Contributions to Agent-Based Modeling and Its Application in Financial Market

Trung-Minh Tran

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à École Pratique des Hautes Études

Contributions à la modélisation à base d'agents et à son application au marché financier

Soutenue par
Trung-Minh TRAN
Le 16 Feb 2023

École doctorale n°472
École doctorale de l'École Pratique des Hautes Études

Spécialité
Informatique, Mathématiques et Applications

Composition du jury :

M. Vu DUONG Professeur des universités, Nanyang Technological University	*Président*
M. Ider TSEVEENDORJ Maître de conférences, Université de Versailles-Saint-Quentin-en-Yvelines	*Rapporteur*
M. Jae Yun JUN KIM Maître de conférences, ECE Paris	*Rapporteur*
M. Soufian BEN AMOR Maître de conférences, Université de Versailles-Saint-Quentin-en-Yvelines	*Examinateur*
M. Marc BUI Professeur des universités, EPHE Paris	*Directeur de thèse*
M. Duc PHAM-HI Professeur des universités, ECE Paris	*CoDirection de thèse*

École Pratique des Hautes Études | PSL

Contributions to Agent-Based Modeling and Its Application in Financial Market

Trung-Minh TRAN

Supervisor: Prof. Marc BUI
Prof. Duc PHAM-HI

École Pratique des Hautes Études
Paris Sciences & Lettres – PSL Research University Paris

This dissertation is submitted for the degree of
Doctor of Philosophy
of
Paris Sciences & Lettres – PSL Research University Paris

February 2023

Acknowledgements

First of all, I would like to express my deepest gratitude to my advisors, Prof. Marc Bui and Prof. Duc Pham-Hi. Your suggestions have provided new streams of thought that have enriched my research a lot, and my dissertation is something I can be proud of having written. May you continue to inspire and enrich your students. I will always remember and be proud that I am one of them.

I would like to extend my sincere thanks to the members of jury who accepted to attend my defense. In particular, I would like to thank my two reviewers who devoted their time to read and to give constructive feedback for my manuscript.

I am also extremely grateful to JVN and EPHE. The institute and the university have provided me with everything I need to complete this dissertation. Not only do they provided funds, courses and seminars but it was really like a second home to me.

I would be remiss not to mention Mrs. Laure CARREAU from EPHE, who has assisted me in administrative procedures for the past 5 years.

Lastly, this PhD dissertation is dedicated to my parents who raised me with unconditional love. Many thanks to my family for always supporting me and being with me whenever I feel happy, sad, pleasant and frustrated. To Julia, thank you for being so nice to me, for your words of encouragement, for all the things you have done and for always being with me. I feel fortunate to have met you on my life's journey.

Abstract

The analysis of complex models such as financial markets helps managers make reasonable policies and traders choose effective trading strategies. Agent-based modeling is a computational methodology to model complex systems and analyze the influence of different assumptions on the behaviors of agents. In the scope of this thesis, we consider a financial market model that includes three types of agents: technical agents, fundamental agents, and noise agents. We start with the technical agent and the challenge of optimizing a trading strategy based on technical analysis through an automated trading system. Then, the proposed optimization methods are applied with suitable objective functions to optimize the parameters for the agent-based model. The study was conducted with a simple agent-based model including only noise agents, then the model was extended to include different types of agents. The first part of the thesis investigates the trading behavior of technical agents. Different approaches are introduced, such as Genetic Algorithm, Bayesian Optimization and Deep Reinforcement Learning. The trading strategies are built based on a leading indicator, the Relative Strength Index, and two lagging indicators, the Bollinger Band and Moving Average Convergence-Divergence. Multiple experiments are performed in different markets, including: cryptocurrency market, stock market, and crypto futures market. The results show that optimized strategies from the proposed approaches can generate higher returns than their typical form and the Buy and Hold strategy. Using the results from the optimization of trading strategies, we propose a new approach to optimize the parameters of the agent-based model. The second part of the thesis presents an application of agent-based modeling to the stock market. As a result, we have shown that agent-based models can be optimized using the Bayesian Optimization method with multiple objective functions. The stylized facts of the actual market can be reproduced by carefully constructing the objective functions of the agent. Our work includes the development of an environment, the behaviors of different agents, and their interactions. The Bayesian optimization method with the Kolmogorov–Smirnov test as an objective function has shown advantages and potential in estimating an optimal set of parameters for an artificial financial market model. The model we propose is capable of reproducing the stylized facts of the real market. Furthermore, a new stylized fact about the proportion of traders in the market is presented. With empirical data from the Dow Jones In-

dustrial Average index, we found that fundamental traders account for 9%–11% of all traders in the stock market. In the future, more research will be done to improve the model and optimization methods, such as applying machine learning models, multi-agent reinforcement learning, or considering the application in different markets and traded instruments.

Table of contents

List of figures

List of tables

Summary

The financial market is a complex system in which the relation between its components cannot be captured analytically. Thereby, computational approaches, such as simulation and modeling, are needed to comprehend this relation. In the last few years, agent-based modeling is a powerful simulation modeling technique for simulating the actions and interactions of autonomous agents in social, economic and financial systems. In this thesis, the agent-based model is studied to build an artificial stock market and novel contributions to the agent-based model in financial markets are also presented. Specifically, the aim of this thesis is to study different approaches to tune the parameters in trading strategy and different method to optimize the agent based model. In addition, two agent-based models are developed to study important stylized facts in real stock markets.

The first part of the thesis investigates the trading behavior of technical agents. Genetic Algorithm is a popular optimization method widely used in previous studies. To meet short-term investment goals and find a method of estimating parameters that can dynamically change depending on the conditions of the environment, we introduce new approaches such as Bayesian Optimization and Deep Reinforcement Learning. The complex objective functions in the trading strategies of technical agents in the real market are also considered. Our approach is to give a simpler definition of investment strategy optimization. Then, an artificial stock trading system with common trading strategies is built. The results in this part can be applied to build an artificial stock market model with multi-agents. When building an agent-based model for the stock market, we face the problem of parameter optimization for the model. This is a complex problem when different agents and their interactions are added to the model. The objective of this part is to find a parameter optimization method for agent-based models of the stock market with low computational cost and high accuracy. To solve the above problems, we introduce an approach to Bayesian Optimization with different objective functions to optimize the parameters of the model. This approach is proven to quickly and flexibly provide optimal parameter sets for the model. We do the experiments on two different markets, the US and Vietnam stock market, thereby analyzing the possibility of reproducing stylized facts in the real market.

The second part of the thesis presents an application of agent-based modeling to the stock market. As a result, we have shown that agent-based models and agents' trading strategies can be optimized using the Bayesian Optimization method. The stylized facts of the actual market can be reproduced by carefully constructing the objective functions of the agents. In detail, we have introduced three types of agents typical of the stock market: noise traders, technical traders and fundamental traders. After building the model, Bayesian Optimization is used to tune the parameters of the strategies of traders as well as of the stock market model. The experimental results on Bayesian Optimization with the Kolmogorov–Smirnov test demonstrated that the proposed set reduced simulation error with plausible estimated parameters. With empirical data of the Dow Jones Industrial Average index, we found that fundamental traders account for 9%–11% of all traders in the stock market. The statistical analysis of simulated data can produce the important stylized facts in real stock markets, such as the leptokurtosis, the heavy tail of the returns, and volatility clustering.

Résumé

Le marché financier est un système complexe dans lequel la relation entre ses composants ne peut être appréhendée analytiquement. Ainsi, des approches informatiques, telles que la simulation et la modélisation, sont nécessaires pour comprendre cette relation. Au cours des dernières années, la modélisation à base d'agents est devenue une technique de modélisation de simulation puissante pour simuler les actions et les interactions d'agents autonomes dans les systèmes sociaux, économiques et financiers. Dans cette thèse, le modèle à base d'agents est étudié pour construire un marché boursier artificiel et de nouvelles contributions au modèle à base d'agents sur les marchés financiers sont également présentées. Plus précisément, l'objectif de cette thèse est d'étudier différentes approches pour ajuster les paramètres de la stratégie de trading et différentes méthodes pour optimiser le modèle basé sur les agents. De plus, deux modèles à base d'agents sont développés pour étudier des faits stylisés importants dans les marchés boursiers réels.

La première partie de la thèse étudie le comportement commercial des agents techniques. L'algorithme génétique est une méthode d'optimisation populaire largement utilisée dans les études précédentes. Pour atteindre les objectifs d'investissement à court terme et trouver une méthode d'estimation des paramètres qui peuvent changer dynamiquement en fonction des conditions de l'environnement, nous introduisons de nouvelles approches telles que l'optimisation bayésienne et l'apprentissage automatique. Les fonctions objectives complexes dans les stratégies commerciales des agents techniques sur le marché réel sont également prises en compte. Notre approche consiste à donner une définition plus simple de l'optimisation de la stratégie d'investissement. Ensuite, un système artificiel de négociation d'actions avec des stratégies de négociation communes est construit. Les résultats de cette partie peuvent être appliqués pour construire un modèle boursier artificiel avec multi-agents. Lors de la construction d'un modèle à base d'agents pour le marché boursier, nous sommes confrontés au problème de l'optimisation des paramètres du modèle. Il s'agit d'un problème complexe lorsque différents agents et leurs interactions sont ajoutés au modèle. L'objectif de cette partie est de trouver une méthode d'optimisation des paramètres pour les modèles à base d'agents du marché boursier avec un faible coût de calcul et une grande précision. Pour résoudre les problèmes ci-dessus, nous introduisons une approche d'optimisation bayésienne

avec différentes fonctions objectifs pour optimiser les paramètres du modèle. Il a été prouvé que cette approche fournit rapidement et de manière flexible des ensembles de paramètres optimaux pour le modèle. Nous faisons les expériences sur deux marchés différents, le marché boursier américain et vietnamien, analysant ainsi la possibilité de reproduire des faits stylisés sur le marché réel.

La deuxième partie de la thèse présente une application de la modélisation multi-agents au marché boursier. En conséquence, nous avons montré que les modèles basés sur les agents et les stratégies de trading des agents peuvent être optimisés en utilisant la méthode d'optimisation bayésienne. Les faits stylisés du marché réel peuvent être reproduits en construisant soigneusement les fonctions objectives des agents. Dans le détail, nous avons introduit trois types d'agents typiques du marché boursier : les traders de bruit, les traders techniques et les traders fondamentaux. Après avoir construit le modèle, l'optimisation bayésienne est utilisée pour régler les paramètres des stratégies des commerçants ainsi que du modèle boursier. Les résultats expérimentaux sur l'optimisation bayésienne avec le test de Kolmogorov–Smirnov ont démontré que l'ensemble proposé réduisait l'erreur de simulation avec des paramètres estimés plausibles. Avec les données empiriques de l'indice Dow Jones Industrial Average, nous avons constaté que les traders fondamentaux représentent 9%–11% de tous les traders du marché boursier. L'analyse statistique des données simulées peut produire les faits stylisés importants dans les marchés boursiers réels, tels que la leptokurtose, la queue lourde des rendements et le regroupement de la volatilité.

Chapter 1

Introduction

1.1 Motivation

The study of complex systems has always been of great interest because of their rich properties and behavior, for example, the solar system [55] or biological systems [67]. By analyzing such natural systems, humans have discovered fundamental laws that can help build other systems such as computer networks [42] and power grid systems [109]. Financial markets can also be considered as complex systems [46, 69]. The way financial markets work is the subject of much debate among researchers and financial experts. Besides, the trading behavior of investors is also a matter of concern.

The system theory in financial markets is divided into system structure and system behavior [50]. The market structure involves trading rules, information management systems, and information communication systems. To describe the market structure, in [62], the following organizational factors are studied: traded instruments, submitted orders, market participants, and market rules. Besides, financial market behavior can be characterized by complex macroeconomic behavior. Many researchers conducted empirical studies to identify a set of common characteristics among financial data that are known as stylized facts [43, 31, 137]. These main facts can be summarized as follows: random walk price behavior, fat-tailed return distributions, power-law tails in the distribution of returns, excess volatility, volatility clustering, and the power-law autocorrelations in absolute returns.

In recent years, agent-based financial modeling has grown into an important research field for developing and understanding the complex phenomena that are observed in financial systems. The agent-based computational economics [131] and the micro-simulation [86] approaches have been proposed to study the emergent properties from the interactions of traders. These approaches attempt to model financial markets as evolving systems of competing, autonomously interacting agents and emphasize their learning dynamics [131].

Agent-based models offer the possibility of transparently modeling behavioral issues and studying in this way the effect of agents' behavior on market prices. Then asset prices in the models can be formed by the interaction of market participants. By using agents for studying market dynamics, the heterogeneous, boundedly rational, and adaptive behavior of market participants can be represented, and their impact on market dynamics can be assessed.

Along with the emergence of more traded assets, such as derivatives, gold, and cryptocurrencies, the need to understand market behavior and investors' trading behavior becomes more urgent. In [115], the authors analyzed the early exercise of the Chicago Board Options Exchange listed calls by different classes of investors over the 1996-1999 period. Their findings provide evidence that prospect theory is operative in the options market and that it applies differentially across various classes of investors. The authors in [81] studied the day-trading policy in the Taiwan futures market and provided evidence that more experienced individual investors exhibit more aggressive day trading behavior, although they do not learn their types or gain superior trading skills that could mitigate their losses. In [30], the authors proposed an agent-based model to analyze the mining processes of two popular assets, Bitcoin and gold.

The results from financial market studies are not exclusive to academic research [45], but also provide important contributions to market participants and policy makers [138]. The artificial market serves as a test environment for policymakers to explore the impact of different regulatory policies on improving decision-making. By identifying the sensitive parameters that affect the financial market either directly or indirectly, many shocks can be analyzed and market crashes can be limited [61]. In [93], the authors emphasized the importance of modeling economic interactions. They argued that microeconomic regularities observed in behavioral economics and strong interactions should build the foundation of more methodologically flexible models for a systemic perspective on globalized economic systems. Furthermore, the authors in [45] highlighted that agent-based models are necessary to provide a quantitative description of how the economy is likely to react to government policies under different scenarios. By testing policies in controlled and repeated simulations, they can guide policymakers. For traders, they need to test and analyze many different strategies and determine the optimal time to make decisions. The authors in [38] argued that it is very important for traders to understand the dynamics of financial markets and the influencing factors. Traders exhibit different investing behaviors corresponding to their beliefs and preferences. This heterogeneity is considered the main source of the complex emergent behavior of markets. The results in [4] validate the predictive and profitability power of technical indicators in the markets of Indonesia, Malaysia, Taiwan, and Thailand. This ability has proven useful in generating returns in excess of Buy and Hold returns even after including

transaction costs, adjustments for risk, and data snooping. In [119], the authors examined the potential profit of technical trading strategies among 10 emerging equity markets in Latin America and Asia and found that Taiwan, Mexico, and Thailand emerge as markets where technical trading strategies can be profitable. Therefore, the analysis of the trading strategies of technical agents in a financial market should also be considered.

1.2 Research questions

Based on the above considerations, an interesting research question was posed, which we have partially investigated: whether we can develop an artificial stock trading system and build a model that includes traders' behavior that can be applied to the real market. This question is difficult to completely answer, and it also requires sufficient understanding of the applications of agent-based modeling in the financial market. Thus, in this thesis, there are three sub-problems that we have targeted and investigated:

(i) How can we build an artificial stock trading system and optimize the strategies of technical agents?

(ii) Whether Bayesian Optimization can help to tune parameters in agent-based financial models?

(iii) How can we apply the optimal results for the technical agent and agent-based model to the financial market?

Related to the first question, we start by analyzing the trading behavior of technical agents. In the literature, different trading strategies of technical agents and parameter optimization methods have been studied. However, in practice, there are very few applicable strategies. Previous studies on parameter optimization methods have provided interesting findings, but they are too generic in nature and fail to address the complexity of objective functions in the real market. First, to study the complex objective functions in the trading strategies of technical agents in practice, we introduce and compare different parameter optimization approaches. These approaches aim to give a simple definition of trading strategy optimization and then be applied to trading strategies commonly used in real markets. Next, to understand the behavior of traders in the market, we design an automated trading system in which traders are technical agents. It can be useful for choosing a trading strategy according to long-term or short-term purposes and comparing different trading strategies. The proposed optimization methods can be applied to build diverse trading behaviors of technical agents in

an artificial stock market model. In addition, the parameters of the agent-based model used in the financial market can also be optimized.

The second main research question is related to the problem of parameter optimization for an agent-based model. The ability to handle a large number of parameters is an important feature of agent-based artificial stock markets in relation to other market models and real stock markets [85]. When the number of parameters increases, the parameter optimization for the model becomes a complicated problem. The selection of the parameter optimization algorithm can affect the approach and performance of the model because it affects the behavior of the agents and their ability to reproduce stylized facts of the real market. Therefore, the basic requirements for parameter optimization of the model are low computational cost and high accuracy. However, applying the Grid Search method with large parameter sets and running simulations for long periods of time [3, 120] are the common limitations of studies on this topic. Even for small models, exploring the stylized facts through all parameter combinations is not possible or is prohibitively costly [82]. For example, in [120], the authors generated 400,000 initial points to calibrate their model. To solve the above problems, this model introduces an approach with Bayesian Optimization and an error measurement function to tune the hyperparameters of the model. This approach can quickly and flexibly provide optimal parameter sets for agents' trading strategies and the agent-based model.

The goal of the third question is an attempt to contribute to the understanding and diversification of agents for financial market modeling. We present applications of agent-based modeling to the stock market. As a result, we have shown that the parameters of agent-based models can be tuned using the Bayesian Optimization method. The stylized facts of the actual market can be reproduced by carefully constructing the objective functions of the agents. In detail, we have introduced three types of agents typical of the stock market: noise traders, technical traders, and fundamental traders. After building the model, Bayesian Optimization is used to tune the parameters of the strategies of traders as well as of the stock market model. The experimental results on Bayesian Optimization with the Kolmogorov–Smirnov test demonstrated that the proposed set reduced simulation error with plausible estimated parameters. With empirical data from the Dow Jones Industrial Average index, we found that fundamental traders account for 9%–11% of all traders in the stock market. The statistical analysis of simulated data can produce the important stylized facts in real stock markets, such as the leptokurtosis, the heavy tail of the returns, and volatility clustering.

1.3 Related works

1.3.1 Algorithms for parameter optimization in trading

Analysis of technical trading strategies is a topic that always attracts the attention of traders in the real market [73]. Besides, the approaches and methods to optimize trading strategies are also studied in many pieces of financial literature [103, 119, 41]. A trading strategy is a predefined set of rules to apply in investing. In the stock market, an important objective of traders is to find or optimize trading strategies that offer optimal profits with minimal risks. Common methods are backtesting [107] and optimizing the parameters [13, 58, 15] of the trading strategies before they are deployed in the real market.

Backtesting and optimization of trading strategies have emerged as interesting research topics and experimental problems in many fields, such as finance [41], data science [16] and machine learning [124]. The optimal trading strategies are the result of finding and optimizing the combination of parameters in the strategy to satisfy the profit or risk conditions. Like model optimization, optimization of trading strategies is a process through which a model learns its parameters [110]. Besides, the trading strategy also has some hyperparameters that cannot be learned but can be tuned. In contrast to model parameters, which are automatically learned from data during training, model hyperparameters are manually set and tuned. They are then used during the training step to help learn the model parameters. The common method used to optimize the model's parameters is to apply a large set of parameters and run the simulation for a long time [13]. However, even for small models, discovering all parameter combinations is an expensive or sometimes impossible process [123].

There are mainly two kinds of parameter optimization methods: manual search and automatic search. Manual search attempts parameter sets manually and is dependent on the experience of proficient researchers [142]. Through visualization tools, the important parameters and the relationship between certain parameters and results are determined [135]. From the above aspects, manual search requires researchers to have professional background knowledge and practical experience in the research field. This makes it difficult for researchers who are not familiar with the models or data in a new field. Furthermore, the process of optimizing parameters is not easily repeatable. Trends and relationships in parameters are often misinterpreted or missed as the number of parameters and range of values increase.

The automatic search algorithms have been proposed, such as Grid Search or Random Search [13], to overcome the drawbacks of manual search. Grid Search prevails as the state of the art despite decades of research into global optimization [105, 76, 116]. This method lists all combinations of parameters and then performs model testing against this list [123].

Although automatic tuning is possible and the global optimal value of the optimization objective function can be obtained, Grid Search is inefficient both in computational time and in computational power. As the number of parameters increases, the number of models to train increases exponentially [52]. To solve this problem, the Random Search algorithm has been proposed. Random Search only randomly selects a finite number of parameters from the list to conduct a model test [13]. By reducing the search space of unimportant parameters, the overall efficiency is improved and the approximate solution to the optimization function can be found. Random Search is efficient in high-dimensional space since it doesn't run enough cases like Grid Search. However, some complex models require a global optimal result of the objective function [13], a new optimization method is needed as an alternative to random search.

In complex models, the objective function of the optimization can be either unknown or a black-box function. A very efficient optimization algorithm that optimizes to solve this problem is Bayesian Optimization [15]. Bayesian Optimization uses the results from the previous iteration to decide the next parameter value candidates. So instead of blindly searching the parameter space like in Grid Search and Random Search, this method advocates the use of intelligence to pick the next set of parameters, which will improve the model's performance [123]. Experimental results show that the Bayesian Optimization algorithm outperforms other global optimization algorithms [71]. Although BO provides superior results for parameter optimization compared to Grid Search and Random Search, this method also has its disadvantages. The high-dimensional problem of parameters is costly and contradicts the objective of BO.

Another optimization method used in many studies is evolutionary computing. In [106], the authors presented a general definition of the parameter optimization problem and discussed a genetic algorithm based on evolutionary computing for the optimization of trading strategies. Evolutionary computing handles the high-dimensional problem well and produces globally optimal or near-optimal solutions efficiently [147].

By focusing on extracting interesting and statistically significant trading strategies [54], demonstrating and promoting the use of specific data mining algorithms [146], many trading strategies were found. However, there are very few applicable strategies in the real market. The reason for the above problem lies in the oversimplification of optimization environment and evaluation fitness. This has created a gap between academic findings and business expectations [5]. The author in [96] pointed out that the factors and constraints of the real market were not fully taken into account. Instead of analyzing some unrealistic algorithms, this thesis tries to propose some practical solutions and results.

1.3.2 Agent-based modeling for financial markets

An agent-based model (ABM) is a computational model for simulating the actions and interactions of agents to understand the behavior of a system and what governs its results. In ABM, autonomous decision-making entities, known as agents, assess their own situations and make decisions on the basis of a set of rules [10]. The complex system is then modeled as a collection of agents. Emerging phenomena of the actual environment can be reproduced in the ABM. This capability is what drives other benefits of the ABM model, such as flexibility and diversity, through which complex systems can be described spontaneously in the ABM [17]. From the above benefits, ABM proves to be superior to other modeling techniques.

Since the financial crisis of 2007-2008 to the present, ABMs have gradually evolved and become an essential tool for understanding and describing financial markets. In traditional economic theory, the complex structures of financial markets are simplified by modeling market participants as though they were rational and had homogeneous beliefs. In [126], the author points out the advantages and disadvantages of this concept. Models built on traditional economic theory can describe many market phenomena, but they fail to describe systems close to bifurcation and phase transitions such as financial bubbles and crashes. The author in [18] argues that behavioral aspects and empirical data should be replaced with axioms such as market efficiency. With that in mind, ABM provides a framework with a "bottom-up" approach to describe an artificial financial market with out-of-equilibrium phenomena that takes into account behavioral aspects and heterogeneity.

Historically, the widespread availability of computers allowed the creation of complex simulation models. Along with the development of computer systems, the development of ABM became widely adapted in the 1990s. Stigler was one of the pioneers in simulating a financial market. In [128], he developed a simulation of an American stock market. The impact of regulating the Securities and Exchange Commission based on empirical data and an artificial order book is then tested. The authors in [26] pointed out that artificial stock markets are composed of many heterogeneous adaptive traders. They further remark that the artificial stock markets are rich in dynamics and emergent properties. This is a promising way to study the stock market as a complex adaptive system. In the financial market, there are many types of agents (investors, banks, central banks, etc.). These agents can interact directly by trading stocks or exchanging derivatives. They can also interact indirectly, because any decision or behavior of an agent affects others [53]. Moreover, the agents have different sensitivities to exogenous factors in the market, and thus different news can influence and impact them [24]. Therefore, financial market modeling needs to define the traded instruments, understand the characteristics of agents such as behavior and interaction between investors, and consider the impact of exogenous factors such as public news on the market [32, 63].

The first aspect that is of interest to many researchers is the types of traded instruments available in the market. The financial instruments are defined as the assets traded in the financial market. Stocks are financial assets that represent ownership of corporate assets [62] and are a popular subject of research in financial markets. Studies on the characteristics of the stock market or the impact of investors' behavior on the stock market have always attracted the attention of researchers [72, 144, 113, 74]. Inspired by those works, many studies have been performed to strengthen the understanding of other assets traded in the financial markets, such as derivatives, gold, and cryptocurrencies [115, 81, 30]. Moreover, in [145, 28], the authors presented an agent-based model of a financial market with multiple assets. Through the approach of the ABM model, the simulation can be used to help regulators determine important factors that contribute to market phenomena. The simulated market is validated against empirically observed characteristics of price returns and volatility.

The second aspect addresses the types of agents and how they impact the dynamics of the market. The authors in [65] developed one of the first heterogeneous agent models of the stock market. This model describes two types of traders: fundamentalists and chartists. Technical traders are considered a factor of instability because their trading often adds a positive response to the dynamics of the market [103]. In contrast, fundamental analysis based on its definition provides a stabilizing impact on market dynamics [60]. The concept of zero-intelligence traders, or noise traders, who decide randomly whether to buy or sell, was introduced in a double auction market [57]. Noise traders often miss the distinction between noise and good information. The existence of noise traders may cause price and risk divergence from expected levels even if they represent a small number of traders. But this definition explains the large trading volume that occurred in real markets.

The third aspect is the diversity of agents. To make every agent unique, they are assigned different memory lengths or reaction intensities according to price and fundamental changes [63]. In [92], the authors proposed an ABM with noise traders and fundamentalists. In this model, the noise traders can switch between an optimistic and pessimistic view of the market as influenced by the opinions of other traders. By comparing the expected returns of the strategies, the traders can also switch between fundamental analysis and a noise strategy. Exogenous factors such as news, social media, and insider information [114] also have different impacts on each agent. In addition, the sentiments of agents, such as risk aversion, play an important role in the decision-making process [89, 48].

The last aspect that should be considered is the interaction between agents. The Santa Fe stock market is a famous ABM in which traders forecast future returns using a combination of classifiers [9]. These classifiers have properties similar to gene mutations, which means that successful strategies are retained while unsuccessful ones are replaced over time. Besides

the interactions that directly affect market prices, the decisions of agents also have an indirect effect on each other. The authors in [36] showed that the interaction of fundamentalists and chartists can lead to chaotic behavior in exchange markets. In [77, 137], the authors introduced the propagation in the decision of agents with a fixed probability. In another study, [63], the interactions and diversity among fundamental traders and technical traders were introduced. The average utility associated with each type was calculated, and agents could rely on profit to come up with an effective investment strategy.

Each of these aspects has been extended in various interesting directions. From there, the stylized facts of financial markets are highlighted, such as heavy tails in stock return distribution and volatility clustering [64, 137]. Another proposed stylized fact is the positive autocorrelation in volatility and trading volume [129, 63]. The authors in [37] described one of the first ABMs in which financial bubbles are created by the noise traders' positive feedback on the asset price. More recently, the authors in [139] analyzed the consequences of predicting and exploiting financial bubbles in an agent-based model and found a proportion of up to 10% of dragon riders have a beneficial effect, reducing volatility, value-at-risk, and average bubble peak amplitudes. In [11], the behavior of agents is introduced with bounded rationality. The authors found that the deflation mechanism of less representative news disappeared after the extreme event.

Based on the above properties, agent-based modeling can be seen as a promising approach for modeling complex dynamical systems. Unfortunately, agent-based models are not perfect either; they also entail problems. The most commonly discussed aspects include validation and calibration. A question posed is how models and data relate to reality and real data. Another problem is how to optimize when there are too many parameters. To solve the above problems, our research focuses on building a simulated stock market. Thereby, we introduce different optimization methods to validate and calibrate the model. The results of the study provide some promising approaches to solving the problems faced by ABM.

1.4 Contribution of the thesis

The thesis is divided into two main parts: (i) theoretical and methodological contributions to the study of agent-based financial modeling and (ii) applications of agent-based models in stock markets.

With regard to the theoretical and methodological aspects, our first contribution is **parameter optimization methods for trading strategies of technical agents**. First, we formulate the problem by providing a definition of trading strategies. Trading indicators in different categories are then introduced. When applying trading strategies to specific indicators, we

consider different objective functions and parameter optimization methods. Besides the classical parameter optimization method based on Genetic Algorithm, we also propose new methods with Bayesian Optimization and machine learning approaches. This study is helpful for technical traders to analyze and choose a reasonable trading strategy for short-term or long-term investment purposes. From the knowledge gained, we execute the trading strategies with specific stocks or cryptocurrencies through an artificial trading system and evaluate the results obtained. The performance of the optimization methods is also evaluated and compared, which can be considered a framework for evaluating the optimization method of investment strategies based on technical analysis. These methods can also be used to optimize parameters for ABMs in financial markets. The second methodological contribution is **the parameter optimization methods for agent-based models in financial markets**. Different from the parameter optimization methods in previous studies, the Bayesian Optimization method is introduced to estimate hyperparameters more quickly and flexibly. We present the advantages of the Bayesian Optimization method for complex models such as financial markets compared with other classical optimization methods such as Grid Search or Random Search. Besides, we propose different objective functions for parameter optimization based on price and return. This variety of objective functions is necessary for an agent-based model in financial markets to represent stylized facts in the market. The algorithm can work with large volumes of data in a short time. In particular, this offers significant contributions when computational costs become expensive. Therefore, Bayesian Optimization can be used to speed up parameter optimization in financial multi-agent models. Finally, we contribute **a new stylized fact about the proportion of traders in financial markets**. The proportion of traders in the market can be considered a stylized fact and has important implications in financial markets. The other important components of the model, such as the environment, the trading behavior of traders, and the interactions between the agents, are also discussed in detail.

In the second part of this thesis, we focus on **the application of ABM in the stock market**. The results can be applied to other markets, such as the forex market or the crypto market. We combine the main results of the pioneering work to propose an ABM with diverse traders. Their direct and indirect interactions and exogenous factors that directly affect trading behavior are also introduced. From there, we conduct an artificial financial market simulation with two main components. The first component to be considered is the environment. The model considers a fixed amount of agents trading a single asset. With these liquidity assumptions, the market price and market return are then calculated. Next, we introduce the agent component, which includes different types of traders based on their predefined trading rules, such as noise traders, technical traders, and fundamental traders. In

there, the trading strategies of the technical agents are focused. In this study, the proportion of technical agents is shown to account for the majority of the stock market. Besides, in the era of development, along with the outstanding power of computer systems, technical analysis is increasingly interesting. From the introduced parameter optimization methods, we apply Bayesian Optimization with various objective functions to provide estimated parameters quickly and introduce interesting stylized facts. This research is useful for understanding and investigating the nature of financial markets in practice. Moreover, it can also be considered the first step toward supporting the construction of more complex and diverse models with outstanding assets other than stocks. The results can be used to discover new trading strategies as well as optimize trading strategies depending on the behavior of agents.

1.5 Organization of the thesis

The structure of the thesis is directed by the stated research questions. Each chapter in this thesis describes a subproblem. To begin with, in Chapter 2, we study the trading behavior of agents based on technical analysis. We give a general definition of the optimization problem for agents' trading strategies. Thereby, the technical indicators and corresponding trading strategies are introduced. After defining the trading strategies, we apply different approaches to optimize the parameters with different objective functions. Through the obtained results, we aim to answer research question (i).

In Chapter 3, parameter optimization with a Bayesian Optimization approach is considered to optimize an ABM for financial markets. In particular, a simple ABM with noise traders is considered. In the design phase of the trading environment, we try to take into account both the general characteristics and different aspects of the real market. Based on the analysis of the market structure and the trading behavior of market participants, we present an approach used to estimate the parameters for the model with different objective functions. We briefly discuss the ideas behind analytical research, briefly mention what stylized facts they can reproduce, and discuss the advantages and shortcomings of each objective function. This analysis aims to answer research question (ii), and the result is a method for parameter optimization for a financial market.

In order to evaluate the proposed ABM and answer research question (iii), Chapter 4 tries to replicate and extend ABMs from the literature. However, our research goal goes beyond replicating these experiments. We aim to improve the understanding of stylized facts in the market. Therefore, taking advantage of the proposed model's properties, we extend the model with multiple agents and their interactions and analyze the market dynamics within it. The main focus of the analysis is to determine the proportion of investors in the market.

The structure of the market and the factors that we consider important are introduced when building the model. We study the trading behavior of market participants with different roles, such as noise traders, technical traders, fundamental traders, and market makers. This model actually extends the real market simulation proposed in Chapter 3.

Finally, in Chapter 5, we evaluate our research objectives and analyze how well we have managed to answer the research questions. We end the thesis with suggestions for future research.

Chapter 2

Parameter Optimization for Trading Algorithms of Technical Agents

The main objective of the thesis is to develop an artificial financial market that can reproduce the characteristics of the real financial market. To support that goal, the focus of this Chapter is on the analysis of market trading indicators, especially those commonly used by technical agents in the financial market. This task proposes that the financial market system should be studied as a trading system that describes a set of actual financial markets. Accordingly, this Chapter includes two tasks: building an automated trading system and studying different approaches to optimizing the parameters of a trading strategy based on technical analysis. In our system, the trading task is formulated as a decision-making problem in a large and complex action space, which is applicable for employing a reinforcement learning algorithm. Our work includes the development of a learning environment, state representation, reward function, and learning algorithm for financial markets. The parameter optimization algorithms considered are Genetic Algorithm (GA), Bayesian Optimization (BO), and Deep Reinforcement Learning (DRL). The trading strategies are built based on a leading indicator, the Relative Strength Index (RSI), and two lagging indicators, the Bollinger Band (BOLL) and the Moving Average Convergence-Divergence (MACD). Multiple experiments are performed in different markets, including the cryptocurrency (also called "crypto") market, stock market, and crypto futures market. The 15-minute Open-High-Low-Close price data is used to backtest trading strategies based on technical analysis. The results show that optimized strategies from the proposed approaches can generate higher returns than their typical form and the Buy and Hold strategy. Among the indicators being studied, RSI can beat all the markets in the testing period and is the best indicator when trading in the crypto market. We also propose the appropriate investment purpose for each approach, BO approach

is suitable for a stable trading strategy over the long-term, while the DRL approach is suitable for short-term traders.

2.1 Introduction

An automated trading system is a type of information-based decision-making system that allows traders to establish specific rules for both entry and exit of trades and is executed automatically through a computer. Various platforms report that about 75% of shares traded on United States stock exchanges come from automatic trading systems [25]. In this context, Reinforcement Learning (RL) is applied to change the way classical trading systems work. RL is a self-training system that involves taking actions with the aim of maximizing rewards and achieving the best results. Therefore, instead of making explicit decisions based on price forecasting, in this study our models are trained to execute trading positions directly. In recent years, the use of RL has been greatly increased in the study of trading in financial markets [143, 88, 87]. The authors in [88] applied different Deep Reinforcement Learning (DRL) techniques to build an automated cryptocurrency trading system. As a result, the Double Deep Q-Learning trading system based on the Sharpe ratio reward function demonstrated to be the most profitable approach for trading bitcoin. In another study, an agent is trained in [143] to learn an adaptive stock trading strategy, and it is shown that the DRL approach outperforms the Buy and Hold strategy in terms of both the Sharpe ratio and cumulative return.

In automated trading systems, analysis based on technical indicators is used to identify trading signals. Besides choosing the right trading indicators, optimizing a trading strategy to maximize profits or minimize risks is an important step for technical traders. One of the implementation methods is to backtest and tune the parameters before they are deployed to the real market. Nowadays, the increasing size of data sets and the complexity of strategies lead to more resource requirements and make parameter optimization a harder challenge. Several solutions have been proposed to tune parameters in terms of computational complexity as well as scalability. Common methods for optimizing parameters automatically are Grid Search [83] and Random Search [13]. Grid Search is a method to list all combinations of parameters within given ranges, then perform a model test with this list. Random Search only randomly selects a finite number of parameters from the list to conduct a model test. The advantage of Grid Search is that it can find the global optimal value. However, as the number of parameters increases, the disadvantage of this method is clearly shown in the time and cost of implementation [83]. The advantage of Random Search over Grid Search is faster execution time because it does not run enough cases, but this also leaves

us wondering if a better combination of parameters exists [13]. Genetic Algorithm can limit computational complexity and quickly converge to a set of appropriate solutions. For high frequency trading purposes, this is an appropriate method, as traders require a quick and good solution instead of going through a long process to find the global optimal solution. In [80], Genetic Algorithm and neural networks are used to design a stock trading system with intelligent decision support. The results showed that the strategies of the trading system gave better results when compared to the Buy and Hold strategy. Bayesian Optimization is another technique that relies on probabilistic models, such as Gaussian processes, the Tree-structured Parzen Estimator and Random Forest [40], to optimize the expected improvement of the surrogate model by treating the response as a black box function. In [123], a framework of BO for hyperparameter tuning is used to show that the Gaussian process plays an important role in obtaining a good optimizer and expert-level performance. Although BO provides superior results for parameter optimization compared to previous methods such as Grid Search or Random Search, this method also has its disadvantages. The high-dimensional problem of parameters is costly and contradicts the objective of BO. Several studies have been conducted in order to address the above problems [104, 101, 35]. In [101], the authors proposed a framework for efficient BO that allows optimization of acquisition functions in low-dimensional feature space based on a non-linear embedding method. Another method for high-dimensional BO is presented with generalized additive kernels based on Thompson sampling in [104]. However, the above solutions are only implemented with Gaussian processes as the surrogate model. This creates some limitations because Tree Parzen Estimators are also a common model used. In recent studies, parameter optimization is also addressed in the RL framework, especially for architectural network design. In [70], the hyperparameter tuning for machine learning models was formulated as a RL problem, and then a novel policy based on Q-learning was proposed for navigating high-dimensional hyperparameter spaces. However, the application of the RL framework requires building a suitable environment for each research problem, not just the financial market. Although there are many studies on the application of DRL in financial markets, these studies have only focused on identifying trading signals [94, 117, 100]. In contrast, there is not much research on parameter optimization approaches for trading strategies. Common methods such as Genetic Algorithm [47] and Bayesian Optimization (BO) [123] still have problems with parameter dimensions or expensive costs. From the above analysis, a new approach to parameter optimization for trading strategies in financial markets with high computational performance becomes an urgent need.

