Computational Modeling and Design of Financial Markets: Towards Manipulation-Resistant and Expressive Markets

by

Xintong Wang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2021

Xintong Wang
xintongw@umich.edu
ORCID iD: 0000-0002-0867-8807

To my mom, dad, and grandparents

ACKNOWLEDGMENTS

The past five years at Michigan have been an unforgettable and rewarding journey. I would not have been able to make through it without the help and support of so many people whom I feel deeply indebted to.

First and foremost, my greatest appreciation goes to my advisor Michael Wellman. When I first started my PhD study, Mike provided me invaluable guidance on how to become a researcher, from choosing an interesting and meaningful research topic to solving a problem with appropriate tools, from writing a paper and delivering a presentation to making the right word choice! Mike offers high-level insights about the field, as well as detailed understanding and comments on specific problems. As I become a senior PhD student, Mike also provides me the freedom to explore different directions that I am passionate about. He is always patient with me and believes in me, even when I do not have the patience and confidence in myself. Now after all these years of PhD study, I have realized how privileged I am to learn from one of the most generous mentors and work with one of the most brilliant minds in the field. I am grateful for his encouragement, patience, and guidance in all aspects!

I would also like to sincerely thank Dave Pennock, Uday Rajan, and Satinder Singh for being on my thesis committee and for their sharing of ideas and insightful comments from different perspectives during my thesis proposal, defense, and beyond. I appreciate their generous offerings of time and efforts. This thesis would not have been possible without their invaluable feedback and support.

During my PhD study, I have done two wonderful internships at Microsoft Research and J.P. Morgan AI Research. I am extremely lucky to have Dave Pennock as my mentor at MSR, who introduced me to exciting topics in financial options market and later prediction markets, which have become part of this thesis. I am grateful for his sharing of ideas, guidance on research projects, and mentorship even after my internship! I would also like to thank Miro Dudík (for introducing me to the prediction market project), David Rothschild, and Nikhil Devanur for many useful discussions, as well as help and feedback on paper writings. At J.P. Morgan AI Research, my mentor Tucker Balch offered not only insights on research projects, but also helpful resources and opportunities for me to discuss problems with traders and people from different teams. I greatly enjoyed these conversations at JPM that have been motivating me to think more about the impact and practicability of research projects and solutions in real-world financial markets.

I have also been extremely fortunate to work with and learn a great deal from many wonderful collaborators: Yevgeniy Vorobeychik, Arunesh Sinha, Junyi Li, Christopher Hoang, Biaoshuai Tao, and many others. I would like to thank them for having inspiring conversations, sharing expertise and ideas, and making my PhD a fun, accompanied, and rewarding journey.

I sincerely thank the whole Strategic Reasoning Group for contributing to a great research environment. I have been extremely lucky to work with (and have fun with) my lovely labmates: Erik Brinkman, Mason Wright, Frank Cheng, Kareem Amin, Megan Shearer, Arunesh Sinha, Thanh Nguyen, Zun Li, Yongzhao Wang, Max Smith, Christine Konicki, Katherine Mayo, and Mithun Chakraborty (who has also been my undergraduate TA!). I missed the times when we could still chat in the office and go out for lunch (or bowling or kayaking) together! I appreciate the selfless help from Erik on all those things/bugs that I would otherwise spend tons of time on (and I hope that I have shared at least some of those I learned from him with my fellow labmates).

I would also like to thank everyone in the NSF manipulation project team (though we still do not have a good name for it). The team has offered invaluable knowledge and understanding on market manipulation and financial regulation from disciplinaries across finance, law, and public policy, making me to think about and evaluate computational solutions from new and important aspects. I am especially grateful for the conversations I have had with Uday Rajan, Gabriel Rauterberg, and Zhen (Zach) Yan.

I would like to extend my special thanks to Sanmay Das who initiated and taught the multi-agent system course at WashU (which introduced me to this field), encouraged me to pursue this graduate study, and has been providing me helpful advice even after my undergraduate study. I would also like to thank Zhuoshu Li, Elaine Wah, Yuqing Kong, Hongyao Ma, Bo Li, Fei Fang, and many other friends in the EC community for their mentorship and encouragement in various ways.

Outside research, I have been extremely lucky to be surrounded by so many great friends in Ann Arbor, in the U.S., and all the way across the ocean. I thank them for helping me navigate through the peaks and valleys of this journey, and for keeping me protected and loved all the time!

Lastly, but most importantly, I want to thank my parents for everything they have done for me! I thank them for always being on my side, even though we could only spend two or three weeks together every year. I thank them for always supporting me, encouraging me, believing in me, and bearing with me as I pursue whimsy and wonder in life. I dedicate this thesis to them.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGMENTS iii

LIST OF FIGURES ix

LIST OF TABLES xi

LIST OF APPENDICES xiii

ABSTRACT xiv

CHAPTER

1. Introduction 1

1.1 Designing Manipulation-Resistant Markets 3
1.1.1 Dynamics between a Manipulator and Investors . . 3
1.1.2 Dynamics between a Manipulator and a Regulator . 5
1.2 Designing Expressive Markets 6
1.3 Dissertation Overview 8

2. Spoofing the Limit Order Book:
A Strategic Agent-Based Analysis 10

2.1 Introduction 11
2.2 Related Work 13
2.2.1 Agent-Based Modeling of Financial Markets 13
2.2.2 Autonomous Bidding Strategies 14
2.2.3 Spoofing in Financial Markets 15
2.3 Market Model 15
2.3.1 Market Mechanism 16
2.3.2 Valuation Model 16
2.3.3 Background Trading Agents 18
2.3.4 The Spoofing Agent 21

