we then verified the trade-off between P&L, risk and transaction fees when changing the risk aversion parameter λ. We showed that, given the high bid-ask spread, it is possible to learn a strategy that beats the practitioner's delta hedge in terms of risk and reward, and generates lower transaction costs. These results were obtained not only when testing on data generated through a GBM, but also when generating the underlying with a Heston process and testing our agents on real market data.

Future works As future works we would like to extend our financial environment to consider hybrid instruments, such as (but not limited to) credit contingent fx options, where the rebalancing costs are enhanced by the correlation between the different underlyings, which can have an adverse effect, known as *wrong way risk*, or even positive effects. Our interest in these issues is motivated by the desire of structuring an automatic operative framework able to efficiently manage second order risks, *i.e.,* the bank's XVA, one of the most formidable task nowadays at hand in the financial industry. In Appendix C, we provide a preliminary approach to DVA hedging and explain the concept of XVA, which can be seen as hybrid products (refer to Appendix C.1.1).

Another future stream of work would aim to combine the hedging task with the market making task of Chapter 7, but considering options instead of bonds. This would result in modeling a "complete" options market maker that prices the options and hedges the delta optimally, possibly also optimizing the hedge execution as in Chapter 9.

CHAPTER 9

Optimal Execution with RL

The majority of financial institutions continuously face the optimal execution problem. Indeed, the sizes of financial instruments they handle are often large enough to have an impact on the market, thus causing the price to move in an adverse manner with respect to the trade. This results higher trading costs than expected. Moreover, the optimal execution problem is relevant for any trade of size larger than what is posted in the first limit of the LOB. The most basic form of optimal execution problem can be stated as the purchase (or sale) of a fixed number of shares within a fixed time horizon minimizing the overall cost paid.

Most asset management firms have entire desks created to focus only on the execution. While portfolio managers decide portfolio allocation, execution desks are then assigned with implementing the trades. In investment banks, there are departments designated with executing client's orders. In other realities, such as proprietary trading desks or market making desks, the traders who decide the investment or hedging strategy are also tasked with executing it optimally. Practitioners mostly use the TWAP strategy (see Section 2.4.3) and the VWAP strategy.

Among all market players, HFTs are those that profit the most from optimal execution. HFTs have low latency colocated infrastructure used to monitor continuously the order book with sophisticated software.[43] In our setting, we assume that we do not have access to such infrastructure, keeping our problem close to the existing literature (Almgren and Chriss, 2001). For these analyses we assume we can only use market orders, possible extensions of this work could consider also limit orders (see Section 2.4.2).

[43] https://www.globalbankingandfinance.com/colocation-and-high-frequency-trading/.

In this chapter we propose the use of RL techniques to optimize execution in a changing market environment. We split the problem in two phases, an offline and an online phase. In the offline phase, we learn multiple optimal execution strategies, one for each simulated market condition using FQI. With the trained execution policies, we then select in an online manner the optimal one to use using Thompson Sampling (TS).

In a single trading day, it is possible to collect thousands of data points from the LOB of a single asset. Unfortunately, this data is not ideal, as to analyze the market impact caused by a trade it is necessary to execute the trade on the market, and, thus, collecting historical data is not sufficient. To overcome this problem, we decided to use the multi-agent market simulator ABIDES (Byrd et al., 2019), described in Section 4.3.2, which offers the possibility of simulating the LOB with great realism.

Optimal execution strategies can potentially be used together with the portfolio optimization approaches of Chapter 5, the trading strategies of Chapter 6, and also the hedging policies of Chapter 8 to further reduce transaction costs and, thus, improve performance.

Chapter outline This chapter begins with an overview of the state-of-the-art regarding optimal execution using RL. We then describe TS in Section 9.2, explaining how it can be used together with FQI to tackle the optimal execution problem. The experimental campaign is described in Section 9.3.

9.1 Background on Optimal Execution with RL

The first studies on optimal execution are based on a stochastic framework in which the dynamics of the assets and execution costs are modeled through SDEs. The first formal analysis of the optimal execution problem is by Bertsimas and Lo (1998). Under suitable conditions and using dynamic programming, they provide a closed-form formula as a solution to the optimal execution problem. Subsequently, an extension of the stochastic model was provided by introducing the permanent and temporary effects due to the impacts of orders on the market and by inserting a risk-aversion parameter by Almgren and Chriss (2001) (see Appendix B.2). Huberman and Stanzl (2005) further extended the optimal execution framework by introducing more complex price impact functions and risk aversion parameters. Furthermore, the book by Guéant (2016) proposes an in depth analysis and extension of these approaches. These methods, however, make strong assumptions on the underlying price movement or distributions.

Thanks to technological advances and the availability of data, Nevmyvaka et al. (2006) applied RL for the first time to optimal execution strategies, with the objective of minimizing implementation shortfall. Subsequently, Hendricks and Wilcox (2014) proposed to combine the Almgren-Chriss model with the Q-learning algorithm (see Equation (3.8)), to create a hybrid framework that executes a proportion of the AC trajectory based on the states in input. To address the high dimensions and the complexity of the underlying dynamics, Ning et al. (2018) used the Deep Q-Network (DQN) (Mnih et al., 2015), a combination of deep neural network and Q-learning, for optimal trade execution, addressing the curse of dimensionality issue faced by tabular Q-learning. Lin and Beling (2020), one of the most recent works, used PPO (Schulman et al., 2017), to propose an end-to-end optimal execution framework that can account for temporal correlations and make decisions based on level II market LOB data (see Section 4.1) using a sparse reward signal. These approaches suffer from the

shortcomings of learning from historical data: the impossibility of reproducing realistically the impact of the orders of the agent. This has paved the way for the use of multi-agent approaches (Balch et al., 2019) and inspired the creation of ABIDES (Byrd et al., 2019). Karpe et al. (2020) are the first to use ABIDES to reproduce a realistic environment, and the DDQL algorithm (Van Hasselt et al., 2016) to learn the optimal execution policy. The work presented in this chapter starts with a similar framework to Karpe et al. (2020), and expands by proposing a preliminary method to confront the non-stationarity of the markets.

9.2 Optimal Execution with FQI and Thompson Sampling

As previously mentioned in Section 2.4.3, specifically in Definition 2.9, the objective of the agent is to learn the optimal way of executing a trade by minimizing market impact. Thus, it is necessary to give the agent both information on the market conditions by means of features deriving from the LOB, internal information, such as size remaining and time remaining, and a reward that leads to the minimization of market impact. With this information, the agent then decides how much to trade at each time-step. Specifically, recalling the notation of Section 2.4.3 the MDP is defined as follows.

State The state contains both private information and features deriving from the LOB. The LOB features are described in Section 2.4.2, and include the volume imbalance, the price imbalance, the total depth, the rolling volatility, and the mid price. The private features are:

- assuming we are at time t_k, the time t_r remaining until T, normalized to be in the range $[-1, 1]$:
$$t_r = 2 \cdot \frac{T - t_k}{T} - 1,$$
- the quantity of remaining inventory to execute at time t_k, depending on the initial inventory X, also normalized:
$$x = 2 \cdot \frac{X - \sum_{t=0}^{t_{k-1}} n_t}{X} - 1,$$

Given the high number of features, we used a feature selection approach as defined in Section 4.4.

Action The action at each time-step represents how much of TWAP *i.e.,* $\frac{X}{N}$ to execute (see Equation (2.26)), where the action $a \in \{0, 0.2, 0.4, ..., 4\}$. So the market order n_t to be traded at each time-step corresponds to:

$$n_t = a_t \frac{X}{N}.$$

If the trade has not been completely executed by time t_{N-1}, the remaining size will be executed as a market order at time T.

Algorithm 9: Thompson Sampling

1 **Initialize:** a Bayesian prior for each arm $f_1, ..., f_n$
2 **for** *each round* t **do**
3 Sample $\theta_1, ..., \theta_n$ from each of the priors $f_1, ..., f_n$
4 Pull the arm i_t with the highest sampled value $i_t = \arg\max_i \theta_i$
5 Observe reward r_{it}
6 Update the prior f_{i_t} corresponding to arm i_t

Reward The reward is:

$$r_{t+1} = \left(1 - \frac{|P_{t_{\text{fill}}} - P_{\text{arrival}}|}{P_{t_{\text{fill}}}}\right) \lambda \frac{n_t}{X}, \tag{9.1}$$

where P_{arrival} is the price of the asset in consideration when the trade is requested, while $P_{t_{\text{fill}}}$ is the average execution price at each time-step. λ is a constant for scaling the effect of the quantity component, which we valued as 10. Implementation shortfall (see Equation (4.7)) is a common reward function, as, similarly to Equation (9.1), it penalizes market impact. Nonetheless, given the disadvantages pointed out in (Lin et al., 2020), we preferred the simpler reward structure defined above in Equation 9.1.

Now that we have defined the MDP, we can learn the optimal policy by interacting with ABIDES. In this case, we use FQI (see Section 6.2.1) with extra tree regressors (see Appendix B.4.2). The issue is that market scenarios vary greatly and there are different optimal policies in different scenarios depending on the current market conditions, the asset in consideration etc. Thus, we train a policy for each market scenario, and propose the use of Thompson Sampling (TS) to select the best policy to use. In the next section we introduce TS, and then illustrate our approach.

Thompson Sampling

Thompson Sampling (Thompson, 1933) is one of the earliest works on MAB problems (see Section 3.3.3). While TS has always given promising experimental results, only recently Agrawal and Goyal (2012); Kaufmann et al. (2012) have proved sublinear regret bounds. TS is an online algorithm, related to the online learning setting explained in Section 3.3.3. In the MAB framework, the agent chooses at each round among a finite set of n arms. The environment responds with a reward r_t, the agent updates its model accordingly. The objective of the agent is to minimize regret as defined in Equation (4.8).

TS is illustrated in Algorithm 9, where each arm is initially associated with a prior distribution f_i. At every round, we get a sample θ_i from each distribution (Line 3) and then pull the arm with the highest sampled value (Line 4). The environment then answers with a reward for the arm that has been pulled. This reward is used to update the distribution of the chosen arm f_{i_t} (Line 6).

9.2.1 Using FQI and TS for Optimal Execution

Algorithm 10 represents our approach to tackle the optimal execution problem. By using ABIDES and a random execution policy, it is possible to create multiple datasets $\mathcal{D}_1, ..., \mathcal{D}_n$

Algorithm 10: Thompson Sampling and FQI for optimal execution

1 **Initialize:** create a dataset $\mathcal{D}_k$ for each market scenario $k \in [1, ..., n]$
2 Learn the optimal policy π_k for each dataset using FQI
3 Set a Gaussian prior $f_1, ..., f_n$ and likelihood $l_1, ..., l_n$ (with known variance) for each arm
4 **for** *each round* t **do**
5 Sample $\theta_1, ..., \theta_n$ from each of the priors $f_1, ..., f_n$
6 Run the policy π_i with the highest sampled value $i_t = \arg\max_i \theta_i$
7 Observe the reward r_i obtained by running π_i on a trading day
8 Calculate the posterior p_i incorporating new information
9 Assign $f_i = p_i$

that represent different market scenarios and have different characteristics. For each scenario, we learn a policy $\pi_1, ..., \pi_n$ using FQI (Line 2). In Line 3, we initialize the Gaussian priors and likelihood. In our case we use non-informative priors with $\mathcal{N}(9.5, 1)$, and a Gaussian likelihood with unknown mean and a variance obtained empirically by running each policy on multiple market scenarios.

Then, the loop starts and for every time-step we sample from each of the priors (Line 5). We then select the arm i_t that has given the highest sample. This arm represents a trained policy, so we run this policy π_i on a trading day on ABIDES and obtain the reward r_i (Line 7). The reward r_i, together with the prior and the likelihood, is then used to calculate the posterior. The posterior becomes the new prior, while the likelihood remains the same, and the process starts over. The variance of the posterior σ_1^2 is calculated as:

$$\sigma_1^2 = \frac{1}{\sigma^{-2} + \sigma_0^{-1}},$$

where σ^2 is the variance of the likelihood and σ_0^2 is the variance of the prior. The mean of the posterior μ_1 is calculated as:

$$\mu_1 = (\mu_0 \sigma_0^{-2} + r\sigma^{-2})\sigma_1^2,$$

where μ_0 is the mean of the prior.

9.3 Experimental Results

We generated the market using ABIDES for the experimental campaign. Specifically, we used the following multi-agent composition:

- 1 exchange agent: it acts as a centralized exchange that keeps the order book and matches orders on the bid and ask sides;
- 75 value agents: they produce an internal and noisy estimate of the real value of the stock and, based on the current price at which the stock is traded in the market, decide if it is undervalued or overvalued. Then, they buy or sell as consequence of their estimate;
- 100 noise agents: they introduce some noise in the simulation acting randomly;

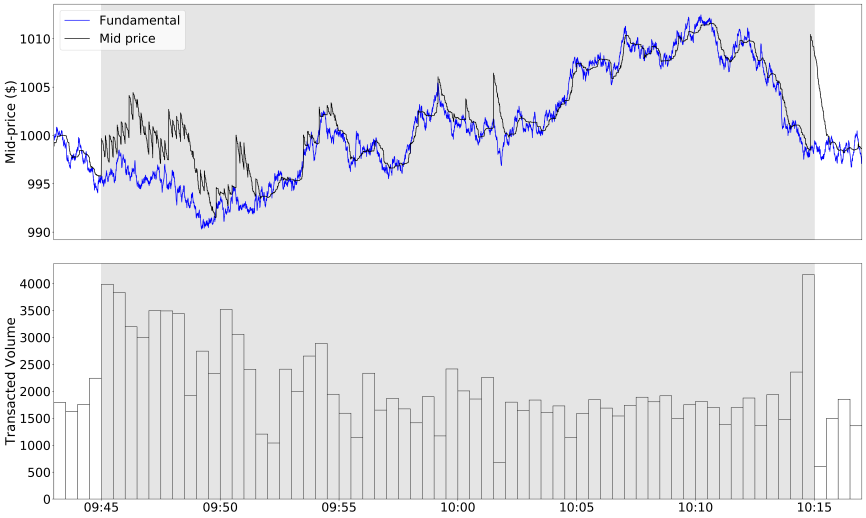

Figure 9.1: *Single execution with the price process on the top graph and the dynamics of the transaction volume dynamics on the bottom graph. Execution in grey shaded area. The market scenario and trained policy are both low volatility low liquidity.*

- 1 market maker: it provides liquidity posting limit orders in bid and ask.

Starting with this configuration, we reproduced two different market situations, one with low volatility and low liquidity, and another with high volatility and high liquidity. These two characterizations can be done by modifying the fundamental curve of the ABIDES configuration (see Section 4.3.2), and the liquidity of the market makers. For each scenario, a dataset of about 2,000 executions, and thus about 350,000 samples, was generated using a random execution policy and used to train the policy through FQI.

Due to computational complexities and the duration of the simulations, we consider $T = 30$ minutes for a single execution (see Figure 9.1). We consider a decision-step every 10 seconds ($\tau = 10$), for a total of 180 decision steps (N = 180). Finally, the total number of shares to execute was $X = 50,000$. This size was chosen so to have an impact on the market but at the same time not consume the entire liquidity of the LOB.

Figure 9.1 shows an example of a single execution in an ABIDES simulation, with the price process on the top graph and the dynamics of the transaction volume on the bottom graph. On the top graph, the fundamental price is depicted in blue while the actual real price is in black. We can see that when there is a high transaction volume the real price tends to detach from the fundamental price. Specifically, we are executing a large buy order, thus the price tends to go higher. This is evident at the last time-step as there probably was a large size remaining to be executed.

In both scenarios (high and low), FQI was run for 180 iterations with 10 executions for each iteration, to tune the number of iterations. The optimal iteration number for the low volatility case was 55, while in the case of high volatility iteration 5 performed best. In the figures shown below, we refer to the two trained agents as “High Volatility Expert” and “Low Volatility Expert”.

Figure 9.2 shows, in the low volatility case, the return of the two FQI agents compared

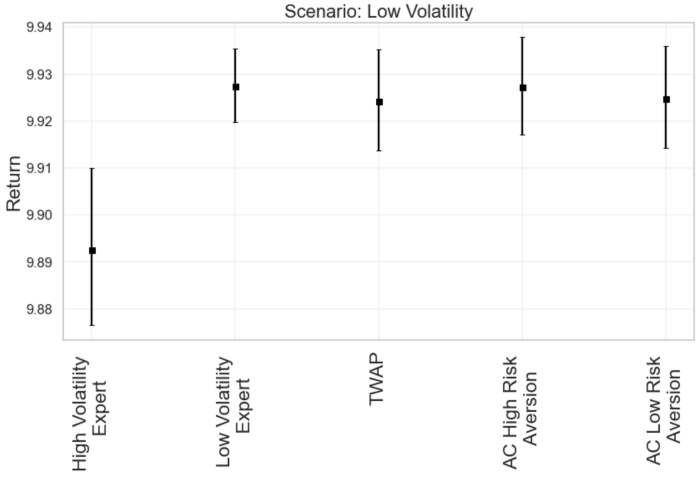

Figure 9.2: *Return of different algorithms in the low volatility, low liquidity scenario with 95% confidence intervals.*

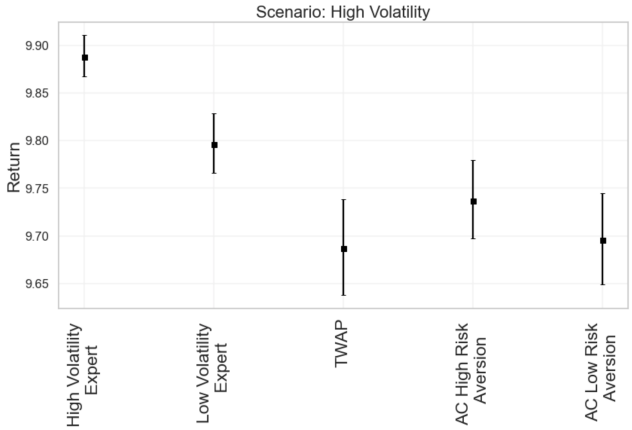

Figure 9.3: *Return of different algorithms in the high volatility, high liquidity scenario with 95% confidence intervals.*

with the TWAP approach described in Section 2.4.3, and two Almgren-Chriss (AC) with varying risk aversion, described in Appendix B.2. Mean and confidence intervals are calculated on 50 executions. The y-axis represents the return over the execution period, specifically, a return of 10 represents the case in which we are capable of executing the order exactly at the level of the arrival price P_{arrival}. This can be seen by considering Equation (9.1) and the fact that $\lambda = 10$. In this figure, the low volatility expert is equivalent to TWAP and AC, which suggests that with low volatility and low liquidity TWAP is the optimal approach. On the opposite, the high volatility expert behaves quite badly, suggesting that the two FQI policies behave differently.