In this Chapter, we introduce the basic concepts and steps involved in the trading process. The trading system based on technical indicators, illustrated in Figure 2.1, aims to improve

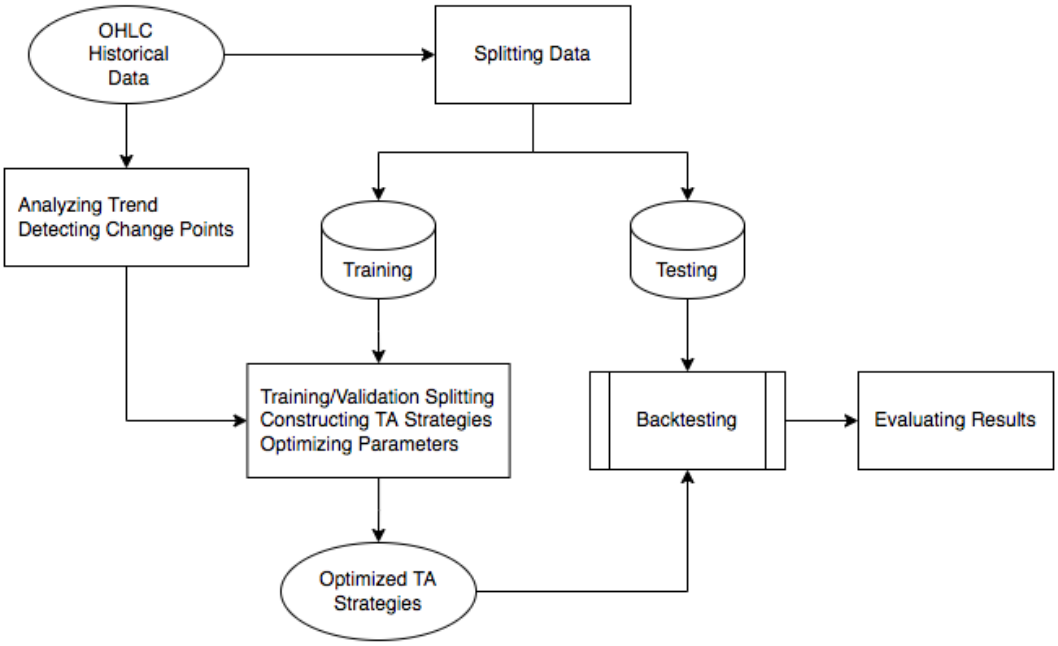


Fig. 2.1 General scheme of the proposed trading system

the technical analysis trading signals by incorporating parameter optimization techniques into the trading strategies. First, the data of a crypto spot market, a stock index market, and a crypto futures market are collected and divided into 3 series corresponding to training, validation, and testing periods. The training data and validation data are used to build and optimize the technical analysis-based strategies. The obtained strategies are then put into the backtesting step, and their performance is evaluated on the test data set. Finally, the results are analyzed to establish the quality of the proposed strategies. The main steps of this workflow are

(i) Analyzing the data and building the technical analysis strategies.

(ii) Optimizing the strategies with different methods.

(iii) Backtesting the strategies and evaluating the results.

Each step is described as follows:
(i) *Analyzing the data and building the technical analysis strategies*. The rolling forward method is applied to split the data into multiple sequences. We analyze trends and identify turning points to get an overview of each market. Based on current ongoing trend data, trend analysis helps to predict the future movement of that trend. When combined with indicators,

traders can clearly identify entry and exit points and earn profits when they trade with the trend and not against it.

(ii) *Optimizing the strategies with different methods*. In this step, the training data and validation data are used for backtesting the technical analysis strategies. For this purpose, two approaches to parameter optimization, Genetic Algorithm and Bayesian Optimization, are applied to find the parameters with the best performance.

(iii) *Backtesting the strategies and evaluating the results*. In this last step, the optimized strategy is backtested on the test data set. The performance of strategies in each market is evaluated and discussed.

Two main experiments were conducted. The first experiment is to compare the Bayesian Optimization approach with approaches from the state-of-the-art, such as the Genetic Algorithm, default parameter set and Buy and Hold strategy. The focus of this experiment is on the analysis of indicators commonly used by technical traders, also called technical agents in this work, with the aim of developing a multi-agent model for the financial market in future research. The trading strategies are built based on various indicators: RSI, BOLL, and MACD. The experiment is performed in the Dow Jones stock market, where 15-minute price data is used to backtest trading strategies. The results show that the parameter optimization method using Bayesian Optimization gives better results in terms of return and execution time than the Genetic Algorithm approach in all periods. The optimized strategies from the Bayesian Optimization approach can generate higher cumulative returns than their typical form and Buy and Hold strategy. Among the indicators being studied, BOLL with BO optimization is a profitable and stable optimal strategy. Two other experiments are performed with 15-minute data of BTC-USDT and BTCUSDT perpetual contracts for short-term trading purposes. With a simple objective function, total return, along with consideration of the cost per transaction, the optimized strategies from the Bayesian Optimization approach can generate higher returns and smaller volatility than the Buy and Hold strategy and their typical form in most cases. Experimental results also show that RSI is proven to outperform MACD with a more profitable and less volatile strategy.

In the second experiment, the highlight of our study is a novel technique based on the proposed DRL approach to optimize parameters for technical analysis strategies. The trading task is formatted as a decision-making problem in a large and complex action space, which is applicable for employing reinforcement learning algorithms. Specifically, we propose a learning environment, state representation, reward function, and learning algorithm for the purpose of strategy optimization in the cryptocurrency market that has not been studied before. The proposed trading system not only focuses on making decisions based on a given strategy but also includes a parameter optimization step in the trading process. Two

configurations are considered to build the artificial intelligence agent in the system: Double Deep Q-Network and Double Deep Q-Network setting. Bayesian Optimization is another approach introduced for comparison purposes. Different objective functions commonly used in trading optimization are also introduced, such as cumulative return and Sharpe ratio. The results showed that the DRL approach with the Double Deep Q-Network setting and the BO approach yield positive average returns for short-term trading purposes, where the system with the DRL approach yields better results. In terms of execution time, the DRL approach also shows outstanding advantages, with an execution time 5.83 times faster than the BO approach. When comparing performance with different settings and objective functions, the Double Deep Q-Network setting with the Sharpe ratio as the reward function is the best Q-Learning trading system with 15.96% monthly return. The trading strategies are built on the simple Relative Strength Index (RSI) indicator, however, the results of this study can be applied to any technical or market indicator.

2.2 Background

Stock analysis is defined as market analysis activities intended to support investment decisions in the stock market. These analytical methods can also be applied to trading cryptocurrencies and derivatives. There are two common analytical methods, namely fundamental analysis and technical analysis [44]. Fundamental analysis is a method of determining the intrinsic value of a stock in the market by examining the fundamental factors that affect or change the price of that stock [1]. Through the financial statements of companies, fundamental analysts perform macro-analyses, such as analyses of the industry in which the company operates or analyses of the state of the economy. They also provide micro-analyzes such as the company's operating model and management performance to make investment decisions. From there, fundamental analysis helps investors evaluate a stock's current value and predict its future price. However, fundamental analysis requires not only good training from a system of experts but also a broad set of data about a company's financial information or its field of activity to be considered. This information is often difficult to collect and has low reliability due to different interpretations caused by the self-interest of company policies [111]. From the point of view of designing an automated system for stock market prediction and analysis, the most common solution is to turn to technical indicators [108], since the information needed is limited to the historical price of stocks. A technical indicator is a series of data points obtained by applying a formula to a series of price and volume data [2]. Price data includes any component of the opening, high, low, or closing value for a period of time. The technical analysis approach tries to predict future stock prices through technical indicators.

In this method, analysts will use charts, graphs, and trading volumes of stocks to analyze the supply (demand) fluctuations for those stocks and, from there, make recommendations to buy (sell). In essence, technical analysis assumes that there are patterns in the past that tend to be so frequent that they can be used to predict future prices.

Indicators are generally classified into two broad categories: leading indicators and lagging indicators. A leading indicator is a type of oscillator that gives signals before a new trend or reversal occurs. Designed to guide price movements, leading indicators represent a form of price momentum over a fixed period of time in the past. Using this time period, traders calculate indicators to apply to their investment strategies when predicting future market conditions. In addition, policymakers also study the indicators before setting monetary policy. For example, the 20-day Stochastic Oscillator will use prices from the past 20 days to calculate; all previous prices will be ignored. Some of the popular indicators used are the Commodity Channel Index, Relative Strength Index, Stochastic Oscillator, and Williams %R. Conversely, the lagging indicator is a momentum indicator that gives a signal after the trend has started. Traders use the lagging indicators to generate trading signals or confirm the strength of a certain trend. Since the lagging indicators are slower than the stock price, a significant change in the market has already taken place before the indicator gives a signal of that change. Therefore, lagging indicators are used to confirm the long-term trend but cannot predict it. An example of a lagging indicator is a moving average crossover, which occurs after a certain price move has occurred. Technical analysts use the short-term moving average crossing above the long-term moving average as a confirmation to place a buy order, as it indicates an increase in volume. The drawback of using the lagging indicator in stock market trading is that a significant move could have occurred, resulting in the trader being placed in a situation that was too late. Lagging indicators that follow price action are also known as trend-following indicators. Trend-following indicators work best when the market develops strong trends. They are designed to attract traders to join and keep them engaged as long as the trend is intact. On the other hand, these indicators are not effective in the case of a sideways market; they can give many false signals. Some of the popular trend-following indicators include Moving Averages and Moving Average Convergence-Divergence, the indicator chosen for study in the article.

The trading indicator can help a trader to identify market conditions; however, it is not a trading strategy. A trading method that uses technical analysis to identify market tendencies is called indicator-based trading. These strategies include trade filters to identify the setup conditions and trade triggers to identify exactly when to make orders. Finally, for a trading strategy to work effectively, it is necessary to backtest and tune parameters. The assumption is that if a strategy has not worked well in the past, it may not work well

in the future. Conversely, a strategy that has worked previously has a good chance but is unlikely to work again in the future. Therefore, backtesting is one of the most important aspects of trading system development. Besides helping traders optimize and improve their strategies, backtesting also helps find technical or theoretical flaws before applying them to the real-world markets. The second step to improving the performance of trading strategies is parameter optimization, or parameter tuning. Let $\Lambda = \Lambda_1 \times ... \times \Lambda_P$ denotes the P-dimensional parameter space that includes continuous or discrete values, and the objective function is a loss function $\mathscr{L}$ over a data set $D \in \mathscr{D}$, where $\mathscr{D}$ is the set of all data sets. The objective of the tuning algorithms is to optimize a trading strategy $M_\lambda \in \mathscr{M}$, from the space of all strategies $\mathscr{M}$ with parameters $\lambda \in \Lambda$. Thus, the task of parameter optimization is to identify an optimal parameter configuration $\lambda^* \in \Lambda$ that results in a strategy M_{λ^*}, such that the objective function on the validation set is minimized.

$$\lambda^* = \arg\min_{\lambda \in \Lambda} \mathscr{L}\left(M_\lambda(D_{train}), D_{valid}\right), \tag{2.1}$$

and $D_{train} \cap D_{valid} = \emptyset$.

The challenges in optimizing trading strategies come from different aspects, such as market dynamics, large volumes of data, the ability to optimize multiple objectives of trading strategies, etc. The process of solving these problems involves high-dimensional searching, high-frequency data flow, and cost optimization. Therefore, a trading system that can optimize profits and minimize execution time is very important.

2.3 Technical trading strategies

In this work, we describe in detail classical strategies based on three indicators: one leading indicator (RSI) and two lagging indicators (BOLL and MACD).

2.3.1 Relative Strength Index (RSI)

The RSI, developed by Wilder in the 1978s [141] is a technical analysis oscillator showing price strength by comparing upward and downward close-to-close movements. It is used to calculate the market's recent gains against its recent losses and translates that information into a number between 0 and 100. The formula for calculating RSI is given by

$$RSI = 100 - \left(\frac{100}{1+RS}\right) \quad \text{with} \quad RS = \frac{AP_w}{AN_w},$$

where
AP_w = Average of the variations of the price that resulted positive for the last w periods.
AN_w = Average of the variations of the price that resulted negative for the last w periods.

A stock is considered fairly priced if its RSI is at 50. RSI values above 70 indicate overbought conditions, while values below 30 indicate oversold conditions. The 30-70 levels are calculated by taking a constant shift value above and below the center line. The standard number of days used for calculating RSI is 14 days [141].

A buy signal is produced when the RSI falls below the oversold zone ($RSI < 30$) and rises above 30 again. When the RSI rises above the overbought zone ($RSI > 70$) and falls below 70 again, a sell signal is obtained. The trading rule based on the RSI indicator is the following:

$$s_{t+1} = \begin{cases} buy & \text{if } RSI_t > 30, \\ sell & \text{if } RSI_t < 70. \end{cases}$$

2.3.2 Bollinger Band (BOLL)

Bollinger band is an indicator invented by John Bollinger in the early 1980s. This indicator is composed of a moving average and a price standard deviation. Specifically, Bollinger bands consist of three bands:

- Middle band (MB) represents an intermediate-term trend. It is calculated as a simple moving average with a period of N. The middle band is a frame of reference for the upper and lower bands.

$$MB = \frac{1}{N} \sum_{i=M-N}^{M} p_i,$$

  where M is the total number of observations, and p_i is the asset price.

- Upper band (UB) is shifted up by K standard deviations above the middle band.

$$UB = MB + K\sigma_N,$$

  where $\sigma_N = \sqrt{\frac{1}{N}\sum_{i=M-N}^{M}(x_i - MB)^2}$ is a standard deviation calculated for the newest N observations.

- Lower band (LB) is shifted down by$K$ standard deviations below the middle band.

$$LB = MB - K\sigma_N.$$

When the market is volatile, the upper and lower bands will widen, conversely, when volatility decreases, the width of the two lines will narrow. The trading rule based on the BOLL indicator is as follows:

$$s_{t+1} = \begin{cases} buy & \text{if } p_t > LB_t, \\ sell & \text{if } p_t > UB_t. \end{cases}$$

Therefore, a buy signal is generated when the price crosses below the LB_t line, and we have a sell signal when the price crosses above the UB_t line.

2.3.3 Moving Average Convergence/Divergence (MACD)

MACD was developed by Gerald Appel in the late 1970s [57] and used to identify the trend direction and duration by calculating the relationship between two Exponential Moving Averages (EMA). The EMA is a moving average that shows the average price movement over a period of time. It was created to solve the problem of slow reactions to exchange rate movements by calculating an exponential formula that weighted the most recent price movements. The formula for calculating EMA is given by

$$EMA_t^m(p_t) = \begin{cases} p_1 & \text{if } t = 1, \\ \alpha \cdot p_t + (1-\alpha) \cdot EMA_{t-1}^m & \text{if } t > 1. \end{cases}$$

In the above formula, p_t refers to the current price, m refers to the number of observations, and α is a smoothing factor, $0 \leq \alpha \leq 1$, that is calculated as $\alpha = \frac{2}{m+1}$.

Given the time periods m, n and q so that $m < n$,

$$MACD_t = EMA_t^m(p_t) - EMA_t^n(p_t),$$
$$Signal_t = EMA_t^q(MACD_t).$$

The MACD line ($MACD_t$) is obtained as the difference between the faster EMA and the slower EMA. The signal ($Signal_t$) is the moving average of the MACD series.

The trading rule based on the MACD indicator is the following:

$$s_{t+1} = \begin{cases} buy & \text{if } MACD_t > Signal_t, \\ sell & \text{if } MACD_t < Signal_t. \end{cases}$$

Therefore, a buy signal is generated when $MACD_t$ crosses above the $Signal_t$ line, and similarly, we have a sell signal in the opposite scenario.

2.4 Parameter optimization

2.4.1 Objective function

The objective functions used to evaluate the performance of the trading strategies are:

(i) **Cumulative return** (R_C). Cumulative return is the aggregate amount that the investment has gained or lost over a given time period from 1 to N:

$$R_C = \frac{p_N - p_1}{p_1}.$$

(ii) **Annualized return** (R_A). Annualized return is the geometric average amount of money earned by an investment each year over a given time period (n days):

$$R_A = (1 + R_C)^{(\frac{252}{n})} - 1.$$

(iii) **Sharpe ratio** (SR). Sharpe ratio is an objective function used to calculate the investment's excess return over the risk-free rate (μ_p is the expected return and the risk-free rate μ_f is assumed to be zero in our work) relative to the standard deviation of returns (σ_p):

$$SR = \frac{\mu_p - \mu_f}{\sigma_p}.$$

(iv) **Maximum drawdown** (D_{max}). The worst decline-from-peak observed in the test period.

(v) **Number of trades (N_T)**. The number of trades done. A trade is composed of a buy and a sell order.

(vi) **Net Profit (NP)**. The amount of money left over after losses has been subtracted from the profits:

$$NP = Profits - Losses.$$

(vii) **Average profit per trade ($\bar{T}$)**. This metric is calculated by dividing the net profit by the number of trades:

$$\bar{T} = NP/N_T.$$

(viii) **Percent profitable (PP)**. This metric is also known as the probability of winning and is calculated by dividing the number of winning trades (N_T^W) by the total number of trades:

$$PP = N_T^W/N_T.$$

These objective functions can be used as fitness functions in Genetic Algorithm, score functions in Bayesian Optimization or reward functions in Deep Reinforcement Learning to optimize the parameters of trading strategies.

2.4.2 Genetic Algorithm

The Genetic Algorithm (GA), a class of adaptive search and optimization techniques, was developed by John Holland and his collaborators in the early 1970s [68] to find suitable solutions to combinatorial optimization problems. GAs are a branch of evolutionary algorithms that apply the principles of evolution such as heredity, mutation, natural selection, and crossover. The main idea is to continuously come up with different solutions to a problem while combining, mutating, and evaluating them. Without the need for assumptions regarding the continuity or differentiability of the loss function, it is able to evaluate loss functions associated with the predictor parameters [140]. The working principles of GAs can be considered an unconstrained optimization problem,

$$\text{maximize } f(x), \quad x_i^l \leq x_i \leq x_i^u, \quad i = 1,2,...,M,$$

where x_i^l and x_i^u are the lower and upper bounds that the variable x_i can take. Although a maximization problem is considered here, a minimization problem can also be handled.

The GA starts with a randomly generated population of individuals. In each iteration, a new population of the same size is generated from the current population to evolve to a state that maximizes its overall fitness. The three basic operations used on the individuals of the population are (i) selection, (ii) crossover, and (iii) mutation. These aspects are introduced in detail below.

Initialization

The search starts with a random population of N individuals. Each of these individuals encodes a sequence of genes representing a particular solution to the problem, and is called a chromosome. Depending on the problem at hand, the genes representing the solution could be bits (0's and 1's) or continuous (real values).

Calculation of fitness value

The fitness function is simply defined as a function that takes a chromosome as input and produces as an output how fit the solution is with respect to the problem under consideration. Therefore, the solution is more optimal when its fitness value is higher. Moreover, the fitness function should be sufficiently fast to compute since the fitness value is done repeatedly in a GA. After the fitness score is calculated, the individuals are sorted to select the fittest ones.

Selection

The selection phase determines which parental individuals are chosen for mating and how many off-springs each selected individual produces. Just like in natural selection, fitter individuals have higher chances of being parents. The main objective of this operator is to focus on good solutions and eliminate bad ones in a population while keeping the population size constant. There are several different selection techniques, including rank-based, Roulette wheel, and Tournament selection (which is used in this study).

In Tournament selection, a number of individuals are chosen at random from the entire population. These individuals compete against each other, and finally, the one with the best fitness is selected. This technique has the advantage that it does not require the individuals to be sorted by fitness first. It also includes less time complexity [58], easy parallel implementation, and low vulnerability to takeover by dominant individuals [34].

Crossover

The crossover operator is similar to biological reconstruction and intersection. This operator creates new off-springs to replace the least fit individuals in the population. The basic idea is

that, by combining different genes, better solutions can be created in fitter individuals. In this process, more than one parent is selected, and more off-springs are produced using the parents' genetic material. In [125], the crossover gene of each off-spring is given by

$$p_1^{new} = \alpha p_d - \alpha(p_m - p_d),$$
$$p_2^{new} = \alpha p_d + \alpha(p_m - p_d),$$

where p_1^{new}, p_2^{new} are the new off-springs, p_d, p_m are the parents, and α will be a random number between 0 and 1.

Mutation

The mutation operator is defined as a small, random change in a chromosome. It explores the search space to maintain and introduce diversity in the population. Given a certain mutation rate, we randomly select a number of individuals and assign a new random number to the positions of those genes.

Using the operators that we defined above, the algorithm can now solve the problem. The Genetic Algorithm procedure can be summarized by the steps in Figure 2.2.

1. Create an initial population of candidates (first generation) randomly.
2. Measure fitness: evaluate the performance of each candidate in terms of an objective function.
3. Selection: select the candidates for recombination.
4. Perform crossover and mutation.
5. Evaluate the performance of the new candidates.
6. Return to step 3, unless a termination criterion is satisfied.

2.4.3 Bayesian Optimization

Bayesian Optimization (BO) is a class of machine-learning-based optimization methods focused on solving the problem.

$$min_{x \in A} F(x),$$

where A is the feasible set and F is the objective function. The input x is in $\mathbb{R}^d$ for a value of d that is not too large. Typically, $d \leq 20$ in most successful applications of Bayesian Optimization. The objective function typically has the following properties:

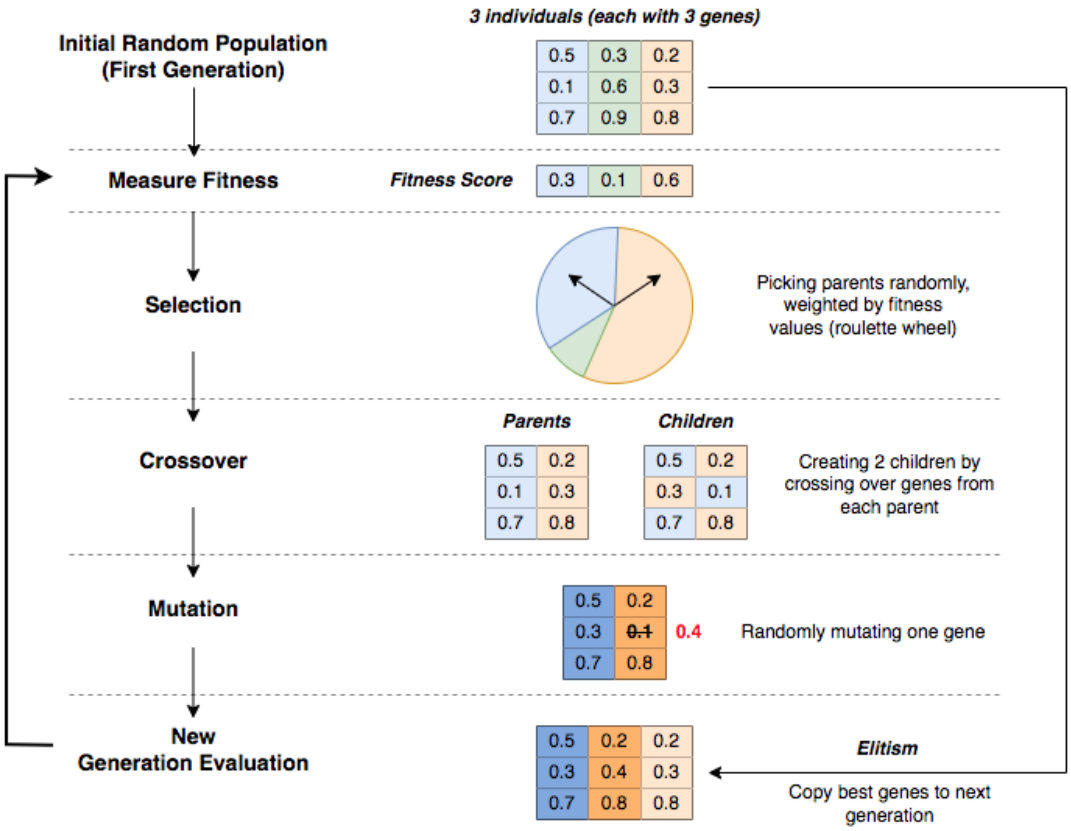


Fig. 2.2 Illustration of the Genetic Algorithm concept and an example

(i) The objective function F is continuous.

(ii) The evaluation cost of F is large.

(iii) F is a "black box", and we do not know its properties, such as linearity, convexity, or concavity.

(iv) Since F is a "black box", when we evaluate F, we observe only $F(x)$ and no first- or second-order derivatives. This prevents the application of first- and second-order methods such as gradient descent or Newton's method.

The basic idea of BO is to spend a little more time selecting the next parameters to make fewer calls to the objective function. By evaluating more promising parameters from past results, BO can find better model settings than a random search with fewer iterations. The BO method consists of two main components: a Bayesian statistical model (surrogate function) for modeling the objective function F, and an acquisition function for deciding where to sample next. The goal is to optimize the surrogate as close to F as possible, from which the global minimum/maximum of F can be easily found by taking the minimum/maximum of the surrogate.

Surrogate function

The surrogate function is the probability representation of the objective function built using previous evaluations. Bayesian methods work by finding the next set of parameters to evaluate on the actual objective function by selecting parameters that perform best on the surrogate function. There are several different forms of the surrogate function, including the Gaussian process, Random Forest regression and Tree-structured Parzen Estimator [12]. In this study, we incorporate Bayesian Optimization into our model using the Tree-structured Parzen Estimator (TPE). TPE is a type of sequential model-based optimization technique that uses the Bayesian approach in an effort to reduce computational time.

Let $P(s|x)$ denote the surrogate probability model expressing the probability of the score s given the proposed set of parameters x. Whereas the Gaussian Process-based approach models $P(s|x)$ directly, the TPE approach models $P(x|s)$ and $P(s)$. The formula of $P(s|x)$ is calculated by the Bayes rule as follows:

$$P(s|x) = \frac{P(x|s)P(s)}{P(x)}.$$

$P(x|s)$ is the probability of the parameters given the score on the objective function and is defined as:

$$P(x|s) = \begin{cases} l(x), & \text{for } s < s^*, \\ g(x), & \text{for } s \geq s^*. \end{cases}$$

where s^* is the threshold value of the objective function, $l(x)$ is the density formed by using the parameters x such that the corresponding loss $F(x)$ was less than s^* and $g(x)$ is the density formed by using the remaining observations. So TPE iteratively models two probability distributions, of which one is used to find the maximum instead of modeling the complete evaluation function.

Acquisition function

Given the surrogate function, the acquisition function is the criteria to find the next parameters. This study uses the most common choice of criteria, which is the Expected Improvement:

$$EI_{s^*}(x) = \int_{-\infty}^{s^*} (s^* - s)P(s|x)ds.$$

The aim of this method is to maximize the Expected Improvement (EI) with respect to x, which means finding the best parameters x under the surrogate function $P(s|x)$. If $EI_{s^*}(x) > 0$, the parameters x are expected to yield a better result than the threshold value. If $P(s|x)$ is zero everywhere that $s < s^*$, the parameters x are not expected to yield any improvement.

In [12], the authors show that, using Bayes' rule and some substitutions, it can be written as:

$$EI_{s^*}(x) \propto \left(\gamma + \frac{g(x)}{l(x)}(1-\gamma)\right)^{-1}, \tag{2.2}$$

where $\gamma = P(s < s^*)$, which is the quantile of the observed s function we use to model $l(x)$. The expression above means that we want to sample points that have a low probability under $g(x)$ and a high probability under $l(x)$ to maximize the improvement. This means that the TPE algorithm weighs exploitation more important than exploration.

The TPE works by drawing sample parameters from $l(x)$ and evaluating them in terms of $l(x)/g(x)$. The set of parameters that yield the highest value below $l(x)/g(x)$, corresponding to the greatest expected improvement, is returned. These parameters are then evaluated using the objective function. If the surrogate function is correct, then these parameters should yield a better value when evaluated.

The Bayesian Optimization procedure is illustrated in more detail in Figure 2.3. In summary, the Bayesian Optimization algorithm follows five main steps:

1. Building surrogate probability model of objective function.

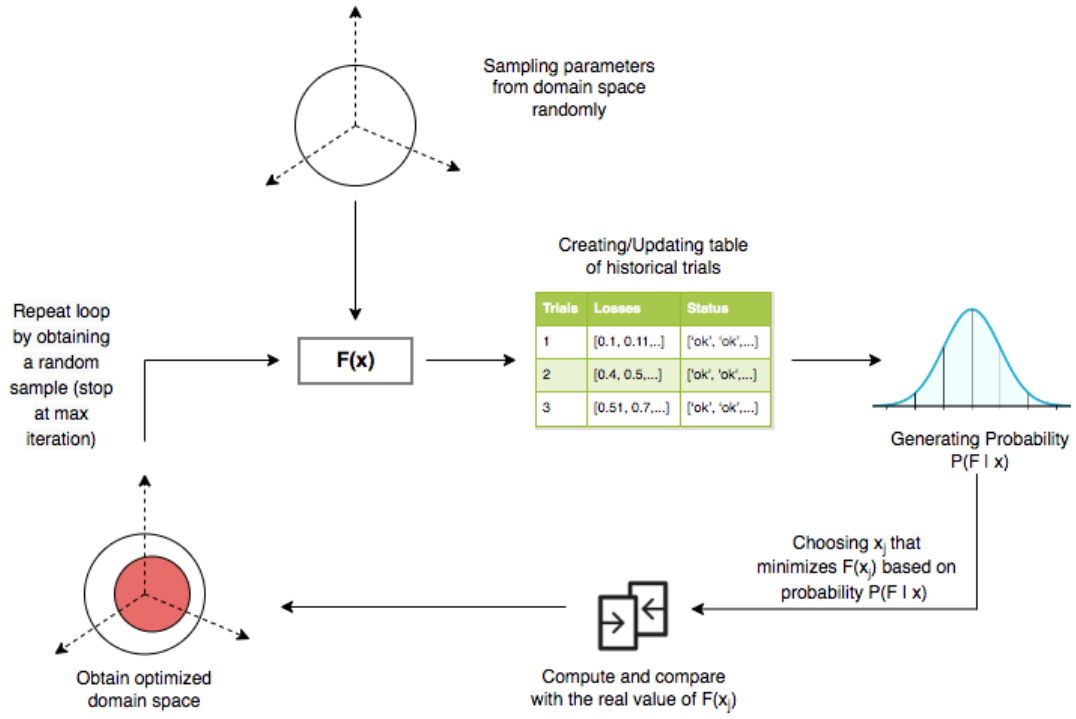


Fig. 2.3 Illustration of the Bayesian Optimization concept

2. From the surrogate function, find the most potential candidate using an acquisition function.

3. Evaluate the score of the candidate with the original objective function.

4. Update the surrogate model incorporating the new candidate and its score.

5. Repeat steps 2-4 until the maximum number of iterations or time is reached.

2.4.4 Deep Reinforcement Learning

Deep reinforcement learning (DRL) is the combination of deep learning and reinforcement learning (RL), allowing agents to make decisions from unstructured input data without manual engineering of the state space. In our problem, the goal is to train the artificial intelligence (AI) agent such that, given a trading scenario, it could give an optimized parameter set of the trading strategy and earn the possibly highest reward after a finite number of iterations, as quickly as possible. Instead of using classical optimization approaches, we adapt the Deep Q-Learning (DQN) algorithm [70] for our learning model. This approach is proposed because it does not require prior knowledge of how to efficiently optimize a trading strategy, and the learning algorithm is able to self-evolve when exposed to unseen scenarios. DRL was selected since it increases the potential for automation for many decision-making problems that were previously intractable because of their high-dimensional state and action spaces. In this section, we briefly describe our learning environment and AI agent, discuss the learning process, and discuss some implementation considerations. Accordingly, an automated trading system is introduced to optimize the trading strategies of the experiments performed in this work.

Learning Environment

The parameter optimization problem is formulated as a Markov Decision Process represented by a tuple $(\mathscr{S}, \mathscr{A}, R, \tau)$, where $\mathscr{S}$ is the set of possible states, $\mathscr{A}$ is the set of legal actions, a reward function $R : \mathscr{S} \times \mathscr{A} \to \mathscr{R}$ and the transition function $\tau : \mathscr{S} \times \mathscr{A} \times \mathscr{R} \to \mathscr{S}$ that generates a new state in a possibly stochastic or deterministic environment $\mathscr{E}$.

The scenario, or state of the environment, is defined as the data sets $\mathscr{D}$ plus the history of evaluated parameter configurations and their corresponding responses:

$$\mathscr{S} = \mathscr{D} \times (\Lambda \times \mathscr{R}). \tag{2.3}$$

The agent navigates the parameter response space through a series of actions, which are simply the next parameter configurations to be evaluated, and thus the action space corresponds to the space of all parameter configurations through the function $g : \mathscr{A} \rightarrow \Lambda$. According to the definition of the action space, the agent executes an action from $\mathscr{A} = \{1, ..., |A|\}$. For example, action $a = 1, a \in \mathscr{A}$, corresponds to parameter set $\lambda = g(a) = \{\lambda_1\}^{dim(\Lambda)}$ and action $a = |A|$ corresponds to parameter set $\lambda = \{\lambda_{|A|}\}^{dim(\Lambda)}$.

The parameter response surface can be any performance metric that is defined by the function $f : \mathscr{D} \times \Lambda \rightarrow \mathscr{R}$. The response surface is to estimate the value from an objective function $\mathscr{L}$ of a strategy $M_\lambda \in \mathscr{M}$, with parameters $\lambda \in \Lambda$, over a data set $D \subset \mathscr{D}$:

$$f(D, \lambda) = \mathscr{L}(M_\lambda, D). \tag{2.4}$$

Considering that the agent's task is to maximize the reward, the reward function is set as the parameter response function, and depends on the data set D and the action selected, as shown below:

$$R(D, a) = -f\Big(D, \lambda = g(a)\Big). \tag{2.5}$$

The observed reward depends solely on the data set and the parameter configuration selected. Once an action is selected, a new parameter configuration is evaluated.

The transition function then generates a new state, $s' \in \mathscr{S}$, by appending the newly evaluated parameter configuration, λ, and the corresponding reward $r \in \mathscr{R}$ observed to the previous state, $s \in \mathscr{S}$:

$$s' = \tau(s, a, r) = \Big(s, (\lambda = g(a), r)\Big). \tag{2.6}$$

The agent reaches the terminal state in the case of exceeding the prescribed budget T. At each step $t \in T$, agent study the data $d \in D$, the state $s_t = (d_t, (\lambda_0, r_0), ..., (\lambda_t, r_t))$ and the next step $s_{t+1} = (d_{t+1}, (\lambda_0, r_0), ..., (\lambda_t, r_t), (\lambda_{t+1}, r_{t+1}))$. This means each state s includes all previously parameter configurations and their corresponding responses. The budget could be reached when the running time/target reward is reached or the same parameter set is selected twice in a row. The last condition causes the agent to keep on exploring the parameter space without getting stuck in a specific reward configuration.

Artificial Intelligent Agent

The agent interacts with the environment $\mathscr{E}$ with the task of maximizing the expected discounted reward. They execute actions from the action space while receiving observations

and rewards. At each time step, which ranges over a set of discrete time intervals, the agent selects an action a at state s. The behavior of the agent is governed by a stochastic policy, $\pi : \mathscr{S} \rightarrow \mathscr{A}$, which tells the agent which actions should be selected for each possible state. As a result of each action, the agent receives a scalar reward r, and observes the next state s'. The policy is used to compute the true state-action value, $Q_\pi(s,a)$, as:

$$Q_\pi(s,a) = E_\pi\Big[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\Big], \tag{2.7}$$

where $\gamma \in [0,1]$ represents the discount factor balancing between immediate and future rewards. This basically helps to avoid infinity as a reward in case the task has no terminal state.

The aim of the agent is to learn an optimal policy that defines the probability of selecting an action that maximizes the discounted cumulative reward, $\pi^*(s) \in argmax_a Q^*(s,a)$, where $Q^*(s,a)$ denotes the optimal action value. One of the most popular value-based methods for solving RL problems is the Q-Learning algorithm [32]. The basic version of the Q-Learning algorithm makes use of the Bellman equation for the Q-value function, whose unique solution is the optimal value function $Q^*(s,a)$:

$$Q^*(s,a) = E_\pi\Big[r + \gamma max_{a'} Q^*(s',a') | s_0 = s, a_0 = a\Big]. \tag{2.8}$$

Learning Mechanism

The interaction between the AI agent and the learning environment is the core mechanism for training and testing in the RL framework. Starting at a random location in the parameter space of a random data set, the agent needs to navigate the parameter response surface, including the parameter configuration and corresponding reward, of a given model. At each step, the agent explores the environment, selects the next best parameter set with the ε-greedy technique, and sends it to the environment. The policy ε-greedy(Q) to select action a_t is defined as follows:

$$a_t = \begin{cases} \sim Unif(\Lambda), & \text{if } x \sim Unif([0,1]) < \varepsilon, \\ argmax_a Q^*(s_t,a), & \text{otherwise.} \end{cases} \tag{2.9}$$

The learning environment will update the current state, evaluate it, and send feedback back to the agent as a reward. When the agent runs out of episodes, he will be relocated to the response surface of another data set. The transitions are stored in the replay memory, where a small batch of experiences is sampled to update the Q-Network. By using experience

replay, it breaks down the successive correlation among samples and also allows the network to make better use of experiences.