2.4 Empirical Game-Theoretic Analysis 22
2.4.1 Profile Search 22
2.4.2 Game Reduction 23
2.5 Spoofing the Limit Order Book 24
2.5.1 Market Environments 24
2.5.2 Games without Spoofing 26
2.5.3 Games with Spoofing 28
2.5.4 Discussion 31
2.6 A Cloaking Mechanism to Mitigate Spoofing 32
2.6.1 A Cloaking Market Mechanism 33
2.6.2 Tradeoff Faced by Cloaking Mechanisms 36
2.6.3 Finding the Optimal Cloaking 38
2.6.4 Probing the Cloaking Mechanism to Spoof 39
2.7 Learning-Based Trading Strategies under the Presence of Market Manipulation 41
2.7.1 Two Variations of HBL 42
2.7.2 Empirical Evaluation 43
2.7.3 Combine Order Blocking and Price Offsets 49
2.8 Conclusions 49

3. Modeling the Evasion of Manipulation Detection: An Adversarial Learning Framework 51

3.1 Introduction 51
3.2 Related Work 54
3.2.1 Agent-Based Modeling of Trading Roles 54
3.2.2 Learning via Adversarial Training 55
3.3 Problem Formulation 55
3.3.1 Trading Strategies and Representations 55
3.3.2 An Adversarial Learning Framework 57
3.4 Experimental Results 60
3.4.1 Dataset and Implementation Details 60
3.4.2 Generating Adapted Manipulation Examples 61
3.5 Conclusions 65

4. Designing a Combinatorial Financial Options Market 67

4.1 Introduction 68
4.2 Related Work 70
4.2.1 Rational Option Pricing 70
4.2.2 Combinatorial Market Design 71
4.3 Background and Notations 71
4.4 Consolidating Standard Financial Options 72
4.4.1 Match Orders on Standard Options 72
4.4.2 Quote Prices for Standard Options 75

4.5 Combinatorial Financial Options 76
4.5.1 Match Orders on Combinatorial Options. 79
4.6 Experiments: OptionMetrics Data 82
4.7 Experiments: Synthetic Combinatorial Options Market 83
4.7.1 Generate Synthetic Orders 83
4.7.2 Evaluation 84
4.8 Discussion 86

5. Log-time Prediction Markets for Interval Securities 87

5.1 Introduction 87
5.2 Formal Setting 90
5.2.1 Cost-Function-Based Market Making 90
5.2.2 Complete Markets and LMSR 91
5.2.3 Interval Securities over $[0, 1)$ 92
5.3 A Log-time LMSR Market Maker 93
5.3.1 An LMSR Tree for $[0, 1)$ 93
5.3.2 Price Queries 95
5.3.3 Buy Transactions 96
5.4 A Multi-Resolution Linearly Constrained Market Maker . . . 99
5.4.1 A Multi-Resolution LCMM for $[0, 1)$ 99
5.4.2 Price Queries 104
5.4.3 Buy and Cost Operations 105
5.5 Discussion 108

6. Conclusion 111

APPENDICES 116

BIBLIOGRAPHY 155

LIST OF FIGURES

Figure

1.1 The decision-making process of an agent in the market. 1
1.2 Example of alleged spoofing. 4
2.1 An agent-based model of spoofing in a CDA market with a single security traded. 12
2.2 Comparisons of background-trader surplus for equilibria in each environment, with and without the HBL strategies available to background traders. 27
2.3 Comparisons of price discovery for equilibrium in each environment, with and without the HBL strategies available to background traders. 27
2.4 Transaction price differences throughout the trading horizon with and without a spoofer against each HBL-and-ZI equilibrium found in non-spoofing games. 29
2.5 HBL adoption rates at equilibria in games with and without spoofing. 31
2.6 Background-trader surplus achieved at equilibria in games with and without spoofing. 31
2.7 HBL adoption rate in equilibrium across different cloaking markets without spoofing. 37
2.8 The impact of cloaking on spoofing effectiveness. 37
2.9 Equilibrium outcomes in games with and without cloaking. 38
2.10 Exploitation payoff and transaction risk as we vary price increment δ and probing limit l. 41
2.11 Price deviations caused by spoof orders placed behind different price levels in the order book. 45
2.12 Correctly blocking spoof orders increases background-trader surplus and decreases manipulation profits. 45
2.13 Average HBL surplus differences and total number of transactions in non-spoofing markets where HBL traders use different price offsets. 47
2.14 Market price deviations caused by spoofing in markets where HBL traders use different price offsets. 47
2.15 Total background-trader surpluses and HBL strategy adoption rates achieved at equilibria across different market settings. 48
3.1 Overview of the proposed adversarial learning framework that reasons about evading a manipulation detector. 54

3.2 Order streams associated with EXP, SP, and MM in a set of controlled simulations. 58
3.3 Comparisons of the respective statistics on the SP order streams, adapted outputs, and MM order streams. 63
3.4 The manipulation effect of order streams associated with the corresponding level of SP strategy. 63
3.5 Examples of adapted manipulation order streams. 64
4.1 Payoffs of the matched options as a function of the value of the underlying asset at expiration. 75
4.2 Payoff of combinatorial options matched in Example 4.3 as a function of $S_{\texttt{AAPL}}$ and $S_{\texttt{MSFT}}$. 78
4.3 Results of using Mechanism M.2 to match orders in synthetic combinatorial options markets. 84
5.1 Left and right rotations with node z as an input. 97
5.2 The price convergence error as a function of the number of trades, measured at two resolution levels. 110
C.1 The price convergence error as a function of liquidity and the number of trades for the three respective market makers. 154