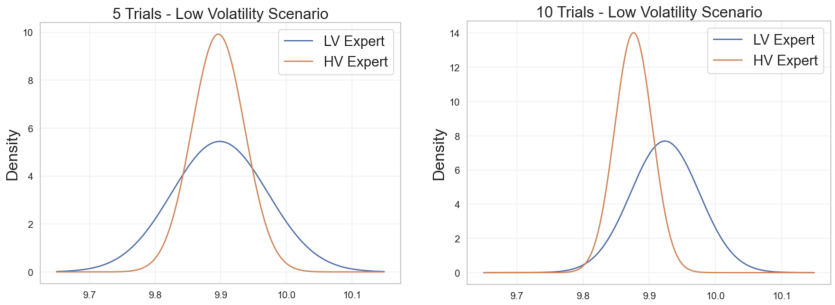

Figure 9.4: *Distributions of each TS arm in low volatility, low liquidity scenario. Left graph is the distribution after 5 iterations and right graph after 10 iterations.*

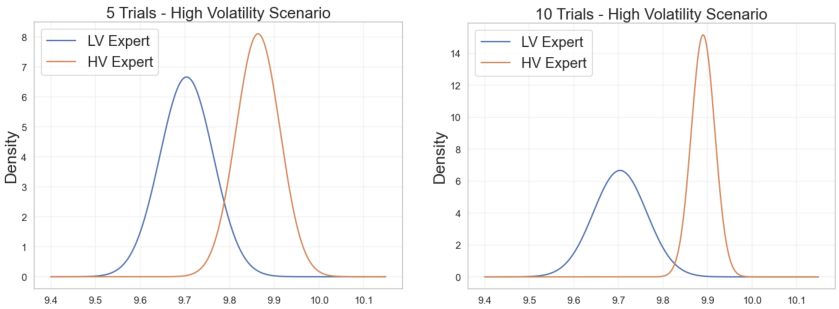

Figure 9.5: *Distributions of each TS arm in high volatility, high liquidity scenario. Left graph is the distribution after 5 iterations and right graph after 10 iterations.*

Figure 9.3 is the equivalent of Figure 9.2, but the scenario in consideration is the one with high liquidity and volatility. In this case, it is evident that the FQI policy trained on the high volatility scenario is superior to all the other approaches.

At this point, we are at Line 2 of Algorithm 10, the priors and the likelihood are defined as in Section 9.2.1. Figures 9.4 and 9.5, represent the *for loop* of Algorithm 10 (from Line 4 to the end).

In Figure 9.4 we are considering the low volatility, low liquidity market scenario. This figure depicts the prior distributions of returns obtained of the two FQI policies. We can see that after 5 TS iterations it is not yet possible to distinguish the two approaches, while by iteration 10 the two distributions start to detach. With respect to the assumptions taken in the experimental campaign, this means that about ten 30-minute executions are necessary to understand which one is the optimal policy for this market scenario.

Figure 9.5 represents the high volatility configuration. In this case, the two distributions can be easily distinguished already at the fifth iteration. This was also expected from Figure 9.3, as the performances of the two trained FQI policies are noticeably different. This means that after at most 5 executions, it is possible to select the optimal execution policy.

9.4 Chapter Summary

The work presented in this chapter considers the optimal execution problem using RL. The main difficulty, when analyzing optimal execution is the fact that it is not possible to realistically simulate market impact by using historical data. For this reason, we decided to use the multi-agent market simulator ABIDES, which creates a multi-agent simulation of the financial markets, enabling the replication of market impact. We started the chapter by defining the MDP, selecting the state features through a feature selection approach. We then proposed the use of FQI to learn the MDP and concentrated on two market scenarios, one with high volatility and high liquidity, and the other with low volatility and low liquidity, comparing with two benchmark strategies: TWAP and Almgren-Chriss. The experimental results show that the FQI approach is superior in the case of high volatility and high liquidity. With low volatility all the approaches obtain similar results. Finally, we devised a method to select in an online manner the optimal policy to use for execution using TS. The results shows that in the high volatility context, it is possible to select the optimal policy after a small number of iterations. This is a first effort to tackle the non-stationarity of the markets.

Interesting future works could be to add the possibility of using also limit orders instead of only market orders and to increase the realism in the order book, using the market replay feature of ABIDES. Finally, it will be important to combine optimal execution with the portfolio optimization, quantitative trading, and hedging tasks to optimize transaction costs in all these domains.

CHAPTER 10

Conclusions

In this dissertation, we explored the use of RL techniques in the financial markets domain, providing theoretical, algorithmic, and experimental contributions. Specifically, in Part I we introduced the financial knowledge and RL notions we deemed necessary to understand Part II. Then, in Part II, we analyzed how to learn in an autonomous fashion the main decision-making processes in the financial markets, namely portfolio optimization, quantitative trading, market making, hedging, and optimal execution. We present below a brief conclusion to each topic.

Online Portfolio Optimization In Chapter 5, we examined two characteristics of the OPO framework, dividing the chapter in two parts. The first part, Section 5.1, focused on controlling transaction costs in the OPO problem by defining a novel algorithm, namely OGDM. We proved that OGDM is capable of achieving a total regret of $\mathcal{O}(\sqrt{T})$. We then verified the analytical results through an extensive experimental campaign comparing with state-of-the-art OPO algorithms.

The second part, Section 5.2, focused on the problem of conservative optimization defining a novel algorithm, namely CP. CP is a "wrapper" that can be applied to any existing OCO algorithm and maintains the same regret order of the OCO algorithm it uses as a subroutine, while satisfying the conservativeness property. We confirmed the theoretical results through the experiments on the OPO framework.

Quantitative Trading In Chapter 6, we proposed two methodologies to create a quantitative trading strategy, specifically, using FQI and using MCTS. Section 6.2 concentrates

on the use of FQI, and is mostly experimental. We compared two different scenarios, a multi-currency framework with EURUSD and USDGBP and a single currency framework considering the FX pairs individually. We observed experimentally the behavior of the resulting trading policy, by changing FQI parameters (min-split and number of training iterations), and action persistence (1-minute, 5-minute, and 10-minute). The results show that the 5-minute persistence achieves a superior performance. Furthermore, the multi-currency setting surpasses the single currency cases - this is expected as it may exploit additional trading opportunities.

Section 6.3 proposed the use of MCTS for trading. We coined a new algorithm, namely QL-OL UCT, that uses open loop planning in order to handle the continuous state space and stochastic transition model and a Q-learning backup operator to reduce the noise of the backups. We also introduced a novel generative model that uses historical data and a clustering approach. We tested the algorithm on the EURUSD FX pair, concentrating on optimizing parameters. While being profitable without considering transaction costs, adding these costs causes the agents to decide not to trade. Further work is necessary in order to improve the performance, also compared to the FQI approach. Such improvement could be achieved by using advanced algorithms like Alphazero (Silver et al., 2017).

Bond Market Making In Chapter 7, we presented a novel solution to the problem of market making in dealer markets. We defined the framework as an N-player stochastic game and proposed a solution using MFGs and RL. This approach is capable of handling the competitive nature of the problem, by learning an equilibrium strategy in a multi-agent framework. The experimental evaluation showed that, in the presence of strategic opponents, the method outperforms other benchmark agents. Instead, in the presence of a generic market configuration, other agents might perform better when considering P&L. However, since the proposed methods guarantee a safe behaviour, they are capable of achieving a lower risk in terms of a higher Sharpe ratio and a smaller inventory.

An interesting future direction to explore is the addition of elements of realism to the problem, e.g., by considering a portfolio of correlated assets, or by training on real data.

Option Hedging In Chapter 8, we learned how to use a risk-averse RL algorithm, namely TRVO, to optimally hedge the delta risk of options. This chapter is composed of two parts, the first on hedging equity options and the second on hedging credit index options. In both cases, the focus is on the experimental campaign, which shows that, without considering costs, the TRVO policy learns to reproduce the delta hedge. With transaction costs, the mean-volatility objective considered, makes it possible to balance risk and return, by deciding the agent's level of risk aversion. We noticed that, when increasing risk aversion, the policy approaches that of the delta hedge, while when decreasing risk aversion the policy becomes smoother and reduces the transaction costs. Furthermore, we learned that the policies are robust, as the agents can efficiently hedge options with different characteristics or markets that behave differently than those used in training. Additionally, in Section 8.4 we obtained positive results when testing on an underlying with a Heston process, and, finally, also with real market data.

As future works, we would like to extend our financial environment to consider hybrid instruments, such as the bank's XVA, one of the most formidable task nowadays at hand in the financial industry. Another future stream of work would aim to combine the hedging

task with the pricing task of Chapter 7.

Optimal Execution The work presented in Chapter 9 considers the optimal execution problem using RL. We used the multi-agent market simulator ABIDES, which creates a multi-agent simulation of the financial markets, enabling a realistic market impact. We proposed FQI to learn a policy on two market scenarios, one with high volatility and high liquidity, and the other with low volatility and low liquidity, and used TS to select in an online manner the optimal policy to use for execution. The experimental results show that the FQI approach is superior to benchmarks such as TWAP and Almgren-Chriss, in the case of high volatility and high liquidity. With low volatility all the approaches obtain similar results. Furthermore, in the high volatility context, it is possible to select the optimal policy after a small number of TS iterations.

Interesting future works could explore the possibility of using also limit orders instead of only market orders, and adding the realism in the order book, using the market replay feature of ABIDES.

Future Synergies We analyzed separately the portfolio management and quantitative trading streams, as they originate from different streams of literature, but highlighted how they are becoming more and more intertwined, as they share the same objective of obtaining a positive return on invested capital. Furthermore, the optimal execution approach presented in Chapter 9 is beneficial when managing a portfolio or implementing a trading or hedging strategy. Hence, future works would include unifying optimal execution with the mentioned approaches.

We evaluated the pricing and hedging tasks of market makers in two separate chapters, since they derive from different streams of literature. One of the future works on this topic is to model through RL a "complete" market maker, covering all the necessary tasks from pricing to hedging to optimal execution.

Moreover, some specialist firms continuously price an asset while also taking a market position with trading strategies, hedging unwanted risks and optimizing the execution. An ambitious project is that of modelling an artificial agent or a team of agents to manage multiple assets with the flexibility of combining all the approaches.

Bibliography

Abernethy, J., Bartlett, P. L., Rakhlin, A., and Tewari, A. (2008). Optimal strategies and minimax lower bounds for online convex games.

Abernethy, J. and Kale, S. (2013). Adaptive market making via online learning. In *Neural Information Processing Systems*, pages 2058–2066, Stateline, Nevada, United States. NeurIPS.

Achdou, Y., Camilli, F., and Capuzzo-Dolcetta, I. (2012). Mean field games: numerical methods for the planning problem. *SIAM Journal on Control and Optimization*, 50(1):77–109.

Achdou, Y. and Capuzzo-Dolcetta, I. (2010). Mean field games: numerical methods. *SIAM Journal on Numerical Analysis*, 48(3):1136–1162.

Adlakha, S., Johari, R., and Weintraub, G. Y. (2015). Equilibria of dynamic games with many players: Existence, approximation, and market structure. *Journal of Economic Theory*, 156:269–316.

Agarwal, A., Hazan, E., Kale, S., and Schapire, R. (2006). Algorithms for portfolio management based on the newton method. In *International Conference on Machine Learning*, pages 9–16, Pittsburgh, Pennsylvania, United States. ICML.

Agrawal, S. and Goyal, N. (2012). Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pages 39–1. JMLR Workshop and Conference Proceedings.

Almgren, R. and Chriss, N. (2001). Optimal execution of portfolio transactions. *Journal of Risk*, 3:5–40.

Alonso, M. N. and Srivastava, S. (2020). Deep reinforcement learning for asset allocation in us equities. *CompSciRN: Other Machine Learning (Topic)*.

Anahtarcı, B., Karıksız, C. D., and Saldi, N. (2019). Fitted q-learning in mean-field games. *arXiv preprint arXiv:1912.13309*.

Antos, A., Szepesvári, C., and Munos, R. (2007). Fitted q-iteration in continuous action-space mdps. *Advances in neural information processing systems*, 20.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Auer, P. and Ortner, R. (2010). Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65.

Aumann, R. J. (1964). Markets with a continuum of traders. *Econometrica: Journal of the Econometric Society*, pages 39–50.

Avellaneda, M. and Stoikov, S. (2008). High-frequency trading in a limit order book. *Quantitative Finance*, 8(3):217–224.

Bacoyannis, V., Glukhov, V., Jin, T., Kochems, J., and Song, D. R. (2018). Idiosyncrasies and challenges of data driven learning in electronic trading. *arXiv preprint arXiv:1811.09549*.

Balch, T. H., Mahfouz, M., Lockhart, J., Hybinette, M., and Byrd, D. (2019). How to evaluate trading strategies: Single agent market replay or multiple agent interactive simulation? *arXiv preprint arXiv:1906.12010*.

Balseiro, S. R., Besbes, O., and Weintraub, G. Y. (2015). Repeated auctions with budgets in ad exchanges: Approximations and design. *Management Science*, 61(4):864–884.

Banach, S. (1922). Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fund. math*, 3(1):133–181.

Banerjee, A., Merugu, S., Dhillon, I., and Ghosh, J. (2005). Clustering with bregman divergences. *J MACH LEARN RES*, 6:1705–1749.

Bank, P., Ekren, I., and Muhle-Karbe, J. (2021). Liquidity in competitive dealer markets. *Mathematical Finance*, 31(3):827–856.

Basel Committee, B. S. (2016). Minimum capital requirements for market risk.

Baxter, J. and Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *JAIR*, 15.

Bellman, R. (1966). Dynamic programming. *Science*, 153(3731):34–37.

Belmega, E., Mertikopoulos, P., Negrel, R., and Sanguinetti, L. (2018). Online convex optimization and no-regret learning: Algorithms, guarantees and applications. *arXiv:1804.04529*, 12 April:1–34.

Bensoussan, A., Frehse, J., Yam, P., et al. (2013). *Mean field games and mean field type control theory*, volume 101. Springer.

Bernasconi de Luca, M., Vittori, E., Trovò, F., and Restelli, M. (2021). Conservative online convex optimization. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML-PKDD 2021*.

Bertsimas, D. and Lo, A. W. (1998). Optimal control of execution costs. *Journal of Financial Markets*, 1(1):1–50.

Besson, L. and Kaufmann, E. (2018). What doubling tricks can and can't do for multi-armed bandits. *arXiv preprint arXiv:1803.06971*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Bisi, L., Liotet, P., Sabbioni, L., Reho, G., Montali, N., Corno, C., and Restelli, M. (2020a). Foreign exchange trading: A risk-averse batch reinforcement learning approach. In *ICAIF 2020*. ACM.

Bisi, L., Sabbioni, L., Vittori, E., Papini, M., and Restelli, M. (2020b). Risk-averse trust region optimization for reward-volatility reduction. In Bessiere, C., editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4583–4589. International Joint Conferences on Artificial Intelligence Organization. Special Track on AI in FinTech.

Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654.

Blum, A. and Kalai, A. (1999). Universal portfolios with and without transaction costs. *MACH LEARN*, 35(3):193–205.

Borodin, A. et al. (2004). Can we learn to beat the best stock. In *Neural Information Processing Systems*, pages 345–352, Vancouver, Canada. NeurIPS.

Bouchaud, J.-P. and Potters, M. (2003). *Theory of financial risk and derivative pricing: from statistical physics to risk management*. Cambridge university press.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *FOUND TRENDS MACH LEARN*, 3(1):1–122.

Briceño-Arias, L., Kalise, D., Kobeissi, Z., Laurière, M., González, A. M., and Silva, F. J. (2019). On the implementation of a primal-dual algorithm for second order time-dependent mean field games with local couplings. *ESAIM: Proceedings and Surveys*, 65:330–348.

Briceño-Arias, L. M., Kalise, D., and Silva, F. J. (2018). Proximal methods for stationary mean field games with local couplings. *SIAM Journal on Control and Optimization*, 56(2):801–836.

Briola, A., Turiel, J., Marcaccioli, R., and Aste, T. (2021). Deep reinforcement learning for active high frequency trading. *arXiv preprint arXiv:2101.07107*.

Buehler, H., Gonon, L., Teichmann, J., and Wood, B. (2019). Deep hedging. *Quantitative Finance*, pages 1–21.

Byrd, D., Hybinette, M., and Balch, T. H. (2019). Abides: Towards high-fidelity market simulation for ai research. *arXiv preprint arXiv:1904.12066*.

Cao, H., Guo, X., and Laurière, M. (2020). Connecting gans and mfgs. *arXiv preprint arXiv:2002.04112*.

Cao, J., Chen, J., Hull, J. C., and Poulos, Z. (2019). Deep hedging of derivatives using reinforcement learning. *Available at SSRN*.

Carapuço, J., Neves, R., and Horta, N. (2018). Reinforcement learning applied to forex trading. *Applied Soft Computing*, 73:783–794.

Cardaliaguet, P. and Hadikhanloo, S. (2017). Learning in mean field games: the fictitious play. *ESAIM: Control, Optimisation and Calculus of Variations*, 23(2):569–591.

Carlini, E. and Silva, F. J. (2014). A fully discrete semi-lagrangian scheme for a first order mean field game problem. *SIAM Journal on Numerical Analysis*, 52(1):45–67.

Carlini, E. and Silva, F. J. (2015). A semi-lagrangian scheme for a degenerate second order mean field game system. *Discrete & Continuous Dynamical Systems*, 35(9):4269.

Carmona, R. and Laurière, M. (2019). Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games: Ii–the finite horizon case. *arXiv preprint arXiv:1908.01613*.

Carmona, R., Laurière, M., and Tan, Z. (2019a). Linear-quadratic mean-field reinforcement learning: convergence of policy gradient methods. *arXiv preprint arXiv:1910.04295*.

Carmona, R., Laurière, M., and Tan, Z. (2019b). Model-free mean-field reinforcement learning: mean-field mdp and mean-field q-learning. *arXiv preprint arXiv:1910.12802*.

Cartea, Á., Jaimungal, S., and Penalva, J. (2015). *Algorithmic and high-frequency trading*. Cambridge University Press.

Cartea, Á., Jaimungal, S., and Ricci, J. (2014). Buy low, sell high: A high frequency trading perspective. *SIAM Journal on Financial Mathematics*, 5(1):415–444.

Cesa-Bianchi, N., Dekel, O., and Shamir, O. (2013). Online learning with switching costs and other adaptive adversaries. In *Neural Information Processing Systems*, pages 1160–1168, Stateline, Nevada, United States. NeurIPS.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge university press.

Chan, N. T. and Shelton, C. (2001). An electronic market-maker.

Chang, H. S. and Marcus, S. I. (2003). Approximate receding horizon approach for markov decision processes: Average reward case. *Journal of Mathematical Analysis and Applications*, 286(2):636–651.

Choueifaty, Y. and Coignard, Y. (2008). Toward maximum diversification. *The Journal of Portfolio Management*, 35(1):40–51.

Chowdhury, G. G. (2003). Natural language processing. *Annual review of information science and technology*, 37(1):51–89.

Corazza, M. and Malliaris, A. T. G. (2002). Multi-fractality in foreign currency markets. *Multinational Finance Journal*, 6(2):65–98.

Couetoux, A. (2013). *Monte Carlo Tree Search for Continuous and Stochastic Sequential Decision Making Problems*. PhD thesis, Université Paris Sud-Paris XI.

Cutkosky, A. and Orabona, F. (2018). Black-box reductions for parameter-free online learning in banach spaces. In *Conference On Learning Theory*, pages 1493–1529. PMLR.

Darley, V., Outkin, A., Plate, T., and Gao, F. (2000). Sixteenths or pennies? observations from a simulation of the nasdaq stock market. In *Proceedings of the IEEE/IAFE/INFORMS 2000 Conference on Computational Intelligence for Financial Engineering (CIFEr)(Cat. No. 00TH8520)*, pages 151–154. IEEE.