Algorithm 1 DRL algorithm for parameter optimization

Initialize network Q and target network $\hat{Q}$
Initialize experience replay memory B
Initialize the Agent to interact with the Environment
for N_e iterations **do** ▷ N_e - number of episodes
    Randomly sampling a data set $D = \{d_0, ..., d_T\}$, $d \sim Unif(\mathscr{D})$
    Get state $s_0 = (d_0, (\{\lambda_{init}\}^{dim(\Lambda)}, 0))$
    for $t \in \{0, ..., T\}$ and while s_t is not terminal **do**
        Determine next action a_t from state s_t using policy ε-greedy(Q) ▷ Eq. 2.9
        Receive reward $r_t = R(d_t, \lambda = g(a_t))$ ▷ Eq. 2.5
        Generate new state $s'_t = \tau(s_t, a_t, r_t)$ ▷ Eq. 2.6
        Store transition (s_t, a_t, r_t, s'_t) in the experience replay memory B
        Replace oldest tuple if $|B| > N_B$ ▷ N_B - replay buffer size
        if enough experience in B **then**
            Sample a random minibatch of N transitions from B
            for every transition (s_i, a_i, r_i, s'_i) in minibatch **do**

$$y_i = \begin{cases} r_i, & \text{if } s'_i \text{ is terminal;} \\ r_i + \gamma max_{a'} \hat{Q}(s'_i, a'), & \text{otherwise.} \end{cases}$$

            end for
            Update Q by minimizing the loss

$$\mathscr{L} = 1/K \sum_{i=0}^{K-1} \left(Q(s_i, a_i) - y_i \right)^2$$

            Copy weights from Q to $\hat{Q}$ for every N_u step
        end if
    end for
end for

Algorithm 1 can be used to describe the learning process of the agent, including the training phase and testing phase. The main purposes of the training phase include generating learning samples and training the Deep Q-Network using the DRL algorithm. During the testing phase, given an unseen scenario, the target network is used to predict an optimal parameter set. The step-by-step algorithm for training is as follows:

1. Learning Environment randomly samples a data set $D = \{d_0, ..., d_T\}$ where each data $d_t \sim Unif(\mathscr{D})$ for $t = 1, ..., T$.

2. The input for the DRL algorithm is represented as a one-hot encoded state vector, $s_0 = (d_0, (\{\lambda_{init}\}^{dim(\Lambda)}, 0))$.

3. Given the state vector s_t, a candidate action a_t is selected with the ε-greedy technique.
4. The parameter λ_t is computed. Then it is sent to the learning environment to compute the reward r_t and generate the next scenario s'_t (or s_{t+1}).
5. The sample tuples (s_t, a_t, r_t, s_{t+1}) are stored in the replay buffer for later use in training.
6. When the replay buffer has stored enough samples ($\geq$ the minimum replay buffer size, N_B), the oldest tuple will be replaced. A batch of samples is sampled randomly from the replay buffer for training.
7. The Q-Network is updated by minimizing the defined loss function, which is similar to a training-supervised learning model.
8. Finally, the target networks are updated after a preset number of steps N_u.
9. If the end of the episode is reached, the searching step will be stopped and we will go back to step 1. Else, increase $t = t + 1$ and go back to step 3.

The testing phase is relative simple since we only need to get the final optimized parameter set for given scenario. However, in practical use, the experiences generated in this phase can also be stored in replay buffer for tuning the model via batch training. This setting can help the model tuning to be faster and keep the model up-to-date with new incoming data. The step-by-step algorithm for testing is described as follows.

1. An unseen data set D from learning environment is given.
2. The state of the environment is defined as the data set D plus the history of evaluated parameter configurations and their corresponding response.
3. Given state vector, an action a_t^* is suggested.
4. The parameter λ_t^* is calculated and sent to the environment. If end of episode is reached, go to next step, else compute next state s_{i+1}, $i = i + 1$ and return to step 3.
5. Given state vector and optimal action, the Q-value, $Q^*(s, a^*)$, could be computed.
6. Finally, the Q-value along with corresponding parameter set λ^* is stored to evaluate performance.

2.5 The Trading System

In this study, we invest with an initial invested capital granted only in the beginning of the investment process. No additional money is infused during the investment period. Therefore, each trade affects the whole process. If the current trade is profitable then the capital will increase, which means the profit is used to reinvest in the next trade. On the contrary, if the current trade is not profitable, the decrease in the invested capital will also affect the following trades. The assumptions about buy and sell signals are also considered. Two consecutive buy or sell signals are not allowed, in other words each buy signal is followed by a sell signal. The number of buys and sells are equal. In addition, the trading strategy always starts with a buy order first. The advantage of this process is that it examines more firmly the capability of the trained parameters to hold on during all the investment periods. The optimization of the technical analysis strategies is proposed in the next subsection.

2.5.1 Parameter Optimization with Genetic Algorithm

We study the research results of [47] and based on the experimental results to give the values for the parameters in this section. Our GA chromosome is composed of three main parts with each part corresponding to one of the indicators to be optimized. Genes are real numbers representing the different parameters.

Table 2.1 An example of a chromosome

RSI			**BOLL**		**MACD**		
14	30	70	20	2	12	26	9

The first part consists of three genes that represent the three parameters of the RSI which are the number of days used for RSI calculation, the lower threshold and the upper threshold. The next two genes represents the two parameters of the BOLL indicator, period N and standard deviations K. The last part consists of three genes corresponding to the time periods of MACD indicator, m, n and q. The numbers presented in Table 2.1 are the typical parameters used for each indicator [15]. Each parameter under study is limited by some boundary constraints. Specifically,

RSI constraints:

- $w \in [3, 25]$
- $lower\_boundary \in [10, 40]$

- $upper\_boundary \in [60, 90]$

As above, w is selected within the range from 3 days to 25 days. In [47], the authors argued that lower threshold values are between 10 and 40 because values below 10 are rarely considered to provide buy signals. The upper threshold values are selected between 60 and 90 to avoid missing profits.

BOLL constraints:

- $N \in [20, 100]$
- $K \in [1, 3]$

where the values of period N and standard deviations K are selected from heuristics. These parameters are the well known parameters used for default settings [112]. Other ranges of values do not lead to better solutions while the cost of execution time increases if the range of values is increased.

MACD constraints:

- $q < m < n$
- $m \in [11, 20]$
- $n \in [21, 50]$
- $q \in [1, 10]$

According to our experiments, other range of values lead to poorer solutions to the data used in the study.

The optimal parameters for GA are selected through preliminary experiments. The selected parameters are presented in Table 2.2.

Table 2.2 Optimal parameters for GA

Population size	100
Number of generations	500
Selection technique	Rank based selection
Crossover probability	0.8
Mutation probability	0.05

2.5.2 Parameter Optimization with Bayesian Optimization

The *hyperopt* software package [12] in the Python programming language is used to implement Bayesian Optimization algorithm in this study. Uniform distribution is used to avoid any assumption about the domain space. In the objective function, the returned score is multiplied with negative one since the used package tries to minimize our defined objective function while our objective is to maximize conflict probability. In Bayesian Optimization, there are two terminating conditions: the convergence of maximum objective values or the maximum number of evaluations is reached. In this study, we use the maximum number of evaluations as the terminating criterion.

We formulate an optimization problem with three main ingredients:

- Domain space: The value of each parameter is selected within the range defined for the GA method (see Section 2.5.1) for comparison purposes.
- Objective function: We define a score function that indicates how well a set of parameters of the indicator. In our model, the cumulative return is used as the evaluation metric. We put a minus sign before the objective function since *hyperopt* by default is defining a function to minimize.
- Surrogate function and selection function: In this model, we follow [123] to use the Tree-structured Parzen Estimator as the surrogate function and the Expected Improvement criterion is used as the acquisition function. The terminating criterion is detected when the maximum number of evaluations is reached. In our experiment, the maximum number of evaluations is 500.

2.5.3 Parameter Optimization with Deep Reinforcement Learning

From the concepts for parameter optimization described, we proceed to build a trading system that can both give trading signals and optimize strategies automatically. To provide the trading environment, we study the state representation and present it in a form that the agent can understand. We propose a definition of an agent's parameter selection and a mapping from the act of choosing parameters for trading strategies to investment decisions. Each decision can be scored using the proposed reward function.The main components of the proposed system are described as follows.

- State representation: The trading scenario or state of environment, s_t, is represented as a one-hot encoded vector and decomposed into two parts: the data price $d_t \in D$, and the sequence of selected parameter configurations and their corresponding rewards, $(\lambda_t, r_t) \in (\Lambda \times \mathscr{R})$.

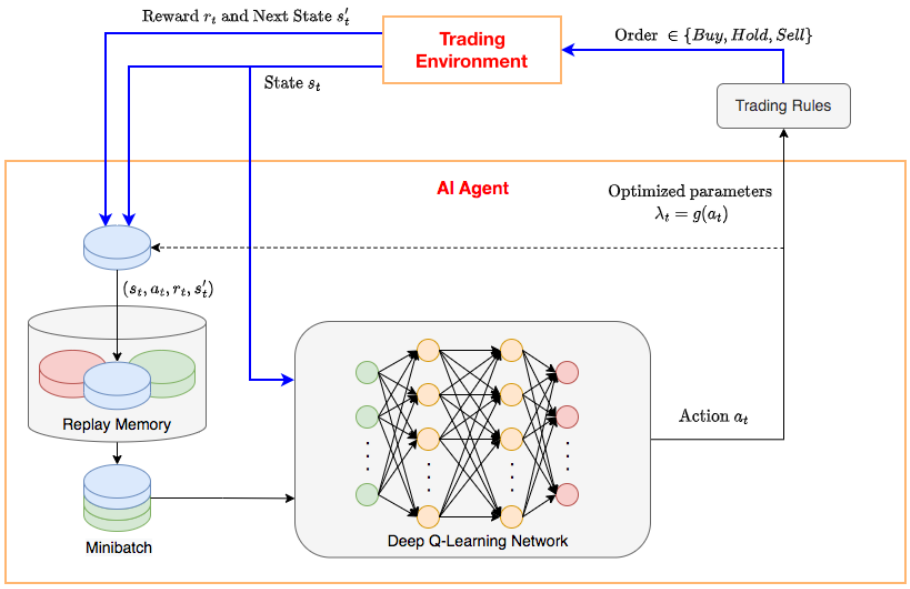


Fig. 2.4 Illustration of the learning mechanism in trading environment

- Action of agent: Given a certain state of the environment, the agent navigates the parameter response surface to select a set of parameters to optimize the reward. He then applies the chosen set of parameters to his trading strategy and executes a sequence of orders (buy, hold or sell) based on the trading rules.

- Reward evaluation: Sharpe ratio, cumulative return or a combination of them can be used as the reward function agent wants to optimize. In this work, each objective function is considered separately for comparison purposes.

- Learning mechanism: Applying the Algorithm 1, the interaction between the trader and the trading environment is presented in Figure 2.4. The agent can take a random action with probability, ε, or follows the policy that is believed to be optimal with probability, $1-\varepsilon$. An initial value for epsilon of the ε-greedy action, ε_{start}, is selected for the first observations and then is set to a new value, ε_{end}, after a number of observations. The learning process of agent can be built on a Deep Q-Learning Network or its very powerful variants like Double Deep Q-Learning Network and Dueling Double Deep Q-Learning Network. During the trading process, trader executes orders and calculates the performance through a backtesting step. Thus, an extra step is added in step 4 of both training and testing phase of the algorithm: the selected parameters are used as input of trading rules to give trading signals (buy, hold or sell).

2.6 Experiments

In this Chapter, two experiments are carried out in turn.

(i) Experiment 1: Comparing the performance of trading strategies in various markets with two approaches to parameter optimization: Genetic Algorithm and Bayesian Optimization.

(ii) Experiment 2: Testing the performance of the proposed trading system with optimized parameters in the Reinforcement Learning framework.

2.6.1 Experiment 1 Setup

For the simplified trading strategies described in this study, we carried out a set of experiments for all the trading strategies described in Section 2.3 to illustrate the effects of each strategy. Without loss of generality, we invest with an initial capital of $1,000,000, and the trading commission is assumed to be 0.1% for each executed transaction. In the proposed trading system, we perform the backtesting of each strategy against the main financial assets from 3 different markets: crypto spot (BTC-USDT), stock index (DJIA), and crypto futures (BTCUSDT perpetual contract), and the 15-minute bar data (open, high, low, and close price) of each asset mentioned above are collected. The data is publicly available at https://www.binance.com/en/landing/data, accessed on November 1, 2022.

To avoid the case where the optimization techniques are over-fitted or the obtained parameters do not yield any profit in other periods, we split the time series into training, validation, and testing periods. We use the rolling forward method to split data on crypto spot and crypto futures into seven sequences. Each sequence contains 6144 data (64 days) and is represented as an independent run. More precisely, the first 3072 data (32 days) are taken as the training period, the next 1536 data (16 days) are taken as the validation period and the last 1536 data are the testing period. For each sequence, we move all the periods 1536 data forward, which means the data in one sequence will overlap the data in the next sequence. The second half of the training period and the entire validation period constitute the next training period. The testing period in one sequence will become the validation period in the next sequence. The crypto markets tend to be available all the time, so traders can access them no matter what time of the day or week it is. On the other hand, the stock market is only open during business hours in their particular country. As a result, the amount of stock data collected during the observation period in this research is less than that of crypto data. The experiment was conducted with a small data set for the stock market case. Thereby,

the impact of data volume on experimental results is also analyzed. The method introduced above is applied with different parameter values for each market, as shown in Table 2.3.

Table 2.3 Experiment description

Asset	**Crypto spot** (BTC-USDT)	**Stock index** (DJIA)	**Crypto futures** (BTCUSDT perpetual)
Period	2022-03-25 to 2022-08-31	2022-03-25 to 2022-07-07	2022-03-25 to 2022-08-31
Total observations	15360	1847	15360
Nb. of sequences	7	7	7
Nb. of data in each sequences	6144	820	6144
Nb. of data in training period	3072	410	3072
Nb. of data in validation period	1536	205	1536
Nb. of data in testing period	1536	205	1536

The parameters are firstly optimized over the training period. The obtained result is used as an initial population for the validation period. Then the final result of the validation period is re-evaluated against the training period and arranged according to the overall average gain of both training and validation periods. The best obtained parameters that match both periods are then finally tested in the testing period.

To test the performance of the work, many benchmarks have been taken into account. Firstly, the obtained results from the optimal indicators are compared against their typical parameters. The second is the Buy and Hold (B&H) strategy, this strategy implies to buy at the start of the investment period and sell at the end. We have implemented 20 independent runs for each of the previously mentioned sequences and then tabulated the average of these runs. This process is repeated for all the indicators under study.

2.6.2 Data Analysis

The trends of various financial assets are analyzed using two techniques: the rolling means and the distribution of price increments. First, the trend of our data is visualized using rolling means on 7-day and 30-day scales. As shown in Figure 2.5, we can see that the overall trend of the 30-day rolling closing price is decreasing over time, which indicates that the markets are in a major downtrend over a large time frame. In a smaller time frame, such as the 7-day rolling closing price, the market shows signs of slight recovery: from early July to mid-August for the crypto market, and from mid-May to early June for the stock market.

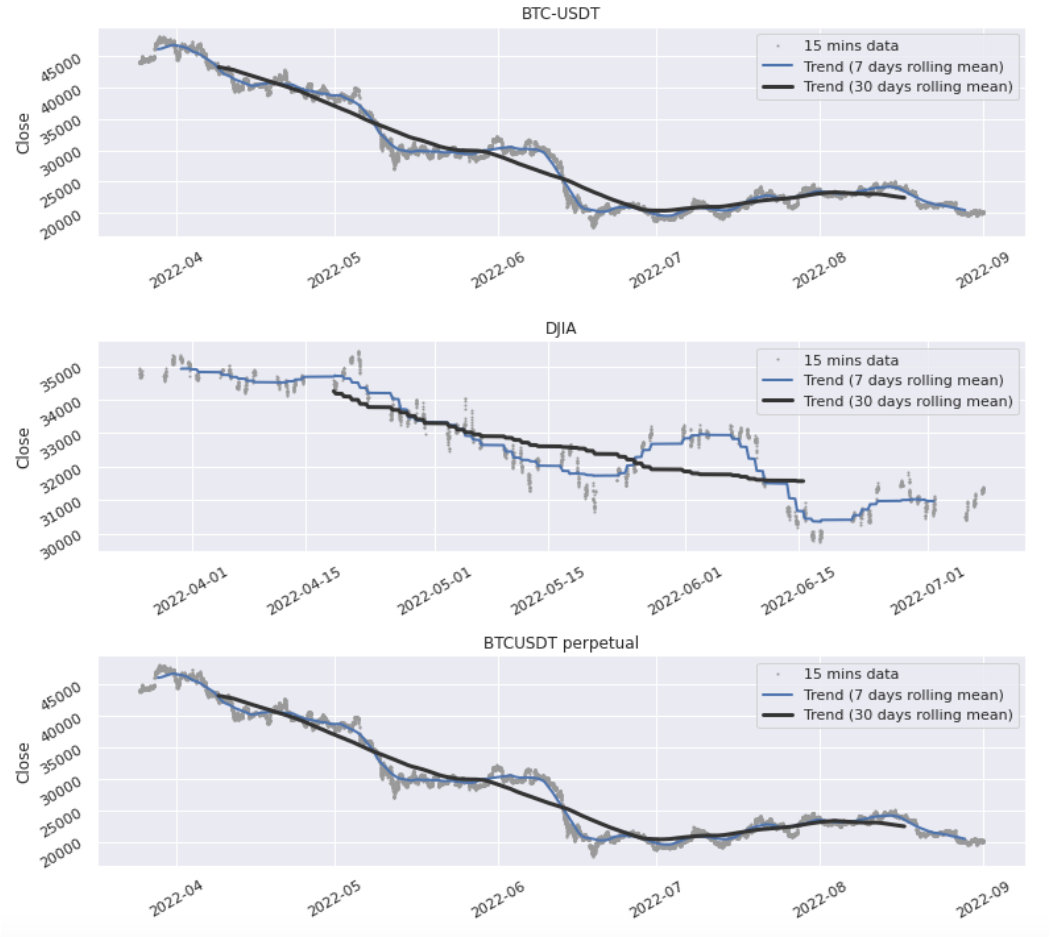


Fig. 2.5 Trend analysis for various market

Beside the rolling means, the median values can also be useful for exploratory analysis of the data set. The distribution of price increments for each weekday is plotted in Figure 2.6. Note that the stock market does not open on Saturday and Sunday, while the crypto market is active all the time. The fluctuation range of all trading days is large, indicating that the seasonal stability is not good. This is consistent with the strong downtrend results of the markets given in the trend analysis step. A good result is that the data does not contain outliers, so we can skip the outlier detection step when preprocessing the data. The median values from Saturday to Tuesday suggest that the crypto market is likely to fall during this time period. Wednesday to Friday is the time when the market goes up again. Saturday's data is marked by high volatility and the market tends to decrease on this day, so traders can build a strategy to buy on Wednesdays and sell on Saturdays. As for the stock market, the daily median statistics show that prices are more volatile on Thursday because this is the end of a week's trading session. On Wednesday, the median value was shifted down lightly and

started to rise again after Thursday; however, the volatility was small. On the contrary, there are strong fluctuations in the minimum and maximum values of the weekdays, which traders can rely on to execute intraday trades.

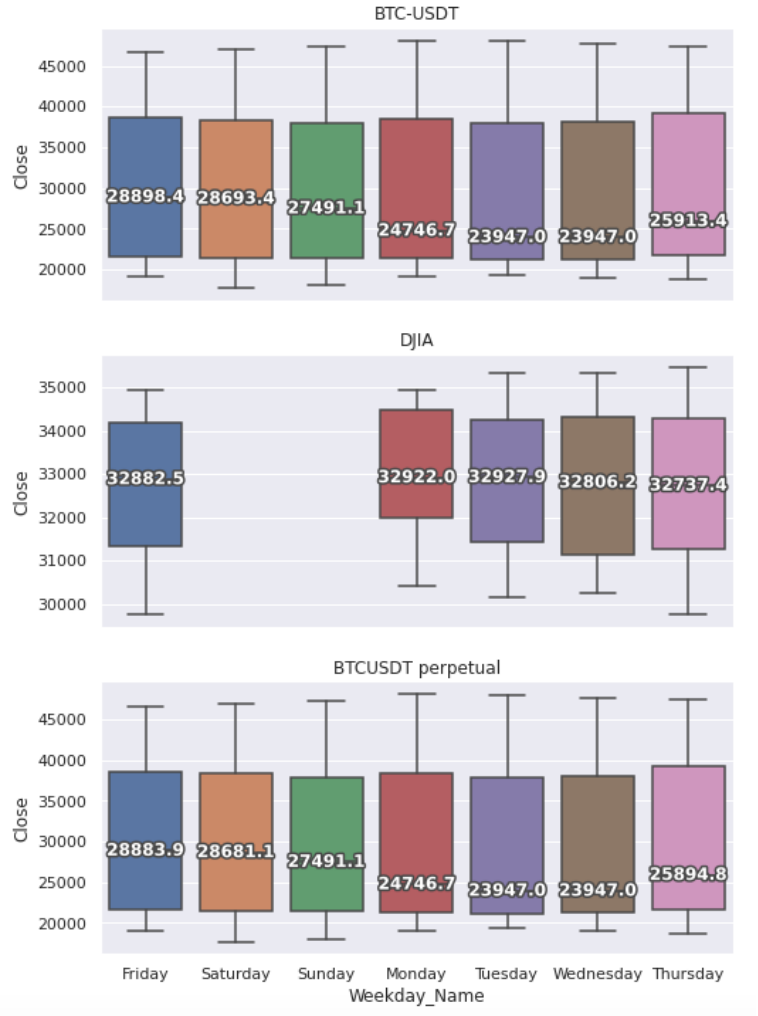


Fig. 2.6 Average price increment ranges by weekday

We also look deeper into shorter cycles of the day, for example hourly cycles, as shown in Figure 2.7. According to the price range of stock data, the volatility by hour is not large, and the median values do not clearly show an upward or downward trend. Therefore, it cannot be concluded that trading is more intensive at some particular hour of the day. For cryptographic data, it is easy to see that the largest range is between 00:00 and 01:00. A slight decrease can be seen in the period after 01:00, which is suitable for a short position.

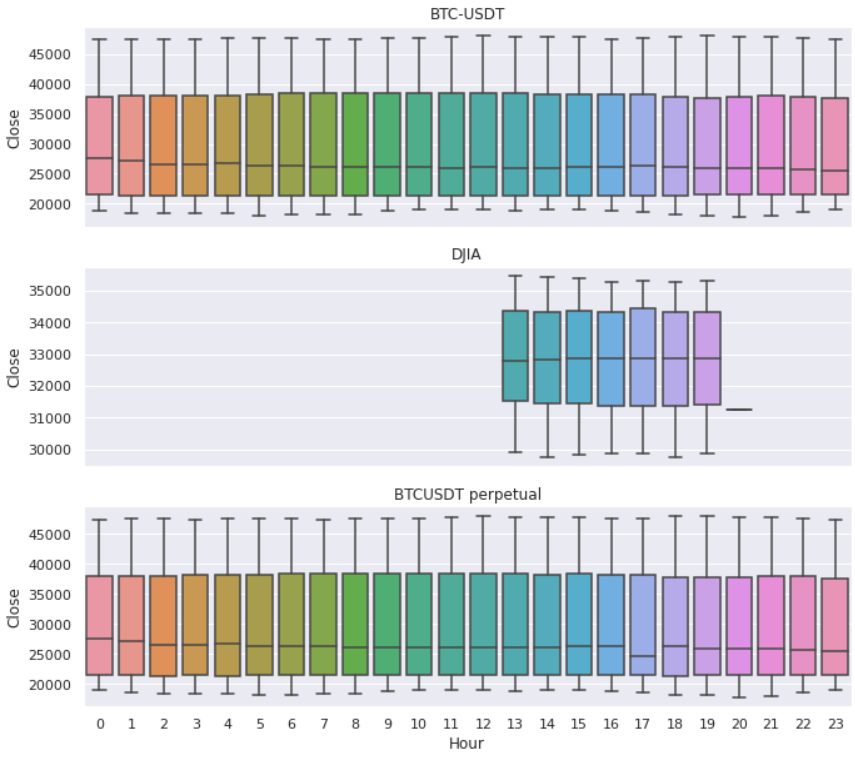


Fig. 2.7 Average price increment ranges by hour (UTC+0)

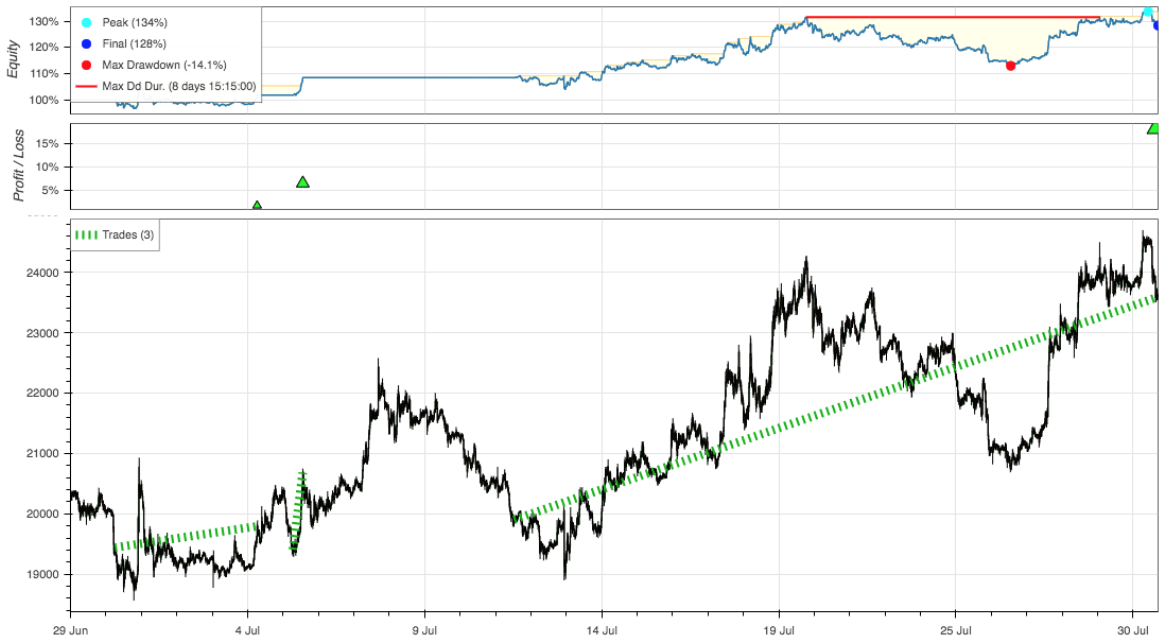
The results obtained can provide a useful overview of the underlying conditions for making trading decisions. In addition, these additional features can be integrated into the trading system to execute more buy or sell trades during certain days or hours.

2.6.3 Results and Discussion

To evaluate the performance of the proposed strategies, experiments are performed and can be summarized based on each market. The results shown in the following tables are the average of average returns for all the training, validation, and testing periods across all indicators.

Results for Experiment 1.1

An example of the backtesting results of a trading strategy based on the RSI indicator is shown in Figure 2.8. The typical RSI parameters are: $w = 14$, $lower\_boundary = 30$, $upper\_boundary = 70$. The optimized parameters from BO are: $w = 10$, $lower\_boundary = 13$, $upper\_boundary = 90$. In this example, the optimal return is 28.36% and the optimal Sharpe ratio is 1.43 while the typical return is -2.47% and the typical Sharpe ratio is -0.12.



	Open	High	Low	Close	Volume	position
2022-06-30 07:15:00+00:00	19374.0	19461.4	19300.0	19421.3	13010.064	1.0
2022-07-04 11:45:00+00:00	19627.1	19838.0	19572.8	19794.2	26845.865	-1.0
2022-07-05 13:00:00+00:00	19364.4	19407.9	19300.0	19386.7	12591.779	1.0
2022-07-05 20:00:00+00:00	20392.9	20749.2	20349.7	20682.7	29848.806	-1.0
2022-07-12 01:00:00+00:00	19805.1	19885.0	19796.0	19882.3	5292.679	1.0

Fig. 2.8 Backtesting results of a trading strategy based on the optimized RSI indicator in the cryptocurrency market

Table 2.4 shows the results of the experiment with the crypto data. In the table, the optimized RSI from both BO and GA methods can beat the typical RSI and the B&H strategy in terms of return for the training and validation periods. However, the optimized RSI from GA and BO can not beat the typical RSI in testing periods. In the testing period, RSI_{GA} provides the return of -4.81%, RSI_{BO} provides the return of -4.86% while the typical RSI return is -2.71% and the B&H return is -5.78%. In [99], the authors argued that positive returns with training data do not necessarily match positive returns with testing data. Therefore, we consider another measure to evaluate the results. The average standard deviation performance of the RSI indicator is shown in Table 2.5.

Table 2.4 Average returns (%) of the RSI indicator in the cryptocurrency market

Period	Method	Optimal	Typical	B&H
Training	GA	8.17	-11.87	-19.57
	BO	13.61		
Validation	GA	3.94	-4.47	-6.32
	BO	4.54		
Testing	GA	-4.81	-2.71	-5.78
	BO	-4.86		

In Table 2.5, we can see the decreasing volatility over the three periods for the two approaches GA and BO. Furthermore the sum return from validation and testing periods also showed that BO approach outperforms the others.

Table 2.5 Standard deviation performance of the RSI indicator in the crypto market

Period	Method	Optimal	Typical	B&H
Training	GA	12.97	8.99	12.43
	BO	13.83		
Validation	GA	6.03	8.48	16.56
	BO	7.56		
Testing	GA	4.54	7.44	15.99
	BO	4.83		

An example of the backtesting results of a trading strategy based on the BOLL indicator is shown in Figure 2.9 and Figure 2.10.

Fig. 2.9 Trading signals with BOLL indicator

	Open	High	Low	Close	Volume	position
2022-06-29 13:30:00+00:00	20110.6	20135.2	19904.3	19931.8	17564.664	1.0
2022-06-29 19:45:00+00:00	20121.7	20270.1	20085.1	20214.1	10558.570	-1.0
2022-06-30 06:15:00+00:00	19977.1	19982.8	19700.0	19776.6	13061.781	1.0
2022-06-30 23:45:00+00:00	19053.0	19965.6	19044.0	19923.5	63531.651	-1.0
2022-07-01 10:45:00+00:00	19139.8	19170.0	18955.3	19062.5	17345.811	1.0
...	...	...	...	...	...	...

Fig. 2.10 Backtesting results of a trading strategy based on the optimized BOLL indicator in the cryptocurrency market

In this example, the typical BOLL parameters are: $N = 21$, $K = 2$ and the optimized parameters from BO are: $N = 30$, $K = 2$. Both configurations are profitable, however, the trading strategy using optimized BOLL indicator provides a higher profit. In particular, the optimal return is 27.42% and the optimal Sharpe ratio is 1.90 while the typical return is 21.63% and the typical Sharpe ratio is 1.54.

Table 2.6 Average returns (%) of the BOLL indicator in the cryptocurrency market

Period	Method	Optimal	Typical	B&H
Training	GA	-4.04	-11.83	-19.57
	BO	8.21		
Validation	GA	-0.34	-1.31	-6.32
	BO	0.77		
Testing	GA	-3.40	1.09	-5.78
	BO	0.75		

As seen from Table 2.6, $BOLL_{GA}$ and $BOLL_{BO}$ provide the greater returns over the B&H strategy in all periods. For example in the testing period, the $BOLL_{GA}$ return is -3.40% and $BOLL_{BO}$ return is 0.75% while the B&H return is -5.78%. Although they can not beat the typical BOLL in testing period, the BO method provides positive returns in all three periods, average return against a typical BOLL during the test period is comparable. With higher profits, the optimization method from BO shows better performance than the one from GA. The positive sum return from validation and testing periods also show that BO is the best approach to optimize BOLL in the crypto market. Since all the results are negative, it can be concluded that the BOLL indicator with GA optimization is not good for the crypto market over the considered period.

Table 2.7 Standard deviation performance of the BOLL indicator in the crypto market

Period	Method	Optimal	Typical	B&H
Training	GA	18.11	18.05	12.43
	BO	9.38		
Validation	GA	11.43	13.75	16.56
	BO	7.29		
Testing	GA	8.50	10.52	15.99
	BO	4.03		

As shown in Table 2.7, both $BOLL_{GA}$ and $BOLL_{BO}$ provide better results with less volatility than the typical BOLL and the B&H strategy over all periods. $BOLL_{BO}$ can beat $BOLL_{GA}$ in all periods.

	Open	High	Low	Close	Volume	position
2022-06-29 15:15:00+00:00	20058.7	20081.2	19970.0	20054.2	7917.093	1.0
2022-06-29 17:00:00+00:00	20012.7	20042.0	19937.3	19957.9	5801.958	-1.0
2022-06-29 17:45:00+00:00	20028.5	20078.6	20003.1	20078.0	4204.484	1.0
2022-06-29 23:15:00+00:00	20116.9	20132.3	20062.5	20078.1	4414.681	-1.0
2022-06-30 12:30:00+00:00	19135.1	19294.0	19132.1	19243.9	19403.296	1.0
...	...	...	...	...	...	...

Fig. 2.11 Backtesting results of a trading strategy based on the optimized MACD indicator in the cryptocurrency market

Next, Figure 2.11 shows an example of the backtesting results of a trading strategy based on the MACD indicator. In this example, the typical parameters are: $m = 12$, $n = 26$, $q = 9$ and the optimized parameters from BO are: $m = 20$, $n = 53$, $q = 10$. The optimal return is positive, 9.33%, and the optimal Sharpe ratio is 0.65 while the typical return is negative, -1.02%, and the typical Sharpe ratio is 0.02.

Table 2.8 Average returns (%) of the MACD indicator in the cryptocurrency market

Period	Method	Optimal	Typical	B&H
Training	GA	-15.05	-17.41	-19.57
	BO	1.33		
Validation	GA	-6.81	-7.14	-6.32
	BO	-0.21		
Testing	GA	-5.28	-6.00	-5.78
	BO	-2.65		

As shown in Table 2.8, both $MACD_{GA}$ and $MACD_{BO}$ provide the greater returns over the typical MACD in all periods. Even though the average results are negative, the optimized MACD from BO still outperforms the market. The results show that doing too many trades (see Figure 2.11) in a bearish market does not bring good results. Losses due to transaction costs are one cause of negative returns. MACD gives lower performance when compared to other indicators in the experiment. Therefore, it can be concluded that MACD is not suitable for a crypto market with a major downtrend. Similar results are shown in Table 2.9 where the methods provide high volatility over the testing period.

Table 2.9 Standard deviation performance of the MACD indicator in the crypto market

Period	Method	Optimal	Typical	B&H
Training	GA	13.57	11.27	12.43
	BO	3.53		
Validation	GA	10.24	9.81	16.56
	BO	0.55		
Testing	GA	6.62	7.33	15.99
	BO	7.01		

Results for Experiment 1.2

Table 2.10 shows the results of the experiment with the stock data. In the table, RSI_{BO} provides positive returns for all the 3 periods. The optimized RSI from BO outperforms the market in terms of average returns. RSI_{GA} can also beat the typical RSI the B&H strategy in training and validation periods, RSI_{GA} returns are 1.98% and 0.14% while the returns from

typical RSI and B&H strategy are negative. For the testing period, RSI_{GA} return is -0.35%, while return of typical RSI is -0.17% and B&H return is -1.22%.

Table 2.10 Average returns (%) of the RSI indicator in the stock market

Period	Method	Optimal	Typical	B&H
Training	GA	1.98	-0.45	-3.04
	BO	3.70		
Validation	GA	0.14	-0.52	-1.46
	BO	1.94		
Testing	GA	-0.35	-0.17	-1.22
	BO	0.03		

Table 2.11 Standard deviation performance of the RSI indicator in the stock market

Period	Method	Optimal	Typical	B&H
Training	GA	2.04	1.97	4.07
	BO	2.32		
Validation	GA	1.84	1.78	5.45
	BO	1.55		
Testing	GA	3.44	1.38	5.47
	BO	3.45		

In Table 2.12, the optimized BOLL from BO method outperforms the market in terms of average returns for all periods. Similar to the results of the RSI indicator, $BOLL_{GA}$ also provides the negative return in the testing period. The optimized BOLL from GA method can not beat the typical BOLL in the testing period. The $BOLL_{GA}$ return is -0.22% while the typical BOLL return is 0.14%.

Table 2.12 Average returns (%) of the BOLL indicator in the stock market

Period	Method	Optimal	Typical	B&H
Training	GA	0.53	-0.76	-3.04
	BO	1.02		
Validation	GA	0.96	-0.36	-1.46
	BO	-0.13		
Testing	GA	-0.22	0.14	-1.22
	BO	0.34		

Table 2.13 Standard deviation performance of the BOLL indicator in the stock market

Period	Method	Optimal	Typical	B&H
Training	GA	2.91	1.90	4.07
	BO	1.11		
Validation	GA	2.46	3.23	5.45
	BO	1.96		
Testing	GA	2.76	3.17	5.47
	BO	1.41		

In Table 2.14, the optimized MACD from BO, $MACD_{BO}$, provides higher returns when compared with $MACD_{GA}$, typical MACD and the B&H strategy for all periods. All of returns are negatives in validation and testing periods, however, this makes sense because in a downtrend market (see Figure 2.5), trying to trade with a typical strategy is rarely profitable. Instead, traders often stay out of the market to avoid risk and preserve their capital. The results also indicate that the MACD is not good compared to the BOLL and the RSI for the stock market.

Table 2.14 Average returns (%) of the MACD indicator in the stock market

Period	Method	Optimal	Typical	B&H
Training	GA	-0.71	-1.58	-3.04
	BO	0.68		
Validation	GA	-0.13	-0.97	-1.46
	BO	-0.10		
Testing	GA	-0.89	-1.05	-1.22
	BO	-0.35		

Table 2.15 Standard deviation performance of the MACD indicator in the stock market

Period	Method	Optimal	Typical	B&H
Training	GA	3.42	3.05	4.07
	BO	1.22		
Validation	GA	3.09	2.71	5.45
	BO	0.23		
Testing	GA	2.97	2.74	5.47
	BO	1.31		

Results for Experiment 1.3

Table 2.16 shows the results of the experiment with the crypto futures data. As shown in the table, the optimized RSI from BO approach, RSI_{BO}, provides positive returns for all the 3 periods and better than RSI_{GA} in training and testing period. RSI_{BO} and RSI_{GA} can beat their typical form and the B&H strategy for all the 3 periods.

Table 2.16 Average returns (%) of the RSI indicator in the crypto futures market

Period	Method	Optimal	Typical	B&H
Training	GA	13.18	-11.86	-19.32
	BO	14.10		
Validation	GA	2.30	-4.55	-6.06
	BO	1.67		
Testing	GA	-0.65	-2.92	-5.52
	BO	0.75		

Table 2.17 Standard deviation performance of the RSI indicator in the crypto futures market

Period	Method	Optimal	Typical	B&H
Training	GA	10.41	8.92	12.61
	BO	9.52		
Validation	GA	10.23	8.56	16.52
	BO	9.26		
Testing	GA	7.28	7.77	15.94
	BO	5.58		

As shown in Table 2.17, RSI_{BO} can beat all opponents in terms of standard deviation in testing period. In training and validation period, RSI_{BO} shows comparable results. RSI_{GA} also provides better results when compared to the B&H strategy.