LIST OF TABLES

<u>**Table**</u>

2.1 Background trading strategies included in EGTA. 25
2.2 Additional background trading strategies included in EGTA for cloaking mechanisms. 36
2.3 Least number of probing attempts required to beat equilibrium performance. 41
2.4 Background trading strategies included to evaluate the two HBL variations. 44
2.5 Average payoffs of learning-based background traders and the exploiter, as they deviate from the equilibrium strategy profiles. . . . 45
3.1 Summary statistics of the respective trading strategy on test dataset. 62
A.1 Background-trader surplus and HBL proportion in equilibrium of markets without spoofing. 117
A.2 Background-trader surplus and HBL proportion in equilibrium of markets with spoofing. 118
A.3 Equilibria for games without spoofing, $N = 28$, calculated from the 4-player DPR approximation. 119
A.4 Equilibria for games without spoofing, $N = 65$, calculated from the 5-player DPR approximation. 120
A.5 Equilibria for games with spoofing, $N = 28$, calculated from the 4-player DPR approximation. 121
A.6 Equilibria for games with spoofing, $N = 65$, calculated from the 5-player DPR approximation. 122
A.7 Equilibria for games where agents are restricted to ZI strategies, $N = 28$, calculated from the 4-player DPR approximation. 123
A.8 Equilibria for games where agents are restricted to ZI strategies, $N = 65$, calculated from the 5-player DPR approximation. 123
A.9 Background trading strategies included in EGTA for cloaking mechanisms. 124
A.10 Equilibria for games where the exploiter does not spoof. 124
A.11 Detailed equilibria for games where the exploiter does not spoof. . . 125
A.12 Equilibria for games where the exploiter strategically chooses to spoof. 126
A.13 Detailed Equilibria for games where the exploiter strategically chooses to spoof. 127

A.14 Background trading strategies used in EGTA for HBL variations. . . 128
A.15 Equilibria for games where the learning-based trading strategy set is restricted to standard HBL. 129
A.16 Equilibria for games where the learning-based trading strategy set is comprised of standard HBL and HBL with price level blocking. . . . 130
A.17 Equilibria for games where the learning-based trading strategy set is comprised of standard HBL and HBL with price offsets. 131
A.18 Equilibria for games where the learning-based trading strategy set is comprised of standard HBL, HBL with price level blocking, HBL with price offsets, and HBL with both price offsets and price level blocking. 132
B.1 Summary statistics (matching) of options on each stock in DJI by consolidating options related to the same underlying asset and expiration date. 142
B.2 Summary statistics (quoting) of options on each stock in DJI by consolidating options related to the same underlying asset and expiration date. 143
B.3 Summary statistics (matching with $L = 0$) of options on each stock in DJI by consolidating options related to the same underlying asset and expiration date. 144
B.4 Summary statistics (quoting with $L = 0$) of options on each stock in DJI by consolidating options related to the same underlying asset and expiration date. 145

LIST OF APPENDICES

Appendix

A. Detailed Equilibrium Results for Chapter 2 117
B. Additional Proofs and Results for Chapter 4 133
C. Deferred Proofs and Additional Experiments for Chapter 5 146

ABSTRACT

Electronic trading platforms have transformed the financial market landscape, supporting automation of trading and dissemination of information. With high volumes of data streaming at high velocity, market participants use algorithms to assist almost every aspect of their decision-making: they learn market state, identify opportunities to trade, and express increasingly diverse and nuanced preferences. This growing automation motivates a reconsideration of market designs to support the new competence and prevent potential risks.

This dissertation focuses on the design of (1) manipulation-resistant markets that facilitate learning genuine market supply and demand, and (2) expressive markets that facilitate delivering preferences in greater detail and flexibility. Advances towards each may contribute to efficient resource allocation and information aggregation.

Manipulation-Resistant Markets. *Spoofing* refers to the practice of submitting spurious orders to deceive others about supply and demand. To understand its effects, this dissertation develops an agent-based model of manipulating prices in limit-order markets. *Empirical game-theoretic analysis* on agent behavior in simulated markets with and without manipulation shows that spoofing hurts market surplus and decreases the proportion of learning traders who exploit order book information. That learning behavior typically persists in strategic equilibrium even in the presence of manipulation, indicating a consistently spoofable market.

Built on this model, a *cloaking mechanism* is designed to deter spoofing via strategically concealing part of the order book. Simulated results demonstrate that the benefit of cloaking in mitigating manipulation outweighs its efficiency cost due to information loss. This dissertation explores variations of the learning-based trading strategy that reasonably compromise effectiveness in non-manipulated markets for robustness against manipulation.

Regulators who deploy detection algorithms to catch manipulation face the challenge that an adversary may obfuscate strategy to evade. This dissertation proposes an adversarial learning framework to proactively reason about how a manipulator might mask behavior. Evasion is represented by a generative model, trained by augmenting manipulation order streams with examples of normal trading. The framework generates adapted manipulation order streams that mimic benign trading patterns and appear qualitatively different from prescribed manipulation strategies.