Das, P. (2014). *Online convex optimization and its application to online portfolio selection*. PhD thesis, The University of Minnesota.

Das, P., Johnson, N., and Banerjee, A. (2013). Online lazy updates for portfolio selection with transaction costs. In *Conference on Artificial Intelligence*, pages 202–208, Bellevue, Washington, United States. AAAI.

Das, S. (2005). A learning market-maker in the glosten–milgrom model. *Quantitative Finance*, 5(2):169–180.

Das, S. (2008). The effects of market-making on price dynamics. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 887–894. Citeseer.

Daskalakis, C., Goldberg, P. W., and Papadimitriou, C. H. (2009). The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259.

Debnath, A. (2019). Global currency trading surges to $6.6 trillion-a-day market.

Degris, T. and Sigaud, O. (2013). Factored markov decision processes. *Markov Decision Processes in Artificial Intelligence*, pages 99–126.

Dempster, M. A., Payne, T. W., Romahi, Y., and Thompson, G. W. (2001). Computational learning techniques for intraday fx trading using popular technical indicators. *IEEE Transactions on neural networks*, 12(4):744–754.

Di Matteo, T., Aste, T., and Dacorogna, M. M. (2003). Scaling behaviors in differently developed markets. *Physica A: Statistical Mechanics and its Applications*, 324(1-2):183–188.

Dochow, R. (2016). *Online algorithms for the portfolio selection problem*. Springer, Berlin, Germany.

Du, J., Jin, M., Kolm, P. N., Ritter, G., Wang, Y., and Zhang, B. (2020). Deep reinforcement learning for option replication and hedging. *The Journal of Financial Data Science*, 2(4):44–57.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the l1-ball for learning in high dimensions. In *International Conference on Machine Learning*, pages 272–279, Helsinki, Finland. ICML.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Tewari, A. (2010). Composite objective mirror descent. In *Conference on Learning Theory*, pages 14–26, Haifa, Israel. COLT.

Economist, T. (2019). The rise of the financial machines.

Efroni, Y., Ghavamzadeh, M., and Mannor, S. (2020). Online planning with lookahead policies. *Advances in Neural Information Processing Systems*, 33.

Elder, T. (2008). Creating algorithmic traders with hierarchical reinforcement learning msc dissertation.

Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *JMLR*, 6(Apr):503–556.

Feng, L. and Seasholes, M. S. (2005). Do investor sophistication and trading experience eliminate behavioral biases in financial markets? *Review of Finance*, 9(3):305–351.

Fermanian, J.-D., Guéant, O., and Pu, J. (2016). The behavior of dealers and clients on the european corporate bond market: the case of multi-dealer-to-client platforms. *Market microstructure and liquidity*, 2(03n04):1750004.

Fischer, T. G. (2018). Reinforcement learning in financial markets-a survey. Technical report, FAU Discussion Papers in Economics.

Fouque, J.-P. M. and Zhang, Z. (2020). Deep learning methods for mean field control problems with delay. *Frontiers in Applied Mathematics and Statistics*, 6:11.

Fu, Z., Yang, Z., Chen, Y., and Wang, Z. (2019). Actor-critic provably finds nash equilibria of linear-quadratic mean-field games. *arXiv preprint arXiv:1910.07498*.

Fudenberg, D., Drew, F., Levine, D. K., and Levine, D. K. (1998). *The theory of learning in games*, volume 2. MIT press.

Ganesh, S., Vadori, N., Xu, M., Zheng, H., Reddy, P., and Veloso, M. (2019). Reinforcement learning for market making in a multi-agent dealer market. *arXiv preprint arXiv:1911.05892*.

Garcelon, E., Ghavamzadeh, M., Lazaric, A., and Pirotta, M. (2020a). Conservative exploration in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1431–1441. PMLR.

Garcelon, E., Ghavamzadeh, M., Lazaric, A., and Pirotta, M. (2020b). Improved algorithms for conservative exploration in bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3962–3969.

García, J. and Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *JMLR*, 16(1):1437–1480.

Garcin, M. (2019). Fractal analysis of the multifractality of foreign exchange rates. Technical report, HAL.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, 63(1):3–42.

Goel, G., Lin, Y., Sun, H., and Wierman, A. (2019). Beyond online balanced descent: An optimal algorithm for smoothed online optimization. *Advances in Neural Information Processing Systems*, 32:1875–1885.

Gold, C. (2003). Fx trading via recurrent reinforcement learning. In *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003. Proceedings.*, pages 363–370. IEEE.

Gomes, D. A. et al. (2014). Mean field games models a brief survey. *Dynamic Games and Applications*, 4(2):110–154.

Gomes, D. A., Mohr, J., and Souza, R. R. (2010). Discrete time, finite state space mean field games. *Journal de mathématiques pures et appliquées*, 93(3):308–328.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *NeurIPS*, pages 2672–2680. Neural Information Processing Systems (NIPS) Foundation.

Greenwald, A., Li, J., Sodomka, E., and Littman, M. (2013). Solving for best responses in extensive-form games using reinforcement learning methods. *RLDM 2013*, page 116.

Grinold, R. and Kahn, R. (2000). *Active portfolio management*. McGraw Hill, New York, New York, United States.

Guéant, O. (2016). *The Financial Mathematics of Market Liquidity: From optimal execution to market making*, volume 33. CRC Press.

Guéant, O. (2017). Optimal market making. *Applied Mathematical Finance*, 24(2):112–154.

Guéant, O., Lehalle, C.-A., and Fernandez-Tapia, J. (2013). Dealing with the inventory risk: a solution to the market making problem. *Mathematics and financial economics*, 7(4):477–507.

Guéant, O. and Manziuk, I. (2019). Deep reinforcement learning for market making in corporate bonds: beating the curse of dimensionality. *Applied Mathematical Finance*, 26(5):387–452.

Guéant, O. and Pu, J. (2017). Option pricing and hedging with execution costs and market impact. *Mathematical Finance*, 27(3):803–831.

Gunia, A. (2019). How machines are taking over the world's stock markets.

Guo, X., Hu, A., Xu, R., and Zhang, J. (2019). Learning mean-field games. *arXiv preprint arXiv:1901.09585*.

Guo, X., Hu, A., Xu, R., and Zhang, J. (2020). A general framework for learning mean-field games. *arXiv preprint arXiv:2003.06069*.

Halperin, I. (2017). Qlbs: Q-learner in the black-scholes (-merton) worlds. *The Journal of Derivatives*.

Halperin, I. (2019). The qlbs q-learner goes nuqlear: fitted q iteration, inverse rl, and option portfolios. *Quantitative Finance*, pages 1–11.

Hazan, E., Agarwal, A., and Kale, S. (2007). Logarithmic regret algorithms for online convex optimization. *MACH LEARN*, 69:169–192.

Hecht-Nielsen, R. (1989). Neural network primer: part i. *AI Expert*, 4(2):61–67.

Hendricks, D. and Wilcox, D. (2014). A reinforcement learning extension to the almgren-chriss framework for optimal trade execution. In *2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, pages 457–464. IEEE.

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. (2018). Stable baselines. `https://github.com/hill-a/stable-baselines`.

Ho, T. and Stoll, H. R. (1981). Optimal dealer pricing under transactions and return uncertainty. *Journal of Financial economics*, 9(1):47–73.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

Ho, T. S. and Stoll, H. R. (1983). The dynamics of dealer markets under competition. *The Journal of finance*, 38(4):1053–1074.

Hongyang, Y., Xiao-Yang, L., Shan, Z., and Anwar, W. (2020). Deep reinforcement learning for automated stock trading: An ensemble strategy. In *ICAIF '20: ACM International Conference on AI in Finance*.

Huang, C. Y. (2018). Financial trading as a game: A deep reinforcement learning approach. *arXiv preprint arXiv:1807.02787*.

Huang, M. (2010). Large-population lqg games involving a major player: the nash certainty equivalence principle. *SIAM Journal on Control and Optimization*, 48(5):3318–3353.

Huang, M., Caines, P. E., and Malhamé, R. P. (2007). Large-population cost-coupled lqg problems with nonuniform agents: individual-mass behavior and decentralized $varepsilon$-nash equilibria. *IEEE transactions on automatic control*, 52(9):1560–1571.

Huang, M., Malhamé, R. P., Caines, P. E., et al. (2006). Large population stochastic dynamic games: closed-loop mckean-vlasov systems and the nash certainty equivalence principle. *Communications in Information & Systems*, 6(3):221–252.

Huberman, G. and Stanzl, W. (2005). Optimal liquidity trading. *Review of finance*, 9(2):165–200.

Hull, J. C. (2003). *Options futures and other derivatives*. Pearson Education India.

Hutter, M. and Poland, J. (2005). Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research*, 6:639–660.

Ito, S., , Hatano, D., Sumita, H., Yabe, A., Fukunaga, T., Kakimura, N., and Kawarabayashi, K. (2018). Regret bounds for online portfolio selection with a cardinality constraint. In *Neural Information Processing Systems*, pages 1–10, MontrÃ©al, Canada. NeurIPS.

Iyer, K., Johari, R., and Sundararajan, M. (2014). Mean field equilibria of dynamic auctions with learning. *Management Science*, 60(12):2949–2970.

Jangmin, O., Lee, J., Lee, J. W., and Zhang, B.-T. (2006). Adaptive stock trading with dynamic asset allocation using reinforcement learning. *Information Sciences*, 176(15):2121–2147.

Jarrow, R. A. and Turnbull, S. M. (1995). Pricing derivatives on financial securities subject to credit risk. *The journal of finance*, 50(1):53–85.

Jiang, Z., Xu, D., and Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*.

Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *ICML*, pages 267–274.

Kalai, A. and Vempala, S. (2002). Efficient algorithms for universal portfolios. *J MACH LEARN RES*, 3(Nov):423–440.

Karpe, M., Fang, J., Ma, Z., and Wang, C. (2020). Multi-agent reinforcement learning in a realistic limit order book market simulation. *arXiv preprint arXiv:2006.05574*.

Kaufmann, E., Korda, N., and Munos, R. (2012). Thompson sampling: An asymptotically optimal finite-time analysis. In *International conference on algorithmic learning theory*, pages 199–213. Springer.

Kazerouni, A., Ghavamzadeh, M., Abbasi-Yadkori, Y., and Van Roy, B. (2016). Conservative contextual linear bandits. *arXiv preprint arXiv:1611.06426*.

Keh (2018). Billionaire robots: Machine learning at renaissance technologies.

Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer.

Koller, D. and Parr, R. (1999). Computing factored value functions for policies in structured mdps. In *IJCAI*, volume 99, pages 1332–1339.

Kolm, P. N. and Ritter, G. (2019). Dynamic replication and hedging: A reinforcement learning approach. *The Journal of Financial Data Science*, 1(1):159–171.

Kondratyev, A. (2018). Curve dynamics with artificial neural networks. *Risk*, 31(6).

Kondratyev, A. and Christian, S. (2019). The market generator. *Available at SSRN*.

Koolen, W. (2013). The pareto regret frontier. *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 1–9.

Lasry, J.-M. and Lions, P.-L. (2007). Mean field games. *Japanese journal of mathematics*, 2(1):229–260.

Lattimore, T. (2015). The pareto regret frontier for bandits. *arXiv preprint arXiv:1511.00048*.

Lecarpentier, E., Infantes, G., Lesire, C., and Rachelson, E. (2018). Open loop execution of tree-search algorithms. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2362–2368. International Joint Conferences on Artificial Intelligence Organization.

Leland, H. E. (1985). Option pricing and replication with transactions costs. *The journal of finance*, 40(5):1283–1301.

Li, B. and Hoi, S. (2014). Online portfolio selection: A survey. *ACM COMPUT SURV*, 46(3):35.

Li, B., Hoi, S., Sahoo, D., and Liu, Z. (2015). Moving average reversion strategy for on-line portfolio selection. *ARTIF INTELL*, 222:104–123.

Li, B., Wang, J., Huang, D., and Hoi, S. (2018a). Transaction cost optimization for online portfolio selection. *QUANT FINANC*, 18(8):1411–1424.

Li, B., Zhao, P., Hoi, S., and Gopalkrishnan, V. (2012). Pamr: Passive aggressive mean reversion strategy for portfolio selection. *MACH LEARN*, 87(2):221–258.

Li, Y., Qu, G., and Li, N. (2018b). Online optimization with predictions and switching costs: Fast algorithms and the fundamental limit. *arXiv:1801.07780*, 7 March:1–24.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lim, Y.-S. and Gorse, D. (2018). Reinforcement learning for high-frequency market making. In *ESANN*.

Lin, A. T., Fung, S. W., Li, W., Nurbekyan, L., and Osher, S. J. (2020). Apac-net: Alternating the population and agent control via two neural networks to solve high-dimensional stochastic mean field games. *arXiv preprint arXiv:2002.10113*.

Lin, M., Wierman, A., Roytman, A., Meyerson, A., and Andrew, L. (2012). Online optimization with switching cost. *PERF E R SI*, 40(3):98–100.

Lin, S. and Beling, P. A. (2020). An end-to-end optimal trade execution framework based on proximal policy optimization. In *IJCAI*, pages 4548–4554.

Lo, A. W. (2019). *The adaptive markets hypothesis*. Princeton University Press.

Loeb, T. F. (1983). Trading cost: the critical link between investment information and results. *Financial Analysts Journal*, 39(3):39–44.

Loh, W.-Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23.

Lopes, R. and Betrouni, N. (2009). Fractal and multifractal analysis: a review. *Medical image analysis*, 13(4):634–649.

Lu, D. and Weng, Q. (2007). A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28(5):823–870.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777.

Mantegna, R. N. and Stanley, H. E. (1999). *Introduction to econophysics: correlations and complexity in finance*. Cambridge university press.

Markowitz, H. (1952). Portfolio selection. *The journal of finance*, 7(1):77–91.

Meng, T. L. and Khushi, M. (2019). Reinforcement learning in financial markets. *Data*, 4(3):110.

Merton, R. C. (1969). Lifetime portfolio selection under uncertainty: The continuous-time case. *The review of Economics and Statistics*, pages 247–257.

Metelli, A. M., Mazzolini, F., Bisi, L., Sabbioni, L., and Restelli, M. (2020). Control frequency adaptation via action persistence in batch reinforcement learning. In *International Conference on Machine Learning*, pages 6862–6873. PMLR.

Meyes, R., Tercan, H., Roggendorf, S., Thiele, T., Büscher, C., Obdenbusch, M., Brecher, C., Jeschke, S., and Meisen, T. (2017). Motion planning for industrial robots using reinforcement learning. *Procedia CIRP*, 63:107–112.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Moody, J. and Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, 12(4):875–889.

Moon, J. and Başar, T. (2016). Robust mean field games for coupled markov jump linear systems. *International Journal of Control*, 89(7):1367–1381.

Nevmyvaka, Y., Feng, Y., and Kearns, M. (2006). Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, pages 673–680.

Ning, B., Lin, F. H. T., and Jaimungal, S. (2018). Double deep q-learning for optimal execution. *arXiv preprint arXiv:1812.06600*.

Papini, M., Pirotta, M., and Restelli, M. (2017). Adaptive batch size for safe policy gradients. In *NeurIPS*, pages 3591–3600.

Papini, M., Pirotta, M., and Restelli, M. (2019). Smoothing policies and safe policy gradients.

Pendharkar, P. C. and Cusatis, P. (2018). Trading financial indices with reinforcement learning agents. *Expert Systems with Applications*, 103:1–13.

Perrin, S., Pérolat, J., Laurière, M., Geist, M., Elie, R., and Pietquin, O. (2020). Fictitious play for mean field games: Continuous time analysis and applications. *arXiv preprint arXiv:2007.03458*.

Pirotta, M., Restelli, M., and Bascetta, L. (2015). Policy gradient in lipschitz markov decision processes. *Machine Learning*, 100(2):255–283.

Pirotta, M., Restelli, M., Pecorino, A., and Calandriello, D. (2013). Safe policy iteration. In Dasgupta, S. and McAllester, D., editors, *ICML*, volume 28 of *Proceedings of Machine Learning Research*, pages 307–315, Atlanta, Georgia, USA. PMLR.

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17.

Prashanth, L. A. and Ghavamzadeh, M. (2014). Actor-critic algorithms for risk-sensitive reinforcement learning. *arXiv preprint arXiv:1403.6530*.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

Riva, A., Bisi, L., Sabbioni, L., Liotet, P., Vittori, E., Trapletti, M., Pinciroli, M., and Restelli, M. (2021). Learning fx trading strategies with fqi and persistant actions. In *ICAIF 2021*. ACM.

Robinson, J. (1951). An iterative method of solving a game. *Annals of mathematics*, pages 296–301.

Rokach, L. and Maimon, O. (2005). Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer.

Roncalli, T. (2013). *Introduction to risk parity and budgeting*. CRC Press.

Ruthotto, L., Osher, S. J., Li, W., Nurbekyan, L., and Fung, S. W. (2020). A machine learning framework for solving high-dimensional mean field game and mean field control problems. *Proceedings of the National Academy of Sciences*, 117(17):9183–9193.

Saldi, N., Başar, T., and Raginsky, M. (2019). Partially-observed discrete-time risk-sensitive mean-field games. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 317–322. IEEE.

Saldi, N., Başar, T., and Raginsky, M. (2020). Approximate markov-nash equilibria for discrete-time risk-sensitive mean-field games. *Mathematics of Operations Research*, 45(4):1596–1620.

Sani, A., Neu, G., and Lazaric, A. (2014). Exploiting easy data in online optimization. *Advances in Neural Information Processing Systems*, 27:810–818.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. (2015). Trust region policy optimization. In *ICML*, volume 37, pages 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Shalev-Shwartz, S. et al. (2011). Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194.

Sharpe, W. (1963). A simplified model for portfolio analysis. *MANAGE SCI*, 9(2):277–293.

Shen, Y., Huang, R., Yan, C., and Obermayer, K. (2014). Risk-averse reinforcement learning for algorithmic trading. In *CIFEr*, pages 391–398.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Soner, H. M. and Touzi, N. (2013). Homogenization and asymptotics for small transaction costs. *Siam journal on control and optimization*, 51(4):2893–2921.

Sornmayura, S. (2019). Robust forex trading with deep q network (dqn). *ABAC Journal*, 39(1).

Spooner, T., Fearnley, J., Savani, R., and Koukorinis, A. (2018). Market making via reinforcement learning. *arXiv preprint arXiv:1804.04216*.

Spooner, T. and Savani, R. (2020). Robust market making via adversarial reinforcement learning. *arXiv preprint arXiv:2003.01820*.

Streeter, M. and McMahan, H. B. (2012). No-regret algorithms for unconstrained online convex optimization. *arXiv preprint arXiv:1211.2260*.

Sui, Y., Burdick, J., Yue, Y., et al. (2018). Stagewise safe bayesian optimization with gaussian processes. In *International Conference on Machine Learning*, pages 4781–4789. PMLR.

Sui, Y., Gotovos, A., Burdick, J., and Krause, A. (2015). Safe exploration for optimization with gaussian processes. In *International Conference on Machine Learning*, pages 997–1005. PMLR.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tadoori, G. (2020). Quantamental investing: A machine learning investment process. *Available at SSRN 3704670*.

Tamar, A., Chow, Y., Ghavamzadeh, M., and Mannor, S. (2015). Policy Gradient for Coherent Risk Measures. *CoRR*, page 9.