Table 2.18 Average returns (%) of the BOLL indicator in the crypto futures market

Period	Method	Optimal	Typical	B&H
Training	GA	-2.18	-12.42	-0.09
	BO	-2.33		
Validation	GA	-2.79	-1.63	-0.04
	BO	1.90		
Testing	GA	-1.64	0.80	-0.01
	BO	0.19		

As seen from Table 2.18, $BOLL_{GA}$ and $BOLL_{BO}$ provide the greater returns over the B&H strategy in the training, validation and testing periods. However, they can not beat the typical BOLL in the testing period. The BO method provides a better performance when compared with the GA method. The return statistics are negative during the training period but positive during the validation and testing periods, so other metrics need to be considered to confirm the results. The standard deviation performance is shown in Table 2.19. The results show that the BO approach to optimizing the BOLL indicator has lower volatility than other approaches.

Table 2.19 Standard deviation performance of the BOLL indicator in the crypto futures market

Period	**Method**	**Optimal**	**Typical**	**B&H**
Training	GA	10.41	18.12	12.61
	BO	14.31		
Validation	GA	10.23	13.78	16.52
	BO	11.81		
Testing	GA	7.28	10.60	15.94
	BO	10.36		

Table 2.20 Average returns (%) of the MACD indicator in the crypto futures market

Period	**Method**	**Optimal**	**Typical**	**B&H**
Training	GA	-15.13	-16.93	-19.32
	BO	1.45		
Validation	GA	-5.22	-6.85	-6.06
	BO	-0.71		
Testing	GA	-6.76	-5.70	-5.52
	BO	-2.32		

Table 2.20 shows negative returns for all parameter sets with MACD indicator, which means that MACD trading strategies are not as effective as RSI trading strategies. Despite providing negative returns, $MACD_{BO}$ can beat other approaches in terms of returns in all three periods. In Table 2.21, the average standard deviations of the MACD indicator from BO approach can beat GA approach, typical parameters and also B&H strategy.

Table 2.21 Standard deviation performance of the MACD indicator in the crypto futures market

Period	Method	Optimal	Typical	B&H
Training	GA	15.22	11.47	12.61
	BO	3.83		
Validation	GA	11.42	9.79	16.52
	BO	1.87		
Testing	GA	7.14	7.29	15.94
	BO	6.15		

Execution Time Comparison

As mentioned in the experiment setup, each trading strategy is repeated 20 times to count the execution time more accurately. The average execution time for trading strategies in each market are shown in Table 2.22.

Table 2.22 Execution time (minutes) for different optimization methods

Market	Method	Indicator	Execution time	Total time
Crypto spot	GA	RSI	144.7	363.5
		BOLL	141.9	
		MACD	76.9	
	BO	RSI	21.2	56.8
		BOLL	20.3	
		MACD	15.3	
Stock index	GA	RSI	78.3	194.6
		BOLL	69.0	
		MACD	47.3	
	BO	RSI	13.5	38.3
		BOLL	11.6	
		MACD	13.2	
Crypto futures	GA	RSI	141.4	370.9
		BOLL	137.6	
		MACD	91.9	
	BO	RSI	15.7	55.1
		BOLL	21.0	
		MACD	18.4	

In the above table, the average execution time of the BO method is 50.0 minutes while the GA method takes 310 minutes, which shows that the BO method outperforms the GA method with 6.2 times less execution time on average. This result makes it possible to research and analyze complex trading strategies against multiple assets and multiple markets.

Summary of Experiment 1

This experiment presents two approaches, Genetic Algorithm and Bayesian Optimization, to parameter optimization for an automated trading system in different assets: BTC-USDT, DJIA, BTCUSDT perpetual contracts. The three indicators considered are RSI, BOLL and MACD. Results show that both approaches give better results than the Buy and Hold strategy. When compared to the typical parameters, the BO approach has less volatility and provides better returns. The Bayesian Optimization approach can beat the Genetic Algorithm approach in terms of return and execution time. Among the indicators being studied, the optimized parameter sets of RSI and BOLL from the Bayesian Optimization approach are better than MACD with lower volatility and greater returns. The results can be used to create a stable trading strategy for long-term trading.

2.6.4 Experiment 2 Setup

In this experiment, we consider the 15-minute historical data (open, high, low, and close price) of BTC-USDT from the 25th of March 2022 to the 31st of August 2022 (15360 observations). The data is publicly available at https://www.binance.com/en/landing/data, accessed on November 1, 2022.

To avoid the case where the optimization techniques are over-fitted or the obtained parameters do not yield any profit in other periods, we consider 100 periods, with the size of each period being 3456 observations (36 days). For each period, the start date and end date are different; the first 80% of the dataset is dedicated for training purposes, and the remaining 20% is used for testing the performance. This ratio is chosen according to the Pareto principle, which is commonly applied to optimization efforts in computer science [56]. Without losing generality, other ratios can also be applied.

Table 2.23 Parameters for training the AI agent

Parameters	Description	Value
N_e	Number of episodes	100
N_B	Max capacity of replay memory	10000
$batch\_size$	Batch size	40
N_u	Period of Q target network updates	10
γ	Discount factor for future rewards	0.98
ε_{start}	Initial value for epsilon of the ε-greedy	1
ε_{end}	Final value for epsilon of the ε-greedy	0.12
$learning\_rate$	Learning rate of ADAM optimizer	0.001

The parameters used for training the agent are shown in Table 2.23. The agent is assumed to start with an initial capital of \$1,000,000 and a cost of 0.1% is applied to each executed transaction for a more realistic study. The values of other parameters in the system are selected from practice and from previous studies [88, 95, 75]. In our case, future rewards play a more important role because of the volatility of the price, so in this study, γ is assigned a value of 0.98. An initial $\varepsilon_{start} = 1$ is selected for the first observations, and then the value is set to a new value $\varepsilon_{end} = 0.12$ after 300 observations ($\varepsilon_{step} = 300$). Studying the experiment in [88], we apply the ADAM algorithm to optimize the weights because of its simplicity in implementation, computational efficiency, and low memory requirements. This algorithm is suitable for large data and parameter problems when compared to other stochastic optimization methods such as RMSProp [132], AdaGrad [39]. Similarly, the Mean Squared Error is used as the loss function for simplicity, and the activation function is set as the Leaky Rectified Linear Units function because of the additional gains in final system performance relative to more commonly used sigmoidal nonlinearities [95]. The learning process of the agent is powered by the Double Deep Q-Network (DDQN) and Dueling Double Deep Q-Network (D-DDQN). In both cases, the networks are composed of 2 Convolutional Neural Network (CNN) layers with 120 neurons each. In the case of D-DDQN, CNN layers are followed by two streams of fully connected layers: the first with 60 neurons dedicated to estimate the value function and the second with 60 neurons to estimate the advantage function. We compare the results of the system against two objective functions: the cumulative return and the Sharpe ratio. The BO setting is used as a benchmark to compare with the DRL setting, so the design is different from Experiment 1. Specifically, 100 different data sets are trained to choose the best set of parameters, and then the optimal set is applied to 100 different testing data sets to evaluate performance.

Three evaluation metrics are introduced to evaluate our results. The first metric is the average reward, which is the average of daily returns over the experimental period. The second metric is the average standard deviation of daily returns. The third metric is the total cumulative reward, which is the total return at the end of the trading episode. The results from the metrics are discussed together to choose the best configuration for the proposed trading system.

2.6.5 Results and Discussion

DDQN and D-DDQN Comparision

The results with different settings are presented in the tables and figures below. First, the trading system with the cumulative return reward function is considered. In Figure 2.12, average returns in percentage over the training data sets are reported, and average returns over testing sets with different start dates and end dates are plotted in Figure 2.13. The DDQN setting can beat the D-DDQN setting in terms of return for all the periods, for example in the training period, DDQN provides the maximum return of 3.95% and maximum loss of -3.48% while the D-DDQN return is 2.87% and loss is -3.61%.

Fig. 2.12 Average returns from DRL approach with return reward function in training period

More specifically, Table 2.24 shows statistical results where the reward function is cumulative return. Comparing with D-DDQN setting, the system based on DDQN achieves higher average returns in both training and testing period. However, the standard deviation is also larger, which indicates the instability of the results when trading with short-term periods. In the real market, trading performance is evaluated by the profit achieved after a larger period of time, e.g. weekly or monthly, while this experiment focus on the daily profit and

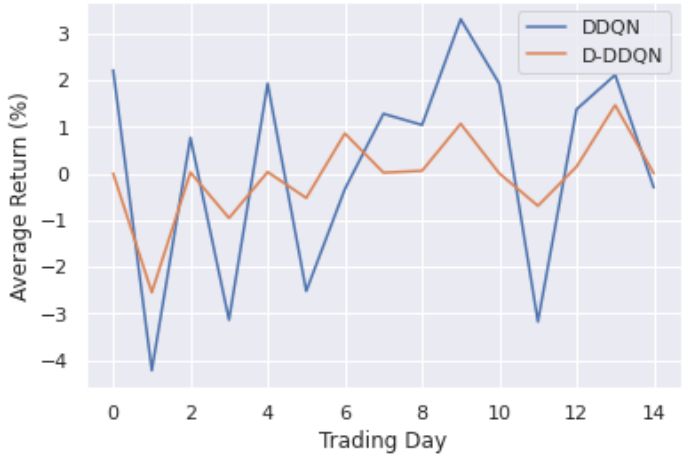


Fig. 2.13 Average returns from DRL approach with return reward function in testing period

thus, high volatility is acceptable. In future work, the system could consider different time intervals to compare the stability of the profit achieved.

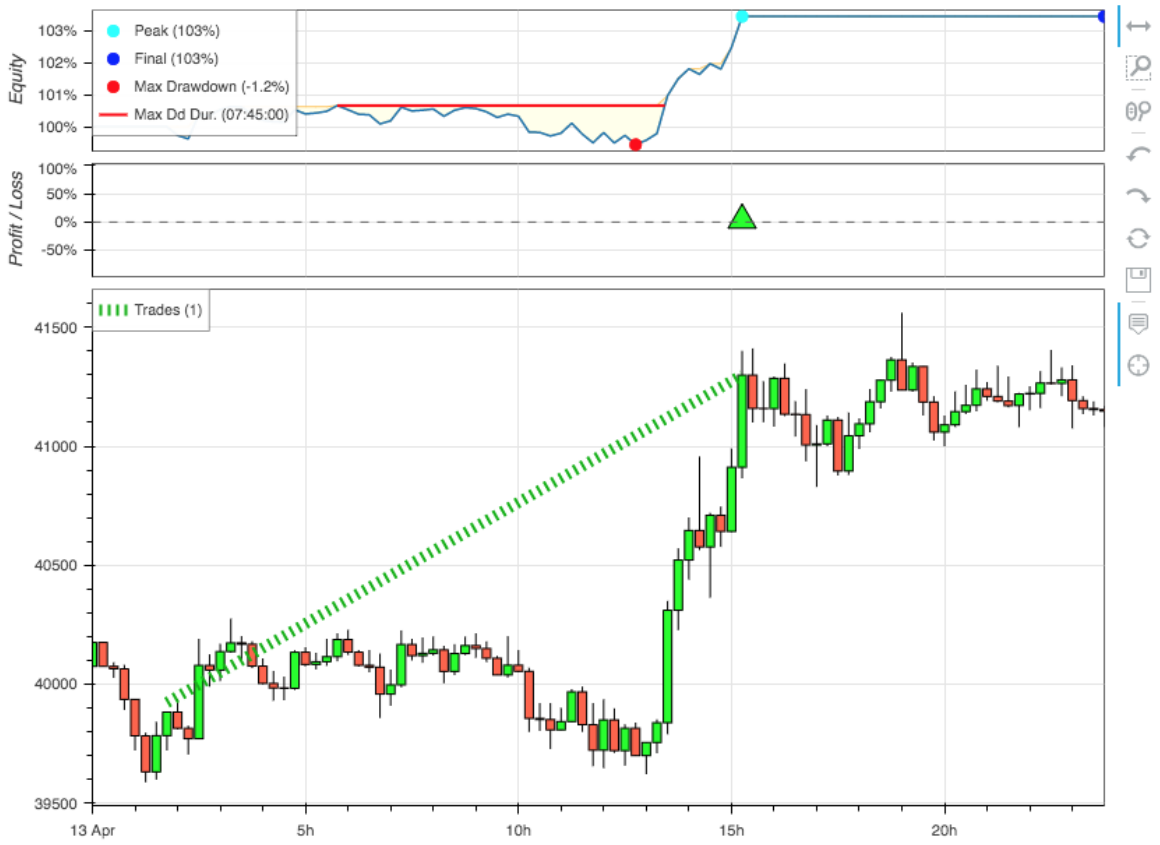
Table 2.24 Average performance with return reward function

Period	Setting	Avg. Return (%)	Max Return (%)	Min Return (%)	SD.
Training	DDQN	0.31	3.95	-3.48	1.39
	D-DDQN	0.06	2.87	-3.61	1.10
Testing	DDQN	0.15	3.30	-4.22	2.26
	D-DDQN	-0.07	1.46	-2.55	0.90

An example of backtesting results and trading details of the trading system in a trading day is shown in Figure 2.14.

To see more clearly the performance of the D-DDQN setting compared with the DDQN setting, we consider the cumulative average returns in all periods. Figure 2.15 shows the cumulative average return achieved when trading for the entire training period and the performance for the entire testing period is plotted in Figure 2.16. The returns of DDQN in the training and testing periods are 6.26% (8.94% per month) and 2.23% (4.46% per month) while the returns of D-DDQN are 1.14% (1.63% per month) and -1.08% (-2.16% per month), respectively. The results show that DDQN setting has better performance than D-DDQN setting with the return reward function.

Next, the trading system with the Sharpe reward function is considered. In Figure 2.17 and Figure 2.18, the average Sharpe value over all the periods is plotted and statistical



	Open	High	Low	Close	Volume	position
2022-04-13 01:45:00+00:00	39783.92	39883.16	39722.36	39882.90	243.66961	1.0
2022-04-13 15:15:00+00:00	40911.78	41400.00	40866.97	41298.81	2091.13220	-1.0

Fig. 2.14 Backtesting results of the trading system with DDQN setting

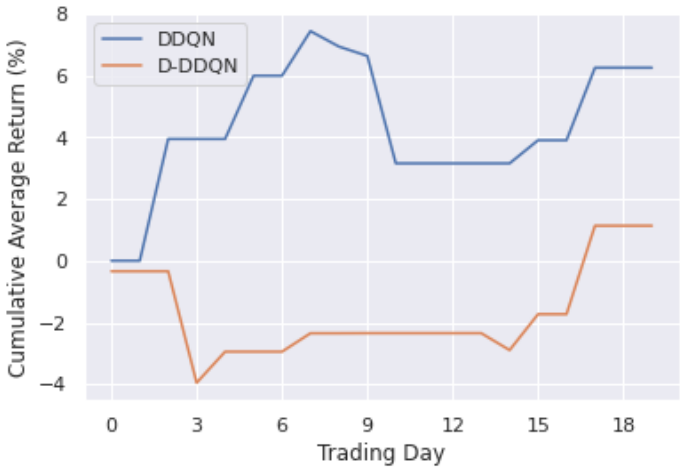


Fig. 2.15 Cumulative average returns from DRL approach with return reward function in training period

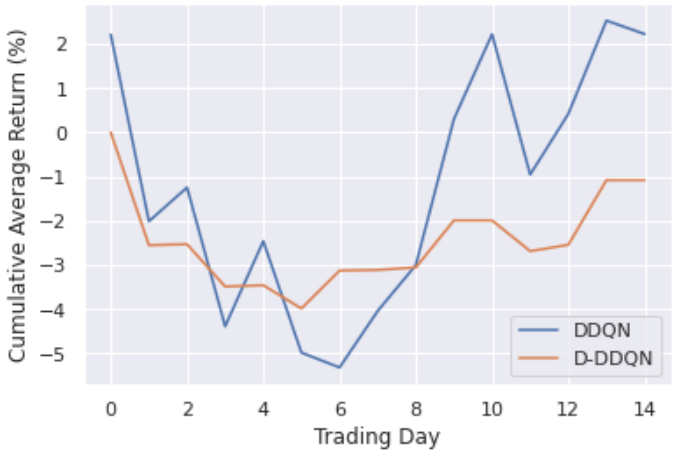


Fig. 2.16 Cumulative average returns from DRL approach with return reward function in testing period

indicators are summarized in Table 2.25. Although D-DDQN setting provides better average returns than DDQN over the training period, other statistical indicators all show that DDQN provides better performance. Furthermore, the DDQN setting provides positive returns and less volatility in all periods.

Fig. 2.17 Average returns from DRL approach with Sharpe reward function in training period

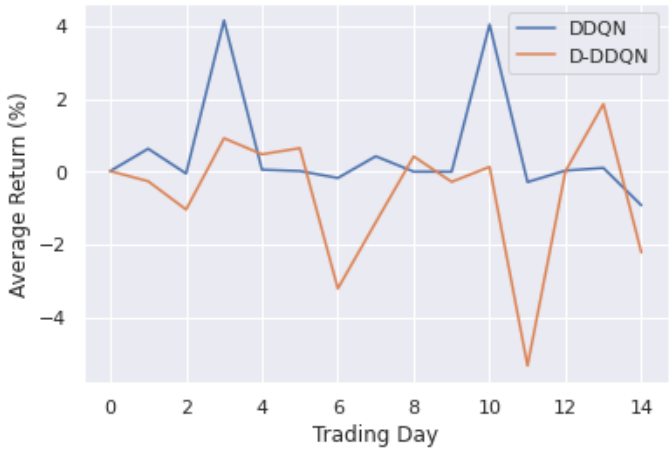


Fig. 2.18 Average returns from DRL approach with Sharpe reward function in testing period

Table 2.25 Average performance with Sharpe reward function

Period	Setting	Avg. Return (%)	Max Return (%)	Min Return (%)	SD.
Training	DDQN	0.05	2.58	-1.34	0.64
	D-DDQN	0.12	1.24	-1.93	0.58
Testing	DDQN	0.53	4.15	-0.93	1.44
	D-DDQN	-0.62	1.8	-5.34	1.76

Figure 2.19 and Figure 2.20 report the cumulative average returns over the entire training and testing periods, respectively. The returns of DDQN in the training and testing periods are 1.08% (1.54% per month) and 7.98% (15.96% per month) while the returns of D-DDQN are 2.60% (3.71% per month) and -9.32% (-18.64% per month), respectively. We can see strong fluctuations in the returns of D-DDQN setting during training and testing period, whereas DDQN setting provides positive returns in both periods.

From the preliminary analyzes above, the DDQN setting with Sharpe ratio as the reward function proved to be the best Q-Learning trading system, this result is consistent with the study of the authors in [88].

DRL and Bayesian Optimization Comparision

As a benchmark for comparison, the performance of the trading system applying Bayesian Optimization to optimize the strategy is presented in the following figures and tables. The cumulative average return with the return fitness function over 100 testing periods is shown

Fig. 2.19 Cumulative average returns from DRL approach with Sharpe reward function in training period

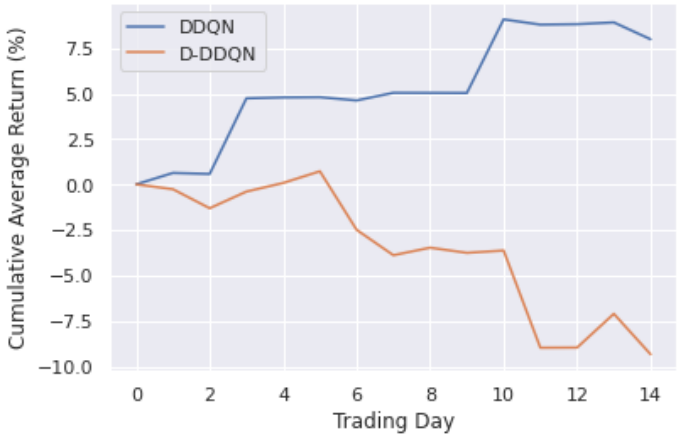


Fig. 2.20 Cumulative average returns from DRL approach with Sharpe reward function in testing period

in Figure 2.21 with a positive return, 1.38%. When compared with the DRL approach, the DDQN setting provides a higher cumulative return (see Figure 2.15 and Figure 2.16).

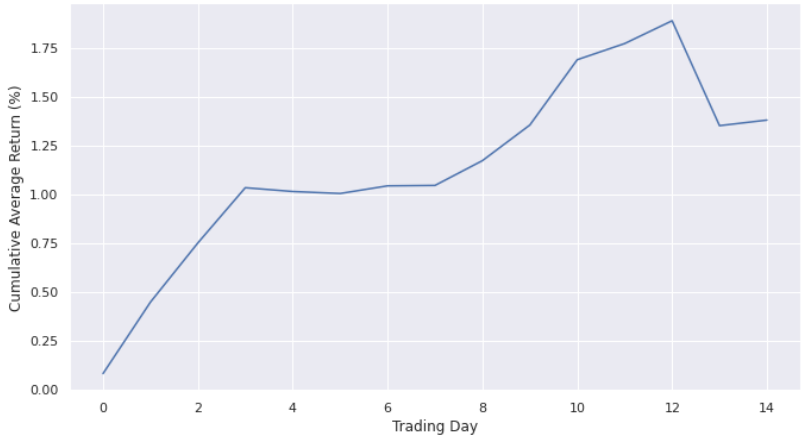


Fig. 2.21 Cumulative average return performance from BO approach in testing period

Next, Table 2.26 summarizes the performance of BO approach over 100 different testing periods, the results show that average return is positive. The highest return is 28.88% and the worse result is -22.45%, which is a big difference indicating the instability of the trading results. The cause of the problem is using only a large data set of the past for training, which is suitable for long-term trading purposes. To solve the problem, the system can regularly update the optimal set of parameters through the rolling training data set which is consistent with the definition of the DRL approach. Despite using a large data set of the past for training, the system with the DRL setting divides the data into states with each state corresponding to data of a trading day. This means the system takes in new information and updates it to make the best decisions every day. Without loss of generality, the system can be changed to smaller intervals for high frequency purpose.

Table 2.26 Statistics of average return performance from BO approach

Avg. Return (%)	**Max Return (%)**	**Min Return (%)**	**SD.**
1.61	28.88	-22.45	9.84

Execution Time Comparison

Finally, the average execution time for trading system with different approaches is shown in Table 2.27.

Table 2.27 Execution time for different optimization methods

Method	Execution time (seconds)
DDQN	6285
D-DDQN	492
BO	1435

The DRL approach with D-DDQN setting has the shortest execution time. DDQN setting provides smaller volatility than BO approach but it takes longer time to execute. Although DDQN setting could not beat BO in terms of running time in this experiment, the DRL approach still has the potential to outperform BO in practice by looking at better settings with recurrent neural network. When trading in the real market, traders need to select or combine multiple trading strategies, which creates a large set of parameters that need to be optimized. DRL approach with Deep Q-Network can solve the above challenge while BO meets the problem of high-dimensional domain. Furthermore, the results from BO also show large variability in return performance (see Table 2.26), so the system needs to be trained continuously as new data sets are added and the validation process should be considered to avoid over-fitting. Let's take an example for this experiment, the system needs to update 100 times to find new optimal parameter sets for 100 different testing sets, and the execution time is doubled when the validation step is included, the total execution time of BO is 2870 seconds, which is 5.83 times higher than D-DDQN.

Summary of Experiment 2

In this experiment, we build an automated trading system and study different approaches to optimizing the parameters of an algorithmic trading strategy with the RSI indicator. Our work includes the development of a learning environment, state representation, reward function, and learning algorithm for the cryptocurrency market. The results showed that a Deep Reinforcement Learning approach with Double Deep Q-Network setting and a Bayesian Optimization approach can provide positive average returns. Among the settings being studied, the Double Deep Q-Network setting with the Sharpe ratio as the reward function is the best Q-Learning trading system. With short-term trading purposes, the system shows outperforming results in terms of cumulative return, volatility, and execution time when compared with the Bayesian Optimization approach. This helps traders to make quick and efficient decisions with the latest information from the market. In long-term trading, Bayesian Optimization is a method of parameter optimization that brings higher profits. In upcoming studies, Deep Reinforcement Learning provides solutions to the high-dimensional problem

of Bayesian Optimization, such as optimizing portfolios with multiple assets and diverse trading strategies.

2.7 Conclusion

This Chapter presented multiple techniques to optimize the parameters of three different technical indicators: RSI, BOLL, and MACD. Two experiments are carried out with two goals: comparing the performance of trading strategies in various markets with different approaches to parameter optimization and testing the performance of the automated AI trading system with optimized parameters in the framework of Reinforcement Learning. The results of this Chapter are consistent with the results of previous studies [99, 88]. This study also provides many new results for the topic of parameter optimization, which are detailed below.

In the first experiment, the results showed that the optimized parameters from Genetic Algorithm and Bayesian Optimization method could beat the indicators with their typical parameters for all indicators over all periods. Moreover, the parameter optimization method using BO provided better results in terms of profit and execution time than the Genetic Algorithm method in most cases. When compared with the Buy and Hold strategy, the optimized parameters from the Bayesian Optimization method provided greater returns in all cases. Among the indicators being studied, RSI and BOLL could beat MACD and BOLL provided the lowest standard deviation in most cases. In addition, standard deviation and total return from validation and testing periods provide better results than average return when comparing the trading performance of strategies.

The second experiment is carried out with the objective of evaluating the performance of an automated AI trading system with optimized parameters in the framework of Reinforcement Learning. For high-frequency trading, the DRL approach with DDQN setting and the Bayesian Optimization approach produced positive average returns. With a daily trading goal, the system with the DRL approach provided better results when compared to the Bayesian Optimization approach. The results also showed that the DDQN setting with the Sharpe ratio as the reward function is the best Q-Learning trading system. These results provide two options for traders. Traders can apply the BO approach with the goal of building a highly profitable trading strategy over the long term. In contrast, the DRL approach can be applied to regularly update strategies when receiving new information from the market, which helps traders make more effective decisions in short-term trading. The system with DRL settings can also solve the high-dimensional problem of parameters in the Bayesian

Optimization approach, so different trading strategies and objective functions as well as new data can be integrated into the system to improve performance.

This research is the first step towards optimizing trading strategies with the Reinforcement Learning framework from popular tools like Double Deep Q-Network and Dueling Double Deep Q-Network. In future research, the proposed approaches should be compared with recent AI techniques, such as the actor-critic algorithm with a deep double-recurrent network, for a more accurate comparison study. Another promising approach is to study the impact of financial news on the price movements of cryptocurrencies and incorporate it into automated trading systems.

Chapter 3

Hyperparameter Tuning in Agent-Based Stock Market Model

Figuring out how to design stock market models requires a deep understanding of the structure and characteristics of the real stock market. An overview of the approaches used to study the stock market has been given in Chapter 1. In this Chapter, our aim is to provide insights into the organization and operation of the stock market. We propose an Agent-Based Model (ABM) by introducing some concepts from the literature on market structure, price formation in the market, and the trading behavior of the agents. From this model, we propose a parameter optimization method with a Bayesian Optimization approach. This method provides a benchmark for optimizing parameters for other ABM models. In particular, we propose different objective functions based on the stock price and stock return to tune hyperparameters in an agent-based simulation of the stock market. To reproduce the stylized facts of the real market, Bayesian optimization is introduced to find the optimized set of parameters. The experimental results have provided a stable and low-cost hyperparameter tuning method for financial market modeling. The Kolmogorov-Smirnov test is shown to outperform the Kullback-Leibler divergence and the Dynamic Time Warping method in achieving a lower variance of the optimal series and a shorter execution time. With empirical data of the Vietnam Stock Index and the Dow Jones Industrial Average Index, statistical analysis of the simulated data showed that the model was able to replicate some of the important stylized facts in real markets, such as random walk price dynamics, absence of autocorrelations, a heavy tail in the distribution, and volatility clustering of the returns.

3.1 Introduction

Hyperparameter tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter of which value is used to control the learning process. Thus, in order to achieve maximal performance, it is important to understand how to optimize them [29]. One prominent application of hyperparameter tuning in financial markets is to help models reproduce some empirical statistical regularities, known as "stylized facts", of the real market [23]. In this study, we provide results to answer the question of how to have a stable and low-cost hyperparameter tuning method in financial market modeling.

Many models use large parameter sets and run simulations for long periods. However, even for small models, exploring the stylized facts through all parameter combinations is not possible or costly [123]. Grid search is a method to list all combinations of hyperparameters within a given range, then perform a model test with this list [83]. Random search only randomly selects a finite number of hyperparameters from the list to conduct a model test [13]. Theoretically, Grid Search can find the global value, however, as the number of parameters increases, this becomes more and more impossible due to the time and cost of implementation. Random search does not run enough cases like Grid search, so it will be significantly faster. However, depending on the corresponding random selection, we don't know if there exists a better combination of hyperparameters. For neural networks, it is possible to compute the gradient for hyperparameters and then optimize the hyperparameters using gradient descent [84]. In [79], evolutionary optimization is used in hyperparameter optimization for statistical machine learning algorithms. Bayesian optimization is an adaptive approach to parameter optimization, trading off between exploring new areas of the parameter space, and exploiting historical information to find the parameters that maximize the function quickly [123]. The optimization function is composed of multiple hyperparameters that are set before the learning process and affect how the learning algorithm fits the model to data. Thereby, Bayesian optimization is used to tune hyperparameters for the optimization function in our model. The main ingredients in Bayesian optimization include: search space, objective function, surrogate function and an acquisition (selection) function.

Following the studies in [121], our approach is to start the domain from a wide range and then focus on specific areas around the optimal parameters calculated from the previous run. We define a score function that indicates how well a set of long-short periods performs. Cumulative returns and Sharpe ratio are usually used to indicate the performance of a technical strategy [98]. In our model, Sharpe ratio is used as the evaluation metric of choice. The surrogate function is used to propose sets of value that increase performance in minimizing the score of objective function. Then, they are selected by applying a criterion.

In this model, we follows [123] to use Tree Parzen Estimator (TPE) as the surrogate function. Expected Improvement (EI) criterion is used as the selection function. We propose two objective functions based on return and price. The return-based objective function considers two tests, the Kullback-Leibler (KL) divergence and the Kolmogorov-Smirnov (KS) test [97], where the statistic in the test quantifies the difference of two probability distributions. Besides, Dynamic Time Warping (DTW), which measures similarity between two series, is introduced to construct the price-based objective function [14].

We synthesize the key components of our previous work to build a simple ABM for the stock market [133]. This model offers a novel perspective on how agents can be influenced in selecting their trading rules. In the financial market, the propensities of traders can be contaminated by the sentiments of their neighbors. The agents also have different sensitivities to exogenous factors in the market and thus different news can influence and impact them [24]. Therefore, modeling financial markets requires a good understanding of the characteristics of market agents, their behaviors, their interaction and their sensitivity to various exogenous factors [8, 91]. The studies in this Chapter will focus on clarifying these properties. In the model of [127], the network of traders is a square lattice and each trader is connected to his four nearest neighbors. In our model, the traders' network is built into groups of different sizes. This makes the level of sentiment contamination more diversified. The objective functions are compared based on cost and stability over many simulations. In addition, we also analyze the efficiency of each objective function through its ability to reproduce the properties of the stock market in reality.

The results in this Chapter show that Bayesian optimization can propose a optimal set of parameters with high stability and low cost. The return-based objective function with the KS test shows better results on convergence rate and execution time. In addition, the ABM with different objective functions can represent some stylized facts of the real market. This provides a benchmarking method for optimizing parameters of agent-based stock market. The DTW algorithm can reproduce dynamics of the prices in the real market but is not good for the absence of autocorrelations in returns. In contrast, the KS test does not show a good simulation of prices but successfully explains some important stylized facts of returns, such as the absence of autocorrelations, fat tails in stock return distribution, and volatility clustering. These results are consistent with the definition of the objective function.

3.2 Objective Function in Bayesian Optimization

The main goal of ABM is to reproduce stylized facts and to identify sufficient conditions for their creations. In financial markets, stylized facts emerge from statistical analysis of prices

and returns. Therefore, the objective function in this study is defined based on the above two variables.

3.2.1 Return-based objective function

The objective function is constructed by comparing simulated returns and actual returns. More specifically, the comparison focuses on the statistical characteristics of the distributions from the two time series. The construction are presented as follows.

Kullback–Leibler divergence

The KL divergence measures how much information is lost when the probability distribution of simulated returns, $Q(x)$ is used to approximate the probability distribution of actual returns, $P(x)$. The formula for the divergence, defined over a random variable $x \in X$, is as follows:

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} P(x) \frac{P(x)}{Q(x)} dx, \tag{3.1}$$

where X is the set of all possible variables for x.

Kolmogorov-Smirnov test

The KS test statistic measures the largest distance between the empirical distribution function of simulated returns, $F_s(x)$ and the empirical distribution function of actual returns, $F_a(x)$. The formula for the test statistic is given by:

$$D_{KS} = \sup_x |F_{s,n}(x) - F_{a,n}(x)|, \tag{3.2}$$

where n is the data size and sup is the supremum function.

3.2.2 Price-based objective function

The objective function uses the DTW algorithm to calculate an optimal match between simulated prices and actual prices with certain restrictions and rules. The cost is computed as the sum of absolute differences, for each matched pair of indices, among their values. Let x and y be two vectors of lengths m and n, respectively, the algorithm is summarized in the following three steps:

- Optimum-value function: Define $D(i, j)$ as the DTW distance between $(x_1, ..., x_i)$ and $(y_1, ..., y_j)$, with the mapping path starting from $(1, 1)$ to (i, j).

- Recursion:

$$D(i,j) = |x(i) - y(j)| + min\Big\{D(i-1,j), D(i-1,j-1), D(i,j-1)\Big\}, \tag{3.3}$$

  with the initial condition $D(0,0) = 0, D(0,j) = \infty, D(0,j) = \infty$.

- Final answer: $D(m,n)$.

3.3 Financial Market Model

In this chapter, we recombine key ingredients of our previous works to come up with a simple ABM that can match the stylized facts of financial markets. For more details, see [133].

3.3.1 The Agent

The agents rely only on the current market situation to make trading decisions: buy, sell or hold. We follow the specification of the agents formulated by [59]. As the key variable, sentiment of the agents implies their reaction on the news they get and the impact of news on future prices. Good sentiment means that agents see the news as good news and hope future prices will increase (bullish) and bad sentiment means that the agents consider the news to be bad and hope prices in the future will decrease (bearish).

The agents in the model are assumed to gather into groups when making investment decisions. In [127], the authors describe the group of traders introduced by is a square lattice and each trader is connected to his four nearest neighbors. In our model, we define the traders' group as a multi-dimensional lattice that allows each agent to be connected to his neighbors in different size groups. This makes the trading behaviors of agents more diverse. Accordingly, the propensity in decision-making behavior of agents is introduced. The term "propensity of agents' decision" defines the probability that one agent will follow the other agents' opinion and that agents will change their decisions on their own.

Let's define the sign function, $sign(x)$, as

$$sign(x) := \begin{cases} 1, & \text{if } x > 0; \\ 0, & \text{if } x = 0; \\ -1, & \text{if } x < 0. \end{cases} \tag{3.4}$$

At each time step t, the final decision of the agent i , denoted as $s_i(t)$, is given by the following equation:

$$s_i(t) = \text{sign}\Big[\eta_i(t)Q_i(t) + \theta_i(t)\sum_{j\in\Omega_i} s_j(t) + \varepsilon_i(t)\Big]. \tag{3.5}$$

If the outcome of this decision is 1, the agent will be bullish and expect the asset price to rise. As such, he will buy one share of the asset. If the outcome of an agent's decision is -1, the agent will be bearish and expect the asset price to fall. In this case, he will sell the one share of the asset. Otherwise, he will hold his assets. The main components in the Eq. (3.5) are introduced in detail below.

- News reaction term $\eta_i(t)Q_i(t)$ - The news reaction of an agent i implies the sensitivity on the news he get, $\eta_i(t)$, and the impact of news on future prices, $Q_i(t)$. We define the inflow of news arriving in the market as a signal $I(t)$ with $I(t) \sim N(0,\sigma_I^2)$. $I(t)$ is a forecast of the future return and each agent has to decide whether the news is significant or not. To describe the heterogeneity, the news sensitivity $\eta_i(t)$ is allowed to float randomly on a uniform distribution,

$$\eta_i(t) = \sigma_I[1 + U(0,1)]. \tag{3.6}$$

  At each time step, each agent i has a probability $0 \leq p_i \leq 1$ of updating $\eta_i(t)$. When an agent updates his news sensitivity, he sets it to be equal to the recently observed absolute return, $|r(t)| = |ln\frac{S(t)}{S(t-1)}|$, where $S(t)$ and $r(t)$ are the market price and market return of stock, respectively. Introducing I.I.D random variables $u_i(t), i \in N$, uniformly distributed on [0, 1], the updating scheme is represented as

$$\eta_i(t) = \begin{cases} |r(t)|, & \text{if } u_i(t) < p_i; \\ \eta_i(t-1), & \text{otherwise.} \end{cases} \tag{3.7}$$

  The news-meaning, $Q_i(t)$, is assumed that news about the stock can either be good or bad. Specifically, the news value presented to the market that is greater than 0 is attributed a value of 1, otherwise, it is set to -1. The formula of $Q_i(t)$ is represented by

$$Q_i(t) = \mathbb{1}_{I(t)>0} - \mathbb{1}_{I(t)<=0}. \tag{3.8}$$

- Propensity term $\theta_i(t)\sum_{j\in\Omega_i} s_j(t)$ - The propensity of agent i implies the trust (decision sensitivity), $\theta_i(t)$, and the sum of decisions of agent i's neighbors, $\sum_{j\in\Omega_i} s_j(t)$. We

assume that the agent is sensitive to the decisions of his neighbors in the agents' group. In this way, the decision of each agent influences, and is influenced by the decisions of other agents. The trust variable is initially set to base trust σ_θ, $\theta_i(t) \sim U(0, \sigma_\theta)$. At each time step t, each agent i also has a probability $0 \leq p_i \leq 1$ of updating $\theta_i(t)$

$$\theta_i(t) = \begin{cases} |r(t)|, & \text{if } u_i(t) < p_i; \\ \theta_i(t-1), & \text{otherwise.} \end{cases} \tag{3.9}$$

Moreover, when an agent updates his trust, he decides to switch to another group.

For each agent i, we denote his neighborhood which contains all other agents that directly connect to him as Ω_i. The sum $\sum_{j \in \Omega_i} s_j(t)$ is the sentiments of all agents j who are in direct contact with agent i.