Expressive Markets. *Financial options* are contracts that specify the right to buy or sell an underlying asset at a *strike price* in the future. Standard exchanges offer options of predetermined strike values and trade them independently, even for those written on the same asset. This dissertation proposes a mechanism to match orders on options related to the same asset, supporting trade of any custom strike. *Combinatorial financial options*—contracts that define future trades of any linear combination of underlying assets—are further introduced to enable the expression of demand based on predicted correlations among assets. Optimal clearing of such markets is coNP-hard, and a heuristic algorithm is proposed to find optimal matches through iterative constraint generation.

Prediction markets that support betting on ranges (e.g., on the price of S&P 500) offer predetermined intervals at a fixed resolution, limiting the ability to elicit fine-grained information. The *logarithmic market scoring rule* (LMSR) used in this setting presents two limitations that prevent its scaling to large outcome spaces: (1) operations run in time linear in the number of outcomes, and (2) loss suffered by the market can grow unbounded. By embedding the modularity properties of LMSR into a binary tree, this dissertation shows that operations can be expedited to logarithmic time. A constant worst-case loss can also be achieved by designing a liquidity scheme for intervals at different resolutions.

CHAPTER 1

Introduction

Financial markets were heretofore perceived as places where *people* gather to trade assets.[1] The design of mechanism or auction rules underlying a market plays a key role that directs the decision-making of participants, the aggregation of information, and the allocation of resources. Over the past few decades, markets have become to operate almost entirely electronically, supporting automation of trading and consequential scaling of volume and speed across geography and asset classes. With data streaming on a short timescale, often below the limits of human response time, *autonomous agents* directed by algorithms operate on behalf of human traders. Such increasing automation has transformed the financial market landscape from a human decision ecosystem to an algorithmic one, motivating a reconsideration of market designs that can support the new competence and prevent potential risks.

Whereas exactly how agents trade is proprietary and unknown, the incorporation of algorithmic assistance in almost every aspect of a decision loop is evident. Figure 1.1 illustrates three main aspects involved: (1) assessing market states, (2) identifying opportunities to trade, and (3) expressing demands and preferences. The first and last involve direct interactions with a market, and the second can be affected by these interactions. This dissertation focuses on the design of market mechanisms and algorithms to better facilitate (1) and (3), with the ultimate goal of ensuring the efficient resource allocation and information aggregation.

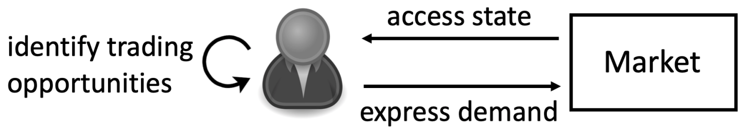


Figure 1.1: The decision-making process of an agent in the market.

[1]As chronicled in *Flash Boys: A Wall Street Revolt* by Lewis (2014).

With the assistance of algorithms, market participants have the unprecedented ability to gather and exploit information from a plethora of sources, from news articles to *order book* information disclosed by many financial exchanges. Learning and assessing market states helps to make informed trades and may contribute to improved market efficiency. However, the prevalent use of automated learning techniques introduces new possibilities for manipulation, and a distortion to one source of information can lead to a cascading effect in the greater marketplace. One common type of manipulative practices is *order-based manipulation*, applied through a series of direct trading actions in a market. Rather than expressing true trading intent, spurious orders are submitted—often also aided by algorithms—to maneuver the market state: they feign a strong buy or sell interest to deceive investors who learn from others' bidding activities. As algorithms respond to information much faster than humans, the market can reflect misled beliefs in milliseconds, with prices moving toward the crafted direction that benefits the manipulator. *This dissertation combines methods from agent-based modeling, game-theoretic analysis, and adversarial learning to examine the mechanics behind such manipulation and propose manipulation-resistant designs.*

With the assistance of algorithms, market participants become more capable of computing and identifying complex but well-defined investment objectives (e.g., achieving a particular return, hedging exposure risks, or speculating the movement of a portfolio of assets). This competence is accompanied with the need of custom financial instruments, as well as a higher level of *expressiveness* in the mechanism—the ability to provide agents the means to express more diverse demands and nuanced preferences. Most current markets, however, fail to tailor to such individualized needs, offering standardized contracts with predetermined characteristics. This keeps the mechanism and its operations clean and simple, but puts burdens on investors who have to craft trades across available markets and bear the risk of execution failure. Increasing expressiveness may in fact benefit a mechanism: the market is able to incorporate information of greater detail to optimize outcome, improving economic efficiency, and obtain high-quality information aggregation. *The second part of this dissertation investigates expressive designs for financial options markets and prediction markets, analyzing computational complexity of increased expressiveness and proposing algorithms to facilitate computationally efficient operations.*

In sections below, I provide more background on the two identified problems, and outline the computational approaches this dissertation adopts.

1.1 Designing Manipulation-Resistant Markets

Market manipulation is defined by the U.S. Securities and Exchange Commission (SEC) as "intentional or willful conduct designed to deceive or defraud investors by controlling or artificially affecting the price of securities, or intentional interference with the free forces of supply and demand". Though it has long been present, the practice has evolved in its forms to exploit automated trading and the dissemination of market information offered by many trading platforms (Lin 2015).