Tamar, A., Chow, Y., Ghavamzadeh, M., and Mannor, S. (2017). Sequential Decision Making With Coherent Risk. *IEEE Transactions on Automatic Control*, 62(7):3323–3338.

Tamar, A., Di Castro, D., and Mannor, S. (2016). Learning the variance of the reward-to-go. *JMLR*, 17(1):361–396.

Tamar, A. and Mannor, S. (2013). Variance adjusted actor critic algorithms. *arXiv preprint arXiv:1310.3697*.

Tembine, H., Zhu, Q., and Başar, T. (2013). Risk-sensitive mean-field games. *IEEE Transactions on Automatic Control*, 59(4):835–850.

Théate, T. and Ernst, D. (2021). An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173:114632.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Trovò, F., Paladino, S., Restelli, M., and Gatti, N. (2016). Budgeted multi-armed bandit in continuous action space. In *European Conference on Artificial Intelligence*, pages 560–568, The Hague, Netherlands. ECAI.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Vittori, E., Bernasconi de Luca, M., Trovò, F., and Restelli, M. (2020a). Dealing with transaction costs in portfolio optimization: Online gradient descent with momentum. In *ICAIF*.

Vittori, E., Likmeta, A., and Restelli, M. (2021). Monte carlo tree search for trading and hedging. In *Proceedings of the International Conference on AI for Finance*.

Vittori, E., Trapletti, M., and Restelli, M. (2020b). Option hedging with risk averse reinforcement learning. In *Proceedings of the International Conference on AI for Finance*.

Vodopivec, T., Samothrakis, S., and Ster, B. (2017). On monte carlo tree search and reinforcement learning. *Journal of Artificial Intelligence Research*, 60:881–936.

Wang, S., Jia, D., and Weng, X. (2018). Deep reinforcement learning for autonomous driving. *arXiv preprint arXiv:1811.11329*.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, King's College, Cambridge.

Wiese, M., Knobloch, R., Korn, R., and Kretschmer, P. (2020). Quant gans: Deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wu, Y., Shariff, R., Lattimore, T., and Szepesvári, C. (2016). Conservative bandits. In *International Conference on Machine Learning*, pages 1254–1262. PMLR.

Yang, J., Ye, X., Trivedi, R., Xu, H., and Zha, H. (2018a). Learning deep mean field games for modeling large population behavior. In *Proceedings of the International Conference on Learning Representations*.

Yang, X., Li, H., Zhang, Y., and He, J. (2018b). Reversion strategy for online portfolio selection with transaction costs. *INT J APP DECIS SCI*, 11(1):79–99.

Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., and Wang, J. (2018c). Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5571–5580. PMLR.

Zaki, M. J. and Meira, W. (2014). *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning*, pages 928–936, Washington D.C., United States. ICML.

APPENDIX $\mathcal{A}$

Proofs and Additional Material

A.1 Proof for Chapter 3

Lemma 3.1. *In the applications considered in this dissertation, let $s = (s_m, s_i)$, where s_m are the market prices and s_i are the agent's internal information, assuming $\mathcal{P}(s'_i|s_i, a)$ is deterministic, $\mathcal{P}(s'_m|s_m, a) = \mathcal{P}(s'_m|s_m)$ i.e., it does not depend on the agent's actions and $r(s, a) = r(s_m, a) + r(s_i, a)$ then:*

$$Q_\pi(s, a) = r(s_m, a) + r(s_i, a) + \gamma \mathop{\mathbb{E}}_{\substack{s'_m \sim \mathcal{P}(\cdot|s_m) \\ a' \sim \pi(\cdot|s')}} \big[Q_\pi(s', a')\big].$$

Futhermore, assuming no transaction costs, then we can exclude s_i from the state, thus $s = s_m$, $r(s_i, a) = 0$ and we obtain:

$$\arg\max_a Q(s, a) = \arg\max_a r(s_m, a).$$

Proof. Expanding the expected value of Q in Equation (3.6), and considering $\mathcal{P}(s'|s, a) = \mathcal{P}(s'_m|s_m)$, we get that:

$$\begin{aligned} Q(s, a) =& \mathcal{R}(s, a) + \sum_{s'} P(s'|s, a) \sum_{a'} \pi(a'|s') Q(s', a'), \\ =& \mathcal{R}(s_m, a) + \mathcal{R}(s_i, a) + \sum_{s'_m} P(s'_m|s_m) \sum_{a'} \pi(a'|s') Q(s', a'). \end{aligned}$$

Now if we consider no transaction costs, we can take out s_i from the state and obtain:

$$Q(s_m, a) = \mathcal{R}(s_m, a) + \sum_{s'} P(s'_m|s_m) \sum_{a'} \pi(a'|s'_m) Q(s'_m, a'),$$

and thus

$$\arg\max_a Q(s_m, a) = \arg\max_a \mathcal{R}(s_m, a).$$

□

A.2 Proofs for Section 5.1

Theorem 5.1. *The OGDM algorithm with $\eta_t = \frac{K_\eta}{\sqrt{t}}$, and $\lambda_t = \frac{K_\lambda}{t}$, for each value of $K_\eta, K_\lambda \in \mathbb{R}^+$, has a total regret of:*

$$R_T^C \leq \left[\frac{D^2}{K_\eta}\left(\frac{1}{2} + K_\lambda\right) + K_\eta \tilde{G}\left(2\gamma\sqrt{M} + \tilde{G}\right)\right]\sqrt{T}, \tag{5.8}$$

where $D = \sup_{\mathbf{a},\mathbf{y}\in X} ||\mathbf{a} - \mathbf{y}||_2$, and $\tilde{G} = \sup_{\mathbf{a}\in X} ||\nabla f_t(\mathbf{a})||_2 + \frac{DK_\lambda}{2K_\eta}$.

Proof. Recall that for a generic loss function $f_t : X \to \mathbb{R}$ and a convex set X the OGDM algorithm has the update rule in Equation (5.6). This formulation can be rewritten as:

$$\mathbf{a}_{t+1} = \Pi_X\left(\mathbf{a}_t - \eta_t \nabla \tilde{f}_t(\mathbf{a}_t)\right), \tag{A.1}$$

by defining: $\tilde{f}_t(\mathbf{a}) = f_t(\mathbf{a}) + \frac{\beta_t}{2}||\mathbf{a} - \mathbf{a}_{t-1}||_2^2$, with $\beta_t = \frac{\lambda_t}{\eta_t}$. Note that, by the triangle inequality $||\nabla \tilde{f}(\mathbf{a}_t)||_2 \leq \tilde{G}$.

Before presenting the main result, we use that, from Zinkevich (2003), given the update in Equation (A.1) we have:

$$\begin{aligned} &||\mathbf{a}_{t+1} - \mathbf{a}^*||_2^2 = ||\Pi_X(\mathbf{a}_t - \eta_t \nabla \tilde{f}_t(\mathbf{a}_t)) - \mathbf{a}^*||_2^2 \\ \leq &||\mathbf{a}_t - \mathbf{a}^*||_2^2 - 2\eta_t \langle \mathbf{a}_t - \mathbf{a}^*, \nabla \tilde{f}_t(\mathbf{a}_t)\rangle + \eta_t^2 ||\nabla \tilde{f}_t(\mathbf{a}_t)||_2^2, \end{aligned}$$

where we consider the fact that the projection operator $\Pi_{\Delta_X}(\cdot)$ is non-expansive. Rearranging the terms, we have:

$$\langle \mathbf{a}_t - \mathbf{a}^*, \nabla \tilde{f}_t(\mathbf{a}_t)\rangle \leq \frac{1}{2\eta_t}\left(||\mathbf{a}_t - \mathbf{a}^*||_2^2 - ||\mathbf{a}_{t+1} - \mathbf{a}^*||_2^2\right) + \frac{\eta_t}{2}\tilde{G}^2. \tag{A.2}$$

Using the above inequality, the total regret $R_T^C(OGDM)$ of the OGDM algorithm is bounded as follows:

$$R_T^C(OGDM) = \sum_{t=1}^{T} f_t(\mathbf{a}_t) - f_t(\mathbf{a}^*) + \gamma \sum_{t=1}^{T} ||\mathbf{a}_t - \mathbf{a}_{t-1}||_1 \tag{A.3}$$

$$
\begin{aligned}
&= \sum_{t=1}^{T} \tilde{f}_t(\mathbf{a}_t) - \tilde{f}_t(\mathbf{a}^*) - \sum_{t=2}^{T} \frac{\beta_t}{2} \left(||\mathbf{a}_t - \mathbf{a}_{t-1}||_2^2 - ||\mathbf{a}^* - \mathbf{a}_{t-1}||_2^2 \right) \\
&\quad + \gamma \sum_{t=1}^{T} ||\mathbf{a}_t - \mathbf{a}_{t-1}||_1 \qquad \text{(A.4)} \\
&\leq \sum_{t=1}^{T} \langle \mathbf{a}_t - \mathbf{a}^*, \nabla \tilde{f}_t(\mathbf{a}_t) \rangle + \sum_{t=1}^{T} \frac{\beta_t}{2} ||\mathbf{a}^* - \mathbf{a}_{t-1}||_2^2 \\
&\quad + \gamma \sum_{t=1}^{T} \sqrt{M} \eta_t ||\nabla \tilde{f}_t(\mathbf{a}_t)||_2 \qquad \text{(A.5)} \\
&\leq \sum_{t=1}^{T} \frac{1}{2\eta_t} \left(||\mathbf{a}_t - \mathbf{a}^*||_2^2 - ||\mathbf{a}_{t+1} - \mathbf{a}^*||_2^2 \right) + \sum_{t=1}^{T} \frac{\eta_t}{2} \tilde{G}^2 \\
&\quad + \sum_{t=1}^{T} \frac{\beta_t}{2} ||\mathbf{a}^* - \mathbf{a}_{t-1}||_2^2 + \gamma \sum_{t=1}^{T} \sqrt{M} \eta_t ||\nabla \tilde{f}_t(\mathbf{a}_t)||_2 \qquad \text{(A.6)} \\
&\leq \frac{D^2}{2\eta_1} + \frac{D^2}{2} \sum_{t=2}^{T} \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} \right) + \sum_{t=1}^{T} \frac{\eta_t}{2} \tilde{G}^2 \\
&\quad + \sum_{t=1}^{T} \frac{\beta_t}{2} ||\mathbf{a}^* - \mathbf{a}_{t-1}||_2^2 + \gamma \sum_{t=1}^{T} \sqrt{M} \eta_t ||\nabla \tilde{f}_t(\mathbf{a}_t)||_2 \qquad \text{(A.7)} \\
&\leq \frac{D^2}{2\eta_T} + \sum_{t=1}^{T} \frac{\eta_t}{2} \tilde{G}^2 + \sum_{t=1}^{T} \frac{\beta_t}{2} D^2 + \gamma \sqrt{M} \tilde{G} \sum_{t=1}^{T} \eta_t, \qquad \text{(A.8)}
\end{aligned}
$$

where we dropped the negative term and used the convexity of $\tilde{f}(\cdot)$ to derive Equation (A.5), and used the result in Equation (A.2) to derive Equation (A.6).

Finally, substituting $\beta_t = \frac{\lambda_t}{\eta_t}$, $\lambda_t = \frac{K_\lambda}{t}$, $\eta_t = \frac{K_\eta}{\sqrt{t}}$ in Equation (A.8), and using the fact that $\sum_{t=1}^{T} \frac{1}{\sqrt{t}} \leq 2\sqrt{T}$ concludes the proof. □

Data used in experimental section Figure A.1 shows the behavior of the 5 assets chosen for the portfolio optimization exercise of Figure 5.1 (b) and (c).

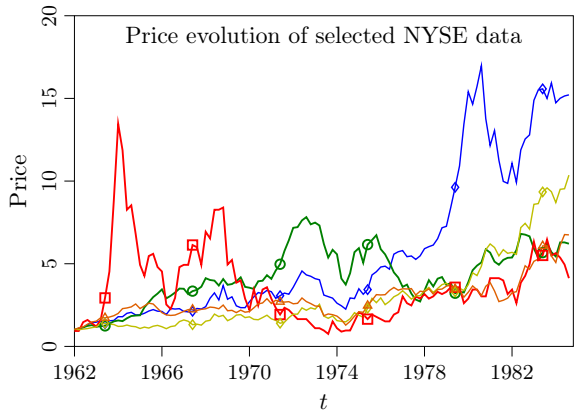

Figure A.1: *Assets from the NYSE dataset used for Figures 5.1(b) and (c). Price process is normalized to start at* 1.

A.3 Proofs and Additional Material for Section 5.2

Theorem 5.3. *In the OCO setting, there is no algorithm* $\mathfrak{U}$ *which obtains* $L_t \leq \tilde{L}_t$*, unless* $\theta_t = \tilde{\theta}$ *for all* $t \in [T]$.

Proof. Let k be the first round in which the algorithm $\mathfrak{U}$ plays $\theta_k \neq \tilde{\theta}$. If the loss function is $f_k(x) := f_k(\tilde{\theta}) + ||\tilde{\theta} - x||_2$, then, by the convexity of the space Θ, we can find $c \in (0,1)$ and $z \in \Theta$ s.t. $\theta_k = c\tilde{\theta} + (1-c)z$. This implies that $f_k(\theta_k) = f_k(\tilde{\theta}) + (1-c)||\tilde{\theta} - z||_2 > f_k(\tilde{\theta})$, showing that $L_t > \tilde{L}_t$. □

Theorem 5.4. *Let* $B(\tilde{\theta}, \omega_t)$ *be the conservative ball defined in Equation* (5.12) *and assume that Equation* (5.11) *is satisfied at round* $t-1$*. Then, each parameter* $\theta \in B(\tilde{\theta}, \omega_t) \cap \Theta$ *satisfies Equation* (5.11) *at round* t.

Proof. Given $\theta \in B(\tilde{\theta}, r_t) \cap \Theta$ we have:

$$f_t(\theta) - (1+\alpha) f_t(\tilde{\theta}) \leq \langle \nabla f_t(\theta), \theta - \tilde{\theta} \rangle - \alpha f_t(\tilde{\theta}) \leq G r_t - \alpha \varepsilon_l, \tag{A.9}$$

where the first inequality is given by the convexity of $f_t(\cdot)$, and the second inequality is given by the Cauchy-Schwarz inequality and by the fact that $\theta \in B(\tilde{\theta}, r_t)$ implies $||\tilde{\theta} - \theta||_2 \leq r_t$. Let us consider two cases: $r_t < D$, and $r_t = D$.

Case $r_t < D$**:** In this case, the value of the radius is $r_t = \frac{(1+\alpha)\tilde{L}_{t-1} - L_{t-1} + \alpha\varepsilon_l}{G}$. By substituting it in Equation (A.9), we conclude that:

$$f_t(\theta) - (1+\alpha) f_t(\tilde{\theta}_t) \leq (1+\alpha)\tilde{L}_{t-1} + L_{t-1}. \tag{A.10}$$

Case $r_t = D$**:** From the fact that $r_t \geq 0$ and using Equation (5.12), we obtain that:

$$\frac{L_{t-1} - (1+\alpha)\tilde{L}_{t-1} - \alpha\varepsilon_l}{GD} + 1 \leq 0 \tag{A.11}$$

$$GD - \alpha\varepsilon_l \leq (1+\alpha)\tilde{L}_{t-1} + L_{t-1}. \tag{A.12}$$

Combining the above result with the inequality in Equation (A.9), provides the same result presented in Equation (A.10).

The proof is concluded by rearranging the terms of Equation (A.10). □

Theorem 5.5. *Consider any OCO algorithm $\mathcal{A}$ that guarantees a regret of $R_T(\mathcal{A}) \leq \xi\sqrt{T}$. The CP algorithm using $\mathcal{A}$ as subroutine has the following regret bound:*

$$R_T(CP) \leq \xi\sqrt{T} + \tau DG, \tag{5.15}$$

for any $T > \tau$, where:

$$\tau = \frac{2\alpha\mu(DG + \alpha\mu) + \xi\left(\sqrt{\xi^2 + 4\alpha\mu(DG + \alpha\mu)} + \xi\right)}{2\alpha^2\mu^2}. \tag{5.16}$$

Proof. Using the convexity of the loss functions on the regret and the definition of θ_t in Equation (5.13), we have:

$$\begin{aligned} L_T - \tilde{L}_T &\leq \sum_{t=1}^{T}[\beta_t f_t(\tilde{\theta}) + (1-\beta_t)f_t(z_t) - f_t(\tilde{\theta})] \\ &= \sum_{t=1}^{T}(1-\beta_t)[f_t(z_t) - f_t(\tilde{\theta})] \qquad \text{(A.13)} \\ &\leq \sup_{\theta\in\Theta}\left(\sum_{t=1}^{T}(1-\beta_t)[f_t(z_t) - f_t(\theta)]\right) \leq \xi\sqrt{T}. \qquad \text{(A.14)} \end{aligned}$$

This shows that the CP algorithm has sublinear regret with respect to an algorithm that always chooses the default parameter $\tilde{\theta}$ over the entire time horizon T.

Combining Equation (5.14) and (5.12), we have:

$$\begin{aligned} \beta_t &\leq 1 - \frac{r_t}{||z_t - \tilde{\theta}||_2} \leq 1 + \frac{L_{t-1} - (1+\alpha)\tilde{L}_{t-1} - \alpha\varepsilon_l}{DG} \qquad \text{(A.15)} \\ &\leq 1 + \frac{\xi\sqrt{t} - (t-1)\mu\alpha}{DG}, \qquad \text{(A.16)} \end{aligned}$$

where we used the bound in Equation (A.14), the fact that the space Θ has radius D, and that $\tilde{\theta}$ is not a no-regret strategy, and, hence, there exists a $\mu > \varepsilon_l > 0$ s.t. $\tilde{L}_{t-1} > \mu(t-1)$.

On the other hand, we assumed that $\mathcal{A}$ is a no-regret strategy and, therefore, the regret of the algorithm $\mathcal{A}$ is sublinear, this means that there exists a round $\tau > 0$ s.t. Equation (A.16) is negative, and, consequently, for $t > \tau$, defined in Equation (5.16) we have $\beta_t = 0$. The value of τ is provided by the solution of the following equation $1 + \frac{\xi\sqrt{\tau} - \tau\mu\alpha}{DG} = 0$.