- Idiosyncratic term $\varepsilon(t)$ - This variable is included to aid in the representation of the heterogeneous preferences held by agents. In other words, this variable is the outlook that an agent brings to the market, whether he is inherently bullish or bearish, optimistic, or pessimistic, regardless of market behavior [59]. We take it as being normally distributed around zero, $\varepsilon_i \sim N(0, \sigma_\varepsilon^2)$.

In summary, the main characteristics of the agents are described as follows. At each time step t:

- Agents receive a public information as a signal $I(t)$.
- Each agent i compares the signal to his news sensitivity $\eta_i(t)$ to make his news reaction.
- The agent then looks at the propensity of neighbors' decisions.
- He make decision by combining the news reaction, the propensity, and his own interpretation.
- After calculating the market return, he updated his sensitivities with a probability p_i. If the agent updates his trust, he decides to switch to another group.

3.3.2 The Environment

The model considers a fixed amount of agents, denoted as N, trading a single stock. At each time step t, a given trader i can choose between three actions: buys/sells one unit of the stock or remains inactive, $s_i(t) \in \{1, -1, 0\}$, respectively. We assume that each agent always has

enough wealth and shares for liquidity purpose. In other words, all agents can always buy or sell.

At each time step t, the excess demand $D(t)$ is calculated as the number of buyers minus the number of sellers, which is formed as

$$D(t) = \sum_{i=1}^{N} s_i(t). \tag{3.10}$$

The market price and market return of stock are then calculated. The market price $S(t)$ is updated according to

$$S(t) = S(t-1)[1 + r(t)], \tag{3.11}$$

and return $r(t)$ is determined according to

$$r(t) = g\left(\frac{D(t)}{N}\right), \tag{3.12}$$

where the price impact function $g : \mathbb{R} \mapsto \mathbb{R}$ is an increasing function with $g(0) = 0$. We follow [32] to define the market depth λ by: $g'(0) = 1/\lambda$. In our model, the chosen price impact function is $g(z) = arctan(z/\lambda)$.

3.4 Stylized Facts of Financial Markets

Many empirical studies have been and are being conducted to determine the efficiency of various markets. Empirical studies collect real data and try to fit them with the tested model to determine if the theoretical models have similar properties to the real market. In contrast to the theoretical approach, empirical studies aim to describe and analyze properties of market data. They do not assume an ideal market but investigate the real one.

Statistical properties that reveal non-random features of historical data are known as stylized facts. A stylized fact is a term used in economics to refer to empirical findings that are so consistent that they are accepted as truth [122]. Many studies of different types of markets, traded assets and participants [23, 31, 32, 43] have proposed a list interesting stylized facts of asset prices and returns. These stylized facts can be summarized as follows:

(i) *The random walk property of price changes*. Randomness implies that successive price changes should be statistically independent and identically distributed [102].

(ii) *Excess volatility*. The sample standard deviation of returns can be much larger than the standard deviation of the input noise representing news arrivals [32].

(iii) *Heavy tails*: The unconditional distribution of the asset returns are considered as a result of power-law or Pareto-like in financial dynamics [31]. This fact reflects the number and magnitude of market bubbles and crashes.

(iv) *Absence of autocorrelations*. The sample autocorrelation function of the returns exhibits an insignificant value at all lags, indicating the absence of linear serial dependence in the returns [43].

(v) *Volatility clustering*. There is a significant autocorrelation between the absolute values of the returns [31]. Large returns often tend to emerge in clusters, that is, large changes in asset prices tend to be followed by large changes and small changes tend to be followed by small changes. This fact is explained that the market behavior can be affected by its behavior a long time ago.

(vi) *Leverage effect*. The volatility of stocks tends to increase when the price drops. This is a negative correlation between past returns and future volatility [19].

(vii) *Downside correlations*. The apparent increase of cross correlations between stocks in highly volatility market conditions, in particular when prices significantly. This increase is thought to be larger for large downward moves than for large upward moves [19].

3.5 Experiment Setup

3.5.1 Data Description

In this section, we set up experiments with data of two indices: the Vietnam Stock Index (VN-Index) and the Dow Jones Industrial Average Index (DJIA). As an empirical basis, we use a data set that covers 4 years of daily quotes of the VN-Index, from January 2017 to January 2021. With the DJIA Index, we use a data set that covers 5 years of daily quotes, from of January 2014 to the of January 2019. The main statistical properties are given in Table 3.1 and Table 3.2. All numerical simulation and graphical visualization are done by using Python 3.6.5 version on Intel Core i7–4790 3.6GHz. The focus of analysis is on statistical properties of the random walk price dynamics, distribution of stock returns, and the volatility clustering in the market.

Table 3.1 Statistical properties of the actual returns

Data	Length	Mean	Standard Deviation	Kurtosis	Skewness
VN-Index	1000	0.05%	1.10%	5.62	-1.04
DJIA	1257	0.03%	8.07%	4.3139	-0.6431

Table 3.2 Statistical properties of the actual returns (cont.)

Data	Min	Quantiles			Max
		25%	*50%*	*75%*	
VN-Index	-6.27%	-0.34%	0.14%	0.59%	4.98%
DJIA	-5.07%	-2.99%	0.06%	4.21%	3.81%

3.5.2 Hyperparameter Description

We construct an artificial stock market of 1000 traders. In order to interpret the trading periods as days and compare the results with properties of daily returns, we choose $\lambda \in [5,20]$ which allows a range of daily returns between 5% and 20%. Denote $1/p_i$ as the typical time period during which an agent i will hold a his news sensitivity $\eta_i(t)$. p is chosen in $[0.001,0.1]$ to be interpreted as days. For example, $p = 0.1$ means the average updating period of agents is 10 days. The standard deviation of the signal, σ_I, can be chosen in the range $[0.001,0.01]$ to reproduce a realistic range of values for the annualized volatility [32]. The value of the trust base is chosen in the range $[0.1,1]$. The value of standard deviation in idiosyncratic term is chosen in the range $[0.01,1]$. The parameters and ranges are summarized in Table 3.3.

Table 3.3 Description of parameters

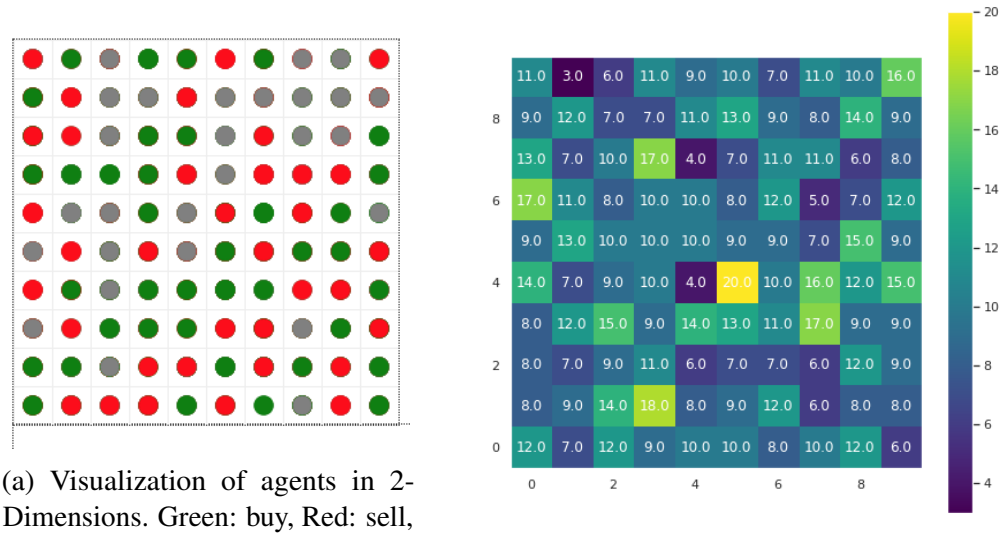
Parameters	Description	Value
N	Number of agents	1000
λ	Market depth	$[5,20]$
p	Fraction of agents updating their news reaction	$[0.001,0.1]$
σ_θ	Trust base	$[0.1,1]$
σ_I	Standard deviation of the inflow of news	$[0.001,0.01]$
σ_ε	Standard deviation of idiosyncratic term	$[0.01,1]$

We formulate the optimization problem using *hyperopt package* of Python with three main ingredients

- Domain space: Each trader is assigned with initial values derived from random uniform distributions within the range of Table 3.3, respectively. Following the studies in [121], our approach is to start the domain from a wide range and then focus on specific areas around the optimal parameters calculated from the previous run.
- Objective function: return-based and price-based objective functions are used to find the best value of hyperparameters that minimize the loss in 100 iterations.
- Surrogate function and selection function: TPE is used as the surrogate function. Then, EI criterion is used as the selection function.

3.6 Result and Discussion

In this section, we study the results from the VN-Index data to compare the performance of the two return-based objective functions: the KL divergence and KS test. The perfomance of the return-based objective function and the price-based objective function are then compared and the stylized facts are discussed. The same experiment can also conducted with DJIA data. We run the model 100 times to ensure the consistency of the results. An example of the simulation visualization is shown in Figure 3.1.



(a) Visualization of agents in 2-Dimensions. Green: buy, Red: sell, Grey: hold.

(b) Distribution of number of agents.

Fig. 3.1 An example of the simulation visualization

Firstly, we compare the performance of the two return-based objective functions. Both KL divergence and KS test found the optimal set of parameters with small simulation error as shown in Table 3.4. Note that we do not compare values of the objective functions due

to differences in measurement definitions. In this work, we picked out the better objective function by investigating their stability over many simulations. KL divergence represents the instability in many simulations, the minimum value can range from 0.07 to 3.99. In contrast, KS test shows stability with small fluctuations in the range of 0.03 to 0.07. The stylized facts which have been found to be present in the simulated returns are discussed below.

Table 3.4 Optimized parameters with the return-based objective functions

Method	**Optimal Value**	**Parameters**				
		λ	p	σ_θ	σ_I	σ_ε
KL	0.0747	11	0.060	0.1	0.001	0.853
KS	0.0350	19	0.047	0.1	0.001	0.999

The follow-up results and discussions are conducted between KS and DTW. The optimal set of parameters and simulation error of two methods are shown in Table 3.5.

Table 3.5 Optimized parameters with different objective functions

Method	**Approach**	**Optimal Value**	**Parameters**				
			λ	p	σ_θ	σ_I	σ_ε
KS	return-based	0.05	12	0.088	0.1	0.001	0.14
DTW	price-based	77.74	17	0.037	0.1	0.004	0.34

After 100 simulations, the DTW algorithm represents the instability, the optimal value can range from 77 to 100 (see Table 3.6). In contrast, the KS test shows stability with small fluctuations in the range of 0.05 to 0.07. Moreover, the execution time of KS is shorter than that of DTW, which means that the KS test costs less than DTW algorithm. Figure 3.2 shows that the KS convergence is faster than the DTW, and generally, the KS is more stable than the DTW algorithm, because the standard deviation of optimal observed objective is smaller after 50 iterations.

Table 3.6 Comparison of cost and stability

Method	**Confidence Interval**	**Running Time**
KS	0.05-0.07	2.78 hrs
DTW	77-100	3.19 hrs

In addition, both algorithms can reproduce some stylized facts of the real market. In the figures below, the actual series is blue and the simulated series is coral. The results are summarized as follows.

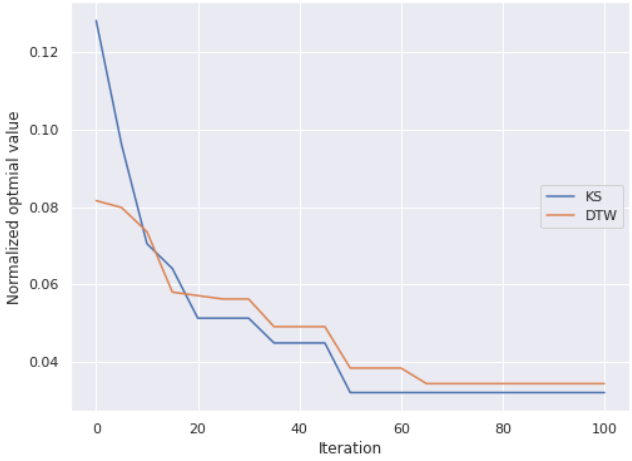


Fig. 3.2 Comparison of convergence rate

3.6.1 Dynamics of the Prices

First, we investigate the random walk price dynamics in financial markets. The simulated prices are computed from price-based optimization algorithm. Figure 3.3 shows the daily movement of real prices and simulated prices. Both series have an upward trend, the mean and variance change over time, and prices do not tend to come back towards their average.

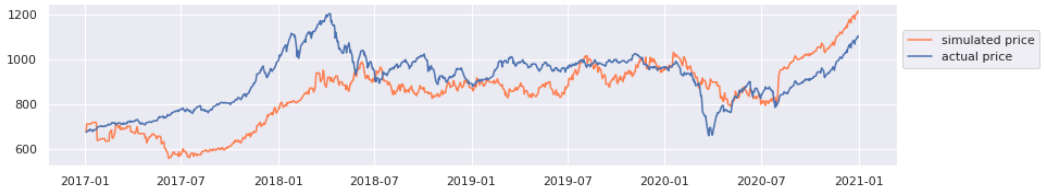


Fig. 3.3 Simulation of prices

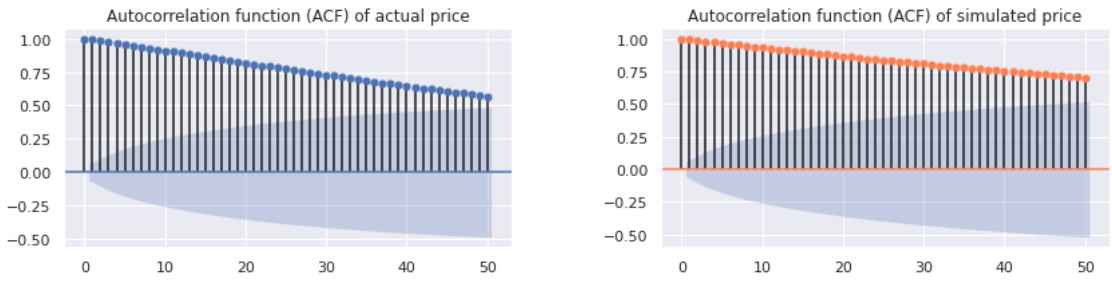


Fig. 3.4 Autocorrelations of prices

The autocorrelation function (ACF) plots in Figure 3.4 demonstrate that the correlation between the time series and its lags fall outside the 95% of the confidence interval of no

autocorrelation for 50 lags. Therefore, the results confirm that both prices are random walk and the simulated prices are similar to the dynamics of prices in the real market.

3.6.2 Distribution of Returns

The simulated returns are computed from a return-based optimization algorithm. The statistical analysis of the actual returns and simulated returns are given in Table 3.7. The simulated returns have similar properties to the actual returns. We consider a significance test if there is evidence for a difference θ between two returns, with the null hypothesis $H_0 : \theta = 0$ and the alternative hypothesis $H_1 : \theta \neq 0$. Let $D := r_{actual} - r_{simulated}$, where r_{actual} and $r_{simulated}$ are the actual returns and simulated returns, respectively. The Bayes factor $BF_{10}(D)$ stands for "H_1 versus H_0", is given by:

$$BF_{10}(D) = \frac{P(D|H_1)}{P(D|H_0)}.$$

The result is computed from the R packages "BayestestR", which gives: $BF_{10} = 0.036$. So, $BF_{01} = 1/0.036 = 27.03$. This value of BF_{01} means that H_0 is more strongly supported by the data under consideration than H_1. Using the scale for interpreting Bayes factors by Harold Jeffreys [66], we obtain a strong evidence that there is no difference between the actual returns and simulated returns.

In financial data, the excess kurtosis of return is larger than 0, indicating that the distribution of returns displays a heavy tail [90]. The quantile-quantile plots in Figure 3.5 show that the actual returns and simulated returns are not normally distributed. The returns produced by the model are consistently leptokurtically distributed with fat tails as shown in Figure 3.6.

Table 3.7 Comparison of statistical properties

Data	**Mean**	**Standard Deviation**	**Kurtosis**	**Skewness**
Real returns	0.05%	1.10%	5.62	-1.04
Simulation	0.03%	1.22%	5.17	-1.16

Data	**Min**	**Quantiles**			**Max**
		25%	***50%***	***75%***	
Real returns	-6.27%	-0.34%	0.14%	0.59%	4.98%
Simulation	-6.19%	-0.43%	0.00%	0.33%	6.60%

Fig. 3.5 Q-Q plot of returns

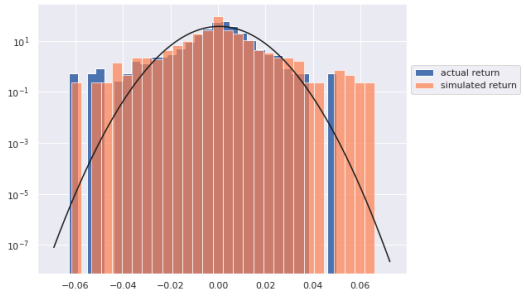


Fig. 3.6 Distribution of returns with logarithmic scaling on vertical axis

3.6.3 Absence of autocorrelations in returns

Figure 3.7 shows the plots of stock returns for both actual data and simulated data. ACF plots in Figure 3.8 demonstrate that there is no correlation between the simulated returns and its lags after the first few lags.

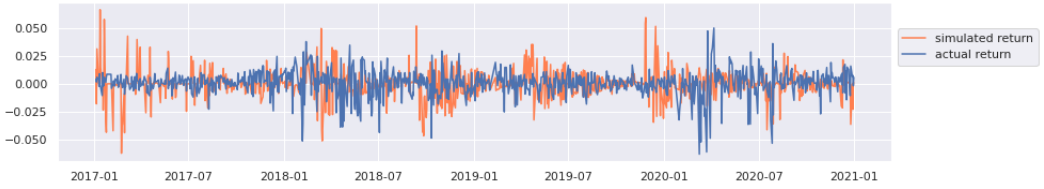


Fig. 3.7 Simulation of returns.

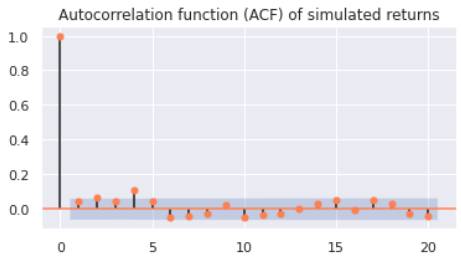


Fig. 3.8 Autocorrelations of simulated returns.

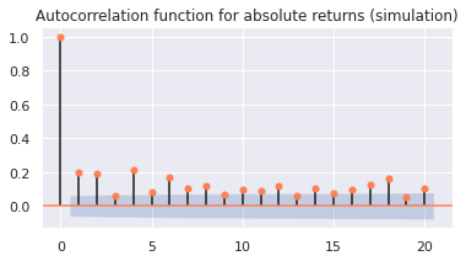


Fig. 3.9 Autocorrelations of absolute value of simulated returns.

3.6.4 Volatility Clustering

The next stylized fact we study is volatility clustering, which describes the fact that the tendency of large changes in prices to cluster together. A slow decay in the ACF of absolute values of returns is a signature of the volatility clustering that can be seen in Figure 3.9. Notice that for both cases, the autocorrelation of absolute returns remains significantly positive over many time lags. This is a clear evidence of volatility clustering.

3.7 Conclusion

In this Chapter, we have proposed different objective functions to tune hyperparameters in an agent-based simulation of the stock market. The Bayesian optimization with Kolmogorov-Smirnov test has proposed an optimal set of parameter with high stability and low cost. The model with a return-based objective algorithm was able to highlight some of the important stylized facts of the stock returns such as the absence of autocorrelations, the heavy tail of the returns, and volatility clustering. Besides, the dynamics of stock prices in the real market are reproduced when simulated with the price-based objective algorithm. In the future, we can adjust to combine the two objective functions to get results for both prices and returns.

This work is only the first step toward fully assessing the properties of agent-based models in the United States stock market. In future research, the model in this Chapter can be applied to other exchange such as the New York Stock Exchange or emerging markets in Asia. The different characteristics of the market and investors provide many interesting results for managers and investors. Moreover, deep learning is a good methodology for dealing with high-dimensional problems. To optimize hyperparameters in ABM, Reinforcement learning can be considered to address Bayesian optimization issues, in which our Bayesian approach can be used as a benchmark method for the comparison.

Chapter 4

Application of Agent-Based Modeling in Stock Market

In this Chapter, an Agent-Based Model (ABM) of the stock market is constructed to detect the proportion of different types of traders. We model a simple stock market which has three different types of traders: noise traders, fundamental traders, and technical traders, trading a single asset. Bayesian optimization is used to tune the parameters of the strategies of traders as well as of the stock market model. The experimental results on Bayesian Optimization with the Kolmogorov–Smirnov (KS) test demonstrated that the proposed set reduced simulation error with plausible estimated parameters. With empirical data of the Dow Jones Industrial Average (DJIA) index, we found that fundamental traders account for 9%–11% of all traders in the stock market. The statistical analysis of simulated data can produce the important stylized facts in real stock markets, such as the leptokurtosis, the heavy tail of the returns, and volatility clustering.

4.1 Introduction

Financial market can be considered as a complex system. For tens of years, researchers have been faced with the problem of the analysis and modeling of financial markets [46, 6, 7]. One of the most important questions is "How can we develop an artificial stock trading system and describe investors' behaviors in a way that is similar to reality?". Previous models are based only on a single fully rational representative agent and fail to reproduce all the properties of real markets [78, 22]. To overcome these problems, a new behavioral approach emerged, referred to as agent-based modeling, characterized by markets populated with bounded rational, heterogeneous agents [64].

An agent-based model (ABM) in the financial market is a class of quantitative models to simulate the decisions and interactions of different traders in order to understand their behaviors and their impact on the market [33]. In recent years, ABM in financial markets is still a prominent topic of interest for researchers [82, 6]. The behavior of the agents is described in a variety of ways; their interaction and the impact of exogenous factors in the market are also introduced [63]. Thereby, studies help managers to understand the behavioral characteristics of investors and explain the phenomena occurring in the stock market [48, 63].

One of the problems of agent-based modeling is how to optimize the parameters of the traders' strategies and tune the hyperparameters of the market model. Typically, many models use large parameter sets and run simulations for long periods of time [120, 3]. However, even for small models, exploring the stylized facts through all parameter combinations is not possible or is prohibitively costly [82]. For example, in [120], the authors generated 400,000 initial points to calibrate their model. To solve the above problems, this model introduces an approach with Bayesian optimization and an error measurement function to optimize the parameters of the environment.

In this work, we combine the main properties of the pioneer works. An agent-based model is introduced to highlight the stylized facts of the stock market. This research offers a novel perspective on how to detect the proportion of each trader in the stock market. The results can be used in selecting effective trading strategies for traders. The model consists of three types of traders: noise traders, technical traders, and fundamental traders, with various behaviors to increase diversity. A noise trader makes decisions based on price changes. They have different emotions and news reactions which play important roles in making decisions. Technical traders are diversified by trading indicators, they make different decisions based on target return and time. Fundamental traders are introduced to balance the market, they are diversified by their confidence level which measures the convergence of the stock price to its fundamental value. The interaction is a function that combines the sentiment change when getting information from the market and the influence of other investors' decisions.

The highlight of our model is that Bayesian optimization is introduced to optimize the parameters of the traders' strategies and tune the hyperparameters of the market model. The results show that Bayesian optimization can propose an optimal set of parameters and reduce simulation error. This provides stylized facts to compare with the real market, providing a bench-marking method for calibrating parameters of agent-based stock market. We used different methods of comparing the two distributions, Kullback-Leibler (KL) divergence and the Kolmogorov-Smirnov (KS) test, to measure the simulation error. In our model, the KS test showed better results by successfully explaining some important stylized facts in real stock markets, such as the leptokurtosis, the heavy tail of the returns, and the volatility

clustering. With extensive simulations, we found that the fundamental traders accounted for 9%–11% of all traders in the stock market. These results are consistent with the real-market research [27].

4.2 Background

In the financial market, there are many types of agents (investors, banks, central banks, etc.). These agents can interact directly through trading, exchanging derivatives. They can also interact indirectly, because any decision or behavior of an agent affects others [53]. Moreover, the agents have different sensitivities to exogenous factors in the market and thus different news can influence and impact them [24]. Therefore, financial market modeling needs to understand the characteristics of agents such as behavior and interaction between investors and the impact of exogenous factors such as public news on the market [32, 63]. The studies in this Chapter will focus on clarifying these properties.

The first approach addresses the types of agents and how they impact the dynamics of the market. Pioneering works have proposed a number of interesting agent-based financial market models. The concept of zero-intelligence traders or noise traders who decide randomly whether to buy or sell was introduced in a double auction market [57]. The studies of market participants using technical and fundamental analysis to assess financial markets creates a strong empirical foundation for building financial market models [51]. Technical traders are considered to be a factor of instability because their trading often adds a positive response to the dynamics of the market [103]. In contrast, fundamental analysis based on its definition provides a stabilizing impact on market dynamics [60].

The second approach that should be considered is the diversity of agents. To make every fundamental agent and technical agent unique, they are assigned with different memory lengths or reaction intensities to price and fundamental changes [63]. Exogenous factors such as news, social media, and insider information [114] also have different impacts on each agent. In addition, the sentiments of agents such as risk aversion play an important role in the decision-making process. [89, 48].

The third approach is the interaction between agents. Besides the interactions that directly affect market prices, investors' decisions also have an indirect effect on each other. In [77, 137], the authors introduced the propagation in the decision of investors with a fixed probability. In another study [63], the interactions and diversity among fundamental traders and technical traders were introduced. The average utility associated with each type was calculated and agents could rely on profit to come up with an effective investment strategy.

Each of these approaches has been extended in various interesting directions. From that, the stylized facts of financial markets are highlighted, such as heavy tails in stock return distribution and volatility clustering [64, 137]. Another proposed stylized fact is the positive autocorrelation in volatility and trading volume [129, 63].

To have an optimal set of parameters for the model, many optimization methods have been considered. Grid search is a method to list all combinations of parameters, then perform a model test with this list [123]. Random search only randomly selects a finite number of parameters from the list to conduct a model test [13]. Theoretically, grid search can find the global value, but as the number of parameters increases, this becomes increasingly impossible due to the time and cost of implementation. Random search does not run as many cases as grid search, so it is significantly faster [13]. However, depending on its randomness, each run will receive a different optimal value because it does not guarantee the optimum value to be global or local only. Bayesian optimization is an adaptive approach to parameter optimization, trading off between exploring new areas of the parameter space and exploiting historical information to find the parameters that maximize the function quickly [123]. Thus, Bayesian optimization is introduced to tune the parameters for the trading strategies of technical traders and the environment of our model.

4.3 Model Description

This section outlines the construction of a new agent-based model. This model adds the understanding of the reason for the stylized facts and the estimated proportion of traders in the stock market.

4.3.1 Environment

The agent-based model in this Chapter focuses on analyzing the behavior of traders and the log return properties which are reflected in the simulation process. The model considers a fixed amount of agents, denoted as N, trading a single asset. The current time is denoted with t and the corresponding price is $S(t)$. At each time step t, trader i can choose between three actions: buy/sell one unit of the stock or hold, $s_i(t) \in \{+1, -1, 0\}$, respectively. For liquidity purposes, the model assumes that every trader always has enough wealth and shares. In other words, they are always able to buy or sell. The market price and market return are then calculated. The market price $S(t)$ is updated according to

$$S(t) = S(t-1)[1 + r(t)],$$

and return $r(t)$ is determined according to

$$r(t) = g\left(\frac{D(t)}{N}\right),$$

where the excess demand $D(t) = \sum_{i=1}^{N} s_i(t)$, the price impact function $g(z) = arctan(z/\lambda)$, and λ measures the market depth or liquidity.

Due to the tendency to form discrete clusters of traders [134], we define different types of traders in this model based on their predefined trading rules. There are three different types of traders operating in the market. Noise traders are people who make decisions at random according to their news reactions and their propensity to sentiment contagion. Technical traders analyze charts and make decisions based on current patterns and trends analysis, and fundamental traders trade based on the fundamental profit-generating potential of the stock.

4.3.2 Noise Agent

Noise agents are traders who trade on the basis of misunderstanding information and news regarding future prices. They make decisions and trade based on inaccurate analysis of the market [130]. In this model, noise agents rely only on the current market situation to make trading decisions: buy, sell, or hold [63]. As the key variable, the sentiment of the noise agents implies their reaction to the news they get [32]. Good sentiment means that agents see the news as good news and hope future prices will increase (bullish) and bad sentiment means that the agents consider the news to be bad and hope prices in the future will decrease (bearish).

The decision of noise agents is determined by the following process. At each time step t:

- Agents receive public information as a signal $I(t)$.
- Each agent i compares the signal to his threshold $\theta_i(t)$ to make a decision.
- After calculating the market return, he updates his threshold with a probability p_i.

The inflow of news arrives to the market as a signal $I(t)$ with $I(t) \sim N(0, \sigma_I^2)$. Each agent will decide whether the news is significant or not, in which case he will make a decision according to the sign of $I(t)$. To describe the heterogeneity, we introduce the sentiment of each agent to the news as a decision threshold $\theta_i(t)$ with the initial trading threshold set between 1 and 2 times the standard deviation of the news, $\theta_i(0) = \sigma_I[1 + U(0,1)]$.

Let N_{noise} denote the number of noise agents. The trading rule of each noise agent $i, i \in N_{noise}$, is represented by

$$s_i^{noise}(t) = \mathbb{1}_{I(t)>\theta_i(t)} - \mathbb{1}_{I(t)<-\theta_i(t)}.$$

Market-related sentiments are formed differently for each agent and each responds differently to the information. To avoid the artificial ordering of agents as in sequential choice models, we follow [32] to generate an asynchronous strategy to update the agents' threshold. At each time step, each agent i will update his threshold $\theta_i(t)$ with probability $0 \leq p_i \leq 1$. The threshold is set equal to the absolute return at time t, $|r(t)| = |ln\frac{S(t)}{S(t-1)}|$. Introducing independent and identically distributed random variables $u_i(t), i \in N_{noise}$, uniformly distributed on [0, 1], the updating scheme is represented as

$$\theta_i(t) = \begin{cases} |r(t)|, & \text{if } u_i(t) < p_i; \\ \theta_i(t-1), & \text{otherwise.} \end{cases}$$

In the next addition to the model, technical agents and fundamental agents are introduced.

4.3.3 Technical Agent

Unlike the noise agents, technical agents are traders who use the past prices to infer private information. They make decisions by calculating the technical indicators from historical prices [21]. The characteristic of technical analysis is to make decisions after identifying the trends at an early stage and reverse the decision when the trend reversal occurs [118]. In this model, the agents are assumed to use a technique in the real world called a moving average-oscillator to predict future price moves [20]. The technique first involves taking two moving averages of the price, a short-term moving average A and a long-term moving average B, of different lengths $l_A < l_B$. The moving average crossover is calculated as the difference $M(t)$ between these two moving averages. $M(t) > 0$ indicates that the price is trending upwards and the agents will make a buy. $M(t) < 0$ indicates that price is trending down and the agents will sell the stock.

In order to account for individual heterogeneity, each agent can use different lengths of historical data, so one moving average strategy may indicate an uptrend while another moving average strategy indicates a downtrend. However, in our model, the difference in window lengths, l_A and l_B, has no effect on agents' trading decision. When the historical stock price data has a strong uptrend or downtrend, all investors will make the same decision.

Thus, we assume that all technical agents can use Bayesian optimization to estimate the optimal length for short and long windows.

We formulate an optimization problem with three main ingredients:

- Domain space: Each agent is assigned a random short window and long window derived from a random uniform distribution in [5, 40] and [50, 100], respectively.
- Objective function: We define a score function that indicates how well a set of long-short periods performs. Cumulative returns and Sharpe ratio are usually used to indicate the performance of a technical strategy [98]. In our model, the Sharpe ratio is used as the evaluation metric of choice. We put a minus sign before the objective function since *hyperopt* by default is defining a function to minimize.
- Surrogate function and selection function: The surrogate function is used to propose sets of values that increase performance in minimizing the score of the objective function. Then, they are selected by applying a criterion. In this model, we follow [123] to use the Tree-structured Parzen Estimator as the surrogate function. The Expected Improvement criterion is used as the selection function.

The optimal lengths for short and long windows with the largest Sharpe ratio are 35 and 51, respectively. The buy/sell signals are shown in Figure 4.1.

Fig. 4.1 Moving average oscillator with buy/sell signals.

To describe the heterogeneity, we introduce $\psi_i(t)$ as the average profit a technical agent i gained in the previous n_i trading day, $\psi_i(t) = \sum_{j=1}^{n_i} \frac{1}{n_i} r(t-j)$, which is considered as a threshold to make a decision. Each agent i will update her threshold $\psi_i(t)$ with probability $0 \leq p_i \leq 1$ at each time step t. The updating scheme is represented as in the description of noise agents.

Technical agents are also affected by the public information like the noise agents. The news reactions $Q_i(t)$ are represented by

$$Q_i(t) = \mathbb{1}_{I(t)>\theta_i(t)} - \mathbb{1}_{I(t)<-\theta_i(t)}.$$

Define the sign function, $sign(x)$, as

$$sign(x) = \begin{cases} 1, & \text{if } x > 0; \\ -1, & \text{if } x < 0; \\ 0, & \text{if } x = 0. \end{cases}$$

Let N_{ta} denote the total number of technical agents. The trading rule of the technical agent $i, i \in N_{ta}$, is represented by

$$s_i(t) = \text{sign}\Big[\psi_i(t)M(t) + Q_i(t)\Big].$$

The amplification of trends and unlimited buying power of technical agents can cause problems because the price can grow to infinity or drop to extremely small values very quickly. Thus, fundamental agents are necessary to keep the market reasonably stable.

4.3.4 Fundamental Agent

Fundamental agents are traders who measure the intrinsic value by analyzing the accounting, finance, and economics of the stock [136]. They make decisions with the belief that the stock price will return to its fundamental value in the long-run. The fundamental value of the stock f_t is public to all agents and is assumed to follow a random walk process

$$f(t) = f(t-1) + \eta,$$

where the fundamental shocks $\eta \sim N(0, \sigma_\eta^2)$.

At each time step t, fundamental agents make a decision based on the difference between $S(t)$ and $f(t)$. Therefore, if the price is below (above) its fundamental value, the agents will make a buy (sell) order. This strategy shortens the difference between price and its fundamental value. They help to stabilize the market and cause the opposite effect on prices for technical agents. In order to account for individual heterogeneity, they are given heterogeneous beliefs about the fundamental price for more interesting dynamics [49].

Let N_{fa} denote the total number of fundamental agents. The trading rule of the fundamental agent $i, i \in N_{fa}$, is represented by

$$s_i^{fa}(t) = sign\Big[\kappa(f(t) - S(t)) + \varepsilon_i\Big],$$

where κ is a positive coefficient that describes the speed of adjustment. The idiosyncratic term ε_i embodies a random term that accounts for each individual's own interpretation. We take it as being normally distributed around zero and with a standard deviation controlled by the user, $\varepsilon_i \sim N(0, \sigma_\varepsilon^2)$.

Finally, all agents are put together in the artificial financial market. This includes N_{noise} noise agents, N_{ta} technical agents, and N_{fa} fundamental agents. The total number of traders is N,

$$N = N_{noise} + N_{ta} + N_{fa}.$$

4.4 Bayesian Parameter Optimization

4.4.1 Data Description

As an empirical basis, we used a data set that covers 6 years of daily quotes of the DJIA Index, from of January 2013 to January 2019. This period contains 1508 daily returns, of which the main statistical properties are given in Table 4.1. The analysis focused on stylized facts of the distribution of stock returns and the volatility clustering in the market. Based on this data set, we estimated the proportion of each type of trader in our artificial stock market.

Table 4.1 Statistical properties of the actual returns

Mean	Standard Deviation	Kurtosis	Skewness	Min	Max
0.03%	8.09%	4.31	−0.64	−5.07%	3.81%

4.4.2 Parameter Optimization

We constructed an artificial stock market of N traders. Each type of trader has a minimum number of 10 and a maximum of 100. The trading period was considered as days, thus the results can be compared with properties of daily returns. We chose $\lambda \in [5, 20]$ which means the minimum value of daily returns was 5% and the maximum value of daily returns was 20%. We denote $1/p_i$ as the latency of an agent i when making his decision $\theta_i(t)$. p was

chosen within $[0.001, 0.1]$ to be interpreted as days, for example, $p = 0.1$ means the average updating period of agents was 10 days. The number of days each technical agent collected to calculate profit was chosen within $[5, 22]$, which describes the interest in the returns of traders in the range from 1 week to 1 month. The short and long windows for the technical strategy of all technical traders are given in Section 2, $l_A = 35, l_B = 51$. The amplitude σ_I of the noise of the news arrival process can be chosen in the range $[0.001, 0.01]$ to reproduce a realistic range of values for the volatility. The standard deviation of the noise process in fundamental value, σ_η, was calculated from the standard deviation of the real price ($\sigma_\eta^p = 3255.43$), which is in $[0.5\sigma_\eta^p, 1.5\sigma_\eta^p]$. The value of the standard deviation in idiosyncratic term was chosen in the range $[0.001, 1]$. The results discussed in the next section are generic within these parameters' ranges.

Table 4.2 Description of parameters

Parameters	**Description**	**Value**
N_{noise}	Number of noise agents	$[10, 100]$
N_{ta}	Number of technical agents	$[10, 100]$
N_{fa}	Number of fundamental agents	$[10, 100]$
λ	Market depth	$[5, 20]$
p	Fraction of agents updating their news reaction	$[0.001, 0.1]$
n	Number of days each technical agent collected to calculate profit	$[5, 22]$
κ	Adjustment speed	$[0.001, 1]$
σ_I	Standard deviation of the noise representing the news arrival process	$[0.001, 0.01]$
σ_η	Standard deviation of the noise process in fundamental value	$[1627.72, 4883.15]$
σ_ε	Standard deviation of idiosyncratic term	$[0.01, 1]$

For clarity, we summarize the parameters and search ranges used in our model in Table 4.2. We formulated an optimization problem using the *hyperopt* package of Python with the three main ingredients:

- Domain space: Each trader was assigned initial values derived from random uniform distributions within the range of Table 4.2, respectively.

- Objective function: KL divergence and KS test were used as the evaluation metrics of choice. For this run, the algorithm found the best value of parameters (the one which minimized the loss) in just under 100 trials.
- Surrogate function and selection function: Tree-structured Parzen Estimator was used as the surrogate function. Then, the Expected Improvement criterion was used as the selection function.