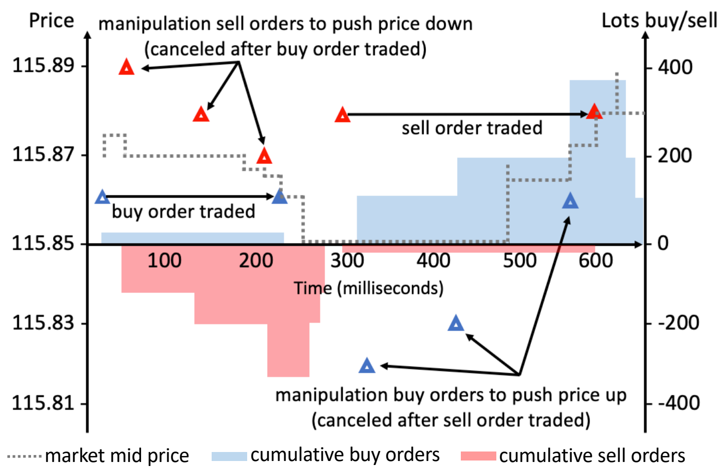
On July 22, 2013, the U.S. Commodity Futures Trading Commission (CFTC) filed charges against Michael Coscia for manipulating a broad spectrum of commodities on the Chicago Mercantile Exchange (Patterson and Trindle 2013). Coscia utilized a computer algorithm that quickly placed and canceled orders to mislead the market about demand and supply for these contracts. The trial evidence suggested that such practice allowed Coscia to buy low and sell high in a market artificially distorted by his actions, purportedly earning him \$1.4 million. Figure 1.2 illustrates an episode of the alleged *spoofing* activity conducted over the course of 0.6 seconds.

The case of Coscia was the first prosecuted under the Dodd-Frank Act passed in 2010, but not the only one occurring in today's marketplace. A recent lawsuit claimed evidence of thousands of manipulation episodes in the U.S. Treasury futures observed during 2013 and 2014 (Hope 2015b). Since 2016, the SEC has brought legal action over a hundred cases of manipulation (U.S. SEC 2017, 2018, 2019), and new allegations have been emerging on a regular basis.

Despite regulatory enforcement efforts, manipulation is hard to eliminate due to (1) the difficulty of determining the manipulation intent behind placement of orders, and (2) the adversarial nature of a manipulator who adapts to evade regulation and detection. This calls for a more comprehensive understanding of the dynamics between a manipulator and other market participants—as well as the dynamics between a manipulator and a regulator—to design deterrent measures, ensuring the general efficiency and integrity of a marketplace.

1.1.1 Dynamics between a Manipulator and Market Participants

Prior work that investigates order-based market manipulation has primarily relied on examination of historical trading data (Lee, Eom, and Park 2013; Wang 2019). Researchers conduct empirical analysis to characterize manipulation patterns and market conditions where manipulation is more likely to occur and be effective. Grounding on historical data, pure data-driven approaches can provide insights to ob-



A sequence of limit orders submitted by Coscia over the course of 0.6 seconds. A series of large out-of-the money manipulation sell orders (red triangles) are first placed to drive the price down and make the buy order accepted (the filled blue triangle). These sell orders are immediately replaced with large buy ones (blue triangles) to push the price up and profit from the sale at a higher price (the filled red triangle).

Figure 1.2: Example of alleged spoofing. Source: UK Financial Conduct Authority Final Notice 2013.

servational questions, delivering findings that may better reflect the real world situation. However, they present fundamental limitations when concerning counterfactual questions (e.g., what would change if a certain action is not taken), and answers to such questions can be highly relevant to identifying manipulation and understanding its impact.

Analytic models can be an useful approach to analyze the mechanics behind market manipulation. Allen and Gale (1992) develop a model of *transaction-based manipulation*, and compute equilibrium based off of this model where the existence of noise traders makes it possible to manipulate prices. Fishman and Hagerty (1995) propose a one-period equilibrium model of *information-based manipulation* where uninformed insiders can make a profit by pretending they are informed and disclosing their trades. Both models rely on highly simplified context and assumptions (e.g., limited number of trading stages, probabilistic information disclosure) to derive equilibrium and demonstrate the theoretical possibility of manipulation. In our case, however, the manipulation practice of interest relies on complex features, such as frequent entries into the market and propagation of order book information between the market and agents. These features are essential to the problem at hand and may not be removed or easily stylized for tractability.

This dissertation adopts a computational approach that lies somewhere in between of the two approaches discussed: it combines agent-based modeling (ABM) and empirical game-theoretic analysis (EGTA) (Wellman 2006) to study the effect of manipulation on trading behavior and market performance in equilibrium. ABM takes a simulation approach to reproduce phenomena of interest through the dynamic interactions of agents. It enables the designer to incorporate any desired level of complexity into the model, and provides the means to acquire counterfactual information. By simulating different scenarios and conducting controlled experiments, one can evaluate how a certain factor affects agent and system behavior. Despite the merits and flexibility, ABM presents a challenge that may affect its practicability: simulated data can vary as one adopts different design choices (e.g., environment, prescribed strategies, agent composition), and thus analysis conducted upon it may or may not accurately reflect situation in reality.

This dissertation alleviates this issue by exploring a wide range of environments and employing EGTA in each to focus on the most relevant strategic context. From the agent-based model, EGTA induces a normal-form game defined by heuristic strategy space and simulated agent utilities, and solves for Nash equilibria (or other game-theoretic solution concepts) to determine agent behavior. By such, rather than prescriptively assigning strategies to agents and exploring all possible combinations, a designer is directed to a strategically stable setting where agents are making the best choices among their available strategies, given an environment and others' choices.