What we showed above also proves that the CP algorithm for $t > \tau$ eventually plays the same parameter as $\mathcal{A}$ since for all $t > \tau$ the pseudo-losses $g_t(\cdot)$ and the true losses $f_t(\cdot)$ coincide. Indeed, the regret of the CP algorithm can be written as:

$$R_T(\text{CP}) \leq \sum_{t=1}^{\tau}\left[\beta_t f_t(\tilde{\theta}) + (1-\beta_t)f_t(z_t) - f_t(\bar{\theta})\right] + \sum_{t=\tau+1}^{T}(f_t(z_t) - f_t(\bar{\theta})) \tag{A.17}$$

$$\leq \sum_{t=1}^{\tau} \beta_t \left[f_t(\tilde{\theta}) - f_t(z_t) \right] + \sum_{t=1}^{T} [f_t(z_t) - f_t(\bar{\theta})] \tag{A.18}$$

$$\leq \sum_{t=1}^{\tau} \beta_t \langle \nabla f_t(\tilde{\theta}), \tilde{\theta} - z_t \rangle + \sum_{t=1}^{T} [f_t(z_t) - f_t(\bar{\theta})] \tag{A.19}$$

$$\leq \tau DG + \xi\sqrt{T}, \tag{A.20}$$

where the inequality in Equation (A.17) uses the convexity of $f_t(\cdot)$. Equation (A.18) comes from the extension of the time horizon from $\{\tau, \ldots, T\}$ to $\{1, \ldots, T\}$. Equation (A.19) follows from the convexity of $f_t(\cdot)$ and the inequality in Equation (A.20) follows from the Cauchy-Schwarz inequality on the first term while the second term is the regret of the used no-regret algorithm $\mathcal{A}$. □

Theorem 5.6. *Consider any OCO algorithm $\mathcal{A}$ that guarantees a regret of $R_T(\mathcal{A}) \leq \rho \log(T)$. The CP algorithm using $\mathcal{A}$ as subroutine has the following regret bound:*

$$R_T(CP) \leq \rho \log(T) + \tau DG, \tag{5.17}$$

for any $T > \tau$, where:

$$\tau := \frac{\alpha e^2 \mu(DG + \alpha\mu) + 2\rho \left(\sqrt{\alpha e^2 \mu(DG + \alpha\mu) + \rho^2} + \rho \right)}{e^2 \alpha^2 \mu^2}. \tag{5.18}$$

Proof. The proof is similar to that of Theorem 5.6, we only report the steps that are significantly different from it. From Equation (A.14), which holds also in this setting, we obtain:

$$L_T - \tilde{L}_T \leq \rho \log(T). \tag{A.21}$$

This shows that the regret with respect to an algorithm which always chooses the default parameter $\tilde{\theta}$ is of the order $\mathcal{O}(\log(T))$. Following the same steps used to derive Equation (A.16), we have that $\beta_t \leq 1 + \frac{\rho \log(t) - t\mu\alpha}{DG}$. Therefore, β_t is zero after τ rounds, where τ is defined in Equation (5.18). The derivation of τ is provided in Lemma A.1. Finally, using the same argument used to derive Equation (A.20), we obtain the bound present in the theorem. □

Lemma A.1. *Consider any OCO algorithm $\mathcal{A}$ that guarantees a regret of $R_T(\mathcal{A}) \leq \rho \log(T)$. The last time the CP algorithm using $\mathcal{A}$ as subroutine plays the default parameter $\tilde{\theta}$,* i.e., *$\beta_t > 0$, is upper-bounded by τ, defined as:*

$$\tau := \frac{\alpha e^2 \mu(DG + \alpha\mu) + 2\rho \left(\sqrt{\alpha e^2 \mu(DG + \alpha\mu) + \rho^2} + \rho \right)}{e^2 \alpha^2 \mu^2}.$$

Proof. The CP algorithm plays at each time: $\theta_t = \beta_t \tilde{\theta} + (1 - \beta_t) z_t$, where $\beta_t = 1 - \frac{r_t}{||z_t - \tilde{\theta}||_2}$, and z_t is generated by a no-regret algorithm $\mathcal{A}$. From Equation (5.12) we know that:

$$\frac{L_{t-1} - (1 + \alpha)\tilde{L}_{t-1} - \alpha\epsilon_l}{GD} + 1 \geq \beta_t.$$

By using Equation (A.21) we have that eventually there is a time t for which:

$$\frac{1}{GD}\left[\rho\log(t) - \alpha\mu(t-1)\right] + 1 = 0, \tag{A.22}$$

as the left hand side goes to zero for t sufficiently large.

Finding the τ for which β_t becomes zero is equivalent to solving an equation of the type $A\log t = Bt - C$ with $A, B, C > 0$, which has no analytical roots. Thanks to the fact that the logarithm is concave, upper-bounding it in Equation (A.22) results in an equation whose result gives an upper bound on the solution of the original equation. Using that $\log x < \frac{2\sqrt{x}}{e}$, holding for each $x > 0$, the upper bound on the solution is of Equation (A.22) is provided by:

$$\frac{1}{GD}\left[\rho\frac{2\sqrt{t}}{e} - \alpha\mu(t-1)\right] + 1 = 0,$$

whose solution concludes the proof. □

A.3.1 Baseline Approaches

In what follows we present the Conservative Switching (CS) and CRDG algorithms, used as baseline for our experiments.

The Conservative Switching Algorithm

In this section, we design a more immediate approach to solve the COCO problem. The algorithm follows the idea by (Wu et al., 2016), and adapts it to the OCO setting: play the action z_t only if the conservativeness constraint is satisfied at round t and the current budget is big enough to sustain any loss at the next iteration; otherwise, it plays the default parameter $\tilde{\theta}$. The complete pseudo-code implementing this approach in the COCO setting, namely *Conservative Switching* (CS), is presented in Algorithm 11. The algorithm works as follows: at each round t, the algorithm computes its budget (Algorithm 11, Line 6). If the current budget is large enough to ensure the conservative constraint is satisfied even after suffering the loss at round t (Line 4), CS queries the action z_t from the no-regret algorithm $\mathcal{A}$ (Line 6). Otherwise, CS plays the default action $\tilde{\theta}$ (Line 9) keeping fixed the optimistic action z_t.

In what follows, we prove that the CS algorithm satisfies the conservativeness constraint in Equation (5.11) and has sublinear regret bound of the same order of the underlying algorithm $\mathcal{A}$.

Theorem A.1. *The CS algorithm applied to a generic online learning algorithm $\mathcal{A}$ is conservative.*

Proof. Let k be a time in which we played the optimistic action z_t given by algorithm $\mathcal{A}$, otherwise the constraint is trivially verified by the fact that the default parameter is inside the conservative ball. In this specific case we have that the following condition (Line 4 in Algorithm 11) is satisfied:

$$\begin{aligned} L_{k-1} + \epsilon_u - (1+\alpha)\epsilon_l &\leq \tilde{L}_{k-1}(1+\alpha) \\ L_{k-1} &\leq \tilde{L}_{k-1}(1+\alpha) + (1+\alpha)\epsilon_l - \epsilon_u. \end{aligned} \tag{A.23}$$

Algorithm 11: Conservative Switching

Initialize: Online learning algorithm $\mathcal{A}$, conservativeness level $\alpha > 0$, default parameter $\tilde{\theta} \in \Theta$
Set $\tilde{L}_0 \leftarrow 0$, $L_0 \leftarrow 0$
for $t \in [T]$ **do**
  if $L_{t-1} + \epsilon_u - (1+\alpha)\epsilon_l \leq \tilde{L}_{t-1}(1+\alpha)$ **then**
    $z_t \leftarrow \mathcal{A}(f_{t-1}(z_{t-1}))$
    Select $\theta_t \leftarrow z_t$
  else
    $z_t \leftarrow z_{t-1}$
    Select $\theta_t \leftarrow \tilde{\theta}$
  Suffer loss $f_t(\theta_t)$
  Observe feedback $f_t(z_t)$ and $f_t(\tilde{\theta})$

Moreover, we have that, due to the fact that the loss function is bounded from below by ϵ_l, we have:

$$\tilde{L}_{k-1}(1+\alpha) + (1+\alpha)\epsilon_l \leq \tilde{L}_{k-1}(1+\alpha) + f_k(\tilde{\theta})(1+\alpha) = \tilde{L}_k(1+\alpha). \tag{A.24}$$

The loss of the CS algorithm becomes:

$$\begin{aligned} L_k &= L_{k-1} + f_k(z_k) \\ &\leq \underbrace{\tilde{L}_{k-1}(1+\alpha) + (1+\alpha)\epsilon_l}_{\leq \tilde{L}_k(1+\alpha)} \underbrace{-\epsilon_u + f_k(z_k)}_{\leq 0} && \text{(A.25)} \\ &\leq \tilde{L}_k(1+\alpha), && \text{(A.26)} \end{aligned}$$

where Equation (A.25) follows from the fact that we played the $\mathcal{A}$ algorithm for the round k, thus the condition in Equation (A.23) holds, and Equation (A.26) is derived using Equation (A.24) and from the fact that the loss function is bounded from below by ϵ_l. This concludes the proof. □

Theorem A.2. *Consider any OCO algorithm $\mathcal{A}$ that guarantees a regret of $R_T(\mathcal{A}) \leq \xi\sqrt{T}$. The CS algorithm using $\mathcal{A}$ as subroutine has the following regret bound:*

$$R_T(CS) \leq \xi\sqrt{T} + \tau DG,$$

where:

$$\tau := \frac{\xi^2 - 2\alpha\mu(\epsilon_u - \epsilon_l)}{2\alpha^2\mu^2} + \frac{1}{2}\sqrt{\frac{\xi^4 + 4\alpha\xi^2\mu(\epsilon_u - \epsilon_l)}{\alpha^4\mu^4}}.$$

Proof. Let us define $C_\alpha := \epsilon_u - (1+\alpha)\epsilon_l$ and let $k \geq 1$ be a time in which we played the default strategy and define S and R as the set of rounds in which the CS algorithm played the default parameter and the parameter chosen by $\mathcal{A}$ up to time k, respectively. Formally:

$$S = \{t \leq k \text{ s.t. } (1+\alpha)\tilde{L}_{t-1} - L_{t-1} \leq C_\alpha\},$$

$$V = \{t < k \text{ s.t. } (1+\alpha)\tilde{L}_{t-1} - L_{t-1} > C_\alpha\}.$$

By definition of the cumulative loss L_{k-1} and since $S \cup V = [k-1]$, we have:

$$\begin{aligned}
\sum_{t\in V} f_t(z_t) &= L_{k-1} - \sum_{t\in S\setminus\{k\}} f_t(\tilde{\theta}) \\
&\geq (1+\alpha)\tilde{L}_{k-1} - C_\alpha - \sum_{t\in S\setminus\{k\}} f_t(\tilde{\theta}) \\
&= (1+\alpha)\tilde{L}_{k-1} - C_\alpha \underbrace{- \sum_{t\in S\setminus\{k\}} f_t(\tilde{\theta}) - \sum_{t\in V} f_t(\tilde{\theta})}_{=-\tilde{L}_{k-1}} + \sum_{t\in V} f_t(\tilde{\theta}) \\
&= \alpha\tilde{L}_{k-1} - C_\alpha + \sum_{t\in V} f_t(\tilde{\theta}),
\end{aligned}$$

where the first inequality follows from the fact that $k \in \tilde{S}$ and, therefore, $L_{k-1} \geq (1+\alpha)\tilde{L}_{k-1} - C_\alpha$. Finally, using that $\tilde{L}_k > \mu k$, since the default parameter $\tilde{\theta}$ is not a no-regret strategy we get:

$$\sum_{t\in V} f_t(z_t) \leq \alpha\tilde{L}_{k-1} - C_\alpha + \sum_{t\in V} f_t(\tilde{\theta}) \tag{A.27}$$

$$\sum_{t\in V} [f_t(z_t) - f_t(\tilde{\theta})] \geq k\alpha\mu - (\epsilon_u - \epsilon_l). \tag{A.28}$$

Since the algorithm $\mathcal{A}$ has been run only on the set V, the left hand side is bounded by the regret of $\mathcal{A}$ on the set V, and, consequently, also on the entire time horizon k. Taking the limit $k \to +\infty$, there will be a time τ in which Equation (A.28) is not verified anymore, proving that the last time the algorithm plays the default parameter satisfies $t_0 \leq \tau < +\infty$.

Solving for τ the following equation:

$$\xi\sqrt{\tau} = \alpha\tau\epsilon_l - (\epsilon_u - \epsilon_l),$$

we get:

$$\tau := \frac{\xi^2 - 2\alpha\mu(\epsilon_u - \epsilon_l)}{2\alpha^2\mu^2} + \frac{1}{2}\sqrt{\frac{\xi^4 + 4\alpha\xi^2\mu(\epsilon_u - \epsilon_l)}{\alpha^4\mu^4}}.$$

With this result we can bound the regret of the CS algorithm as follows:

$$R_T(CS) \leq \xi\sqrt{T} + \tau DG.$$

□

Theorem A.3. *Consider any OCO algorithm $\mathcal{A}$ that guarantees a regret of $R_T(\mathcal{A}) \leq \rho\log(T)$. The CS algorithm using $\mathcal{A}$ as subroutine has the following regret bound:*

$$R_T(CS) \leq \rho\log(T) + \tau DG,$$

Algorithm 12: RD-$1D$

Initialize: Learning rate η_1, upper bound $\bar{H}$, initial parameter θ_0
Set $i \leftarrow 1$, $Q_1 \leftarrow 0$
for $t \in [T]$ **do**
  Play θ_t and suffer loss $f_t(\theta)$
  $Q_i \leftarrow Q_i - f_t(\theta_t)$
  if $Q_i < \eta_i \bar{H}$ **then**
    $\theta_{t+1} \leftarrow \theta_t - \eta_1 \nabla f_t(\theta_t)$
  else
    $i \leftarrow i + 1$
    $Q_i \leftarrow 0$
    $\eta_i \leftarrow 2\eta_{i-1}$
    $\theta_t = \theta_0 - \eta_1 \nabla f_t(\theta_t)$

where:

$$\tau := \frac{2\rho^2 + \alpha e^2 \mu(\epsilon_u - \epsilon_l)}{\alpha^2 e^2 \mu^2} + 2\sqrt{\frac{\rho^4 + \alpha e^2 \rho^2 \mu(\epsilon_u - \epsilon_l)}{\alpha^4 e^4 \mu^4}}.$$

Proof. The proof follows the same steps as the one of Theorem A.2 up to Equation (A.28).

Now the left hand side can be bounded by $\rho \log(k)$ that, on its turn, is bounded as $\rho \log k \leq \rho \frac{2\sqrt{k}}{e}$, which holds for $k > 1$. Thanks to this inequality the value of an upper bound τ on the value of the last instant CS plays the default parameter $\tilde{\theta}$ is provided by the analytical solution to the following equation:

$$\rho \frac{2\sqrt{\tau}}{e} = \alpha \tau \epsilon_l - (\epsilon_u - \epsilon_l).$$

With the above result we can bound the regret of the CS algorithm as follows:

$$R_T(CS) \leq \rho \log T + \tau DG,$$

This concludes the proof. □

Even if from these results it is not possible to state that CP attains a strictly better regret than CS. The intuition behind this superior performance is that during the first phase of the optimization, *i.e.,* $r_t < D$, we are less constrained using the CP algorithm since we are allowed to select the parameter for the next round on the conservative ball $B(\tilde{\theta}, r_t)$ border. Conversely, the CS algorithm plays the default parameter $\tilde{\theta}$ until enough budget is collected. Concerning the computational cost of the CS algorithm, it has a constant computational overhead with respect to the original algorithm $\mathcal{A}$ due to the evaluation of the losses $f_t(\theta_t)$, and $f_t(\tilde{\theta})$.

The Constrained Reward Doubling Guess Algorithm

In this section we provide the description of the Conservative Reward Doubling Guess (CRDG). The pseudo-code of the CRDG algorithm is provided in Algorithms 12-15.

Algorithm 13: RD-$1D$-Guess

Initialize: Learning rate ε, initial parameter θ_0
Set $i \leftarrow 1$, $H_i = 1$, $\eta_i = \varepsilon$, $H = 0$
while $t \in [T]$ **do**
  $\mathcal{A} = RD\text{-}1D(\eta_i, H_i, \theta_0)$
  while $H \leq H_i$ **do**
    Play θ_t from algorithm $\mathcal{A}$ and suffer loss $f_t(\theta_t)$
    $H \leftarrow H + \nabla f_t(\theta_t)^2$
    $t \leftarrow t + 1$
  $i \leftarrow i + 1$
  $H = 0$
  $H_i = 2H_{i-1}$
  $\eta_i = \eta_{i-1}/4$

Algorithm 14: RD-ND-Guess

Initialize: Learning rate ε, initial parameter θ_0
for $k \in [d]$ **do**
  Set $\mathcal{A}_k = RD\text{-}1D$-Guess$(\varepsilon/k, \theta_{0,k})$
while $t \in [T]$ **do**
  for $k \in [d]$ **do**
    Get $\theta_{t,k}$ from $\mathcal{A}_k$
    Play θ_t and suffer loss $f_t(\theta_t)$

Algorithm 15: $CRDG$

Initialize: Learning rate ε, initial parameter θ_0, parameter set Θ
Set $\mathcal{A} = RD\text{-}ND$-Guess$(\varepsilon, \theta_0)$
for $t \in [T]$ **do**
  Get z_t from $\mathcal{A}$
  Play $\theta_t = \Pi_\Theta(z_t)$ and suffer loss $f_t(\theta)$
  Observe the gradient of the loss $\nabla f_t(\theta_t)$
  Update $\mathcal{A}$ using $\nabla g_t(\theta_t)$

In particular, the RD-1D algorithm, presented in Algorithm 12, performs a search, using a gradient descend approach, on the space $\mathbb{R}$ (Lines 7 and 12), and restarts from the point θ_0 (Line 12) every times it collects enough wealth, formally, if $Q_i \geq \eta_i \bar{H}$. At every restart, it doubles its learning rate (Line 11). Notice that the RD-1D algorithm requires the knowledge of an upper bound on the variance of the loss gradients $\bar{H} \geq \sum_{t=1}^{T} (\nabla f_t)^2$. Conversely, if the quantity $\bar{H}$ is unknown, one can resort to the RD-1D-Guess algorithm, presented in Algorithm 13. This algorithm performs the doubling trick (Besson and Kaufmann, 2018) on the quantity $\bar{H}$, using the RD-1D algorithm as a subroutine.

The extension of the RD-1D-Guess algorithm to parameter spaces $\Theta \subseteq \mathbb{R}^d$, with $d > 1$, is provided by RD-ND-Guess, which uses an instance RD-1D-Guess as subroutine, applying it to each one of the d coordinates separately. The pseudo-code of the RD-ND-Guess is

presented in Algorithm 15, which, at the beginning, sets d instances $\{\mathcal{A}_1, \dots, \mathcal{A}_D\}$ of the RD-1D-Guess algorithm, and at round t, selects the k-th component $\theta_{t,k}$ of the parameter θ_t querying the algorithm $\mathcal{A}_k$. Each Algorithm $\mathcal{A}_k$ is run by providing it with the k-th coordinate of the gradient of the loss $\nabla f_t(\theta_{t,k})$.

The aforementioned algorithms have been designed to work in unconstrained domains. However, they can be adapted to work in a convex parameter space Θ, by utilizing the *Constrained Set Reduction* (CSR) Algorithm described in (Cutkosky and Orabona, 2018). The CRDG algorithm, presented in Algorithm 15, describes the CSR meta-algorithm applied to Algorithm 15. It works by projecting the parameter predicted by the CRDG algorithm into the set Θ (Line 4). Moreover, it requires that the losses fed to the CRDG algorithm are redefined to penalize parameter outside the parameter space Θ using the following pseudo-loss function:

$$g_t(x) := \frac{1}{2}\left[\langle x, \nabla f_t(\theta_t)\rangle + ||\nabla f_t(\theta_t)||_2 S_\Theta(x)\right],$$

where $S_\Theta(x) := \arg\inf_{y\in\Theta} ||x-y||_2$ is the distance between x and the set Θ. Finally, the gradient of such a function is used to update the subroutine (Line 7).