Both KL divergence and KS test found the optimal set of parameters with small simulation error as shown in Table 4.3. The proportion of each type of trader with KL and KS tests were: $N_{noise} : 12.2\%, N_{ta} : 78.3\%, N_{fa} : 9.6\%$ and $N_{noise} : 13.6\%, N_{ta} : 75.6\%, N_{fa} : 10.6\%$, respectively. Note that we did not compare values of KL and KS due to differences in measurement definitions. In this work, we picked out the better objective function by investigating their stability over many simulations. KL divergence represents the instability in many simulations; the minimum value ranged from 0.08 to 4.5. In contrast, the KS test showed stability with small fluctuations in the range of 0.03 to 0.06. The stylized facts which we found to be present in the simulated returns are discussed below.

Table 4.3 Optimized parameters of two objective functions

Method	**Optimal Value**	**Parameters**									
		N_{noise}	N_{ta}	N_{fa}	λ	p	n	κ	σ_I	σ_η	σ_ε
KL	0.08	14	90	11	20	0.061	16	1	0.001	2700	0.004
KS	0.03	18	100	14	15	0.1	9	1	0.001	2000	0.003

Moreover, we considered many different sets, which contained fixed numbers of different types of traders in ABM. The results in Table 4.4 show that the KS test had the lowest value when there were three types of traders in the market. Thus, the KS test is an effective objective function to measure simulation errors in the Bayesian optimization algorithm of our model.

Table 4.4 Comparing the KS test results of different fixed numbers of types of traders

KS Test	Trader Set		
	N_{noise}	N_{ta}	N_{fa}
0.10	100	0	0
0.22	0	100	0
0.13	100	100	0
0.25	0	100	100
0.06	100	100	100

4.5 Result and Discussion

In this section, we consider a stock market with three types of trader: noise , technical, and fundamental traders, with the KS test as the objective function. In the first stage, only the noise traders joined the market since the technical traders and fundamental traders need to collect the historical data. In the second stage, after a few days (here we chose the number of days equal to the optimal long window), technical traders and fundamental traders joined the market. The daily movement of actual return and simulated return is shown in Figure 4.2.

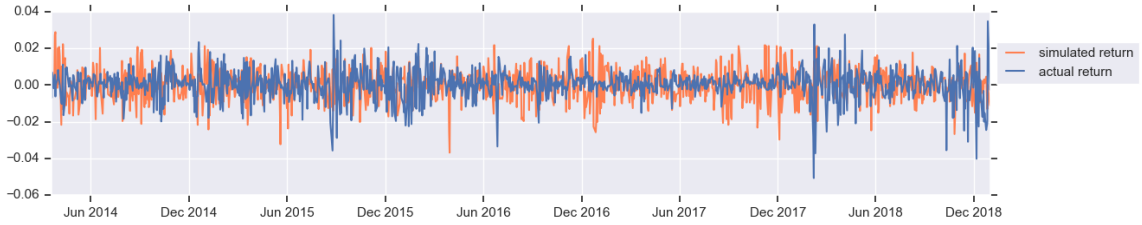


Fig. 4.2 Simulation of returns

4.5.1 Distribution of Returns

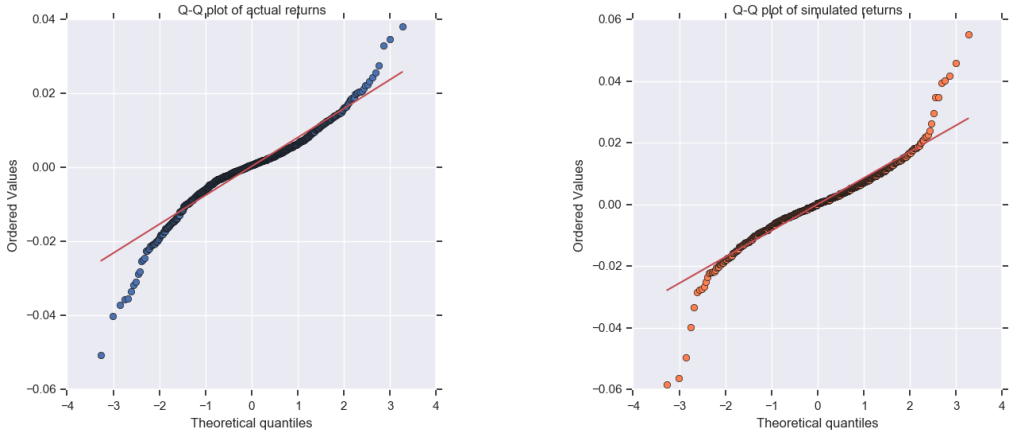
The statistical analysis of the actual returns and simulated returns is given in Tables 4.5. The simulated returns had similar properties to the real ones. In financial data, the heavy tail of the distribution of returns is displayed by a positive kurtosis [90]. We found that the Shapiro-Francia test on actual returns and simulated returns rejected the normality at a significance level of 0.1% with p-values of 2.2×10^{-22} and 1.5×10^{-12}, respectively. The quantile-quantile (Q-Q) plots in Figure 4.3 are also show the same results. In Figure 4.4a, the distribution of actual and simulated returns do not display the heavy tail clearly on linear axes. Thus, we change the vertical axis to use logarithmic scaling in Figure 4.4b to zoom in

on the tails. The returns produced by the model were found to be consistently leptokurtically distributed with a heavy tail.

Table 4.5 Comparing statistical properties of the actual returns and simulated returns

Data	**Mean**	**Standard Deviation**	**Kurtosis**	**Skewness**
Real returns	0.03%	8.09%	4.31	−0.64
Simulation	0.02%	8.09%	2.32	−0.08

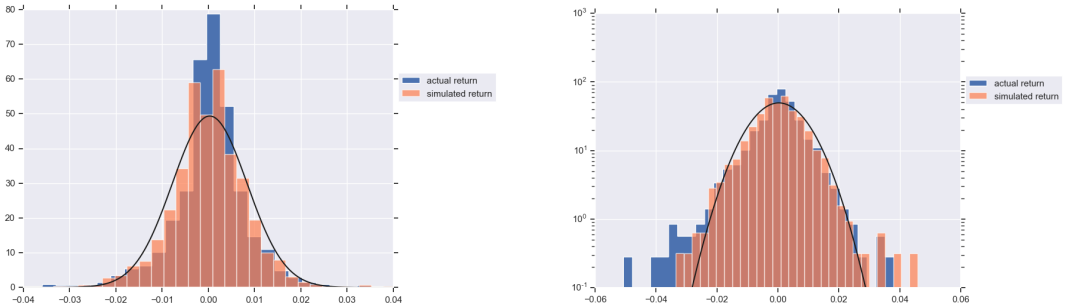
Data	**Min**	**Quantiles**			**Max**
		25%	**50%**	**75%**	
Real returns	−5.07%	−2.99%	0.06%	4.14%	3.81%
Simulation	−3.35%	−4.30%	0.03%	4.92%	4.57%



(a) Q-Q plot of actual returns

(b) Q-Q plot of simulated returns

Fig. 4.3 Quantile-quantile plot of returns



(a) Linear scales on vertical axis (b) Logarithmic scales on vertical axis

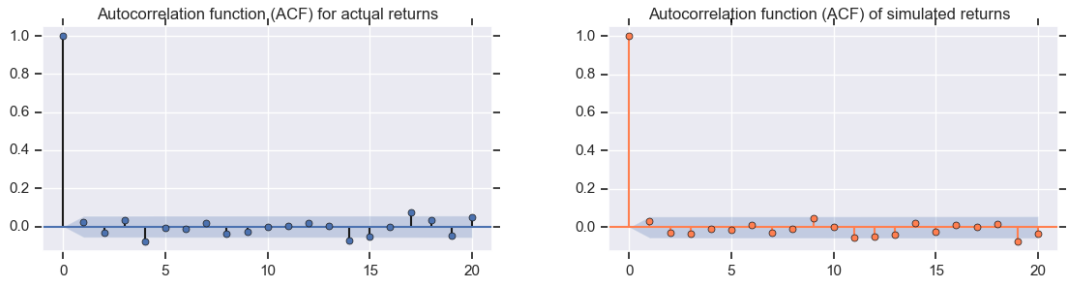
Fig. 4.4 Distribution of actual and simulated returns

4.5.2 Absence of Autocorrelations in Returns

Figure 4.2 shows the plots of stock returns for both actual data and simulated data. We studied the stationarity of the stock returns by using an augmented Dickey–Fuller (ADF) unit root test. The results are shown in Table 4.6. The p-values of the ADF test statistic for the real returns and the simulated returns are 2.31×10^{-30} and 3.44×10^{-30}, respectively. Since both values are smaller than 0.01, we rejected the null hypothesis that they are non-stationary. In addition, the autocorrelation function (ACF) plots in Figure 4.5 show the same results, and so we can conclude that there was no correlation between the returns and its lags.

Table 4.6 ADF unit root test for real returns and simulated returns

Data	**ADF Statistic**	***p*-Value**	**Critical Values**		
			1%	**5%**	**10%**
Real Returns	-18.28	2.31×10^{-30}	-3.44	-2.86	-2.57
Simulation	-17.73	3.44×10^{-30}	-3.44	-2.86	-2.57

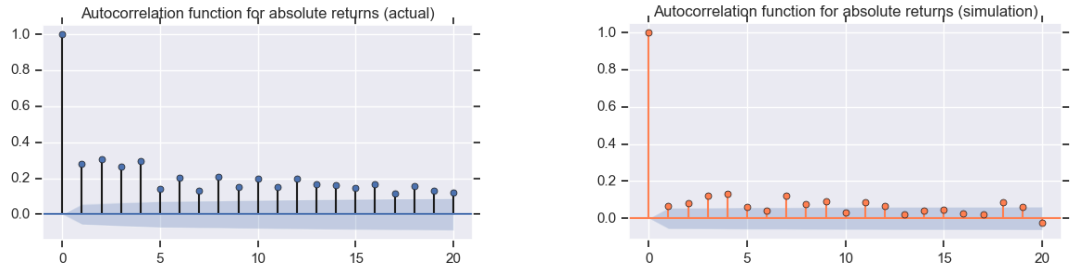


(a) Autocorrelations of actual returns (b) Autocorrelations of simulated returns

Fig. 4.5 Autocorrelations of actual and simulated returns

4.5.3 Volatility Clustering

The next stylized fact we studied was volatility clustering, which describes the the tendency of large changes in prices to cluster together. The volatility clustering can be seen in Figure 4.6, where the ACFs of the absolute value of returns decay slowly. Notice that for both cases, the autocorrelations of absolute returns remained significantly positive over many time lags. This is clear evidence of volatility clustering.



(a) Autocorrelations of absolute actual returns (b) Autocorrelations of absolute simulated returns

Fig. 4.6 Autocorrelation of absolute returns

4.6 Conclusion

In this work we proposed a novel approach to optimize the parameters in an agent-based model with different types of traders. The Bayesian optimiation algorithm with Kolmogorov–Smirnov test proposed an optimal set of parameters. The model was able to highlight some of the important stylized facts of stock returns, such as the leptokurtosis, the heavy tail of the returns, and volatility clustering.

Our Bayesian approach differs from recent studies on the parameter optimization step of ABM in financial markets [120, 82, 3]. In the financial market model, the parameter optimization process considers large parameter sets which can become quite a complex and costly operation. In particular, in [120], the authors generated 400,000 initial points to calibrate their model. In [82], the amount of data was reduced due to the high computational complexity of the Island model. Our approach is based on Bayesian rules and looks at previous experience to scale down the search space. Thus, the algorithm requires less time and iterations to get the optimal parameters. Moreover, the optimized results showed that fundamental traders accounted for 9%–11% of all traders in the United States stock market. The proportion of traders in the stock market provides an additional indicator so that traders can come up with effective strategies.

Although Bayesian approaches provide several advantages, the complexity of the objective function could be a problem since it still has its own costs. In our model, the Sharpe ratio used as the objective function is very simple to calculate. However, if we want to consider a complex model with more types of investors and interactions between them, the dimension of domain space will increase. Optimal models require different settings between datasets, and with a high-dimensional space, the correlation between parameters is difficult to study.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis concentrates on studying multiple approaches to parameter optimization for agent-based models and their application in financial markets. The research objective is divided into three sections.

First of all, we studied the trading behavior of agents based on technical analysis. The basic concepts and steps involved in the trading process of a trader are introduced. We analyzed trends to get an overview of the market. When combined with indicators, traders can clearly identify entry and exit points and earn profits when they trade with the trend and not against it. With common trading strategies and simple objective functions, Bayesian Optimization and Deep Reinforcement Learning are two proposed approaches to find the parameter set with the best performance. These new approaches are compared with approaches from previous studies such as Genetic Algorithms, parametric modeling, and Buy and Hold strategies. Three common indicators, RSI, BOLL and MACD, are used to build simple strategies and are suitable for real markets. The buying and selling behavior as well as the transaction cost value are also retained. We also conducted many experiments with the data of different markets, such as the crypto spot market, stock market, and crypto futures market. The experiment has yielded some interesting results. For short-term trading purposes, the Deep Reinforcement Learning method shows superior results in terms of cumulative returns, volatility, and execution time when compared to the Bayesian Optimization method. This helps traders to make quick and efficient decisions with the latest information from the market. In long-term trading, Bayesian Optimization is a method of parameter optimization that yields higher returns. These results do not change when comparing results on data sets of different lengths. On the extension aspect, Deep Reinforcement Learning can provide a solution to the height problem of Bayesian Optimization, so the studies can be extended

with more complex strategies and objective functions. However, in order to apply the Deep Reinforcement Learning algorithm to models other than trading strategies, it is necessary to rebuild the environment and define completely new states, actions, and rewards. Therefore, Bayesian Optimization is used to optimize the parameters in the agent-based model for financial markets in our researches. With this knowledge, we add technical traders with optimized trading strategies to our proposed agent-based model as technical agents.

Secondly, we have proposed a novel approach for parameter optimization in an agent-based model of the financial market. We have proposed different objective functions to tune hyperparameters in an agent-based simulation of the stock market. The Bayesian optimization with the Kolmogorov-Smirnov test has proposed an optimal set of parameters with high stability. The algorithm can handle large volumes of data with shorter execution times than previous methods in this field. In particular, this offers significant contributions when computational costs become expensive. Therefore, Bayesian optimization can be used to speed up parameter optimization in financial multi-agent models. The model with a return-based objective algorithm was able to highlight some of the important stylized facts of the stock returns, such as the absence of autocorrelations, the heavy tail of the returns, and volatility clustering. Besides, the dynamics of stock prices in the real market are reproduced when simulated with the price-based objective algorithm. In the future, we can adjust to combine the two objective functions to get results for both prices and returns. This work is only the first step toward fully assessing the properties of agent-based models in the United States stock market. In future research, the model can be applied to other exchanges such as the New York Stock Exchange or emerging markets in Asia.

Finally, we tried to replicate and extend ABMs from the literature. However, our research goal goes beyond replicating these experiments. We aim to improve the understanding of stylized facts in the market. Therefore, taking advantage of the proposed model's properties, we extend the model with multiple agents and their interactions, and analyze the market dynamics within it. The main focus of the analysis is to determine the proportion of traders in the market. The structure of the market and the factors that we consider important are introduced when building the model. We study the trading behavior of market participants with different roles, such as noise traders, technical traders, fundamental traders, and market makers. From the introduced parameter optimization methods, we apply Bayesian optimization with various objective functions to provide estimated parameters quickly and introduce interesting stylized facts. The optimized results showed that fundamental traders accounted for 9%–11% of all traders in the United States stock market. This ratio is an interesting piece of information that can serve as an indicator for the trader to come up with effective strategies. Although Bayesian approaches provide several advantages, the complexity of

the objective function could be a problem since it still has its own costs. In our model, the Sharpe ratio used as the objective function is very simple to calculate. However, if we want to consider a complex model with more types of investors and interactions between them, the dimension of domain space will increase. Optimal models require different settings between datasets, and with a high-dimensional space, the correlation between parameters is difficult to study.

Publications related to this thesis include:

1. Tran, M., Duong, T., Pham-Hi, D., & Bui, M. (2020). Detecting the proportion of traders in the stock market: an agent-based approach. Mathematics, 8(2), 198.

2. Tran, M., Ngo, M., Pham-Hi, D., & Bui, M. (2020). Bayesian Calibration of Hyperparameters in Agent-Based Stock Market. In 2020 RIVF International Conference on Computing and Communication Technologies (RIVF) (pp. 1-6). IEEE.

3. Tran, M., Pham-Hi, D., & Bui, M. (2022). Hyperparameter Tuning with Different Objective Functions in Financial Market Modeling. In International Econometric Conference of Vietnam 10-12 January 2022 (pp. 733-745). Springer, Cham.

4. Tran, M., Pham-Hi, D., & Bui, M. (2022). Parameter Optimization for Trading Algorithms of Technical Agents. In 2022 International Conference on Computing and Communication Technologies (RIVF). IEEE.

5. Tran, M., Pham-Hi, D., & Bui, M. (2023). Optimizing Automated Trading Systems with Deep Reinforcement Learning. Algorithms, 16(1), 23.

5.2 Future Researches

The research work in this thesis can be extended in the following directions in the future:

1. Multi-objective optimization: Multi-objective optimization problems arise regularly in the real world where two or more objectives are required to be optimized simultaneously. In the financial market in general or in the trader's trading strategy in particular, the combination of profit and risk objective functions always attracts a lot of attention. To optimize these complex objective functions, carefully crafted evolutionary strategies and heuristics can improve performance. However, recent advances in machine learning algorithms have shown the ability to replace humans as the algorithms' engineers to solve various problems. While deep neural networks focus on making predictions, deep reinforcement learning is mainly used to learn how to make decisions. We therefore

believe that deep reinforcement learning is a viable way to learn how to solve various optimization problems automatically, thus not requiring human-designed evolutionary strategies and experiences.

2. Deep learning in agent-based models: In the current hot topics of machine learning, the development of AI, machine learning, and theories of human decision making are closely related. When building an ABM model, the concept of an artificial agent can be introduced through behaviors simulated by machine learning models. Moreover, when optimizing parameters for the ABM model, a Deep Reinforcement Learning approach can be applied. The objective of this research is to develop new computational methods to improve the applicability of macroeconomic ABMs to economic policy analysis. When successful, we will significantly reduce the complexity and computational load of ABM simulations and offer new methods for modeling the behavior of economic agents.

3. Considering a complex fast-time trading simulator with a variety of strategies: Simple and popular algorithms have been presented in our study. From the results obtained, a more complex trading system can be considered, such as optimizing a multi-asset portfolio, incorporating social media information and macroeconomic market news into investment strategies.

References

[1] Abarbanell, J. S. and Bushee, B. J. (1997). Fundamental analysis, future earnings, and stock prices. *Journal of accounting research*, 35(1):1–24.

[2] Achelis, S. B. (2001). Technical analysis from a to z.

[3] Akram, W., Niazi, M. A., Iantovics, L. B., and Vassilakos, A. V. (2019). Towards agent-based model specification of smart grid: A cognitive agent-based computing approach. *Interdisciplinary Description of Complex Systems: INDECS*, 17(3-B):546–585.

[4] Alhashel, B. S., Almudhaf, F. W., and Hansz, J. A. (2018). Can technical analysis generate superior returns in securitized property markets? evidence from east asia markets. *Pacific-Basin Finance Journal*, 47:92–108.

[5] Ali, F. Ö. G. and Wallace, W. A. (1997). Bridging the gap between business objectives and parameters of data mining algorithms. *Decision Support Systems*, 21(1):3–15.

[6] Anderson, P. W. (2018). *The economy as an evolving complex system*. CRC Press.

[7] Arthur, W. B. (2018). *The economy as an evolving complex system II*. CRC Press.

[8] Arthur, W. B., Holland, J. H., LeBaron, B., Palmer, R., and Tayler, P. (1996). Asset pricing under endogenous expectations in an artificial stock market. *The economy as an evolving complex system II*, 27.

[9] Arthur, W. B., Holland, J. H., LeBaron, B., Palmer, R., and Tayler, P. (2018). *Asset pricing under endogenous expectations in an artificial stock market*. CRC Press.

[10] Axelrod, R. (1997). Advancing the art of simulation in the social sciences. In *Simulating social phenomena*, pages 21–40. Springer.

[11] Bazzana, D., Colturato, M., Savona, R., et al. (2021). Learning about unprecedented events: Agent-based modelling and the stock market impact of covid-19. Technical report, JSTOR.

[12] Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24.

[13] Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).

[14] Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, USA:.

[15] Betrò, B. (1991). Bayesian methods in global optimization. *Journal of Global Optimization*, 1(1):1–14.

[16] Bigiotti, A. and Navarra, A. (2018). Optimizing automated trading systems. In *The 2018 International Conference on Digital Science*, pages 254–261. Springer.

[17] Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the national academy of sciences*, 99(suppl 3):7280–7287.

[18] Bouchaud, J.-P. (2008). Economics needs a scientific revolution. *Nature*, 455(7217):1181–1181.

[19] Bouchaud, J.-P. and Potters, M. (2001). More stylized facts of financial markets: leverage effect and downside correlations. *Physica A: Statistical Mechanics and its Applications*, 299(1-2):60–70.

[20] Brock, W., Lakonishok, J., and LeBaron, B. (1992). Simple technical trading rules and the stochastic properties of stock returns. *The Journal of finance*, 47(5):1731–1764.

[21] Brown, D. P. and Jennings, R. H. (1989). On technical analysis. *The Review of Financial Studies*, 2(4):527–551.

[22] Campbell, J. Y. and Cochrane, J. H. (1999). By force of habit: A consumption-based explanation of aggregate stock market behavior. *Journal of political Economy*, 107(2):205–251.

[23] Challet, D., Marsili, M., and Zhang, Y.-C. (2001). Stylized facts of financial markets and market crashes in minority games. *Physica A: Statistical Mechanics and its Applications*, 294(3-4):514–524.

[24] Challet, D., Marsili, M., and Zhang, Y.-C. (2004). *Minority games: interacting agents in financial markets*. OUP Oxford.

[25] Chan, E. P. (2021). *Quantitative trading: how to build your own algorithmic trading business*. John Wiley & Sons.

[26] Chen, S.-H. and Yeh, C.-H. (2002). On the emergent properties of artificial stock markets: the efficient market hypothesis and the rational expectations hypothesis. *Journal of Economic Behavior & Organization*, 49(2):217–239.

[27] Cheng, E. (2017). Just 10% of trading is regular stock picking, JPMorgan Estimates. Available from: https://www.cnbc.com/2017/06/13/death-of-the-human-investor-just-10-percent-of-trading-is-regular-stock-picking-jpmorgan-estimates.html.

[28] Cividino, D., Westphal, R., and Sornette, D. (2021). Multi-asset financial bubbles in an agent-based model with noise traders' herding described by an n-vector ising model. *Swiss Finance Institute Research Paper*, (21-76).

[29] Claesen, M. and De Moor, B. (2015). Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*.

[30] Cocco, L., Tonelli, R., and Marchesi, M. (2019). An agent based model to analyze the bitcoin mining activity and a comparison with the gold mining industry. *Future Internet*, 11(1):8.

[31] Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative finance*, 1(2):223.

[32] Cont, R. (2007). Volatility clustering in financial markets: empirical facts and agent-based models. In *Long memory in economics*, pages 289–309. Springer.

[33] Cristelli, M., Pietronero, L., and Zaccaria, A. (2011). Critical overview of agent-based models for economics. *arXiv preprint arXiv:1101.1847*.

[34] Datta, D., Deb, K., and Fonseca, C. M. (2006). Solving class timetabling problem of iit kanpur using multi-objective evolutionary algorithm. *KanGAL, Report*, 2006006:1–10.

[35] Daulton, S., Eriksson, D., Balandat, M., and Bakshy, E. (2022). Multi-objective bayesian optimization over high-dimensional search spaces. In *Uncertainty in Artificial Intelligence*, pages 507–517. PMLR.

[36] De Grauwe, P. and Dewachter, H. (1993). A chaotic model of the exchange rate: the role of fundamentalists and chartists. *Open economies review*, 4(4):351–379.

[37] De Long, J. B., Shleifer, A., Summers, L. H., and Waldmann, R. J. (1990). Noise trader risk in financial markets. *Journal of political Economy*, 98(4):703–738.

[38] Dimic, N., Kiviaho, J., Piljak, V., and Äijö, J. (2016). Impact of financial market uncertainty and macroeconomic factors on stock–bond correlation in emerging markets. *Research in International Business and Finance*, 36:41–51.

[39] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

[40] Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., Leyton-Brown, K., et al. (2013). Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, volume 10.

[41] Ehrentreich, N. (2006). Technical trading in the santa fe institute artificial stock market revisited. *Journal of Economic Behavior & Organization*, 61(4):599–616.

[42] Faloutsos, M., Faloutsos, P., and Faloutsos, C. (1999). On power-law relationships of the internet topology. In *ACM SIGCOMM computer communication review*, volume 29, pages 251–262. ACM.

[43] Fama, E. F. (2021). Efficient capital markets a review of theory and empirical work. *The Fama Portfolio*, pages 76–121.

[44] Fang, F., Ventre, C., Basios, M., Kanthan, L., Martinez-Rego, D., Wu, F., and Li, L. (2022). Cryptocurrency trading: a comprehensive survey. *Financial Innovation*, 8(1):1–59.

[45] Farmer, J. D. and Foley, D. (2009). The economy needs agent-based modelling. *Nature*, 460(7256):685–686.

[46] Farmer, J. D., Gallegati, M., Hommes, C., Kirman, A., Ormerod, P., Cincotti, S., Sanchez, A., and Helbing, D. (2012). A complex systems approach to constructing better models for managing financial markets and the economy. *The European Physical Journal Special Topics*, 214(1):295–324.

[47] Fayek, M. B., El-Boghdadi, H. M., and Omran, S. M. (2013). Multi-objective optimization of technical stock market indicators using gas. *International Journal of Computer Applications*, 68(20).

[48] Fenton-O'Creevy, M., Soane, E., Nicholson, N., and Willman, P. (2011). Thinking, feeling and deciding: The influence of emotions on the decision making and performance of traders. *Journal of Organizational Behavior*, 32(8):1044–1061.

[49] Ferreira, F. F., de Oliveira, V. M., Crepaldi, A. F., and Campos, P. R. (2005). Agent-based model with heterogeneous fundamental prices. *Physica A: Statistical Mechanics and its Applications*, 357(3-4):534–542.

[50] Forrester, J. W. (2009). Some basic concepts in system dynamics. *Sloan School of Management, Massachusetts Institute of Technology, Cambridge*, 9.

[51] Frankel, J. A., Froot, K. A., et al. (1986). Understanding the us dollar in the eighties: the expectations of chartists and fundamentalists. *Economic record*, 62(1):24–38.

[52] Fu, W., Nair, V., and Menzies, T. (2016). Why is differential evolution better than grid search for tuning defect predictors? *arXiv preprint arXiv:1609.02613*.

[53] Garifullin, M., Borshchev, A., and Popkov, T. (2007). Using anylogic and agent-based approach to model consumer market. In *Proceedings of the 6th EUROSIM Congress on Modelling and Simulation*, pages 1–5.

[54] Gavrilov, M., Anguelov, D., Indyk, P., and Motwani, R. (2000). Mining the stock market (extended abstract) which measure is best? In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 487–496.

[55] Gazit, E., Yair, Y., and Chen, D. (2005). Emerging conceptual understanding of complex astronomical phenomena by using a virtual solar system. *Journal of Science Education and Technology*, 14(5-6):459–470.

[56] Gen, M. and Cheng, R. (1999). *Genetic algorithms and engineering optimization*, volume 7. John Wiley & Sons.

[57] Gode, D. K. and Sunder, S. (1994). Human and artificially intelligent traders in computer double auctions. *Computational organization theory*, pages 241–262.

[58] Goldberg, D. E. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of genetic algorithms*, volume 1, pages 69–93. Elsevier.

[59] Gonçalves, C. (2003). Artificial financial market model. *Available from WWW:< http://ccl. northwestern. edu/netlogo/models/community/Artificial Financial Market Model*.

[60] Graham, B., Dodd, D. L. F., Cottle, S., et al. (1934). *Security analysis*. McGraw-Hill New York.

[61] Guessoum, Z. (2004). Adaptive agents and multiagent systems. *IEEE Distributed Systems Online*, 5(7).

[62] Harris, L. (2003). *Trading and exchanges: Market microstructure for practitioners*. OUP USA.

[63] Hessary, Y. K. and Hadzikadic, M. (2017). Role of behavioral heterogeneity in aggregate financial market behavior: An agent-based approach. *Procedia Computer Science*, 108:978–987.

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

[65] Isnard, C. and Zeeman, E. C. (1976). Some models from catastrophe theory in the social sciences. *The use of models in the social sciences*, pages 44–100.

[66] Jeffreys, H. (1998). *The theory of probability*. OUP Oxford.

[67] Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N., and Barabási, A.-L. (2000). The large-scale organization of metabolic networks. *Nature*, 407(6804):651.

[68] John, H. (1992). Holland. genetic algorithms. *Scientific american*, 267(1):44–50.

[69] Johnson, N. F., Jefferies, P., Hui, P. M., et al. (2003). Financial market complexity. *OUP Catalogue*.

[70] Jomaa, H. S., Grabocka, J., and Schmidt-Thieme, L. (2019). Hyp-rl: Hyperparameter optimization by reinforcement learning. *arXiv preprint arXiv:1906.11527*.

[71] Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383.

[72] 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*.

[73] Katz, J. O. and McCORMICK, D. L. (2000). *The encyclopedia of trading strategies*. McGraw-Hill New York.

[74] Khan, M. A., Ahmed, M., Popp, J., and Oláh, J. (2020). Us policy uncertainty and stock market nexus revisited through dynamic ardl simulation and threshold modelling. *Mathematics*, 8(11):2073.

[75] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[76] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

[77] Kirman, A. (1991). Epidemics of opinion and speculative bubbles in financial matkets. *Money and financial markets chap*, 17.

[78] Kirman, A. P. (1992). Whom or what does the representative individual represent? *Journal of economic perspectives*, 6(2):117–136.

[79] Kousiouris, G., Cucinotta, T., and Varvarigou, T. (2011). The effects of scheduling, workload type and consolidation scenarios on virtual machine performance and their prediction through optimized artificial neural networks. *Journal of Systems and Software*, 84(8):1270–1291.

[80] Kuo, R. J., Chen, C., and Hwang, Y. (2001). An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network. *Fuzzy sets and systems*, 118(1):21–45.

[81] Kuo, W.-Y. and Lin, T.-C. (2013). Overconfident individual day traders: Evidence from the taiwan futures market. *Journal of banking & Finance*, 37(9):3548–3561.

[82] Lamperti, F., Roventini, A., and Sani, A. (2018). Agent-based model calibration using machine learning surrogates. *Journal of Economic Dynamics and Control*, 90:366–389.

[83] Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480.

[84] Larsen, J., Hansen, L. K., Svarer, C., and Ohlsson, M. (1996). Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*, pages 62–71. IEEE.

[85] LeBaron, B. (2001). A builder's guide to agent-based financial markets. *Quantitative finance*, 1(2):254.

[86] Levy, H., Levy, M., and Solomon, S. (2000). *Microscopic simulation of financial markets: from investor behavior to market phenomena*. Elsevier.

[87] Liu, Y., Liu, Q., Zhao, H., Pan, Z., and Liu, C. (2020). Adaptive quantitative trading: An imitative deep reinforcement learning approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2128–2135.

[88] Lucarelli, G. and Borrotti, M. (2019). A deep reinforcement learning approach for automated cryptocurrency trading. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 247–258. Springer.

[89] Lucey, B. M. and Dowling, M. (2005). The role of feelings in investor decision-making. *Journal of economic surveys*, 19(2):211–237.

[90] Lux, T. and Alfarano, S. (2016). Financial power laws: Empirical evidence, models, and mechanisms. *Chaos, Solitons & Fractals*, 88:3–18.

[91] Lux, T. and Marchesi, M. (1999). Scaling and criticality in a stochastic multi-agent model of a financial market. *Nature*, 397(6719):498.

[92] Lux, T. and Marchesi, M. (2000). Volatility clustering in financial markets: a microsimulation of interacting agents. *International journal of theoretical and applied finance*, 3(04):675–702.

[93] Lux, T. and Westerhoff, F. (2009). Economics crisis. *Nature Physics*, 5(1):2–3.

[94] Ma, C., Zhang, J., Liu, J., Ji, L., and Gao, F. (2021). A parallel multi-module deep reinforcement learning algorithm for stock trading. *Neurocomputing*, 449:290–302.

[95] Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, Georgia, USA.

[96] Madhavan, A. (2000). Market microstructure: A survey. *Journal of financial markets*, 3(3):205–258.

[97] Marsaglia, G., Tsang, W. W., Wang, J., et al. (2003). Evaluating kolmogorov's distribution. *Journal of statistical software*, 8(18):1–4.

[98] Martin, A. D. (2001). Technical trading rules in the spot foreign exchange markets of developing countries. *Journal of Multinational Financial Management*, 11(1):59–68.

[99] Matias Pinto, J., Ferreira Neves, R., and Horta, N. (2014). Multi-objective optimization of investment strategies. In *Proceedings of the 16th International Conference on Enterprise Information Systems-Volume 1*, pages 480–488.

[100] Millea, A. (2021). Deep reinforcement learning for trading—a critical survey. *Data*, 6(11):119.

[101] Moriconi, R., Deisenroth, M. P., and Sesh Kumar, K. (2020). High-dimensional bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109(9):1925–1943.

[102] Müller, U. A., Dacorogna, M. M., Olsen, R. B., Pictet, O. V., Schwarz, M., and Morgenegg, C. (1990). Statistical study of foreign exchange rates, empirical evidence of a price change scaling law, and intraday analysis. *Journal of Banking & Finance*, 14(6):1189–1208.

[103] Murphy, J. J. (1999). *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin.

[104] Mutny, M. and Krause, A. (2018). Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. *Advances in Neural Information Processing Systems*, 31.

[105] Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, 7(4):308–313.

[106] Ni, J., Cao, L., and Zhang, C. (2008). Evolutionary optimization of trading strategies. In *Applications of Data Mining in E-Business and Finance*, pages 11–24. IOS Press.

[107] Ni, J. and Zhang, C. (2005). An efficient implementation of the backtesting of trading strategies. In *International Symposium on Parallel and Distributed Processing and Applications*, pages 126–131. Springer.

[108] Nti, I. K., Adekoya, A. F., and Weyori, B. A. (2020). A systematic review of fundamental and technical analysis of stock market predictions. *Artificial Intelligence Review*, 53(4):3007–3057.

[109] Pagani, G. A. and Aiello, M. (2013). The power grid as a complex network: a survey. *Physica A: Statistical Mechanics and its Applications*, 392(11):2688–2700.

[110] Pardo, R. (2011). *The evaluation and optimization of trading strategies*, volume 314. John Wiley & Sons.

[111] Petrusheva, N. and Jordanoski, I. (2016). Comparative analysis between the fundamental and technical analysis of stocks. *Journal of Process Management and New Technologies*, 4(2):26–31.

[112] Pinakin, S. N. and Manubhai, P. T. (2015). A comparative study on technical analysis by bollinger band and rsi. *International Journal in Management & Social Science*, 3(6):234–251.

[113] Plieger, T., Grünhage, T., Duke, É., and Reuter, M. (2021). Predicting stock market performance: The influence of gender and personality on financial decision making. *Journal of Individual Differences*, 42(2):64.

[114] Plott, C. R. and Sunder, S. (1982). Efficiency of experimental security markets with insider information: An application of rational-expectations models. *Journal of political economy*, 90(4):663–698.

[115] Poteshman, A. M. and Serbin, V. (2003). Clearly irrational financial market behavior: Evidence from the early exercise of exchange traded stock options. *The Journal of Finance*, 58(1):37–70.

[116] Powell, M. J. (1994). A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*, pages 51–67. Springer.

[117] Pricope, T.-V. (2021). Deep reinforcement learning in quantitative algorithmic trading: A review. *arXiv preprint arXiv:2106.00123*.

[118] Pring, M. J. (2002). *Study guide for technical analysis explained*, volume 5. McGraw-Hill New York.

[119] Ratner, M. and Leal, R. P. (1999). Tests of technical trading strategies in the emerging equity markets of latin america and asia. *Journal of Banking & Finance*, 23(12):1887–1905.

[120] Recchioni, M. C., Tedeschi, G., and Gallegati, M. (2015). A calibration procedure for analyzing stock price dynamics in an agent-based framework. *Journal of Economic Dynamics and Control*, 60:1–25.

[121] Schwager, J. D. (1984). *A complete guide to the futures markets: fundamental analysis, technical analysis, trading, spreads, and options*. John Wiley & Sons.

[122] Sewell, M. (2011). Characterization of financial time series. *Rn*, 11(01):01.

[123] Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

[124] Snow, D. (2020). Machine learning in asset management—part 1: Portfolio construction—trading strategies. *The Journal of Financial Data Science*, 2(1):10–23.

[125] Soni, N. and Kumar, T. (2014). Study of various crossover operators in genetic algorithms.

[126] Sornette, D. (2004). Why stock markets crash: critical events in complex financial systems. *Physics Today*, 57(3):78–79.

[127] Sornette, D. and Zhou, W.-X. (2006). Importance of positive feedbacks and overconfidence in a self-fulfilling ising model of financial markets. *Physica A: Statistical Mechanics and its Applications*, 370(2):704–726.

[128] Stigler, G. J. (1963). Public regulation of the securities markets. *Bus. Law.*, 19:721.

[129] Tauchen, G. E. and Pitts, M. (1983). The price variability-volume relationship on speculative markets. *Econometrica: Journal of the Econometric Society*, pages 485–505.

[130] Teall, J. L. (2018). *Financial trading and investing*. Academic Press.

[131] Tesfatsion, L. and Judd, K. L. (2006). *Handbook of computational economics: agent-based computational economics*. Elsevier.

[132] Tieleman, T. and Hinton, G. (2012). Neural networks for machine learning. *Coursera (Lecture 65-RMSprop)*, 138.

[133] Tran, M., Duong, T., Pham-Hi, D., and Bui, M. (2020). Detecting the proportion of traders in the stock market: An agent-based approach. *Mathematics*, 8(2):198.

[134] Tumminello, M., Lillo, F., Piilo, J., and Mantegna, R. N. (2012). Identification of clusters of investors from their real trading activity in a financial market. *New Journal of Physics*, 14(1):013041.

[135] Vassar, M., Atakpo, P., and Kash, M. J. (2016). Manual search approaches used by systematic reviewers in dermatology. *Journal of the Medical Library Association: JMLA*, 104(4):302.

[136] Walraven, J. C. (1981). Fundamental analysis of aggregate interlock. *Journal of the Structural Division*, 107(11):2245–2270.