Combining ABM and EGTA, Chapter 2 develops the first computational model that reproduces spoofing in a dynamic limit-order mechanism, and demonstrates the effectiveness of manipulating against approximate-equilibrium traders. The model offers a constructive basis to quantify the effect of manipulation practices and evaluate any preventive or deterrent proposals under strategic settings. Chapter 2 proceeds to explore variations of trading strategies that may exploit market information in less vulnerable ways, and proposes a mechanism to disincentive manipulation via strategic disclosure of the order book.

1.1.2 Dynamics between a Manipulator and a Regulator

Deterrent mechanisms intend to render manipulative strategies uneconomical; a more direct approach is to detect any manipulation activity. The automated and high-frequency nature of many manipulation practices has led efforts to automate detection. Nasdaq announced an AI-based surveillance system trained with historical data and spotted patterns of market-abuse techniques to detect suspect equities trading episodes (Rundle 2019). Despite recent advances in pattern recognition algorithms, developing high-fidelity detection systems faces the all-time challenge that an adversary may obfuscate its strategies to escape detection (e.g., manipulating in a way that appears as normal trading activity). This causes regulators to play a costly game of cat-and-mouse with manipulators who constantly innovate to evade.

Such a contest resembles the workings of *generative adversarial nets* (GANs) (Goodfellow et al. 2014): the generative model—analogous to the manipulator—learns to fool a discriminative model—the detector, by producing novel candidates that the discriminator believes are part of the true data distribution (e.g., normal trading patterns). Building on this connection, the dissertation proposes an adversarial learning framework to proactively reason about how a manipulator might mask its behavior to evade detection. The framework differs from a vanilla GAN model in two main aspects. First, it takes prescribed spoofing traces as inputs as opposed to randomized inputs in GANs. Second, it attempts to resemble a target distribution while preserving a comparable manipulation effect, whereas resemblance is the only objective in GANs. Ultimately, the generated "unseen" manipulative examples can serve to train more robust detection algorithms.

Biggio, Fumera, and Roli (2014) identify four stages in the scheme of proactive security: (1) model adversary, (2) simulate attack, (3) evaluate attack's impact, and (4) develop countermeasures, if the attack has relevant impact. The proposed adversarial framework combines a variant of GAN and the developed agent-based model of manipulation to perform the four steps iteratively. Evasion (or the adversary) is modeled by a generator that learns to adapt original manipulation activities to resemble trading patterns of a normal trader. The agent-based simulator performs (2) and (3), and the generative and discriminative model respectively conducts (1) and (4). Whereas such an adversarial framework cannot capture all changing aspects of an adversary, it is generally believed that proactive reasoning delays each step of the reactive arms race, forcing the adversary to exert greater efforts to find vulnerabilities (Biggio, Fumera, and Roli 2014).

1.2 Designing Expressive Markets

The second part of this dissertation studies mechanisms and algorithms to improve the expressiveness of financial markets. By giving participants greater flexibility to express preferences and beliefs, a market mechanism can incorporate more inputs to optimize for outcome, increase economic efficiency, and obtain high-quality information aggregation. However, a higher-level of expressiveness may come at the cost of a more intricate mechanism that is computationally expensive.[2] This dissertation inves-

[2]Several works have formally described and quantified tradeoffs of this form (Benisch, Sadeh, and Sandholm 2008; Golovin 2007).

tigates the use of optimization methods and computationally-efficient data structures to facilitate and expedite key operations required by an expressive market.

Take *financial options markets* as an example. An option is a contract that specifies the contract holder the right to buy or sell of an underlying asset at some agreed *strike price* in the future. On standard exchanges, markets for options written on a specific underlying asset feature a selective set of predetermined strike prices. For example, as of this writing, the Chicago Board Options Exchange (CBOE) offers around forty distinct strike prices, ranging from \$100 to \$320 at intervals of \$5 or \$10, for `MSFT` options expiring on September 17, 2021. While one can engineer custom contracts (e.g., a `MSFT` call option with strike price \$202) by simultaneously purchasing multiple available options at appropriate proportions, it requires monitoring several markets to ensure that a bundle can be constructed at a desired price. Often, execution risk and transaction costs prevent traders from carrying out such strategies. As a result, the exchange may fail to aggregate supply and demand requests of greater detail, leading to a loss of economic efficiency.

For the set of offered strikes, standard exchanges operate separate markets, having each independently aggregate and match orders of a designated strike price, despite the interconnectedness and their common dependency on the underlying asset. Such independent market design fails to match options with different strike prices, and may introduce arbitrage opportunities. Moreover, investments get diluted across independent markets even when participants are interested in the same underlying asset. This can cause the problem of *thin markets*, where few trades happen and bid-ask spreads become wide. Empirical evidence has shown that even for some of the most actively traded options, liquidity can vary much across option types and strikes (Cao and Wei 2010).

Besides financial options market, prediction markets that facilitate trading the outcome of events share a similar limitation due to their predetermined designs. For instance, markets that elicit predictions of an outcome variable, such as the time FDA will approve a vaccine or the threshold S&P 500 will hit by the end of the year, often restrain the outcome space to some pre-defined intervals at a certain resolution (e.g., quarters of a year, or ranges of every thousand dollars). Such prescriptive design, by clustering betting interest, attenuates the thin market problem. However, it prevents agents with expert knowledge from expressing more accurate information (e.g., the month, date, or even time of a vaccine release). The popular *logarithmic market scoring rule* (LMSR) (Hanson 2003) has been used in this setting to subsidize trading and aggregate information at different granularity levels in a single market.