The Reward Doubling Guess (RDG) algorithm has been proposed by (Streeter and McMahan, 2012) to solve an instance of the unconstrained online optimization problem. In this framework the guarantees are given with respect to a so called *comparator parameter* $\mathring{\theta} \in \mathbb{R}^d$. When the algorithm starts from the origin of $\mathbb{R}^d$, the RDG algorithm provides a regret bound of:

$$R_T(\mathring{\theta}) \leq ||\mathring{\theta}||_2 \sqrt{T} \log\left[\frac{d(1+||\mathring{\theta}||_2)T}{\varepsilon}\right], \tag{A.29}$$

where $\varepsilon > 0$ is the learning rate of the procedure. Without loss of generality the algorithm can start from a generic point in $\mathbb{R}^d$, and restate the bound in terms of the distance from the starting point and the general comparator parameter $\mathring{\theta}$. In the case the comparator parameter is the starting point of the algorithm the RDG algorithm, it guarantees a regret of:

$$R_T(\mathring{\theta}) \leq \varepsilon.$$

The use of the RDG algorithm in a constrained setting, *i.e.,* over a convex parameter set $\Theta \subset \mathbb{R}^d$, requires to use a conversion provided by the CSR algorithm. We named *Constraint Reward Doubling Guess* (CRDG) the combination of this algorithm together with the RDG algorithm. Thanks to the reduction algorithm we project onto the set Θ, and convert any algorithm with regret R_T in the unconstrained setting to an algorithm that guarantees $2R_T$ in the constrained one. Overall, the resulting algorithm provides the following guarantee:

$$R_T(\mathring{\theta}) \leq 2||\mathring{\theta}||_2 \sqrt{T} \log\left[\frac{d(1+||\mathring{\theta}||_2)T}{\varepsilon}\right], \quad \forall \mathring{\theta} \in \Theta, \tag{A.30}$$

and

$$R_T(\tilde{\theta}) \leq 2\varepsilon.$$

for a specific known parameter $\tilde{\theta} \in \Theta$.

To guarantee the budget constraint of Equation (5.11) as required by the COCO framework, we need to set $\varepsilon = \mu\alpha/2$ in the CRDG algorithm. Two main differences emerge from this analysis. First, setting such ε requires the a priori knowledge of the parameter μ, which conversely is not necessary to run either the CP or the CS algorithms. Second, the regret bound in Equation (A.30) has an extra term of $\log(T)$ compared with ones provided by the CP and CS algorithms. Moreover, the choice of the learning rate as $\varepsilon = \mu\alpha/2$ is too conservative to provide a performing algorithm in practice.

A.4 Additional Material for Chapter 7

A.4.1 Q-learning for GMFG

In this section we describe the algorithm used to train the Q_2 and Q_4 agents used in the experimental section. In particular, in the work by (Guo et al., 2020) the authors present an algorithm designed for finite action and state spaces $\mathcal{A}$ e $\mathcal{S}$, that builds on the idea of Algorithm 6. We report it in Algorithm 16 for completeness. The algorithm exploits the Q-learning algorithm (Equation (3.8)) to update the approximated equilibrium strategy and uses a simulation approach to evolve the population distribution.

Algorithm 16: Q-learning for GMFG

Initialize: initial state-action distribution $\mathcal{L}_0$, environment simulator $\mathcal{E}(\cdot,\cdot;\mathcal{L})$, number of iterations I, W and J, learning rate α and ε greedy parameter
for $k \in [K]$ **do**
  Initialize $\hat{Q}_{w,0}(s,a) = 0 \; \forall a \in \mathcal{A}, s \in \mathcal{S}$
  for $j \in [J]$ **do**
    $a_t \leftarrow \begin{cases} \arg\max_a Q_{w,j}(s_t,a) & \text{with probability } 1-\varepsilon \\ \text{uniform random} & \text{with probability } \varepsilon \end{cases}$
    $(s_{t+1}, r_{t+1}) = \mathcal{E}(s_t, a_t; \mathcal{L}_{k-1})$
    $Q_{w,j+1}(s_t,a_t) =$
     $Q_{w,j}(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_{a\in\mathcal{A}} Q_{w,j}(s_{t+1},a) - Q_{w,j}(s_t,a_t)\right]$
  Extract $\pi_w(s) = \frac{\exp(\tau \hat{Q}_{w,J}(s,\cdot))}{\sum_{a\in\mathcal{A}} \exp(\tau \hat{Q}_{w,J}(s,a))} \; \forall s \in \mathcal{S}$
  $\mu_w \leftarrow \int_{\mathcal{A}} \mathcal{L}_{w-1}(s,a)\, da$
  Initialize $\mathcal{L}_w(s,a) \leftarrow 0 \; \forall s \in \mathcal{S}, a \in \mathcal{A}$
  for $i \in [I]$ **do**
    $s_i \sim \mu_w, a_i \sim \pi_w(s_i)$
    $(s_i', r_i') \leftarrow \mathcal{E}(s_i, a_i; \mathcal{L}_{w-1})$
    $\mathcal{L}_w(s_i', a_i) \leftarrow \mathcal{L}_w(s_i', a_i) + 1/I$
Return π_W and $\mathcal{L}_W$

Algorithm Description

Algorithm 16 is similar to Algorithm 7, we highlight here the main differences. Algorithm 16 works by using the Q-learning algorithm to update the Q function. Specifically, it

initializes to 0 the Q-function for each state and action (Line 3), then, for J steps, it either selects the action that currently has the best Q-value with probability $1 - \varepsilon$, or it selects a random action with probability ε (Line 5), a.k.a. ε-greedy algorithm. Using the simulator $\mathcal{E}(\cdot, \cdot; \mathcal{L})$, it extracts the next state s_{t+1} and the reward r_{t+1}, that are used to update of the Q-function (Line 7). After that, the algorithm (namely the update of the population $\mathcal{L}_k$) is identical to Algorithm 7.

Parameter Setting For the experimental evaluation of Q_2 and Q_4, we used $K = 5$ outer iterations and $J = 10 \cdot 10^5$ inner iterations, we fixed the learning rate to $\alpha = 0.8$ and the discount factor to $\gamma = 0.95$.

A.5 Proof for Chapter 8

Lemma 8.1. *Consider the return variance σ_π^2 (Equation* (4.9)*) and the reward volatility ν_π^2 defined in (Equation* (4.11)*). The following inequality holds:*

$$\sigma_\pi^2 \leq \frac{\nu_\pi^2}{(1-\gamma)^2},$$

Proof. Expanding the square term in Equation (4.9),

$$\sigma_\pi^2 = \mathbb{E}_\tau\big[(\sum_t \gamma^t R_t)^2\big] - J_\pi^2/(1-\gamma)^2.$$

As a consequence of the Cauchy-Schwarz inequality,

$$\mathbb{E}_\tau\big[(\sum_t \gamma^t R_t)^2\big] \leq \mathbb{E}_\tau\big[(\sum_t \gamma^t R_t^2)\big]/(1-\gamma).$$

Rearranging the terms the lemma is proven. □

APPENDIX B

Additional Financial Material and ML Tools

In this supplementary material chapter, we provide information on topics that are relevant for the dissertation, but not the main focus.

B.1 Additional Material on Financial Instruments

This section adds detail to the concepts of Section 2.2.

B.1.1 Stocks

Stocks are the most popular type of financial asset. They represent the ownership of a corporation and a single unit is referred to as a share.

Although stocks are traded on stock exchanges, they are easily accessible even to retail investors through online brokers. Trading stocks may seem more restrictive than trading derivatives as they are funded. However, it is possible to increase one's buying power with the use of *margin*, *i.e.,* a short-term loan offered directly by the broker. Moreover, it is possible to borrow a stock to sell it, this is called shorting. Once the short position is closed, the stock is given back to the lender. To borrow the stock, the holder is required to pay an interest rate. Thus holding short positions can be quite costly.

B.1.2 Bonds

The price of a bond P_t is quoted on the markets, given the price it is possible to calculate the yield i by implicitly solving:

$$P_t = \sum_{n=1}^{N} \frac{C}{(1+i)^n} + \frac{M}{(1+i)^N},$$

where M is the value payed at maturity (thus the price to which P converges at maturity), C is the coupon payment, defined as $M \times i_F$ where i_F is the contractual interest rate. The yield i is composed of credit risk and interest rate risk. Specifically, knowing the risk-free market interest rate r_f, the credit risk can be calculated as $i - r_f$. As we can see from the price equation, if interest rates increase, then P decreases, this means that by buying a bond we are also hoping interest rates will decrease (we are short rates). Credit risk is present in the context of corporate bonds, or specific government bonds that are not risk-free (such as the italian BTPs). Furthermore, while for a public company there is usually a relatively small number of types of stock that can be traded (*e.g.*, preferred or common stock), the same company may issue a variety of different bonds with different maturities as well as other diverse characteristics.

B.1.3 Forwards

Forwards contracts are OTC instruments and belong to the category of derivatives, as their price depends on an underlying asset (also referred to as spot). They represent an agreement to buy or sell an asset for a certain price at a certain future time, in contrast with a spot contract, which is an agreement to make a trade today. The payoff of a long position in a forward contract is $S_T - K$ where K is the agreed upon price and S_T is the spot price of the asset at maturity.

B.1.4 Futures

Futures contracts can be thought of as standardized forwards: they share all the same properties with the only difference of being actively traded on regulated exchanges. Futures are leveraged instruments, which means that it is not necessary to fund the future contract but there is an initial margin requirement to pay to the broker. In fact, since futures contracts are ruled at delivery, there is a strict regulation aiming to avoid large losses in case of financial difficulties of one of the counterparties. The increase or decrease in value of the contract is exchanged at the end of each trading day. To handle this exchange, brokers usually require a margin to be posted initially and maintained if necessary. When the value of an investor's margin account falls below the minimum required amount, the investor receives a margin call asking to deposit additional money as a means to bring the account above the minimum amount required.

There are multiple futures contracts, with different delivery dates, which are active at the same time on a single asset. A futures contract is specified using the delivery or expiry month, and, in general, the most liquid future contract is the one that is closest to delivery. A futures contract is defined as *on the run* when it is the closest one to delivery, the others are referred to as *off the run*. Before the on the run contract delivers, on the *Last Trade*

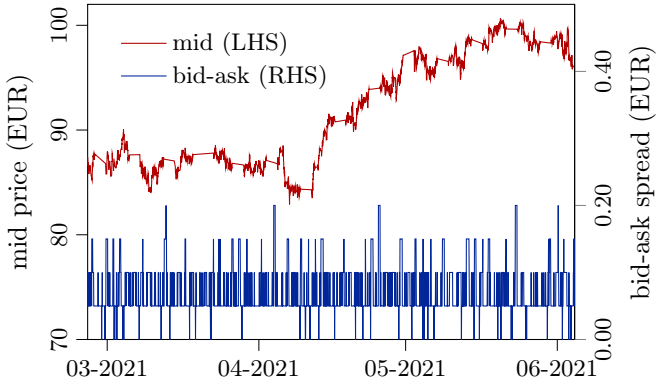

Figure B.1: *The evolution of the SX7E mid price (EUR, left axis) and bid-ask spread (EUR, right axis) from mid 03-2021 to 06-2021.*

Date (LTD), it is necessary to switch the position to the upcoming on the run contract, this means closing the current position and re-opening it on the new series. This procedure is referred to as *roll*. If the future is not rolled, it is either cash settled or physically settled. If it is physical delivery, one will receive the underlying bond in case of bond futures, or even the physical commodity in case of commodity futures.

Trading Futures Futures may exhibit different characteristics depending on their underlying, on the exchange, and so forth. Let's take the EURO STOXX Banks Index Futures (SX7E) as an example. The SX7E is a capitalization weighted index that includes banks and financial institutions located in the European Monetary Union. In Figure B.1 we can see the evolution of mid prices and the bid-ask spread during 4 months of trading. As we can see from the figure, the bid-ask spread is generally 0.05 or 0.1, which corresponds to one or two tick sizes (see Section 2.4.2). These future contracts deliver every March, June, September and December. Trading hours are between 2:10am and 10:00pm CET, even though most of the trading happens between 9:00am and 6:00pm.[44] Futures are not traded as single units, but as lots and each lots contains multiple units, in the SX7E the number of units is 50.

B.1.5 Equity Options

In this section, we add detail to equity options as defined in Section 2.2.3. It is often complex to decide which option to trade as it is necessary to pick not only the underlying risk but also the strike and expiry. For each underlying instrument there may be up to around 100 different strikes, and usually the most liquid are those close to the At The Money (ATM), *i.e.*, the strikes that are closest to the current price of the underlying. Expiry dates vary depending on the underlying instrument, if we consider again the SX7E, there is an expiry every month for the next 6 months, then they thin out. From a practical point of view, once the specific option to trade has been chosen, the trading is the same as in the

[44] Central European Time.

case of a futures contract.

B.1.6 Credit Default Swaps

In this section, we add detail to CDS indexes as defined in Section 2.2.4.

A CDS is a financial derivative or contract that allows an investor to swap or offset her/his credit risk with that of another investor. Every 6 months, on the 20/09 and 20/03, or the business day immediately thereafter if it is not a business day, a new Series of the index is originated. The new Series will be called on the run, until a new one is generated. Different maturities are traded for this CDS index (3, 5 and 10 years) with the maturity date that is the 20/12 or 20/06, respectively. For our purposes we consider the CDS index with 5Y maturity because it is the most liquid and there are many more options compared to the other maturities. The index composition may be different from one series to the other either in the number of constituents or in the CDS reference entities considered.[45] At the present time (December 2021), the SNFRIN index on the run is the Series 35, started the 22 March 2021 and with maturity date 20 June 2026.

Each CDS index has a premium leg and a protection leg. The premium leg has standardized coupon dates: 20/03, 20/06, 20/09 and 20/12 (or the business day immediately thereafter if it is not a business day). The coupon C is defined as:

$$C = \mathrm{N}\,\tau(t_{i-1}, t_i)\,1\%,$$

where N is the notional expressed in Euro, $\tau(t_{i-1}, t_i)$ is the year fraction (the number of days between the present t_i and the previous t_{i-1} coupon date divided by 360) and 1% is the standardized coupon, which is paid on t_i.[46]

In the event of a default of one of the j-th Series constituents occurred before the Series' maturity, the protection leg pays, an amount equal to $\frac{\mathrm{LGD}_j\,\mathrm{N}}{n}$, where LGD_j is the loss given default and n the number of constituents at the default time.[47] Upon default of a constituent and the subsequent settlement of the relative protection leg, a new version of the Series is spun-off including the surviving constituents and the original notional N is rescaled accordingly. The upfront is defined in Equation (2.3).

B.1.7 Interest Rate Swaps

Interest Rate Swaps (IRS) are contracts where a fixed coupon payment F is swapped with a variable payment f that depends on the current market interest rate. The price of an IRS can be defined as:

$$\mathrm{P}_t = F \sum_{t<\{t_i\}\leq t_n} \tau(t_{i-1}, t_i) D(t, t_i) - \sum_{t<\{t_i\}\leq t_n} f(t, t_i)\tau(t_{i-1}, t_i) D(t, t_i),$$

where $f(t, t_i)$ is the forecast variable rate at time t_i, $D(t, t_i)$ is the forecast discount factor and $\tau(t_{i-1}, t_i)$ is the year fraction of the days between t_i and t_{i-1}. IRS are OTC

[45] For index versions originated before March 2015 the number of constituents was 25.

[46] The only caveat is about the last coupon date, which corresponds to the index maturity equal to the 20/06 or 20/12 even in case that day is a holiday, with a year fraction including an extra day.

[47] LGD_j is equal to $1 - R_j$, where R_j is the recovery rate determined at the end of the ISDA CDS auction triggered by the credit event.

instruments, they can be traded for example on the Bloomberg MTF. There is a specific terminology, "being long interest rates" means we are paying fixed, and vice versa "being short" means receiving fixed. The sensitivity to 1 basis point of movement of the interest rate in consideration can be approximated as $N \times \tau(t_0, T)/10^3$ where N is the notional.

B.1.8 Sensitivities

Sensitivities are used as features in the state in the DVA hedging problem of Appendix C. The sensitivity of bonds or derivative instrument represent how much the price changes with respect to the movement of the underlying risk factor. Formally, let X be the price of each derivative instrument, and y the underlying risk, the sensitivity is generally defined as $\frac{\partial X}{\partial y}$. As we saw in Section 2.2.3, in the case of options, this is referred to as delta risk. To be consistent with regulatory environments, we use the same definition as in the *Minimum capital requirements for market risk* (Basel Committee, 2016).

- let r_t be the interest rate at tenor t of the risk-free yield curve in a given currency, the sensitivity is defined as: (for BTP and Bund)

$$\frac{X(r_t + 0.0001) - X(r_t)}{0.0001},$$

- let c_t be the credit spread at tenor r, the sensitivity, also referred to as CS01, is defined as: (for SNRFIN and BTP)

$$\frac{X(c_t + 0.0001) - X(c_t)}{0.0001},$$

- let EQ be the market value of the cash equity asset taken into consideration, the sensitivity is defined as: (for SX7E)

$$\frac{X(1.01EQ) - X(EQ)}{0.01},$$

These formulas, represent the way in which the sensitivities must be represented to calculate the risk weighted assets, necessary to quantify the cost of capital as mentioned in Section 2.1.2.

B.2 Almgren-Chriss for Optimal Execution

In this section, we describe the Almgren-Chriss (AC) optimal execution framework (Almgren and Chriss, 2001). It is used as a baseline to compare our approach in Chapter 9. Recalling the notation defined in Section 2.4.3, a trading trajectory is defined as a list $\{x_0, ..., x_N\}$, where x_k is the number of units held at time t_k. We fix $x_0 = X$ and liquidation at time T requires $x_N = 0$. Equivalently, it is possible to specify a "trade list" $\{n_1, ..., n_N\}$, where $n_k = x_{k-1} - x_k$ is the number of units sold between times t_{k-1} and t_k.

AC assumes that the security price evolves according to the following discrete random process:

$$P_k = P_{k-1} + \sigma\tau^{\frac{1}{2}}\varepsilon_k - \tau g\left(\frac{n_k}{\tau}\right), \tag{B.1}$$

where P_k is the mid price at time k, σ is the volatility of the security, τ the length of the time interval, n_k is the volume traded at time k, $g(\cdot)$ is the permanent price impact and ε_k are draws from guassian independent random variables with zero mean and unit variance. Trades induce also a temporary impact modelled by the following function:

$$\hat{P}_k = P_{k-1} - h\left(\frac{n_k}{\tau}\right),$$

meaning that the price of the security considered is $\hat{P}_k$.