[137] Westerhoff, F. (2010). A simple agent-based financial market model: direct interactions and comparisons of trading profits. In *Nonlinear Dynamics in Economics, Finance and Social Sciences*, pages 313–332. Springer.

[138] Westphal, R. (2021). *Agent-based models to understand, exploit and prevent financial bubbles*. PhD thesis, ETH Zurich.

[139] Westphal, R. and Sornette, D. (2020). Market impact and performance of arbitrageurs of financial bubbles in an agent-based model. *Journal of Economic Behavior & Organization*, 171:1–23.

[140] Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85.

[141] Wilder, J. W. (1978). *New concepts in technical trading systems*. Trend Research.

[142] Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., and Deng, S.-H. (2019). Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, 17(1):26–40.

[143] Xiong, Z., Liu, X.-Y., Zhong, S., Yang, H., and Walid, A. (2018). Practical deep reinforcement learning approach for stock trading. *arXiv preprint arXiv:1811.07522*.

[144] Yagi, I., Hoshino, M., and Mizuta, T. (2020). Analysis of the impact of maker-taker fees on the stock market using agent-based simulation. *arXiv preprint arXiv:2010.08992*.

[145] Yang, H., Wang, H. J., Sun, G. P., and Wang, L. (2015). A comparison of us and chinese financial market microstructure: heterogeneous agent-based multi-asset artificial stock markets approach. *Journal of Evolutionary Economics*, 25(5):901–924.

[146] Zhang, D. and Zhou, L. (2004). Discovering golden nuggets: data mining in financial application. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(4):513–522.

[147] Zhi-Hua, Z. (2008). Applications of data mining in e-business and finance: introduction. *Applications of Data Mining in E-Business and Finance*, 177:1.

Appendix A

Résume Long

A.1 Introduction

A.1.1 Motivation

L'étude des systèmes complexes a toujours été d'un grand intérêt en raison de la richesse de leurs propriétés et de leurs comportements, par exemple le système solaire [55] ou les systèmes biologiques [67]. En analysant ces systèmes naturels, les humains ont découvert des lois fondamentales qui peuvent aider à construire d'autres systèmes tels que les réseaux informatiques [42] et les réseaux électriques [109]. Les marchés financiers peuvent également être considérés comme des systèmes complexes [46, 69]. Le fonctionnement des marchés financiers fait l'objet de nombreux débats parmi les chercheurs et les experts financiers. En outre, le comportement sur les marchés financiers des investisseurs est également un sujet de préoccupation.

Ces dernières années, la modélisation financière basée sur les agents est devenue un domaine de recherche important pour développer et comprendre les phénomènes complexes observés dans les systèmes financiers. La simulation par ordinateurs des comportements économiques des agents ("Agent-based Computational Economics" ou ABCE), [131] et les approches de micro-simulation [86] ont été proposées pour étudier les propriétés émergentes des interactions des opérateurs de marchés (communément appelés "Traders"). Ces approches tentent de modéliser les marchés financiers comme des systèmes évolutifs d'agents concurrents interagissant de manière autonome, et mettent l'accent sur leur dynamique d'apprentissage [131]. Les modèles multi-agents offrent la possibilité de modéliser de manière transparente les problèmes de comportement et d'étudier ainsi l'effet du comportement des agents sur les prix des marchés. Ensuite, les prix des actifs dans les modèles peuvent être formés par l'interaction des acteurs du marché. En utilisant des agents pour étudier la

dynamique des marchés, le comportement hétérogène, la rationalité limitée et l'adaptativité des acteurs des marchés peuvent être représentés, et leurs impacts sur la dynamique des marchés peuvent être évalués.

Les résultats des études sur les marchés financiers ne sont pas exclusifs à la recherche universitaire [45], mais apportent également des contributions importantes aux acteurs du marché et aux décideurs politiques [138]. Le marché artificiel ainsi simulé sert d'environnement de test aux décideurs pour explorer l'impact de différentes politiques réglementaires sur l'amélioration de la prise de décision. En identifiant les paramètres sensibles qui affectent directement ou indirectement le marché financier, de nombreux chocs peuvent être analysés et les krachs boursiers peuvent être limités [61]. Dans [93], les auteurs soulignent l'importance de la modélisation des interactions économiques. Ils ont fait valoir que les régularités microéconomiques observées dans l'économie comportementale et les interactions fortes devraient constituer la base de modèles plus flexibles sur le plan méthodologique pour une perspective systémique sur les systèmes économiques mondialisés. En outre, les auteurs de [45] ont souligné que les modèles basés sur les agents sont nécessaires pour fournir une description quantitative de la manière dont l'économie est susceptible de réagir aux politiques gouvernementales dans différents scénarios. En testant les politiques dans des simulations contrôlées et répétées, ils peuvent guider les décideurs. Quant aux traders, ils doivent tester et analyser de nombreuses stratégies différentes et déterminer le moment optimal pour prendre des décisions. Les auteurs de [38] ont fait valoir qu'il est très important pour les traders de comprendre la dynamique des marchés financiers et les facteurs d'influence. Les traders présentent différents comportements d'investissement correspondant à leurs croyances et préférences. Cette hétérogénéité est considérée comme la principale source du comportement émergent complexe des marchés. Les résultats de [4] valide le pouvoir prédictif et la rentabilité des indicateurs techniques sur les marchés d'Indonésie, de Malaisie, de Taïwan et de Thaïlande. Cette capacité s'est avérée utile pour générer des rendements supérieurs aux rendements "Buy and Hold" même après avoir inclus les coûts de transaction, les ajustements pour le risque et l'espionnage des données. Dans [119], les auteurs ont examiné le profit potentiel des stratégies de trading techniques sur 10 marchés boursiers émergents d'Amérique latine et d'Asie et ont constaté que Taïwan, le Mexique et la Thaïlande apparaissent comme des marchés où les stratégies de trading techniques peuvent être rentables. Par conséquent, l'analyse des stratégies de trading des agents techniques sur un marché financier doit également être envisagée.

A.1.2 Questions de recherche

Sur la base des considérations ci-dessus, une question de recherche intéressante a été posée, et nous l'avons partiellement étudiée : est-ce que nous pouvons développer un système artificiel de transactions d'actions et construire un modèle qui inclut le comportement des traders, et qui peut être appliqué au marché réel. Il est difficile de répondre complètement à cette question, et elle nécessite également une compréhension suffisante des applications de la modélisation à base d'agents sur le marché financier. Ainsi, dans cette thèse, il y a trois sous-problèmes que nous avons ciblés et investigués :

(i) Comment construire un système artificiel de transactions d'actions et optimiser les stratégies des agents techniques ?

(ii) L'optimisation bayésienne peut-elle aider à ajuster les paramètres des modèles financiers basés sur les agents ?

(iii) Comment pouvons-nous appliquer les résultats optimaux de l'agent technique et du modèle basé sur les agents au marché financier ?

En lien avec la première question, nous commençons par analyser le comportement dans le marché des agents techniques. Dans la littérature, différentes stratégies de trading des agents techniques et des méthodes d'optimisation des paramètres ont été étudiées. Cependant, en pratique, il existe très peu de stratégies applicables. Des études antérieures sur les méthodes d'optimisation des paramètres ont fourni des résultats intéressants, mais ils sont de nature trop générique et ne parviennent pas à répondre à la complexité des fonctions-objectifs sur le marché réel. Premièrement, pour étudier les fonctions objectifs complexes dans les stratégies de trading des agents techniques en pratique, nous introduisons et comparons différentes approches d'optimisation des paramètres. Ces approches visent à donner une définition simple de l'optimisation des stratégies de trading, puis à les appliquer aux stratégies de trading couramment utilisées sur les marchés réels. Ensuite, pour comprendre le comportement des traders sur le marché, nous concevons un système de trading automatisé dans lequel les traders sont des agents techniques. Cela peut être utile pour choisir une stratégie de trading en fonction d'objectifs à long terme ou à court terme et pour comparer différentes stratégies de trading. Les méthodes d'optimisation proposées peuvent être appliquées pour construire divers comportements d'achats/ventes par les agents techniques dans un modèle boursier artificiel. De plus, les paramètres du modèle à base d'agents utilisé sur le marché financier peuvent également être optimisés.

La deuxième question de recherche principale est liée au problème d'optimisation des paramètres pour un modèle à base d'agents. La capacité à gérer un grand nombre de

paramètres est une caractéristique importante des marchés boursiers artificiels basés sur des agents par rapport aux autres modèles de marché et aux marchés boursiers réels [85]. Lorsque le nombre de paramètres augmente, l'optimisation des paramètres pour le modèle devient un problème compliqué. Le choix de l'algorithme d'optimisation des paramètres peut affecter l'approche et les performances du modèle car il affecte le comportement des agents et leur capacité à reproduire des faits stylisés du marché réel. Par conséquent, les exigences de base pour l'optimisation des paramètres du modèle sont d'avoir un faible coût de calcul et de permettre une grande précision. Cependant, l'application de la méthode Grid Search avec de grands ensembles de paramètres et l'exécution de simulations pendant de longues périodes [3, 120] sont les limites courantes des études sur ce sujet. Même pour les petits modèles, l'exploration des faits stylisés à travers toutes les combinaisons de paramètres n'est pas possible ou coûte trop cher [82]. Par exemple, dans [120], les auteurs ont généré 400 000 points initiaux pour calibrer leur modèle. Pour résoudre les problèmes ci-dessus, ce modèle introduit une approche avec optimisation bayésienne et une fonction de mesure d'erreur pour régler les hyperparamètres du modèle. Cette approche peut fournir rapidement et de manière flexible des ensembles de paramètres optimaux pour les stratégies de trading des agents et le modèle basé sur les agents.

L'objectif de la troisième question est de tenter de contribuer à la compréhension et à la diversification des agents pour la modélisation des marchés financiers. Nous présentons des applications de la modélisation à base d'agents au marché boursier. En conséquence, nous avons montré que les paramètres des modèles à base d'agents peuvent être ajustés à l'aide de la méthode d'optimisation bayésienne. Les faits stylisés du marché réel peuvent être reproduits en construisant soigneusement les fonctions-objectifs des agents. Dans le détail, nous avons introduit trois types d'agents typiques du marché boursier : les traders "en bruits" (noise traders), les traders techniques et les traders fondamentaux. Après avoir construit le modèle, l'optimisation bayésienne est utilisée pour régler les paramètres des stratégies des traders ainsi que du modèle boursier. Les résultats expérimentaux sur l'optimisation bayésienne avec le test de Kolmogorov–Smirnov ont démontré que l'ensemble proposé réduisait l'erreur de simulation avec des paramètres estimés plausibles. Avec des données empiriques de l'indice Dow Jones Industrial Average, nous avons constaté que les traders fondamentaux représentent 9%–11% de tous les traders du marché boursier. L'analyse statistique des données simulées peut produire les faits stylisés importants dans les marchés boursiers réels, tels que la leptokurtose, la queue épaisse des rendements et le regroupement de la volatilité.

A.1.3 Contribution de la thèse

La thèse est divisée en deux parties principales : (i) contributions théoriques et méthodologiques à l'étude de la modélisation financière basée sur les agents et (ii) applications des modèles basés sur les agents dans les marchés boursiers.

En ce qui concerne les aspects théoriques et méthodologiques, notre première contribution consiste en **méthodes d'optimisation des paramètres pour les stratégies de trading des agents techniques**. Tout d'abord, nous formulons le problème en fournissant une définition des stratégies de trading. Nous présentons ensuite les indicateurs de transactions dans différentes catégories. Lors de l'application des stratégies de trading à des indicateurs spécifiques, nous considérons différentes fonctions-objectifs et méthodes d'optimisation des paramètres. Outre la méthode classique d'optimisation des paramètres basée sur l'algorithme génétique, nous proposons également de nouvelles méthodes basées sur l'optimisation bayésienne et les approches d'apprentissage automatique. Cette étude est utile aux traders techniques pour analyser et choisir une stratégie de trading raisonnable à des fins d'investissement à court ou à long terme. À partir des connaissances acquises, nous exécutons les stratégies de trading sur des actions ou des crypto-monnaies spécifiques à travers un système de trading artificiel et évaluons les résultats obtenus. La performance des méthodes d'optimisation est également évaluée et comparée, ce qui peut être considéré comme un cadre d'évaluation de la méthode d'optimisation des stratégies d'investissement basées sur l'analyse technique. Ces méthodes peuvent également être utilisées pour optimiser les paramètres des "ABM" (modèles à base d'agents) sur les marchés financiers. La deuxième contribution méthodologique réside dans **les méthodes d'optimisation des paramètres pour les modèles à base d'agents sur les marchés financiers**. Contrairement aux méthodes d'optimisation des paramètres utilisées dans les études précédentes, la méthode d'optimisation bayésienne est introduite pour estimer les hyperparamètres plus rapidement et avec plus de souplesse. Nous présentons les avantages de la méthode d'optimisation bayésienne pour les modèles complexes tels que les marchés financiers par rapport à d'autres méthodes d'optimisation classiques telles que la recherche par grille ou la recherche aléatoire. En outre, nous proposons différentes fonctions-objectifs pour l'optimisation des paramètres en fonction du prix et du rendement. Cette variété de fonctions-objectifs est nécessaire pour qu'un modèle basé sur des agents dans les marchés financiers puisse représenter des faits stylisés sur le marché. L'algorithme peut travailler avec de grands volumes de données en peu de temps. En particulier, cela offre des contributions significatives lorsque les coûts de calcul deviennent onéreux. Par conséquent, l'optimisation bayésienne peut être utilisée pour accélérer l'optimisation des paramètres dans les modèles financiers multi-agents. Enfin, nous apportons **un nouveau fait stylisé concernant la proportion de traders sur les marchés financiers**. La proportion

de traders sur le marché peut être considérée comme un fait stylisé et a des implications importantes sur les marchés financiers. Les autres composantes importantes du modèle, telles que l'environnement, le comportement des traders et les interactions entre les agents, sont également examinées en détail.

Dans la deuxième partie de cette thèse, nous nous concentrons sur **l'application de la technique des ABM au marché boursier**. Les résultats peuvent être appliqués à d'autres marchés, tels que le marché des changes ou le marché des crypto-monnaies. Nous combinons les principaux résultats des travaux pionniers pour proposer un ABM avec divers traders. Leurs interactions directes et indirectes et les facteurs exogènes qui affectent directement le comportement des traders sont également introduits. À partir de là, nous menons une simulation de marché financier artificiel avec deux composantes principales. La première composante à prendre en compte est l'environnement. Le modèle considère un nombre fixe d'agents négociant un seul actif. Avec ces hypothèses de liquidité, le prix et le rendement du marché sont ensuite calculés. Ensuite, nous introduisons la composante "agent", qui comprend différents types de traders en fonction de leurs règles de négociation prédéfinies, tels que les "noise traders", les traders techniques et les traders fondamentaux. Nous nous concentrons sur les stratégies de trading des agents techniques. Cette étude montre que la proportion d'agents techniques représente la majorité du marché boursier. En outre, à avec le développement de simulations, avec la puissance exceptionnelle des systèmes informatiques, l'analyse technique est de plus en plus intéressante. À partir des méthodes d'optimisation des paramètres introduites, nous appliquons l'optimisation bayésienne avec diverses fonctions-objectifs pour fournir rapidement des paramètres estimés et introduire des faits stylisés intéressants. Cette recherche est utile pour comprendre et étudier la nature des marchés financiers dans la pratique. En outre, elle peut être considérée comme un premier pas vers la construction de modèles plus complexes et plus diversifiés avec des actifs exceptionnels autres que les actions. Les résultats peuvent être utilisés pour découvrir de nouvelles stratégies de trading ainsi que pour optimiser les stratégies de trading en fonction du comportement des agents.

A.1.4 Organisation de la thèse

La structure de la thèse est orientée par les questions de recherche énoncées. Chaque chapitre de cette thèse décrit un sous-problème. Pour commencer, dans le Chapitre 2, nous étudions le comportement de trading des agents basé sur l'analyse technique. Nous donnons une définition générale du problème d'optimisation des stratégies de trading des agents. Nous introduisons ensuite les indicateurs techniques et les stratégies de trading correspondantes. Après avoir défini les stratégies de trading, nous appliquons différentes approches pour

optimiser les paramètres avec différentes fonctions-objectifs. Les résultats obtenus nous permettent de répondre à la question de recherche (i).

Dans le Chapitre 3, l'optimisation des paramètres avec une approche d'optimisation bayésienne est envisagée pour optimiser un ABM pour les marchés financiers. En particulier, un ABM simple avec des noise traders est traité. Dans la phase de conception de l'environnement de trading, nous essayons de prendre en compte à la fois les caractéristiques générales et les différents aspects du marché réel. Sur la base de l'analyse de la structure du marché et du comportement de trading des participants au marché, nous présentons une approche utilisée pour estimer les paramètres du modèle avec différentes fonctions-objectifs. Nous discutons brièvement des idées qui sous-tendent la recherche analytique, nous mentionnons brièvement les faits stylisés qu'elles peuvent reproduire et nous discutons des avantages et des lacunes de chaque fonction-objectif. Cette analyse vise à répondre à la question de recherche (ii), et le résultat est une méthode d'optimisation des paramètres pour un marché financier.

Afin d'évaluer l'ABM proposé et de répondre à la question de recherche (iii), le Chapitre 4 reproduit et étend les ABM de la littérature. Cependant, notre objectif de recherche va au-delà de la reproduction de ces expériences. Nous visons à améliorer la compréhension des faits stylisés sur le marché. Par conséquent, en tirant parti des propriétés du modèle proposé, nous étendons le modèle à de multiples agents et à leurs interactions et nous analysons la dynamique du marché dans ce cadre. L'objectif principal de l'analyse est de déterminer la proportion d'investisseurs sur le marché. La structure du marché et les facteurs que nous considérons comme importants sont introduits lors de la construction du modèle. Nous étudions le comportement de trading des participants au marché avec différents rôles, tels que les noise traders, les traders techniques, les traders fondamentaux et les teneurs de marché. Ce modèle étend en fait la simulation du marché réel proposée au Chapitre 3.

Enfin, dans le Chapitre 5, nous évaluons nos objectifs de recherche et analysons dans quelle mesure nous avons réussi à répondre aux questions de recherche. Nous terminons la thèse par des suggestions de recherches futures.

A.2 Optimisation des paramètres pour les algorithmes de trading des agents techniques

L'objectif principal de la thèse est de développer un marché financier artificiel capable de reproduire les caractéristiques du marché financier réel. Pour soutenir cet objectif, ce Chapitre se concentre sur l'analyse des indicateurs de négociation du marché, en particulier

ceux qui sont couramment utilisés par les agents techniques sur le marché financier. Cette tâche propose que le système des marchés financiers soit étudié comme un système de transactions, qui décrit un ensemble de marchés financiers réels. Par conséquent, ce Chapitre comprend deux tâches : la construction d'un système de trading automatisé et l'étude de différentes approches pour optimiser les paramètres d'une stratégie de trading basée sur l'analyse technique. Dans notre système, la tâche de trading est formulée comme un problème de prise de décision dans un espace d'action vaste et complexe, qui est applicable à l'utilisation d'un algorithme d'apprentissage par renforcement. Notre travail comprend le développement d'un environnement d'apprentissage, d'une représentation de l'état, d'une fonction de récompense et d'un algorithme d'apprentissage pour les marchés financiers. Les algorithmes d'optimisation des paramètres considérés sont l'algorithme génétique (GA), l'optimisation bayésienne (BO) et l'apprentissage par renforcement profond (DRL). Les stratégies de trading sont élaborées sur la base d'un indicateur avancé, l'indice de force relative (RSI), et de deux indicateurs retardés, la bande de Bollinger (BOLL) et la moyenne mobile de convergence et de divergence (MACD). Plusieurs expériences sont réalisées sur différents marchés, notamment le marché des crypto-monnaies (également appelé "crypto"), le marché des actions et le marché des contrats à terme sur crypto-monnaies. Les données de prix Open-High-Low-Close de 15 minutes sont utilisées pour backtester des stratégies de trading basées sur l'analyse technique. Les résultats montrent que les stratégies optimisées à partir des approches proposées peuvent générer des rendements plus élevés que leur forme typique et la stratégie Buy and Hold. Parmi les indicateurs étudiés, le RSI peut battre tous les marchés au cours de la période de test et est le meilleur indicateur pour les transactions sur le marché des crypto-monnaies. Nous proposons également l'objectif d'investissement approprié pour chaque approche, l'approche BO convient à une stratégie de trading stable à long terme, tandis que l'approche DRL convient aux traders à court terme.

Dans ce Chapitre, nous présentons les concepts de base et les étapes du processus de trading. Le système de trading basé sur les indicateurs techniques, illustré dans la Figure A.1, vise à améliorer les signaux de trading de l'analyse technique en incorporant des techniques d'optimisation des paramètres dans les stratégies de trading. Tout d'abord, les données d'un marché au comptant de crypto-monnaies, d'un marché d'indices boursiers et d'un marché à terme de crypto-monnaies sont collectées et divisées en trois séries correspondant aux périodes de formation, de validation et de test. Les données d'entraînement et de validation sont utilisées pour construire et optimiser les stratégies basées sur l'analyse technique. Les stratégies obtenues sont ensuite soumises à l'étape de backtesting, et leur performance est évaluée sur l'ensemble des données de test. Enfin, les résultats sont analysés pour déterminer la qualité des stratégies proposées. Les principales étapes de ce processus sont les suivantes

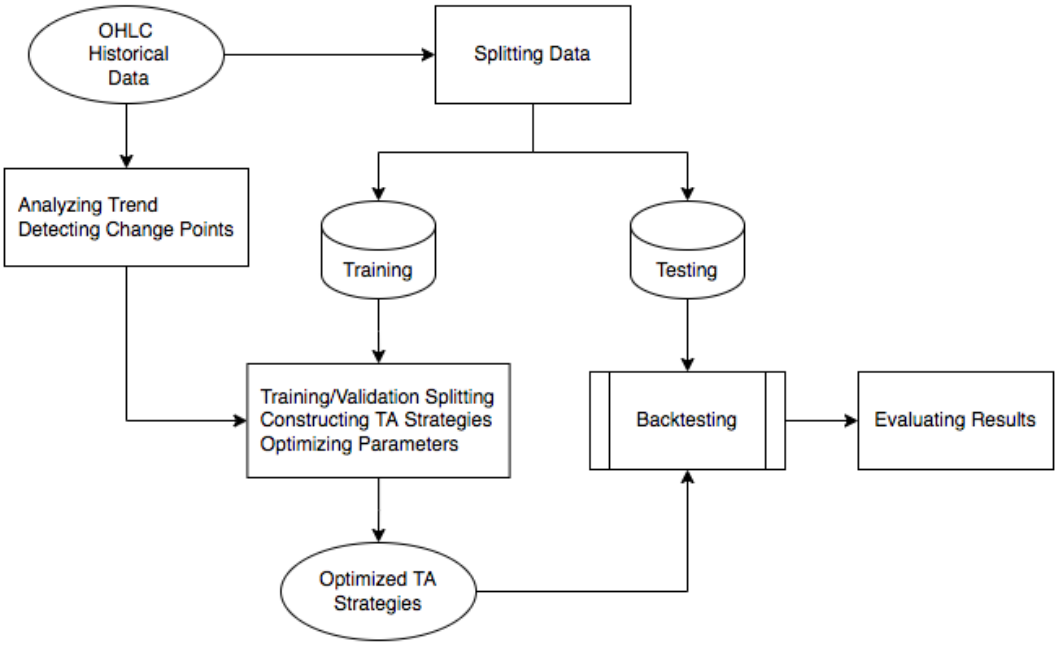


Fig. A.1 Schéma général du système commercial proposé

(i) *Analyse des données et élaboration des stratégies d'analyse technique*. La méthode "rolling forward" est appliquée pour diviser les données en plusieurs séquences. Nous analysons les tendances et identifions les points d'inflexion pour obtenir une vue d'ensemble de chaque marché. Sur la base des données de tendances actuelles, l'analyse des tendances permet de prédire l'évolution future de cette tendance. Combinée à des indicateurs, elle permet aux traders d'identifier clairement les points d'entrée et de sortie et de réaliser des bénéfices lorsqu'ils négocient dans le sens de la tendance et non à contre-courant.

(ii) *Optimisation des stratégies à l'aide de différentes méthodes*. Dans cette étape, les données de formation et de validation sont utilisées pour tester à rebours les stratégies d'analyse technique. À cette fin, deux approches d'optimisation des paramètres, l'algorithme génétique et l'optimisation bayésienne, sont appliquées pour trouver les paramètres offrant les meilleures performances.

(iii) *Backtesting des stratégies et évaluation des résultats*. Dans cette dernière étape, la stratégie optimisée est testée à rebours sur l'ensemble des données de test. La performance des stratégies sur chaque marché est évaluée et discutée.

Dans cette étude, nous investissons avec un capital initial accordé uniquement au début du processus d'investissement. Aucun argent supplémentaire n'est injecté pendant la période d'investissement. Par conséquent, chaque transaction affecte l'ensemble du processus. Si la transaction en cours est rentable, le capital augmente, ce qui signifie que le profit est utilisé

pour réinvestir dans la transaction suivante. Au contraire, si la transaction en cours n'est pas rentable, la diminution du capital investi affectera également les transactions suivantes. Les hypothèses concernant les signaux d'achat et de vente sont également prises en compte. Deux signaux d'achat ou de vente consécutifs ne sont pas autorisés, c'est-à-dire que chaque signal d'achat est suivi d'un signal de vente. Le nombre d'achats et de ventes est égal. En outre, la stratégie de trading commence toujours par un ordre d'achat. L'avantage de ce processus est qu'il permet d'examiner plus fermement la capacité des paramètres formés à se maintenir pendant toutes les périodes d'investissement. L'optimisation des stratégies d'analyse technique est proposée dans la sous-section suivante.

A.2.1 Algorithme génétique

La procédure de l'algorithme génétique peut être résumée par les étapes de la Figure A.2.

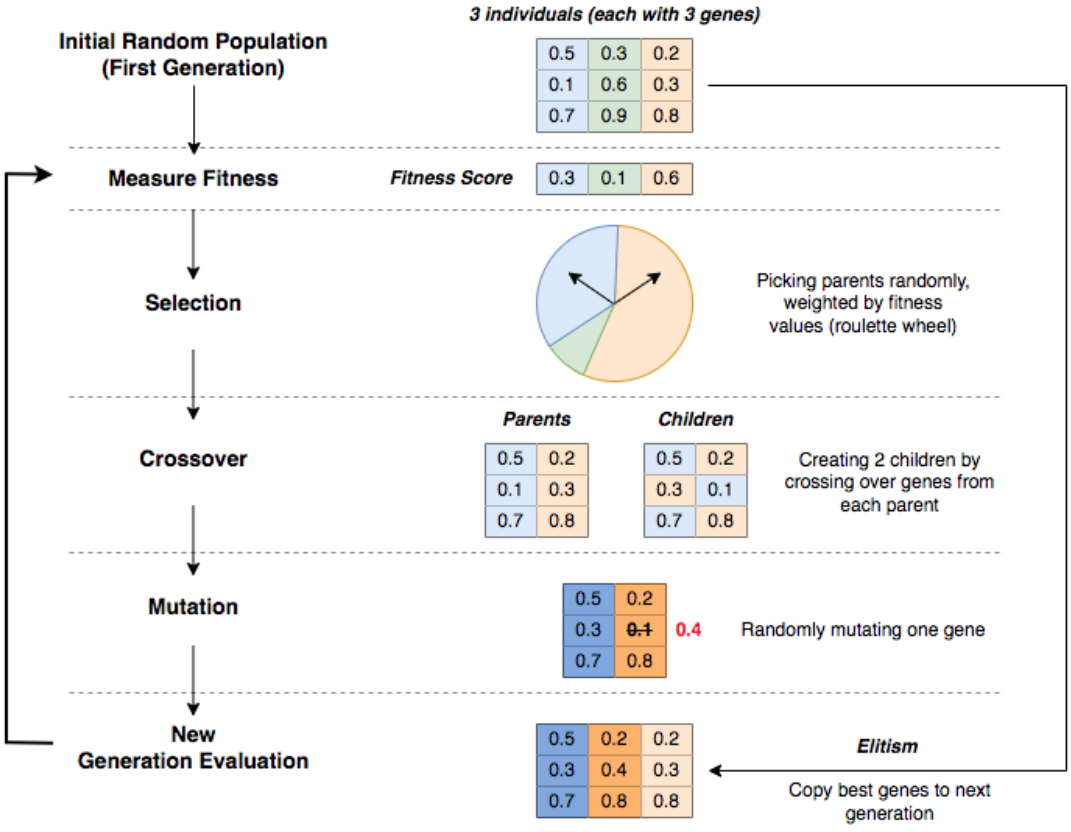


Fig. A.2 Illustration du concept d'algorithme génétique et un exemple

1. Créer une population initiale de candidats (première génération) de manière aléatoire.
2. Mesure de l'aptitude : évaluer la performance de chaque candidat en fonction d'une fonction-objectif.
3. Sélection : sélectionner les candidats pour la recombinaison.
4. Effectuer le croisement et la mutation.
5. Évaluer la performance des nouveaux candidats.
6. Retour à l'étape 3, à moins qu'un critère de terminaison ne soit satisfait.

Nous étudions les résultats de recherche de [47] et nous nous basons sur les résultats expérimentaux pour donner les valeurs des paramètres dans cette section. Notre chromosome GA est composé de trois parties principales, chaque partie correspondant à l'un des indicateurs à optimiser. Les gènes sont des nombres réels représentant les différents paramètres.

Table A.1 Un exemple de chromosome

RSI			**BOLL**		**MACD**		
14	30	70	20	2	12	26	9

La première partie consiste en trois gènes qui représentent les trois paramètres du RSI, à savoir le nombre de jours utilisés pour le calcul du RSI, le seuil inférieur et le seuil supérieur. Les deux gènes suivants représentent les deux paramètres de l'indicateur BOLL, la période et les écarts types. La dernière partie est constituée de trois gènes correspondant aux périodes de l'indicateur MACD. Les nombres présentés dans le tableau A.1 sont les paramètres typiques utilisés pour chaque indicateur [15]. Chaque paramètre étudié est limité par certaines contraintes.

Les paramètres optimaux pour l'AG sont sélectionnés par des expériences préliminaires. Les paramètres sélectionnés sont présentés dans le tableau A.2.

A.2.2 Optimisation bayésienne

La procédure d'optimisation bayésienne est illustrée plus en détail dans la Figure A.3. En résumé, l'algorithme d'optimisation bayésienne suit cinq étapes principales :

1. Construction d'un modèle de probabilité de substitution de la fonction-objectif.

Table A.2 Paramètres optimaux pour l'AG

Taille de la population	100
Nombre de générations	500
Technique de sélection	Sélection basée sur le rang
Probabilité de croisement	0.8
Probabilité de mutation	0.05

2. À partir de la fonction de substitution, trouver le candidat le plus potentiel en utilisant une fonction d'acquisition.
3. Évaluer le score du candidat avec la fonction-objectif originale.
4. Mettre à jour le modèle de substitution en incorporant le nouveau candidat et son score.
5. Répéter les étapes 2 à 4 jusqu'à ce que le nombre maximum d'itérations ou de temps soit atteint.

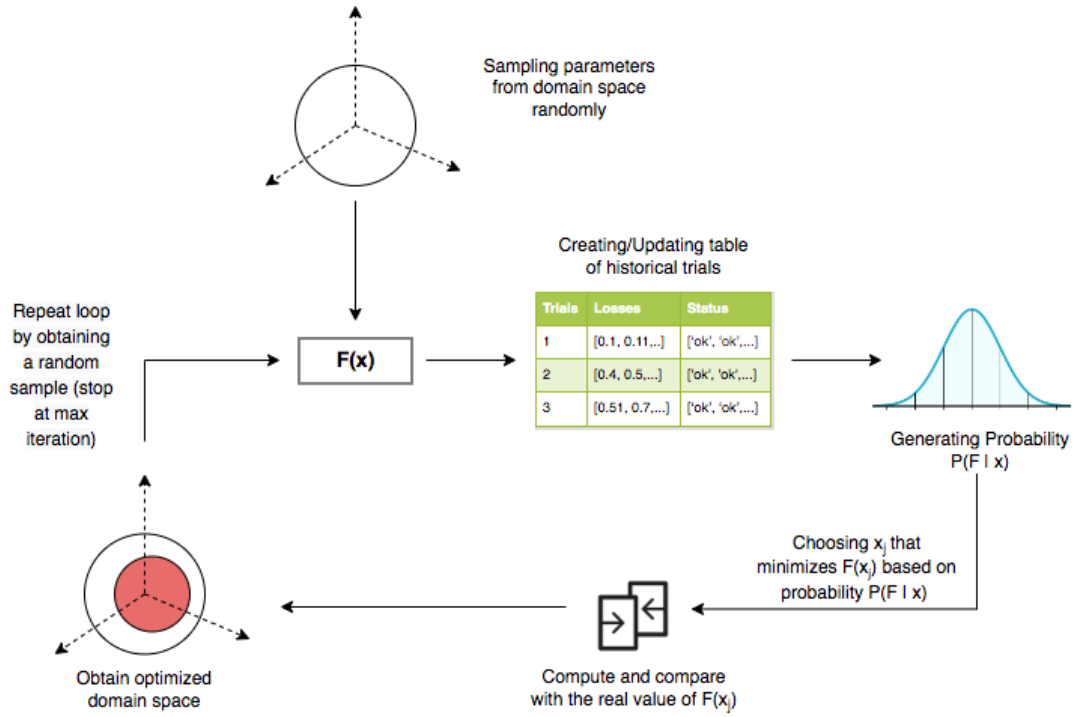


Fig. A.3 Illustration du concept d'optimisation bayésienne

Le progiciel *hyperopt* [12] dans le langage de programmation Python est utilisé pour mettre en œuvre l'algorithme d'optimisation bayésienne dans cette étude. Une distribution uniforme est utilisée pour éviter toute hypothèse sur l'espace du domaine. Dans la fonction-objectif, le score retourné est multiplié par le nombre "-1" car le logiciel utilisé tente de minimiser la fonction-objectif définie alors que notre objectif est de maximiser la probabilité de conflit. Dans l'optimisation bayésienne, il existe deux conditions de terminaison : la convergence des valeurs objectives maximales ou l'atteinte du nombre maximal d'évaluations. Dans cette étude, nous utilisons le nombre maximum d'évaluations comme critère de terminaison.

Nous formulons un problème d'optimisation comportant trois éléments principaux :

- Espace du domaine : La valeur de chaque paramètre est sélectionnée dans la plage définie pour la méthode GA (voir la section A.2.1) à des fins de comparaison.
- Fonction-objectif : Nous définissons une fonction de score qui indique l'efficacité d'un ensemble de paramètres de l'indicateur. Dans notre modèle, le rendement cumulé est utilisé comme mesure d'évaluation. Nous avons placé un signe moins devant la fonction-objectif car *hyperopt* définit par défaut une fonction à minimiser.
- Fonction de substitution et fonction de sélection : Dans ce modèle, nous suivons [123] pour utiliser l'estimateur de Parzen structuré en arbre comme fonction de substitution et le critère d'amélioration attendue est utilisé comme fonction d'acquisition. Le critère de terminaison est détecté lorsque le nombre maximal d'évaluations est atteint. Dans notre expérience, le nombre maximal d'évaluations est de 500.

A.2.3 Apprentissage par renforcement profond

L'algorithme 2 peut être utilisé pour décrire le processus d'apprentissage de l'agent, y compris la phase de formation et la phase de test. Les principaux objectifs de la phase de formation comprennent la génération d'échantillons d'apprentissage et la formation du réseau Q profond à l'aide de l'algorithme DRL. Au cours de la phase de test, le réseau cible est utilisé pour prédire un ensemble de paramètres optimal dans un scénario inédit. L'algorithme de formation étape par étape est le suivant :

1. Learning Environment échantillonne aléatoirement un ensemble de données $D = \{d_0, ..., d_T\}$ où chaque donnée $d_t \sim Unif(\mathscr{D})$ pour $t = 1, ..., T$.
2. L'entrée de l'algorithme DRL est représentée par un vecteur d'état codé à un instant, $s_0 = (d_0, (\{\lambda_{init}\}^{dim(\lambda)}, 0))$.

3. Étant donné le vecteur d'état s_t, une action candidate a_t est sélectionnée à l'aide de la technique ε-greedy.

4. Le paramètre λ_t est calculé. Il est ensuite envoyé à l'environnement d'apprentissage pour calculer la récompense r_t et générer le scénario suivant s'_t (ou s_{t+1}).

5. Les tuples échantillons (s_t, a_t, r_t, s_{t+1}) sont stockés dans le tampon de relecture pour une utilisation ultérieure lors de l'apprentissage.

6. Lorsque la mémoire tampon de relecture a stocké suffisamment d'échantillons ($\geq$ la taille minimale de la mémoire tampon de relecture, N_B), le n-uplet le plus ancien est remplacé. Un lot d'échantillons est prélevé au hasard dans la mémoire tampon de relecture pour l'entraînement.

7. Le réseau Q est mis à jour en minimisant la fonction de perte définie, ce qui est similaire à un modèle d'apprentissage supervisé.

8. Enfin, les réseaux cibles sont mis à jour après un nombre prédéfini d'étapes N_u.

9. Si la fin de l'épisode est atteinte, l'étape de recherche sera arrêtée et nous reviendrons à l'étape 1. Sinon, on augmente $t = t + 1$ et on retourne à l'étape 3.

La phase de test est relativement simple puisqu'il suffit d'obtenir l'ensemble des paramètres optimisés pour un scénario donné. Toutefois, dans la pratique, les expériences générées au cours de cette phase peuvent également être stockées dans la mémoire tampon de relecture afin d'ajuster le modèle par le biais d'un entraînement par lots. Ce réglage peut accélérer la mise au point du modèle et lui permettre de rester à jour avec les nouvelles données entrantes. L'algorithme de test étape par étape est décrit comme suit.

1. Un ensemble de données inédites D provenant de l'environnement d'apprentissage est donné.

2. L'état de l'environnement est défini comme l'ensemble de données D plus l'historique des configurations de paramètres évaluées et leur réponse correspondante.

3. Étant donné le vecteur d'état, une action a_t^* est suggérée.

4. Le paramètre λ_t^* est calculé et envoyé à l'environnement. Si la fin de l'épisode est atteinte, passer à l'étape suivante, sinon calculer l'état suivant s_{i+1}, $i = i + 1$ et revenir à l'étape 3.