However, it suffers two limitations that prevent its scaling to markets with large outcome spaces. First, the worst-case loss of an LMSR market can grow unbounded if agents select outcomes with prior probability approaching zero (Gao, Chen, and Pennock 2009). Second, standard implementations of LMSR operations run in time linear in the number of outcomes or distinct future values agents define, which can be arbitrarily many in a continuous outcome space.

To address issues identified above, this dissertation proposes mechanisms and algorithms to improve market expressiveness. In many cases, computational techniques can help to exploit certain payoff properties or outcome structures present in a market to enjoy a desired level of expressiveness without compromising computational efficiency. Chapter 4 presents a mechanism that utilizes a linear program to consolidate and match orders on standard options related to the same underlying asset, while providing traders the flexibility to specify any custom strike value. Market operations, including match and price quotes, require time polynomial in the number of orders. Chapter 5 proposes a balanced-binary tree data structure that successfully decomposes LMSR calculations along the tree nodes, thus expediting market operations exponentially faster than previous designs.

This dissertation also demonstrates the case when a higher level of expressiveness renders a market computationally intractable. Chapter 4 generalizes standard options to *combinatorial financial options*, which specify the right to buy or sell any defined linear combination of underlying assets at some agreed strike price. Such contracts provide investors the means to speculate relative movements among stocks, thus enabling the elicitation of future correlations among underlying assets. This increased expressiveness of the mechanism, however, comes at the cost of higher computational complexity: optimal clearing of such a market is coNP-hard. Chapter 4 demonstrates that with a proposed heuristic algorithm, the computational hardness may be surmountable in practice.

1.3 Dissertation Overview

This introductory chapter has provided a broad perspective on the two categories of problems the dissertation aims to address and a brief description of the employed methodology and computational techniques. The remainder of this dissertation provides the details.

Chapter 2 presents the computational agent-based model of spoofing (Section 2.3), proposes deterrent mechanisms (Section 2.6), and explores trading strategies to improve learning robustness against manipulation (Section 2.7). Section 2.4 details the EGTA methodology, which is adopted in all three studies to provide strategic analysis. Some of the material in this chapter has appeared in published work (Wang, Hoang, and Wellman 2020; Wang, Vorobeychik, and Wellman 2018; Wang and Wellman 2017).

Chapter 3 describes the adversarial learning framework developed to reason about evading any manipulation detection. It uses the built agent-based market simulator in Chapter 2 to generate spoofing and non-manipulative order streams (served as training data) and to evaluate the manipulation effect of adapted outputs. Part of the material from this chapter has appeared in published work (Wang and Wellman 2020).

Chapter 4 studies expressive designs for financial options markets. Section 4.4 specifies the mechanism that improves matching standard options. Section 4.5 defines combinatorial financial options, and investigates matching mechanisms and computational complexity for such a market. Material from this chapter is under submission (Wang et al. 2020).

Chapter 5 presents two efficient designs of prediction markets that recover a complete and fully general probability distribution of a random variable. Section 5.3 details the balanced tree construction that embeds LMSR calculations and expedites market operations. Section 5.4 describes a different binary tree structure, augmented with a liquidity scheme to enable a constant loss bound. Some of the material in this chapter is in paper to appear at the *20th International Conference on Autonomous Agents and Multiagent Systems* (Dudík, Wang, Pennock, and Rothschild 2020).

Chapter 6 concludes with a summary of contribution and a discussion of limitations and future directions.

CHAPTER 2

Spoofing the Limit Order Book: A Strategic Agent-Based Analysis

This chapter presents an agent-based model of manipulating prices in financial markets through *spoofing*; it is a form of order-based manipulation that operates by submitting spurious orders to mislead traders who learn from the order book. Built around the limit-order mechanism, the model captures a complex market environment with combined private and common values, the latter represented by noisy observations upon a dynamic fundamental time series. In this model, we consider background agents following two types of trading strategies: the non-spoofable *zero intelligence* (ZI) that ignores the order book and the manipulable *heuristic belief learning* (HBL) that exploits the order book to predict price outcomes. We conduct *empirical game-theoretic analysis* upon simulated agent payoffs across parametrically different environments, and measure the effect of spoofing on market performance in approximate strategic equilibria.

We demonstrate that HBL traders can improve price discovery and social welfare, but their existence in equilibrium renders a market vulnerable to manipulation: simple spoofing strategies can effectively mislead traders, distort prices, and reduce total surplus. Based on this model, the chapter further proposes to mitigate spoofing from two aspects: (1) mechanism design to disincentivize manipulation and (2) trading strategy variations to improve the robustness of learning from market information. We evaluate the proposed approaches, taking into account potential strategic responses of agents, and characterize the conditions under which these approaches may deter manipulation and benefit market welfare. The model proposed here provides a way to quantify the effect of spoofing on trading behavior and market efficiency, and thus can help to evaluate the effectiveness of various market designs and trading strategies in mitigating an important form of market manipulation.

2.1 Introduction

On April 21, 2015, nearly five years after the "Flash Crash",[1] the U.S. Department of Justice charged Navinder Singh Sarao with 22 criminal counts, including fraud and spoofing. Prior to the Flash Crash, Sarao allegedly used an algorithm to place orders amounting to about \$200 million seemingly betting that the market would fall, and later replaced or modified those orders 19,000 times before cancellation. The U.S. Commodity Futures Trading Commission (CFTC) concluded that Sarao's manipulative practice was responsible for significant order imbalances. Though recent analysis has cast doubt on the causal role of Sarao on the Flash Crash (Aldrich, Grundfest, and Laughlin 2017), many agree that such manipulation could increase the vulnerability of markets and exacerbate market fluctuations. An illustrative execution trace of a similar spoofing strategy has been presented and discussed in Chapter 1 Figure 1.2, demonstrating how quickly and effectively such manipulation behavior can affect the market and profit from the spoofed belief.