To optimize the execution, the IS of Equation (4.7) is used to measure the performance:

$$IS = XP_0 - \sum_{k=1}^{N} n_k \hat{P}_k.$$

Inserting Equation (B.1) in the price process, it is possible to write the IS as:

$$\sum_{k=1}^{N} n_k h\left(\frac{n_k}{\tau}\right) - \sum_{k=1}^{N} \left((\sigma\tau^{\frac{1}{2}}\varepsilon_k) - \tau g\left(\frac{n_k}{\tau}\right)\right) x_k.$$

Now the objective is to minimize both the expected IS and its variance. Since the IS is a random variable, it is possible to compute its expectation and variance as:

$$\mathbb{E}[IS] = \mathbb{E}\left[\sum_{k=1}^{N} \tau x_k g\left(\frac{n_k}{\tau}\right) + \sum_{k=1}^{N} n_k h\left(\frac{n_k}{\tau}\right)\right],$$
$$\text{Var}[IS] = \sigma^2 \sum_{k=1}^{N} \tau x_k^2.$$

At this point, optimizing the execution, consists in finding the minimum of the problem:

$$\min(\mathbb{E}[IS] + \lambda \text{Var}[IS]).$$

By defining the permanent price impact $g(\cdot)$ and the temporary price impact $h(\cdot)$ as linear functions:

$$g\left(\frac{n_k}{\tau}\right) = \gamma\frac{n_k}{\tau};$$
$$h\left(\frac{n_k}{\tau}\right) = \epsilon\ \text{sgn}(n_k) + \frac{\eta}{\tau}n_k;$$

it is possible to find a closed form solution that depends on the risk aversion parameter λ, as the optimization problem is a quadratic and strictly convex function:

$$n_j = \frac{2\sinh(\frac{1}{2}k\tau)}{\sinh(kT)} \cosh\left(k(T - t_{j-\frac{1}{2}})\right) X \text{ for } j \in \{1, ..., N\}, \tag{B.2}$$

where $k = \frac{1}{\tau}\cosh^{-1}\left(\frac{\tau^2}{2}z^2 + 1\right)$ and $z^2 = \frac{\lambda\sigma^2}{\eta(1-\frac{\gamma\tau}{2\eta})}$.

In Figure B.2, we can see the execution trajectories resulting from the solution to the AC model (Equation (B.2)), while varying the risk-aversion coefficient. A higher risk aversion leads to a quicker execution.

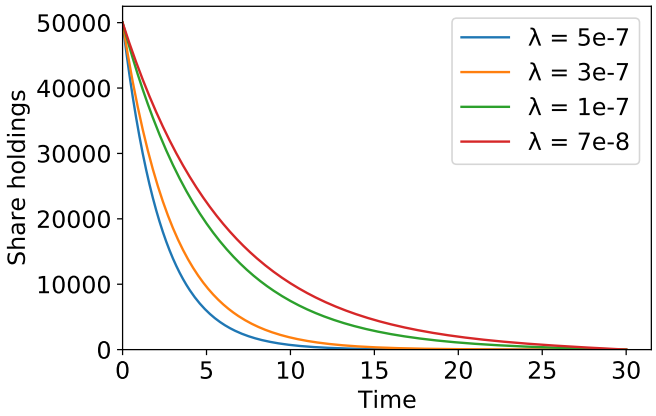

Figure B.2: *Almgren-Chriss execution trajectories, y-axis represents* x_k. $X = 50000$, $T = 30$ *minutes.*

B.3 Additional Material for Data Simulation

As an extension to Section 4.3, it is also possible to generate financial data using econometric models.

B.3.1 Econometric Models

To use econometric models, the time series data must be differenced until it becomes stationary before fitting the model to the data. An econometric model, specifically the ARMA model, was tested as a generative model for minute-by-minute FX data in the MCTS trading approach of Section 6.3, but the result was that with this method it is not possible to find and thus reproduce any relevant patterns.

ARMA ARMA stands for autoregressive integrated moving average, and is defined as:

$$S_t = c + \epsilon_t + \sum_{i=1}^{p} \psi_i S_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i},$$

where $S_t = c + \epsilon_t + \sum_{i=1}^{p} \psi_i S_{t-i}$ is the autoregressive model or order p: AR(p). Where $\psi_1, ..., \psi_p$ are parameters, c is a constant and ϵ is white noise. Instead $S_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$ is the moving average model of order q: MA(q). Where $\theta_1, ..., \theta_q$ are the parameters of the model and μ is the expectation of S_t.

B.4 Regression through Random Forests and Neural Networks

It is common to use random forests or Neural Networks (NNs) in RL and also online planning algorithms. For example, in TRPO or TRVO (see Section 8.2), the parametrized policy is usually a NN. In FQI (see Section 6.2.1), the approximate Q-function can be

represented by a random forest or a NN. In Alphazero (Silver et al., 2016), a NN represents both the policy and value function. In this section we briefly present these two tools for the supervised learning realm. A random forest is a combination of tree predictors, hence it is firstly necessary to introduce decision trees (Loh, 2011; Zaki and Meira, 2014).

B.4.1 Decision Trees

A decision tree is a supervised learning method based on a tree model that, in regression problems, predicts the value of the target y_i, given a datum $x_i \in \mathbb{R}^d$. Denoting with X the data space, a decision tree iteratively produces axis-parallel hyperplanes to recursively split the data space partitioning it, until the points inside each set of the partition are relatively homogeneous in terms of target y_i. Once the tree is trained, X is partitioned collecting train data into subgroups. It is then possible to assign each test point to one subgroup (a leaf of the tree), and its target can be predicted as mean value of the targets of the train data belonging to the same set. Summing up, a decision tree consists of internal nodes that split the data depending on the value of a feature selected for that node, and leaf nodes that represent a set of the partition of X and they are labeled with the predicted value of the target of data in that set, computed in regression as the mean of train targets.

At each internal node, one attribute is selected to split training samples in two subgroups, maximizing a measure of similarity of the targets among data in the same subgroup and minimizing the similarity between them in different subgroups. Therefore, it is necessary to select a splitting method capable of choosing the attribute on which the splitting is based on and to determine the value of the selected attribute: the threshold for the split. In classification there are different methods to perform the splits, like *information gain* or *information gain ratio*, which are based on entropy, or the *Gini index*, that always produces binary splits. In regression the mean squared error or the standard deviation are the common indices of impurity used to determine the split, so the best feature and its best threshold are selected as the ones that minimize the chosen impurity measure. Independently from the split procedure adopted, they are all focused on maximizing the purity among data so that in the same set their target values are as similar as possible. Clearly it is always possible to produce a regression decision tree that predicts exactly all the training target values by splitting until each leaf is made by a single datum, but this is an evident example of overfitting, that will perform very poorly in testing. To overcome this issue, the tree must be *pruned*:

- it is possible to adopt a *prepruning* approach so that the partitioning is stopped if a certain depth of the tree is reached, if data in a group have a standard deviation smaller than a chosen tolerance, or if the number of data in a subgroup is smaller than a certain minimum number;
- another possibility is a *postpruning* approach, that once the tree is built starts from last splits and removes them if there is no statistical evidence that they increase the performance on the evaluation of the target.

In conclusion, a decision tree iteratively splits data into subgroups maximizing the similarity among their target inside each subgroup, paying attention to splits, because too many subgroups lead to overfitting. At the end, leaf nodes of the tree predict the target value associated to each set of the produced partition of the data space, which is computed as the

mean of the target values of the training data in the set, assigning that value to each test datum belonging to that group.

B.4.2 Random Forests and Extra Trees

A random forest regression algorithm is a model ensemble method consisting in a large number of unpruned decision trees with a random selection of features at each split.[48] The idea is to use many weak learners as uncorrelated as possible, that together form a strong learner.

The correlation between decision trees (hence the variance of the model) is reduced in two ways:

- each decision tree in the forest has training set composed by the same number of data N of the original training set extracted with bootstrap technique, which consists in sampling with replacement N data from the original set;
- at each split of a node, the feature on which the split is based on can be selected only between m variables, randomly selected from the d available features.

Summing up, all decision trees of the random forest are trained in this way: the training set is extracted with bootstrap, at each split only some features can be selected and the trees are unpruned. Each one of them is a weak regressor, since its training is not optimized and it will probably overfit, but they all together become a strong learner, since their learning procedure, which singularly is not optimal, reduces the correlation among trees, decreasing the variance. Moreover, since there is no pruning and at each split a smaller number of features is available, decision trees in the random forest are much faster to train. Finally, the prediction of the target of a test datum is computed in the random forest regression as the average value of the predictions of each decision tree.

A possible variant is called extremely randomized trees or extra trees (Geurts et al., 2006), where randomness is even more exploited. In particular, a third random procedure is performed to decrease the variance: as in random forests, a random subset of m features is available at each node to perform the split, but instead of looking for the best threshold, some values are randomly chosen for each available feature and the best of these randomly generated thresholds is used to perform the split.

It is possible to control the complexity of this approximator, by setting a number of parameters that regulate the forest dimensions. We cite here only the main ones we used in our experiments:

- *the number of trees*: setting a higher number of trees it is possible to average out noise from relevant information, hence, the forest would be less dependent from its internal randomization;
- *the minimum sample split*: also referred to as *min-split* this parameter regulates the minimum number of samples that are necessary to allow a node to split. Increasing this parameter, we help the forest selecting features that are common to a large number of samples, hence we improve its generalization capability. On the other

[48] Classically in random forests, weak learners are unpruned, nevertheless, in pratice, it is common to prune the weak learners.

hand, decreasing this parameter allow to *overfit* more, allowing trees to specialize on patterns that are common among a lower number of samples.

Random forests also present two peculiar characteristics:

- the first is that it is possible to compute a validation error directly through the training set. Exploiting the fact that not all the data appears in all the datasets, the out of bag approach estimates the value of the target averaging only the output of trees corresponding to bootstrap samples where does not appear the selected datum. Repeating this for all the data in the training set allows to measure the performance of the learner in a more robust way than training error without the use of a cross validation set;
- another important property of random forests, that makes them widely applied to perform feature selection, is that they produce an estimate of the importance of the features in predicting the target. Indeed decision tree methods calculate their splits by mathematically determining which split will most effectively distinguish groups of data with similar target. Therefore in random forests the importance of a feature is measured as the sum of the improvements (in terms of *Gini index*, *information gain* or *mean squared error*) in any node in which the feature is used to split, weighted by the percentage of training data reaching that node.

In conclusion, to train a random forest it is necessary to set the number of trees, to decide the percentage of features available at each split and the minimum number of data required in a node to be splittable. These parameters can be tuned using a cross validation approach.

B.4.3 Neural Networks

In policy based algorithms, such as TRVO (Section 8.2), the policy is parametrized by feed-forward NN. The simplest definition of a NN is provided by the inventor of one of the first neurocomputers, Dr. Robert Hecht-Nielsen, who defines it (Hecht-Nielsen, 1989) as "[...] a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs".

They constitute a class of flexible nonlinear models designed to mimic biological neural systems, elaborating signals through several layers, each with a large number of neural units (neurons) that can process the information in a parallel manner. So a NN has a multilayer structure such that every layer is built upon many simple nonlinear functions, playing the role of neurons in a biological system. By allowing the complexity of the structure to increase indefinitely, multilayered NNs are able to approximate any continuous function with any desired degree of accuracy. Thanks to their representation power, they are said to be *universal approximators*, and became very popular in the fields related to machine learning.

The number of applications of NNs grew larger and larger in the last decade along with the evolution of GPUs and distributed computed systems, capable of supporting the computational power required to perform tasks in a short time. A feed-forward NN is a generalization in multiple layers of one of the simplest models user for regression, the *perceptron*. For this reason, it is called also MultiLayer Perceptron (MLP).[49]

[49] Perceptron generally refers to using heavyside activation function, nevertheless it is common to use also other activation functions (*e.g.*, ReLu, tanh, and sigmoid in feed-forward NNs).

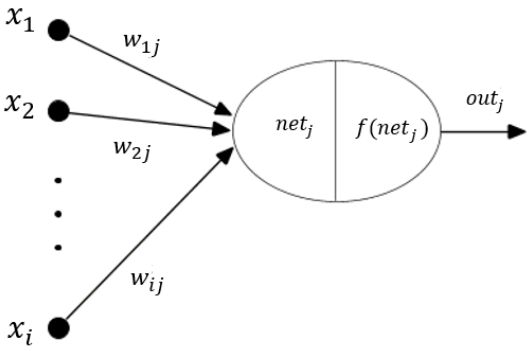

Figure B.3: *Composition scheme of a neuron.*

Building blocks: Neurons

Every layer is made of several *neurons*, the building blocks of the overall structure. Each neuron receives as input a linear combination of the data elaborated in the previous layer, and then transforms it to generate a neural signal to be forwarded to the neurons in the next layer. For example, consider the j^{th} neuron in one of the layers; suppose that it is connected to N neurons of the previous layer with values called $\{x_i\}_{i=1}^{N}$. Then, the j^{th} neuron will have as input (also known as *net value*):

$$net_j = \sum_{i=1}^{N} w_{ij} x_i + w_{0j},$$

in particular, the coefficient w_{ij} corresponds to the weight of the connection between the input i and the neuron j. The last weight w_{0j} is called *bias*, and it behaves like a connection with a fictitious input always equal to 1. The value that this neuron returns as output, also called *activation value*, is simply

$$out_j = \Psi(net_j),$$

where $\Psi(\cdot)$ is called *activation function*.

The architecture

A NN is built by hooking together many simple neurons in several layers, so that the output of a neuron can be the input of another. For example Figure B.4 illustrates a simple representation of a NN.

In Figure B.4 it is possible to see the different roles of the layers. The leftmost one is called the *input layer*, and it is the one that directly considers the input data (eventually elaborated through a preprocessing procedure). The rightmost is the *output layer*; which is the final step where the results can be observed. The middle layers of the network are called *hidden layers*, because their values are not directly accessible. In general, their number

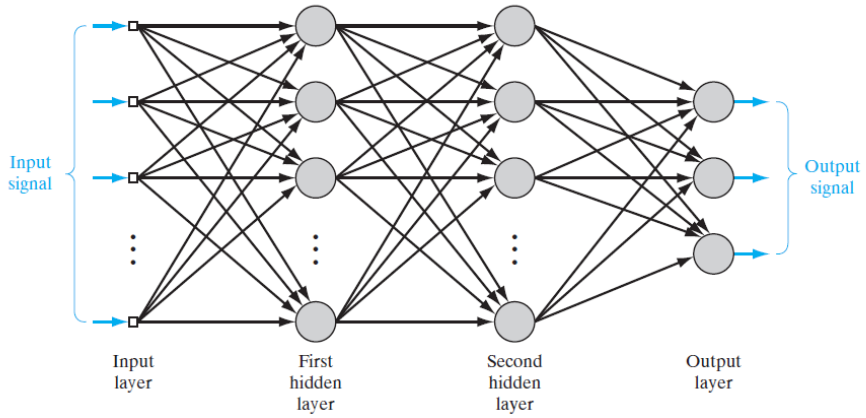

Figure B.4: *Scheme of a fully connected NN*

may vary, but in most cases only one or two hidden layers are used. It is also possible to have no hidden layers, and in this case the NN degenerates into a simple perceptron.

NNs usually exhibit a high degree of connectivity, determined by the presence of weights in the network. In this example, there is a connection between any couple of neurons in consecutive layers, hence this NN is said to be *fully connected*. Moreover, in feed-forward NNs the sample information is passed only forward from the input layer to the output.

NNs are used to parametrize a policy in the following manner: the input layer is the state state vector s_t, whereas the output layer dimension coincides with the dimension of the action space $\mathcal{A}$. To be more precise, to ensure the exploration of the stochastic policy, it is modelled as a multivariate Gaussian distribution, where the mean is the output vector of the NN. Thus, in mathematical terms: $\pi(s_t) \sim \mathcal{N}(\mu(s_t), \sigma^2)$. Consequently, $a_t \in \mathcal{A}$ is the action at time t and it is a realization of the multivariate Gaussian $\pi(s_t)$; and its mean $\mu(s_t)$ is the output vector of the NN.

APPENDIX C

DVA Hedging with RL

C.1 CVA and DVA

The Credit Valuation Adjustments (CVA) and Debt Valuation Adjustments (DVA) are not financial instruments that can be individually traded, but are risks that come paired with non collateralized OTC derivatives (see Section 2.1.1). OTC derivatives mostly represent contingent claims that entail bilateral payment obligations. Credit risk associated with these payment obligations is normally negligible, as it is substantially mitigated by the frequent exchange of collateral margin security between the two parties; this is especially true for OTC derivatives traded between regulated financial institutions, where such credit support practices are normally mandatory (see Appendix B.1.4). For this reason, OTC derivatives in the interdealer market are usually priced as if they were counterparty "risk-free". However, when dealing with non-financial clients, banks may trade without collateral margin security. Dealing on uncollateralized lines is required for instance by most corporate treasurers, due to the costs and sophistication required to set up and run collateral exchange operations. These adjustments are priced and charged by investment banks to the corporate clients. The lack of collateral exposes banks to potential credit risk associated with the impossibility of fully recovering unrealised profits in case of a default of the corporate client.

Such risk is often hybrid in nature, as it depends on both the underlying risk drivers for the contingent claim and on counterparty's default probability. From the 1990s, dealers have started adjusting the price of uncollateralized OTC derivatives to take into account counterparty risk, by adding a (negative) term called Credit Value Adjustment (CVA) to the *risk-free* value of the derivative.

Hence, CVA gradually became a generally accepted practice in the manufacturing reality of derivative products, to the extent that it was subsequently recognized as a component of the fair value of derivatives both in international accounting standards and by prudential capital regulators.[50]

Formalizing CVA We introduce the notion of *Credit Exposure* (CE), which represents the close-out amount from the point of view of the bank. As anticipated, the effect of the client's default is different depending on the sign of the MtM (see Section 2.2) at the time of default:

- if the $MtM > 0$, then the (defaulted) client owes the close-out amount to the bank. The bank may expect to recover this amount only partially via insolvency proceedings upon liquidation of the client's assets.
- if the $MTM < 0$, then the bank owes the balance to the client. The bank has not defaulted, so it can fully honour this obligation.

Mathematically, CE can be expressed by:

$$CE(\tau_C) = \max[MtM_{\tau_C}, 0], \tag{C.1}$$

where, at time t, an OTC derivative with maturity T is considered and $\tau_C \in (t, T)$ is the default time of the counterparty C. A related quantity is the *Expected Exposure* (EE) at time $t^* \in (t, T)$:

$$EE(t^*) = \mathbb{E}_t[MtM_{t^*} | MtM_{t^*} > 0].$$

As discussed, experienced loss upon default is only part of CE, as the bank may expect to recover part of the close-out claim amount in the insolvency proceedings. The recovered fraction of CE is estimated by the recovery rate $R_C \in [0, 1]$ of the client. Correspondingly, the lost fraction is the Loss Given Default $LGD_C = 1 - R_C$. The formula for the CVA of an OTC derivative at time t under the risk-neutral measure Q:

$$\mathsf{CVA}(t) = -\mathbb{E}_t^Q \left[LGD_C \, \mathbb{1}_{\{\tau_C \leq T\}} \, \mathbb{1}_{\{\tau_C < \tau_I\}} \, D(t, \tau_C) \, CE(\tau_C) \right], \tag{C.2}$$

where:

- the loss given default LGD_C is usually set to a deterministic quantity;
- τ_C is a random variable that represents the time of arrival of the counterparty's default (its "credit event"), typically modelled as the first jump of a Poisson process;
- τ_I is the bank's default time (to account for the bilateral nature of the risk);
- T is the maturity date of the contract;
- $D(t, \tau_C)$ is the risk-free stochastic discount factor evaluated at counterparty's time to default;

[50] Fair value is defined by International Financial Reporting Standard (IFRS) 13 as "the price that would be received to sell an asset or paid to transfer a liability in an orderly transaction between market participants at the measurement date (i.e., an exit price)".

- $CE(\tau_C)$ is the bank's Credit Exposure (C.1) at counterparty default time.