Algorithm 2 Algorithme DRL pour l'optimisation des paramètres

Initialize network Q and target network $\hat{Q}$
Initialize experience replay memory B
Initialize the Agent to interact with the Environment
for N_e iterations **do** ▷ N_e - number of episodes
    Randomly sampling a data set $D = \{d_0, ..., d_T\}$, $d \sim Unif(\mathscr{D})$
    Get state $s_0 = (d_0, (\{\lambda_{init}\}^{dim(\Lambda)}, 0))$
    for $t \in \{0, ..., T\}$ and while s_t is not terminal **do**
        Determine next action a_t from state s_t using policy ε-greedy(Q) ▷ Eq. 2.9
        Receive reward $r_t = R(d_t, \lambda = g(a_t))$ ▷ Eq. 2.5
        Generate new state $s'_t = \tau(s_t, a_t, r_t)$ ▷ Eq. 2.6
        Store transition (s_t, a_t, r_t, s'_t) in the experience replay memory B
        Replace oldest tuple if $|B| > N_B$ ▷ N_B - replay buffer size
        if enough experience in B **then**
            Sample a random minibatch of N transitions from B
            for every transition (s_i, a_i, r_i, s'_i) in minibatch **do**

$$y_i = \begin{cases} r_i, & \text{if } s'_i \text{ is terminal;} \\ r_i + \gamma max_{a'} \hat{Q}(s'_i, a'), & \text{otherwise.} \end{cases}$$

            end for
            Update Q by minimizing the loss

$$\mathscr{L} = 1/K \sum_{i=0}^{K-1} \left(Q(s_i, a_i) - y_i \right)^2$$

            Copy weights from Q to $\hat{Q}$ for every N_u step
        end if
    end for
end for

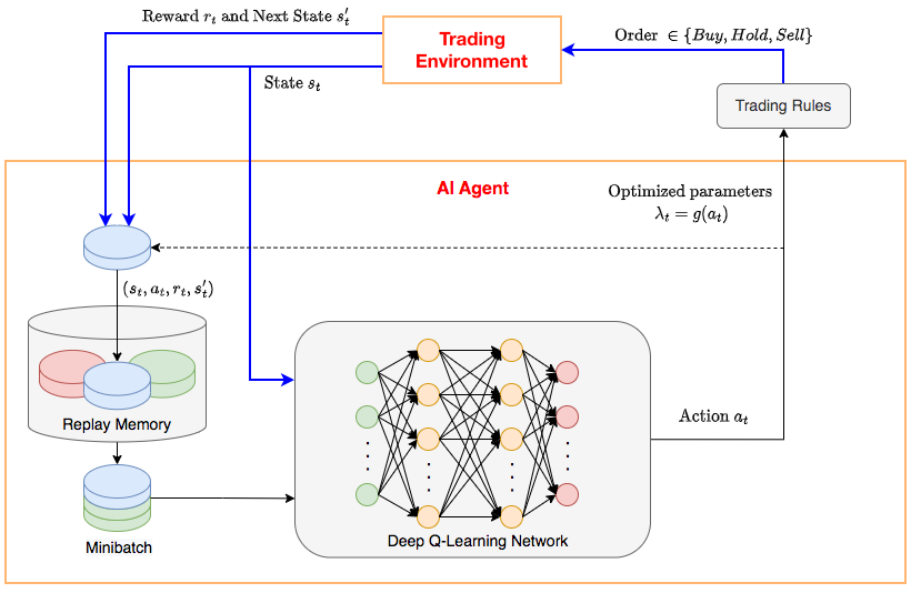


Fig. A.4 Illustration du mécanisme d'apprentissage dans un environnement commercial

5. Étant donné le vecteur d'état et l'action optimale, la valeur Q, $Q^*(s,a^*)$, peut être calculée.

6. Enfin, la valeur Q ainsi que le jeu de paramètres correspondant λ^* sont stockés pour évaluer les performances.

À partir des concepts d'optimisation des paramètres décrits, nous procédons à la construction d'un système de négociation qui peut à la fois donner des signaux de négociation et optimiser les stratégies automatiquement. Pour fournir l'environnement de négociation, nous étudions la représentation de l'état et la présentons sous une forme que l'agent peut comprendre. Nous proposons une définition de la sélection des paramètres d'un agent et une correspondance entre le choix des paramètres des stratégies de négociation et les décisions d'investissement. Chaque décision peut être notée à l'aide de la fonction de récompense proposée. Les principaux composants du système proposé sont décrits comme suit.

- Représentation de l'état : Le scénario commercial ou l'état de l'environnement, s_t, est représenté sous la forme d'un vecteur codé à un coup et décomposé en deux parties : le prix des données $d_t \in D$, et la séquence des configurations de paramètres sélectionnées et leurs récompenses correspondantes, $(\lambda_t, r_t) \in (\Lambda \times \mathscr{R})$.

- Action de l'agent : Étant donné un certain état de l'environnement, l'agent navigue sur la surface de réponse des paramètres afin de sélectionner un ensemble de paramètres

permettant d'optimiser la récompense. Il applique ensuite l'ensemble de paramètres choisi à sa stratégie de négociation et exécute une séquence d'ordres (achat, maintien ou vente) sur la base des règles de négociation.

- Évaluation de la récompense : Le ratio de Sharpe, le rendement cumulé ou une combinaison de ceux-ci peuvent être utilisés comme fonction de récompense que l'agent souhaite optimiser. Dans ce travail, chaque fonction-objectif est considérée séparément à des fins de comparaison.

- Mécanisme d'apprentissage : En appliquant l'algorithme 2, l'interaction entre le trader et l'environnement de trading est présentée dans la Figure A.4. L'agent peut prendre une mesure aléatoire avec une probabilité de ε, ou suivre la politique jugée optimale avec une probabilité de $1-\varepsilon$. Une valeur initiale pour epsilon de l'action ε-greedy, ε_{start}, est sélectionnée pour les premières observations et est ensuite fixée à une nouvelle valeur, ε_{end}, après un certain nombre d'observations. Le processus d'apprentissage de l'agent peut s'appuyer sur un réseau d'apprentissage Q profond ou sur ses variantes très puissantes telles que le réseau d'apprentissage Q profond double et le réseau d'apprentissage Q profond double duel. Au cours du processus de négociation, le trader exécute les ordres et calcule la performance par le biais d'une étape de backtesting. Ainsi, une étape supplémentaire est ajoutée à l'étape 4 des phases de formation et de test de l'algorithme : les paramètres sélectionnés sont utilisés comme entrée des règles de négociation pour donner des signaux de négociation (achat, maintien ou vente).

Dans ce Chapitre, deux expériences sont menées successivement.

(i) Expérience 1 : Comparaison des performances des stratégies de trading sur différents marchés avec deux approches d'optimisation des paramètres : Algorithme génétique et Optimisation bayésienne.

(ii) Expérience 2 : Tester la performance du système de négociation proposé avec des paramètres optimisés dans le cadre de l'apprentissage par renforcement.

La première expérience consiste à comparer l'approche d'optimisation bayésienne avec des approches de pointe, telles que l'algorithme génétique, l'ensemble de paramètres par défaut et la stratégie d'achat et de maintien. Cette expérience se concentre sur l'analyse des indicateurs couramment utilisés par les traders techniques, également appelés agents techniques dans ce travail, dans le but de développer un modèle multi-agents pour le marché financier dans le cadre de recherches futures. Les stratégies de trading sont construites sur

la base de différents indicateurs : RSI, BOLL et MACD. L'expérience est réalisée sur le marché boursier Dow Jones, où des données de prix de 15 minutes sont utilisées pour tester à rebours les stratégies de négociation. Les résultats montrent que la méthode d'optimisation des paramètres utilisant l'optimisation bayésienne donne de meilleurs résultats en termes de rendement et de temps d'exécution que l'approche de l'algorithme génétique dans toutes les périodes. Les stratégies optimisées par l'approche de l'optimisation bayésienne peuvent générer des rendements cumulés plus élevés que leur forme typique et que la stratégie Buy and Hold. Parmi les indicateurs étudiés, BOLL avec l'optimisation BO est une stratégie optimale rentable et stable. Deux autres expériences sont réalisées avec des données de 15 minutes des contrats perpétuels BTC-USDT et BTCUSDT à des fins de négociation à court terme. Avec une fonction-objectif simple, le rendement total, et en tenant compte du coût par transaction, les stratégies optimisées à partir de l'approche d'optimisation bayésienne peuvent générer des rendements plus élevés et une volatilité plus faible que la stratégie Buy and Hold et leur forme typique dans la plupart des cas. Les résultats expérimentaux montrent également que le RSI est plus performant que le MACD, avec une stratégie plus rentable et moins volatile.

Dans la deuxième expérience, le point fort de notre étude est une nouvelle technique basée sur l'approche DRL proposée pour optimiser les paramètres des stratégies d'analyse technique. La tâche de négociation est présentée comme un problème de prise de décision dans un espace d'action vaste et complexe, qui est applicable à l'utilisation d'algorithmes d'apprentissage par renforcement. Plus précisément, nous proposons un environnement d'apprentissage, une représentation de l'état, une fonction de récompense et un algorithme d'apprentissage dans le but d'optimiser la stratégie sur le marché des crypto-monnaies, ce qui n'a jamais été étudié auparavant. Le système de négociation proposé ne se concentre pas seulement sur la prise de décisions basées sur une stratégie donnée, mais inclut également une étape d'optimisation des paramètres dans le processus de négociation. Deux configurations sont envisagées pour construire l'agent d'intelligence artificielle dans le système : Double Deep Q-Network et Double Deep Q-Network. L'optimisation bayésienne est une autre approche introduite à des fins de comparaison. Différentes fonctions-objectifs couramment utilisées dans l'optimisation des transactions sont également introduites, telles que le rendement cumulé et le ratio de Sharpe. Les résultats ont montré que l'approche DRL avec le paramètre Q-Network doublement profond et l'approche BO produisent des rendements moyens positifs à des fins de négociation à court terme, le système avec l'approche DRL produisant de meilleurs résultats. En termes de temps d'exécution, l'approche DRL présente également des avantages remarquables, avec un temps d'exécution 5,83 fois plus rapide que l'approche BO. Lorsque l'on compare les performances avec différents paramètres et

fonctions-objectifs, le paramètre Double Deep Q-Network avec le ratio de Sharpe comme fonction de récompense est le meilleur système de trading Q-Learning avec un rendement mensuel de 15,96%. Les stratégies de trading sont basées sur l'indicateur RSI (Relative Strength Index), mais les résultats de cette étude peuvent être appliqués à n'importe quel indicateur technique ou de marché.

A.3 Optimisation des hyperparamètres dans un modèle de marché boursier basé sur des agents

Pour savoir comment concevoir des modèles de marché boursier, il faut comprendre en profondeur la structure et les caractéristiques du marché boursier réel. Une vue d'ensemble des approches utilisées pour étudier le marché boursier a été donnée au chapitre 1. Dans ce chapitre, notre objectif est de donner un aperçu de l'organisation et du fonctionnement du marché boursier. Nous proposons un modèle basé sur les agents (ABM) en introduisant certains concepts issus de la littérature sur la structure du marché, la formation des prix sur le marché et le comportement des agents en matière de négociation. À partir de ce modèle, nous proposons une méthode d'optimisation des paramètres avec une approche d'optimisation bayésienne. Cette méthode sert de référence pour l'optimisation des paramètres d'autres modèles ABM. En particulier, nous proposons différentes fonctions-objectifs basées sur le prix et le rendement des actions pour ajuster les hyperparamètres dans une simulation à base d'agents du marché boursier.

Suivant les études de [121], notre approche consiste à commencer le domaine à partir d'un large éventail et à se concentrer ensuite sur des domaines spécifiques autour des paramètres optimaux calculés à partir de l'exécution précédente. Nous définissons une fonction de score qui indique la performance d'un ensemble de périodes long-court. Les rendements cumulés et le ratio de Sharpe sont généralement utilisés pour indiquer les performances d'une stratégie technique. Dans notre modèle, le ratio de Sharpe est utilisé comme mesure d'évaluation de choix. La fonction de substitution est utilisée pour proposer des ensembles de valeurs qui augmentent la performance en minimisant le score de la fonction-objectif. Ils sont ensuite sélectionnés en appliquant un critère. Dans ce modèle, nous suivons [123] pour utiliser Tree Parzen Estimator (TPE) comme fonction de substitution. Le critère d'amélioration attendue (AE) est utilisé comme fonction de sélection. Nous proposons deux fonctions-objectifs basées sur le rendement et le prix. La fonction-objectif basée sur le rendement prend en compte deux tests, la divergence de Kullback-Leibler (KL) et le test de Kolmogorov-Smirnov (KS) [97], où la statistique dans le test quantifie la différence de deux distributions de probabilité.

En outre, le Dynamic Time Warping (DTW), qui mesure la similarité entre deux séries, est introduit pour construire la fonction-objectif basée sur le prix [14].

A.3.1 Fonction-objectif basée sur le rendement

La fonction-objectif est construite en comparant les rendements simulés et les rendements réels. Plus précisément, la comparaison porte sur les caractéristiques statistiques des distributions des deux séries temporelles. La construction est présentée comme suit.

- Divergence de Kullback-Leibler : La divergence de KL mesure la quantité d'informations perdues lorsque la distribution de probabilité des rendements simulés est utilisée pour approximer la distribution de probabilité des rendements réels.
- Test de Kolmogorov-Smirnov : La statistique du test KS mesure la plus grande distance entre la fonction de distribution empirique des rendements simulés et la fonction de distribution empirique des rendements réels.

A.3.2 fonction-objectif basée sur le prix

La fonction-objectif utilise l'algorithme DTW pour calculer une correspondance optimale entre les prix simulés et les prix réels avec certaines restrictions et règles. Le coût est calculé comme la somme des différences absolues, pour chaque paire d'indices appariés, entre leurs valeurs.

A.3.3 Modèle du marché financier

Dans ce Chapitre, nous recombinons les ingrédients clés de nos travaux précédents pour proposer un ABM simple qui peut correspondre aux faits stylisés des marchés financiers. Pour plus de détails, voir [133].

L'agent

Les agents s'appuient uniquement sur la situation actuelle du marché pour prendre des décisions de négociation : acheter, vendre ou conserver. Nous suivons la spécification des agents formulée par [59]. En tant que variable clé, le sentiment des agents implique leur réaction aux nouvelles qu'ils reçoivent et l'impact des nouvelles sur les prix futurs. Un bon sentiment signifie que les agents considèrent les nouvelles comme bonnes et espèrent que les prix futurs augmenteront (haussier) et un mauvais sentiment signifie que les agents

considèrent les nouvelles comme mauvaises et espèrent que les prix futurs diminueront (baissier).

Les agents du modèle sont supposés se rassembler en groupes lorsqu'ils prennent des décisions d'investissement. Dans [127], les auteurs décrivent un groupe de traders par un treillis carré et chaque trader est connecté à ses quatre voisins les plus proches. Dans notre modèle, nous définissons le groupe de traders comme un treillis multidi-mensionnel qui permet à chaque agent d'être connecté à ses voisins dans des groupes de tailles différentes. Les comportements d'achats/ventes/conservation des agents sont ainsi plus diversifiés. En conséquence, la propension dans le comportement décisionnel des agents est introduite. Le terme "propension de la décision des agents" définit la probabilité qu'un agent suive l'opinion des autres agents et que les agents changent leurs décisions de leur propre chef.

En résumé, les principales caractéristiques des agents sont décrites comme suit. À chaque pas de temps :

- Les agents reçoivent une information publique comme signal.
- Chaque agent compare le signal à sa sensibilité à l'information pour réagir à l'information.
- L'agent regarde ensuite la propension des décisions de ses voisins.
- Il prend sa décision en combinant la réaction à la nouvelle, la propension et sa propre interprétation.
- Après avoir calculé le rendement du marché, il met à jour ses sensibilités avec une probabilité. Si l'agent actualise sa confiance, il décide de passer à un autre groupe.

L'environnement

Le modèle considère un nombre fixe d'agents, noté N, négociant une seule action. À chaque pas de temps t, un agent i peut choisir entre trois actions : acheter/vendre une unité de l'action ou rester inactif, $s_i(t)\ dans\{1,-1,0\}$, respectivement. Nous supposons que chaque agent dispose toujours d'une richesse et d'actions suffisantes à des fins de liquidité. En d'autres termes, tous les agents peuvent toujours acheter ou vendre.

Nous synthétisons les éléments clés de nos travaux précédents pour construire un ABM simple pour le marché boursier [133]. Ce modèle offre une nouvelle perspective sur la manière dont les agents peuvent être influencés dans le choix de leurs règles de négociation. Sur le marché financier, les propensions des traders peuvent être contaminées par les sentiments de leurs voisins. Les agents ont également des sensibilités différentes aux facteurs exogènes du marché et, par conséquent, différentes nouvelles peuvent les influencer et les

impacter [24]. Par conséquent, la modélisation des marchés financiers nécessite une bonne compréhension des caractéristiques des agents du marché, de leurs comportements, de leurs interactions et de leur sensibilité à divers facteurs exogènes [8, 91]. Les études présentées dans ce Chapitre s'attacheront à préciser ces propriétés. Dans le modèle de [127], le réseau de traders est un treillis carré et chaque commerçant est connecté à ses quatre voisins les plus proches. Dans notre modèle, le réseau de traders est constitué de groupes de tailles différentes. Le niveau de contamination du sentiment est ainsi plus diversifié. Les fonctions-objectifs sont comparées sur la base du coût et de la stabilité au cours de nombreuses simulations. En outre, nous analysons également l'efficacité de chaque fonction-objectif en fonction de sa capacité à reproduire les propriétés du marché boursier dans la réalité.

Les résultats présentés dans ce Chapitre montrent que l'optimisation bayésienne peut proposer un ensemble optimal de paramètres avec une grande stabilité et un faible coût. La fonction-objectif basée sur le rendement avec le test KS présente de meilleurs résultats en termes de taux de convergence et de temps d'exécution. En outre, l'ABM avec différentes fonctions-objectifs peut représenter certains faits stylisés du marché réel. Il s'agit d'une méthode de référence pour l'optimisation des paramètres d'un marché boursier basé sur des agents. L'algorithme DTW peut reproduire la dynamique des prix sur le marché réel, mais n'est pas adapté à l'absence d'autocorrélations dans les rendements. En revanche, le test KS ne montre pas une bonne simulation des prix mais explique avec succès certains faits stylisés importants des rendements, tels que l'absence d'autocorrélations, les queues épaisses dans la distribution des rendements boursiers et le regroupement de la volatilité. Ces résultats sont cohérents avec la définition de la fonction-objectif.

A.4 Application de la modélisation basée sur les agents au marché boursier

Dans cette section, un modèle basé sur des agents du marché boursier est construit pour détecter la proportion des différents types de traders. Nous modélisons un marché boursier simple qui compte trois types de négociateurs différents : les noise traders, les traders fondamentaux et les traders techniques, qui traitent un seul actif.

L'un des problèmes de la modélisation basée sur des agents est de savoir comment optimiser les paramètres des stratégies des traders et ajuster les hyperparamètres du modèle de marché. En règle générale, de nombreux modèles utilisent de grands ensembles de paramètres et effectuent des simulations pendant de longues périodes [120, 3]. Cependant, même pour les petits modèles, l'exploration des faits stylisés à travers toutes les combinaisons

de paramètres n'est pas possible ou est d'un coût prohibitif [82]. Par exemple, dans [120], les auteurs ont généré 400 000 points initiaux pour calibrer leur modèle. Pour résoudre les problèmes susmentionnés, ce modèle introduit une approche avec optimisation bayésienne et une fonction de mesure d'erreur pour optimiser les paramètres de l'environnement. Dans ce travail, nous combinons les principales propriétés des travaux pionniers. Un modèle basé sur des agents est introduit pour mettre en évidence les faits stylisés du marché boursier. Cette recherche offre une nouvelle perspective sur la manière de détecter la proportion de chaque trader sur le marché boursier. Les résultats peuvent être utilisés pour sélectionner des stratégies de trading efficaces pour les traders. Le modèle comprend trois types de traders : les noise traders, les traders techniques et les traders fondamentaux, avec différents comportements pour accroître la diversité.

Cette section décrit la construction d'un nouveau modèle basé sur des agents. Ce modèle permet de mieux comprendre la raison des faits stylisés et la proportion estimée de traders sur le marché boursier. En raison de la tendance à former des groupes discrets de traders, nous définissons différents types de traders dans ce modèle en fonction de leurs règles de négociation prédéfinies. Il existe trois types différents de traders opérant sur le marché. Les noise traders sont des personnes qui prennent des décisions au hasard en fonction de leurs réactions aux nouvelles et de leur propension à la contagion des sentiments. Les traders techniques analysent les graphiques et prennent des décisions sur la base de l'analyse des modèles et des tendances actuels, et les traders fondamentaux négocient sur la base du potentiel fondamental de génération de profits de l'action.

A.4.1 Agent basé sur du bruit

Les agents basés sur le bruit sont des traders qui négocient sur la base d'une mauvaise compréhension des informations et des nouvelles concernant les prix futurs. Ils prennent des décisions et négocient sur la base d'une analyse inexacte du marché [130]. Dans ce modèle, les agents basés sur du bruit se fondent uniquement sur la situation actuelle du marché pour prendre des décisions de négociation : acheter, vendre ou conserver [63]. En tant que variable clé, le sentiment des agents du bruit implique leur réaction aux nouvelles qu'ils reçoivent [32]. Un bon sentiment signifie que les agents considèrent les nouvelles comme bonnes et espèrent que les prix augmenteront à l'avenir (haussier) et un mauvais sentiment signifie que les agents considèrent les nouvelles comme mauvaises et espèrent que les prix diminueront à l'avenir (baissier).

A.4.2 Agent technique

Contrairement aux agents basés sur le bruits, les agents techniques sont des traders qui utilisent les prix passés pour déduire des informations privées. Ils prennent des décisions en calculant les indicateurs techniques à partir des prix historiques [21]. La caractéristique de l'analyse technique est de prendre des décisions après avoir identifié les tendances à un stade précoce et d'inverser la décision lorsque le renversement de tendance se produit [118]. Dans ce modèle, les agents sont censés utiliser une technique réelle appelée oscillateur de moyenne mobile pour prédire l'évolution future des prix [20]. La technique consiste tout d'abord à prendre deux moyennes mobiles du prix, une moyenne mobile à court terme A et une moyenne mobile à long terme B, de longueurs différentes $l_A < l_B$. Le croisement des moyennes mobiles est calculé comme la différence $M(t)$ entre ces deux moyennes mobiles.

- $M(t) > 0$ indique que le prix est orienté à la hausse et que les agents effectueront un achat.
- $M(t) < 0$ indique que le prix est orienté à la baisse et que les agents vendront le titre.

Afin de tenir compte de l'hétérogénéité individuelle, chaque agent peut utiliser différentes longueurs de données historiques, de sorte qu'une stratégie de moyenne mobile peut indiquer une tendance à la hausse tandis qu'une autre stratégie de moyenne mobile indique une tendance à la baisse. Cependant, dans notre modèle, la différence de longueur des fenêtres, l_A et l_B, n'a pas d'effet sur la décision de trading des agents. Lorsque les données historiques du prix des actions présentent une forte tendance à la hausse ou à la baisse, tous les investisseurs prendront la même décision. Nous supposons donc que tous les agents techniques peuvent utiliser l'optimisation bayésienne pour estimer la longueur optimale des fenêtres courtes et longues.

A.4.3 Agent fondamental

Les agents fondamentaux sont des traders qui mesurent la valeur intrinsèque en analysant la comptabilité, la finance et l'économie de l'action [136]. Ils prennent des décisions en pensant que le cours de l'action reviendra à sa valeur fondamentale à long terme.

Un trader basé sur du bruit prend des décisions basées sur les changements de prix. Ils ont des émotions différentes et des réactions aux nouvelles qui jouent un rôle important dans la prise de décision. Les traders techniques sont diversifiés par des indicateurs de trading, ils prennent des décisions différentes en fonction de l'objectif de rendement et du temps. Les traders fondamentaux sont introduits pour équilibrer le marché, ils sont diversifiés par leur niveau de confiance qui mesure la convergence du prix de l'action vers sa valeur

fondamentale. L'interaction est une fonction qui combine le changement de sentiment lors de l'obtention d'informations sur le marché et l'influence des décisions des autres investisseurs.

Le point fort de notre modèle est l'introduction de l'optimisation bayésienne pour optimiser les paramètres des stratégies des traders et ajuster les hyperparamètres du modèle de marché. Les résultats montrent que l'optimisation bayésienne peut proposer un ensemble optimal de paramètres et réduire l'erreur de simulation. Cela permet d'obtenir des faits stylisés à comparer avec le marché réel, en fournissant une méthode de référence pour calibrer les paramètres d'un marché boursier basé sur des agents. Nous avons utilisé différentes méthodes de comparaison des deux distributions, la divergence de Kullback-Leibler (KL) et le test de Kolmogorov-Smirnov (KS), pour mesurer l'erreur de simulation. Dans notre modèle, le test KS a donné de meilleurs résultats en expliquant avec succès certains faits stylisés importants sur les marchés boursiers réels, tels que la leptokurtose, la queue épaisse des rendements et le regroupement de la volatilité. Grâce à des simulations approfondies, nous avons constaté que les traders fondamentaux représentaient 9% à 11% de l'ensemble des traders sur le marché boursier. Ces résultats sont cohérents avec la recherche sur le marché réel.

A.5 Conclusion et travaux futurs

A.5.1 Conclusion

Cette thèse se concentre sur l'étude de multiples approches de l'optimisation des paramètres pour les modèles basés sur des agents et leur application dans les marchés financiers. L'objectif de la recherche est divisé en trois sections.

Tout d'abord, nous avons étudié le comportement de trading des agents basés sur l'analyse technique. Les concepts de base et les étapes impliquées dans le processus de négociation d'un trader sont introduits. Nous avons analysé les tendances pour obtenir une vue d'ensemble du marché. Combinés à des indicateurs, les traders peuvent clairement identifier les points d'entrée et de sortie et réaliser des bénéfices lorsqu'ils négocient avec la tendance et non contre elle. Avec des stratégies de trading communes et des fonctions-objectifs simples, l'optimisation bayésienne et l'apprentissage par renforcement profond sont deux approches proposées pour trouver le jeu de paramètres offrant les meilleures performances. Ces nouvelles approches sont comparées à des approches issues d'études antérieures telles que les algorithmes génétiques, la modélisation paramétrique et les stratégies Buy and Hold. Trois indicateurs courants, RSI, BOLL et MACD, sont utilisés pour élaborer des stratégies simples et adaptées aux marchés réels. Le comportement d'achat et de vente ainsi que

la valeur des coûts de transaction sont également retenus. Nous avons également mené de nombreuses expériences avec les données de différents marchés, tels que le marché au comptant des crypto-monnaies, le marché boursier et le marché des crypto-monnaies à terme. L'expérience a donné des résultats intéressants. Pour les transactions à court terme, la méthode d'apprentissage par renforcement profond donne des résultats supérieurs en termes de rendements cumulés, de volatilité et de temps d'exécution par rapport à la méthode d'optimisation bayésienne. Cela permet aux traders de prendre des décisions rapides et efficaces en s'appuyant sur les informations les plus récentes du marché. Dans le cadre du trading à long terme, l'optimisation bayésienne est une méthode d'optimisation des paramètres qui permet d'obtenir des rendements plus élevés. Ces résultats ne changent pas lorsque l'on compare les résultats sur des ensembles de données de différentes longueurs. En ce qui concerne l'extension, l'apprentissage par renforcement profond peut apporter une solution au problème de hauteur de l'optimisation bayésienne, de sorte que les études peuvent être étendues à des stratégies et des fonctions-objectifs plus complexes. Toutefois, pour appliquer l'algorithme d'apprentissage par renforcement profond à des modèles autres que les stratégies commerciales, il est nécessaire de reconstruire l'environnement et de définir des états, des actions et des récompenses entièrement nouveaux. Par conséquent, l'optimisation bayésienne est utilisée pour optimiser les paramètres du modèle à base d'agents pour les marchés financiers dans nos recherches. Forts de ces connaissances, nous ajoutons des traders techniques dotés de stratégies de négociation optimisées à notre modèle basé sur des agents proposé en tant qu'agents techniques.

Deuxièmement, nous avons proposé une nouvelle approche pour l'optimisation des paramètres dans un modèle de marché financier basé sur des agents. Nous avons proposé différentes fonctions-objectifs pour ajuster les hyperparamètres dans une simulation du marché boursier basée sur des agents. L'optimisation bayésienne avec le test de Kolmogorov-Smirnov a proposé un ensemble optimal de paramètres avec une grande stabilité. L'algorithme peut traiter de grands volumes de données avec des temps d'exécution plus courts que les méthodes précédentes dans ce domaine. En particulier, il apporte des contributions significatives lorsque les coûts de calcul deviennent onéreux. Par conséquent, l'optimisation bayésienne peut être utilisée pour accélérer l'optimisation des paramètres dans les modèles financiers multi-agents. Le modèle avec un algorithme objectif basé sur les rendements a pu mettre en évidence certains faits stylisés importants des rendements boursiers, tels que l'absence d'autocorrélations, la queue épaisse des rendements et le regroupement de la volatilité. En outre, la dynamique des prix des actions sur le marché réel est reproduite lorsqu'elle est simulée à l'aide de l'algorithme objectif basé sur les prix. À l'avenir, nous pourrons combiner les deux fonctions-objectifs afin d'obtenir des résultats à la fois pour les

prix et les rendements. Ce travail n'est qu'une première étape vers une évaluation complète des propriétés des modèles basés sur des agents sur le marché boursier des États-Unis. Dans le cadre de recherches futures, le modèle pourra être appliqué à d'autres marchés boursiers, tels que le New York Stock Exchange ou les marchés émergents d'Asie.

Enfin, nous avons essayé de reproduire et d'étendre les ABM de la littérature. Cependant, notre objectif de recherche va au-delà de la reproduction de ces expériences. Nous visons à améliorer la compréhension des faits stylisés du marché. Par conséquent, en tirant parti des propriétés du modèle proposé, nous étendons le modèle à de multiples agents et à leurs interactions, et nous analysons la dynamique du marché dans le cadre de ce modèle. L'objectif principal de l'analyse est de déterminer la proportion de négociants sur le marché. La structure du marché et les facteurs que nous considérons comme importants sont introduits lors de la construction du modèle. Nous étudions le comportement de négociation des participants au marché avec différents rôles, tels que les traders de bruit, les traders techniques, les traders fondamentaux et les teneurs de marché. À partir des méthodes d'optimisation des paramètres introduites, nous appliquons l'optimisation bayésienne avec différentes fonctions-objectifs pour fournir rapidement des paramètres estimés et introduire des faits stylisés intéressants. Les résultats optimisés montrent que les traders fondamentaux représentent 9%–11% de l'ensemble des traders sur le marché boursier américain. Ce ratio est une information intéressante qui peut servir d'indicateur au trader pour élaborer des stratégies efficaces. Bien que les approches bayésiennes présentent plusieurs avantages, la complexité de la fonction-objectif pourrait être un problème car elle a toujours ses propres coûts. Dans notre modèle, le ratio de Sharpe utilisé comme fonction-objectif est très simple à calculer. Cependant, si nous voulons considérer un modèle complexe avec plus de types d'investisseurs et d'interactions entre eux, la dimension de l'espace du domaine augmentera. Les modèles optimaux nécessitent des paramètres différents entre les ensembles de données, et avec un espace à haute dimension, la corrélation entre les paramètres est difficile à étudier.

Les publications liées à cette thèse comprennent :

1. Tran, M., Duong, T., Pham-Hi, D., & Bui, M. (2020). Detecting the proportion of traders in the stock market: an agent-based approach. Mathematics, 8(2), 198.

2. Tran, M., Ngo, M., Pham-Hi, D., & Bui, M. (2020). Bayesian Calibration of Hyperparameters in Agent-Based Stock Market. In 2020 RIVF International Conference on Computing and Communication Technologies (RIVF) (pp. 1-6). IEEE.

3. Tran, M., Pham-Hi, D., & Bui, M. (2022). Hyperparameter Tuning with Different Objective Functions in Financial Market Modeling. In International Econometric Conference of Vietnam 10-12 January 2022 (pp. 733-745). Springer, Cham.

4. Tran, M., Pham-Hi, D., & Bui, M. (2022). Parameter Optimization for Trading Algorithms of Technical Agents. In 2022 International Conference on Computing and Communication Technologies (RIVF). IEEE.

5. Tran, M., Pham-Hi, D., & Bui, M. (2023). Optimizing Automated Trading Systems with Deep Reinforcement Learning. Algorithms, 16(1), 23.

A.5.2 Recherches futures

Les travaux de recherche menés dans le cadre de cette thèse peuvent être étendus à l'avenir dans les directions suivantes :

1. Optimisation multi-objectifs : Les problèmes d'optimisation multi-objectifs se posent régulièrement dans le monde réel lorsque deux objectifs ou plus doivent être optimisés simultanément. Sur le marché financier en général ou dans la stratégie de négociation du trader en particulier, la combinaison des fonctions-objectifs de profit et de risque attire toujours beaucoup d'attention. Pour optimiser ces fonctions-objectifs complexes, des stratégies évolutives et des heuristiques soigneusement conçues peuvent améliorer les performances. Toutefois, les progrès récents des algorithmes d'apprentissage automatique ont montré leur capacité à remplacer les humains en tant qu'ingénieurs en algorithmes pour résoudre divers problèmes. Alors que les réseaux neuronaux profonds se concentrent sur les prédictions, l'apprentissage par renforcement profond est principalement utilisé pour apprendre à prendre des décisions. Nous pensons donc que l'apprentissage par renforcement profond est un moyen viable d'apprendre à résoudre automatiquement divers problèmes d'optimisation, sans avoir besoin de stratégies évolutives et d'expériences conçues par l'homme.

2. L'apprentissage profond dans les modèles basés sur les agents : Le développement de l'IA, l'apprentissage automatique et les théories de la prise de décision humaine sont étroitement liés aux sujets brûlants actuels de l'apprentissage automatique. Lors de la construction d'un modèle ABM, le concept d'agent artificiel peut être introduit par le biais de comportements simulés par des modèles d'apprentissage automatique. De plus, lors de l'optimisation des paramètres du modèle ABM, une approche d'apprentissage par renforcement profond peut être appliquée. L'objectif de cette recherche est de développer de nouvelles méthodes de calcul pour améliorer l'applicabilité des ABM macroéconomiques à l'analyse des politiques économiques. En cas de succès, nous réduirons considérablement la complexité et la charge de calcul des simulations ABM

et offrirons de nouvelles méthodes de modélisation du comportement des agents économiques.

3. Envisager un simulateur complexe de négociation en temps réel avec une variété de stratégies : Des algorithmes simples et populaires ont été présentés dans notre étude. Les résultats obtenus permettent d'envisager un système de négociation plus complexe, tel que l'optimisation d'un portefeuille multi-actifs, l'intégration des informations des médias sociaux et des nouvelles macroéconomiques du marché dans les stratégies d'investissement.

RÉSUMÉ

L'analyse de modèles complexes tels que les marchés financiers aide les gestionnaires à élaborer des politiques raisonnables et les commerçants à choisir des stratégies de négociation efficaces. La modélisation basée sur les agents est une méthodologie de calcul pour modéliser des systèmes complexes et analyser l'influence de différentes hypothèses sur les comportements des agents. Dans le cadre de cette thèse, nous nous concentrons sur l'étude des enjeux d'optimisation des stratégies de trading des agents techniques à travers un système de trading automatisé. Différentes approches sont introduites telles que: l'algorithme génétique, l'optimisation bayésienne et l'apprentissage par renforcement profond. Les stratégies de trading sont construites sur la base d'un indicateur avancé, Relative Strength Index, et de deux indicateurs retardés, Bollinger Band et Moving Average Convergence-Divergence. De multiples expériences sont réalisées sur différents marchés, notamment: le marché des crypto-monnaies, le marché boursier et le marché des contrats à terme cryptographiques. Les résultats montrent que les stratégies optimisées à partir des approches proposées peuvent générer des rendements plus élevés que leur forme typique et la stratégie Buy and Hold. En utilisant les résultats de l'optimisation des stratégies de trading, nous proposons une nouvelle approche pour optimiser les paramètres du modèle à base d'agents. Enfin, une application du modèle à base d'agents au marché boursier est présentée. Notre travail comprend le développement d'un environnement, les comportements des différents agents et leurs interactions. La méthode d'optimisation bayésienne a montré des avantages et un potentiel dans l'estimation d'un ensemble optimal de paramètres pour un modèle de marché financier artificiel. Le modèle que nous proposons est capable de reproduire les faits stylisés du marché réel. En outre, un nouveau fait stylisé sur la proportion de commerçants sur le marché est présenté. À l'avenir, davantage de recherches seront menées pour améliorer le modèle et les méthodes d'optimisation, telles que l'application de modèles d'apprentissage automatique, l'apprentissage par renforcement multi-agents ou l'examen de l'application sur différents marchés et instruments négociés.

MOTS CLÉS

Optimisation bayésienne, trading automatisé, apprentissage par renforcement profond, modèle à base d'agents, système complexe, marché financier.

ABSTRACT

The analysis of complex models such as financial markets helps managers make reasonable policies and traders choose effective trading strategies. Agent-based modeling is a computational methodology to model complex systems and analyze the influence of different assumptions on the behaviors of agents. In the scope of this thesis, we concentrate on studying the challenges of optimizing the trading strategies of technical agents through an automated trading system. Different approaches are introduced, such as Genetic Algorithm, Bayesian Optimization, and Deep Reinforcement Learning. The trading strategies are built based on a leading indicator, the Relative Strength Index, and two lagging indicators, the Bollinger Band and the Moving Average Convergence-Divergence. Multiple experiments are performed in different markets, including the cryptocurrency market, stock market, and crypto futures market. The results show that optimized strategies from the proposed approaches can generate higher returns than their typical form and the Buy and Hold strategy. Using the results from the optimization of trading strategies, we propose a new approach to optimize the parameters of the agent-based model. Finally, an application of the agent-based model to the stock market is presented. Our work includes the development of an environment, the behaviors of different agents, and their interactions. The Bayesian optimization method has shown advantages and potential in estimating an optimal set of parameters for an artificial financial market model. The model we propose is capable of reproducing the stylized facts of the real market. Furthermore, a new stylized fact about the proportion of traders in the market is presented. In the future, more research will be done to improve the model and optimization methods, such as applying machine learning models, multi-agent reinforcement learning, or considering the application in different markets and traded instruments.

KEYWORDS

Bayesian optimization, automated trading, deep reinforcement learning, agent-based model, complex system, financial market.