If we assume that the Bank cannot default, then it follows that $\tau_I = \infty$, so the corresponding indicator function vanishes. If we make the further assumption that credit exposure and default probability are independent, then Equation (C.2) becomes:

$$\begin{aligned}\mathsf{CVA}(t) =& \mathbb{E}_t^Q\left[LGD_C\, \mathbb{1}_{\{\tau_C \leq T\}}\, D(t,\tau_C)\, CE(\tau_C)\right],\\ =& LGD_C \int_t^T B(t,s)\, EE(s)\, dPD(s),\end{aligned}$$

where:

- $PD(s)$ is the counterparty's probability of default at time s;
- $B(t,s)$ is the expected value of the stochastic discount, *i.e.,* $\mathbb{E}_t[D(t,s)]$.

Adding time discretization and assuming that the default probability of the counterparty is known in a finite interval $[t_{i-1}, t_i)$, with $t_0 = t$ and $t_m = T$, the CVA of an OTC derivative in this simplified context becomes:

$$\mathsf{CVA}(t) \approx LGD_C \sum_{i=1}^{m} B(t,t_i)\, EE(t_i)\, PD(t_i).$$

This equation is easier to compute and highlights the two main components of CVA: EE and default probability of the counterparty.

Formalizing DVA Since Counterparty Credit Risk (and, consequently, CVA) is bilateral, another concept arises: Debt Value Adjustment (DVA). Similarly to what has been done for CVA, DVA can be defined as the difference between the risk-free instrument value and the true instrument value that takes into account the possibility of default of the investor. From a different point of view, it is the CVA from the perspective of the counterparty looking at the institution. Furthermore, remembering that a positive MtM value has a negative impact on the CVA, in a symmetrical way, a negative MtM value affects positively the DVA: in fact, it could be considered as the gain inducted by the counterparty's loss due to the institution's default.

Similarly to CE for CVA, it is necessary to introduce the notion of *Negative Credit Exposure* (NCE). Using a similar notation, NCE is defined as:

$$NCE(\tau_I) = \min[MtM_{\tau_I}, 0],$$

while its expected value at time $t^* \in (t,T)$ is the *Expected Negative Exposure* (ENE):

$$ENE(t^*) = \mathbb{E}_t[MtM_{t^*} | MtM_{t^*} < 0].$$

Even in this case, the actual impact of NCE depends on institution's recovery rate R_I.

Under the risk-neutral measure Q, at time t the DVA can be computed using:

$$\mathsf{DVA}(t) = \mathbb{E}_t^Q\left[LGD_I\, \mathbb{1}_{\{\tau_I \leq T\}}\, \mathbb{1}_{\{\tau_I < \tau_C\}}\, D(t,\tau_I)\, NCE(\tau_I)\right]. \tag{C.3}$$

If we assume, similarly to the CVA case, that the counterparty cannot default (i.e., $\tau_C = \infty$) and following the same steps as before, a simplified formula for DVA is obtained:

$$\begin{aligned}\mathsf{DVA}(t) =& LGD_I \int_t^T B(t,s)\, ENE(s)\, dPD(s),\\ \approx& LGD_I \sum_{i=1}^{m} B(t,t_i) ENE(t_i) PD(t_i).\end{aligned} \tag{C.4}$$

To simplify this expression further, we decided to consider the DVA generated by a liability of a single cash flow N that the institution has to pay at maturity time T. With this financial instrument, the value of the cash flow at default time τ_I becomes:

$$NCE(\tau_I) = N\, D(\tau_I, T),$$

as a consequence, the DVA formula in Equation (C.4) becomes:

$$\mathsf{DVA}(t) = N\, LGD_I B(t,T)[1 - S(t,T)], \tag{C.5}$$

where $S(t,T)$ is the survival probability between t and T.

Survival probability is built upon an intensity-based model and calculated starting from the CDS curve of the bank, it is defined as:

$$S(t,T) = e^{-\int_t^T \lambda_s ds},$$

where λ_s is called *intensity rate* at time s. For the detailed methodology, we invite the reader to follow (Jarrow and Turnbull, 1995).

C.1.1 Banks and the Corporate Derivatives Business

In this section, we describe one of the characteristics of the *corporate derivatives business*. The function of this business is to sell OTC derivatives to other corporate companies. This business requires the interaction of several functions of the bank starting with the derivatives sales team that is in contact with the clients and tries to understand their needs and give advice on which financial assets are more fitting. Once a deal needs to be priced, the sales team calls the bank's market makers that calculate the fair market price of the derivative. When these derivatives are uncollateralized, they have further risks and costs, such as the CVA, DVA (see Section C.1), Funding Valuation Adjustment (FVA) *i.e.*, how much the bank needs to pay to fund such a position, and the cost of capital or KVA *i.e.*, how many capital reserves the bank needs to have for this specific deal.

We focus on one of the tasks of the XVA desk, hedging the DVA. DVA is very difficult to hedge: in fact, recalling Equation (C.3) we can see that it depends on multiple, possibly correlated, risk factors. If we simplify and neglect the possibility of counterparty's default, while part of the DVA's risk can be hedged through the risk drivers of the underlying OTC derivative, the default probability of the institution requires, in principle, to sell protection through its own Credit Default Swaps (CDS). Obviously, this is not possible, since no counterparty would accept to hedge the risk related to the institution's default with a CDS on the institution itself. For this reason, a perfect hedge of the DVA is impossible: anyway, good results can be achieved by using instruments that are highly correlated with the institution's CDS. In our case, the institution we are dealing with is Intesa Sanpaolo. Hedging instruments that can be considered include (see Section 4.2 for further details):

- CDS on correlated entities, *e.g.,* the SNRFIN index;
- government bonds issued by institution's country of origin, *e.g.,* BTP futures and specifically the BTP-Bund spread;
- futures and options on financial entities, *e.g.,* the SX7E Index,
- bonds (or stocks) on the institution's own name. However, this last operation cannot be easily done without constraints, thus we do not consider this last possibility in our analyses.

The profit of a trader hedging the DVA is defined as the sum of the DVA variation and the P&L of the hedging strategy:

$$r_{t+1} = \underbrace{\mathsf{DVA}(t+1) - \mathsf{DVA}(t)}_{\text{DVA variation}} - \underbrace{\sum_{k=0}^{5} a_t^k \times (S_{t+1}^k - S_t^k)}_{\text{hedge}} - \underbrace{c(\boldsymbol{a}_t - \boldsymbol{a}_{t-1})}_{\text{transac. costs}}, \tag{C.6}$$

where a_i^k represents the current portfolio position or notional of each instrument, and S^k their price. $c(\boldsymbol{a}_t - \boldsymbol{a}_{t-1})$ represent the transaction costs corresponding to the sum of the transaction costs of the various hedging instruments.

C.2 DVA hedging with RL

DVA hedging is one of the most complex hedging problems as DVA is a hybrid risk that depends on the risk drivers of the underlying derivative and the institution's default probability and because it is not possible to use the institution's CDS to hedge this risk. The basics of DVA are explained in Section C.1 and the requirements of banks that price and hedge DVA are described in Section C.1.1. Finally, Section 4.2 illustrates the data and how it was collected and prepared. In this chapter, we describe a preliminary approach to DVA hedging, where the institution pricing the DVA is Intesa Sanpaolo.

Notation

In Table C.1, we present some common notation used throughout the chapter. In general, a_i^k represents the notional *i.e.,* the actions chosen by the agent at time t_i, finally the cash account a_i^0 and collateral account a_i^1 are calculated to obtain a realistic gain process G_i.

Conventions

There are some conventions used throughout Appendix D.

- The sign $-$ usually indicates a cash outflow, instead $+$ a cash inflow.
- $(.)_i$ means that the simulator consider the value of the variable at step i, before it is modified by the action taken in i^+. Instead, $(.)_{i^+}$, represent the value of the variable after the action A_{i^+} has been taken into account, *e.g.,* $a_{i^+}^2$ represents the SNRFIN notional at time t_i^+.

π_i^{xy}	Credit spread at time t_i of Intesa Sanpaolo CDS with x-years maturity.
k	Instrument index ($k=0$: cash account, $k=1$: collateral account, $k=2$: SNRFIN, $k=3$: BTP, $k=4$: Bund, $k=5$: SX7E).
Q_i^k	Market quotation of instrument k, at time t_i.
X_i^k	Unit price at time t_i referring to the market quotation Q_i^k: $X_i^{\{0,1\}}=1$; $X_i^2=X_i^2(Q_i^2)$ (i.e., the upfront amount); $X_i^{\{3,4,5\}}=Q_i^{\{3,4,5\}}$.
Y_i^k	Unit dividend at time t_i of instrument k.
L_i^k	Number of lots at time t_i of instrument k.
M^k	Lot size: $\mathsf{M}^{\{0,1,2\}}=1$, $\mathsf{M}^{\{3,4\}}=100{,}000$, $\mathsf{M}^5=50$.
a_i^k	Notional at time t_i; $a_i^k=L_i^k\times\mathsf{M}^k$ and $a_i^{\mathsf{DVA}}=$€400mln $\forall i$.
$y(t_i,t_{i-1})$	Year fraction with the convention Act/360.
$\Delta(.)_i$	$(.)_i-(.)_{i-1}$.
G_i	Gain process at time t_i.
R_i	Reward at time t_i.
s_i^{BTP}	BTP-Bund (yield) spread at time t_i.
D_i^k	Unit sensitivity at time t_i.

Table C.1: *Notation for the DVA problem.*

- For $k\in\{3,4,5\}$, $a_i^k>0$ indicates that the agent has a long position with respect to the instrument k (he bought a certain quantity of instrument k). Instead, for $k=2$, $a_i^k>0$ indicates that the agent has a long position with respect to the risk (he sold protection).
- Positive collateral $a_i^1>0$ means that the collateral is within Intesa Sanpaolo collateral account; vice versa $a_i^1<0$ if the collateral is within the counterparty account.
- The agent buys at ask price and sells at bid price, so it is not necessary to explicitly consider transaction costs. Ask prices are referred as X^{ask}, bid prices as X^{bid}.

C.2.1 Price and Dividend Processes

In this section we define the price and dividend processes X_i^k and Y_i^k for each of the different instruments. The data collection process and some details on these instruments can be found in Section 4.2.

SNRFIN We assign index $k=2$ to SNRFIN. As we saw in the Section 2.2.4, specifically Equation 2.3 the upfront is obtained from the credit spread, which here we call Q_t^k through an evaluation function $X_t^k(Q_t^k)$. The price process increments are thus given by

$$\Delta X_i^2=X_i^2-X_{i-1}^2,$$

while the dividend process (the quarterly coupon) is given by

$$\Delta Y_i^2 = Y_i^2 - Y_{i-1}^2 = C_i.$$

To determine the SNRFIN value the simulator uses the mid price $X_i^2 = (X_i^{2,bid} + X_i^{2,ask})/2$. When buying SNRFIN $A_i^2 < 0$ (to indicate a short risk position) we use the bid price, while when selling $A_i^2 > 0$ we use the ask price.

BTP and Bund Futures For BTP and Bund Futures we assign $k = 3$ and $k = 4$ respectively (see Section 2.2.2 and Appendix B.1.4 for more details on futures). In this case price processes exactly match market quotes, meaning that $X_i^k = Q_i^k$, while there is no dividend process (i.e., $Y_i^k = 0 \; \forall i$).

To implement the BTP-Bund spread trade previously described, the amount of Bund Futures to be bought or sold is determined implicitly from the amount of BTP Futures chosen by the trader following a delta neutral strategy:

$$L^4 = -\left\lfloor L^3 \times \frac{D^3}{D^4} + \frac{1}{2} \right\rfloor,$$

where L^k represents the number of lots (i.e., the minimum negotiable size) related to the k-th instrument and D^k its sensitivity (computed with respect to its yield, see Appendix B.1.8 for further details).

SX7E We assign $k = 5$ to this instrument and, as for the previous case, price process is $X_i^5 = Q_i^5$ and the dividend process is $Y_i^5 = 0 \; \forall i$.

DVA We refer to X_i^{DVA} as the unitary DVA amount, thus $\Delta\mathsf{DVA}$ at time t_i is computed as:

$$\Delta\mathsf{DVA}_i = a_i^{\mathsf{DVA}} \times \Delta X_i^{\mathsf{DVA}},$$

where DVA is calculated as defined in Equation (C.5). The dividend Y_i^{DVA} represents the payment to the counterparty made at the beginning of the derivative contract.

Rolling Instruments For futures prices BTP, Bund, SX7E, it is necessary to consider the roll. If t_i is a roll date for one future instrument, the agent observes two prices: one referring to the old series (indicated by X_i^{ko}) and a one referring to the new instrument with a longer maturity (indicated by X_i^{ka}). In the day t_i, there isn't any change in the simulator mechanism and $\Delta X_i^k = X_i^{ko} - X_{i-1}^{ko}$. Instead the following day t_{i+1} in Equations (C.7) and (C.8), X_i is the new series price: $\Delta X_{i+1}^k = X_{i+1}^{ka} - X_i^{ka}$.

C.2.2 Collateral and Cash Accounts

With collateralized derivative contracts, if the contract increases in value for the bank (or vice versa) then the counterparty is required to immediately recognize this increase by exchanging a collateral of equal ammount. This collateral in the case of OTC derivatives like the SNRFIN is inserted in a collateral account that must be remunerated, while for listed futures contracts, the collateral is not remunerated and thus ends up in the cash account.

Both accounts are calculated at every step (so every 5 minutes), the dividends are always zero except for the final step at the daily closure.

Collateral Account In our case the only instrument that influences the collateral account is the SNRFIN. Thus the collateral balance at step i is:

$$a_i^1 = a_i^2 \times X_i^2,$$

where: $X_{i-1}^2 = (X_{i-1}^{2,bid} + X_{i-1}^{2,ask})/2$.

The collateral is remunerated at a rate of r_t^1 (which can be approximated with a constant risk-free rate r_f) through the dividend process described by the following ordinary differential equation:

$$dY_t^1 = -r_t^1 dt.$$

The dividend process is negative if the collateral is within Intesa Sanpaolo collateral account (and interest rate $r^1(t_i)$ is positive) and vice versa. The above equation can be simply discretized as:

$$\Delta Y_i^1 = -r_{i-1}^1 \times y(t_i, t_{i-1}).$$

Notice that if X_i^2 is positive and the portfolio has a long risk position ($a_i^2 > 0$, see Table C.1) the agent will pay to the counterpart $-a_i^2 \times X_i^2$ (cash outflow). To balance this outflow the agent will receive as collateral the same amount: $a_i^2 \times X_i^2$ (cash inflow).

Cash Account The cash account used to manage the collateral of the futures contracts, but also the gains or losses of the hedging activity. Similarly to the collateral the dividend process can be described by the following ordinary differential equation:

$$dY_t^0 = r_t^0 dt.$$

But differently from before, the dividend process will be positive if the cash account is positive, negative in the opposite situation. It can be discretized as follows:

$$\Delta Y_i^0 = r_{i-1}^0 \times y(t_i, t_{i-1}).$$

Since DVA exists, we cannot assume that the dividend process of the cash account is remunerated by the risk-free rate. Instead, it must grow at a rate r_t^0 consistent with the Intesa Sanpaolo unsecured financing rate, so that:

$$r_t^0 = r_f + \pi_t^{1y},$$

where π_t^{1y} is the Intesa Sanpaolo 1Y CDS credit spread.

Moreover, given that collateral is rehypothecable, only the net cash position between a_t^0 and a_t^1 is actually remunerated at r_t^0 through the cash account dividend process, we can insert the collateral change $a_i^1 - a_{(i-1)^+}^1 = a_i^2 \times \Delta X_i^2$ directly in the formula for the delta cash, obtaining:

$$\Delta a_i^0 = \sum_{k=2}^{5} a_i^k \times \Delta X_i^k + \sum_{k=0}^{2} a_{(i-1)^+}^k \times \Delta Y_i^k. \tag{C.7}$$

Where:

- $a_{(i-1)^+}^k = a_i^k, \quad k \in \{2, 3, 4, 5\}$,
- $a_{(i-1)^+}^1 = a_{(i-1)^+}^2 \times X_{i-1}^2$.

C.2.3 Gain and P&L

A *gain process* G is linked to the hedging strategy:

$$G(t,T,a) = \sum_{k=0}^{K+1} \left(\int_t^T a_u^k \, dX_u^k + \int_t^T a_u^k \, dY_u^k \right),$$

assuming $X_t^0 = X_t^1 = 1$ for every t.

that can be discretized as:

$$G_i(a) = \sum_{k=2}^{5} a_i^k \times \Delta X_i^k + \sum_{k=0}^{2} a_{(i-1)^+}^k \times \Delta Y_i^k, \tag{C.8}$$

which is exactly the formulation for the change in cash account. The step P&L is defined as the sum of the hedging strategy gain and the DVA variation:

$$\rho_i = \underbrace{G_i}_{\text{hedge}} + \underbrace{(\Delta \mathsf{DVA}_i + a_i^{\mathsf{DVA}} \times \Delta Y_i^{\mathsf{DVA}})}_{\text{DVA variation}}. \tag{C.9}$$

This formulation is equivalent to Equation (C.6), with the difference that both the hedge gain process (Equation (C.8)) and the DVA are split between the market movement and the dividend process. Another noticeable difference is that the transaction costs are not explicit in the reward formulation as we are considering buying at the ask price and selling at the bid price.

C.2.4 DVA Hedging as an MDP

We can now define the MDP.

State At time t_i, the agent observes the following information to decide next allocation:

- SNRFIN (price: X_i^2; sensitivity: D_i^2; total allocation: L_i^2 and delta allocation: ΔL_i^2);
- BTP-BUND yield spread: s_i^{BTP};
- BTP (sensitivity: D_i^3; total allocation: L_i^3 and delta allocation: ΔL_i^3);
- BUND (sensitivity: D_i^4; total allocation: L_i^4 and delta allocation: ΔL_i^4);
- SX7E (price: Q_i^5; sensitivity: D_i^5; total allocation: L_i^5 and delta allocation: ΔL_i^5);
- Time to roll for different instruments;
- VIX Index and V2X Index.

Action After receiving a state from the environment, the agent selects an action $\mathbf{a}$, which is a three-dimensional vector: the first component refers to the quantity of Futures on SX7E, the second one to the Futures on BTP (and, as a consequence, to the Futures on Bund), the third one to the quantity of SNRFIN. These quantities are subsequently added (or subtracted) to the total allocations stored inside the environment.

Reward The reward is Equation (C.9).

C.2.5 Experimental Approach

With this formulation, it is possible to learn the MDP using RL. Given this is a hedging problem, like those in Chapter 8, we have run preliminary tests using TRVO (see Section 8.2), but the results are not yet satisfactory. This is not surprising as the DVA hedging problem is extremely challenging. It is a hedging problem as those of Chapter 8, but with the difference that it is not possible to use the underlying that generates the counterparty risk, it is thus necessary to use a number of correlated instruments. The dataset used is described is Section 4.2, and consists of intra-day snapshots of 5 minute frequency. In what follows we describe the preliminary steps taken.

- The first step is to learn a near-optimal risk-neutral policy in the in-sample context, in other words: trying to overfit the financial data, by using all the available data for training. This step is meant to be a way to select the more suitable tools to approach the problem (*e.g.,* neural network architecture) and to highlights the possible issues in a more controllable context. The selected algorithm for this task is TRPO.
- Learning a near-optimal *risk-averse* policy in the in-sample context. In this way it should be possible to understand the effects of controlling *reward-volatility* on the obtained policies. The selected algorithm for this task is TRVO.
- Learning a near-optimal risk-averse policy that can generalize on unseen data.

Unfortunately, learning in this scenario resulted to be challenging. We leave as future work the solution of the DVA hedging problem with RL.