Specifically, spoofing operates through a series of direct trading actions in a market. Traders interact with the market by submitting orders to buy or sell. Orders that do not transact immediately rest in the *order book*, a repository for outstanding orders to trade. At any given time, the order book for a particular security reflects the market's expressed supply and demand. A spoofer (or manipulator) submits large *spurious* buy or sell orders with the intent to cancel them before execution. The orders are spurious in that instead of expressing genuine trading intent, they feign a strong buy or sell interest in the market, thus corrupting the order book's signal on supply and demand. Such orders can be viewed as *targeted attacks* (Huang et al. 2011), designed to mislead others who learn from the order book to believe that prices may soon rise or fall and subsequently alter their trading behavior in a way that will directly move the price. To profit on its feint, the manipulator can submit a real order on the opposite side of the market and as soon as the real order transacts, cancel all the spoof orders.

In 2010, the Dodd-Frank Wall Street Reform and Consumer Protection Act was signed into U.S. law, outlawing spoofing as a deceptive practice. In describing its concern about spoofing, the CFTC notes that "many market participants, relying on the information contained in the order book, consider the total relative number of bid and ask offers in the order book when making trading decisions". In fact, spoofing

[1]The Flash Crash was a sudden trillion-dollar dip in U.S. stock markets on May 6, 2010, during which stock indexes collapsed and rebounded rapidly (Kirilenko, Kyle, Samadi, and Tuzun 2017).

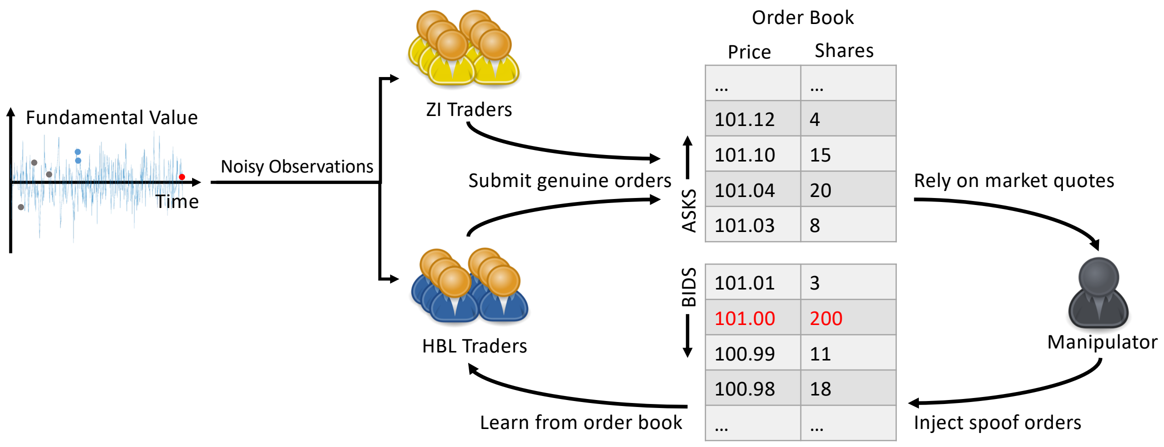


Figure 2.1: An agent-based model of spoofing in a CDA market with a single security traded.

can be effective only to the extent that traders actually use order book information to make trading decisions. In ideal markets without manipulation, traders may extract useful information from the order book, making more informed decisions over those that neglect such information. A manipulator exploits such learning process, minimizing its own risk in the process. Spoof orders are typically placed at price levels just outside the current best quotes to mislead other investors, and withdrawn with high probability before any market movement could trigger a trade (Hope 2015a; Montgomery 2016).

This chapter reproduces spoofing in a computational model, as a first step toward developing more robust measures to characterize and prevent spoofing. We adopt an agent-based modeling approach to simulate the interactions among players with different strategies. Figure 2.1 gives an overview of our proposed agent-based market model. The model implements a *continuous double auction* (CDA) market with a single security traded. The CDA is a two-sided mechanism adopted by most financial and commodity markets (Friedman 1993). Traders can submit limit orders at any time, and whenever an incoming order matches an existing one they trade at the incumbent order's limit price.

The market is populated with multiple background traders and in selected treatments, one manipulator who executes the spoofing strategy. Background traders are further divided to follow two types of trading strategies: *zero intelligence* (ZI) that ignores the order book and *heuristic belief learning* (HBL) that learns from the order book to predict price outcomes. Upon each arrival to trade, a background trader receives a noisy observation of the security's fundamental value. Based on a series of fundamental observations and its private value, a ZI agent computes the limit-order price by shading a random offset from its valuation, and thus is non-manipulable. An HBL agent, on the other hand, is susceptible to spoofing: it considers information about orders recently submitted to the market, estimates the probability that orders at various prices would be transacted, and chooses the optimal price to maximize expected surplus. The manipulator in our model executes a spoofing strategy similar to that illustrated in Figure 1.2. The spoofer injects and maintains large spurious buy orders at one tick behind the best bid, designed to manipulate the market by misleading others about the level of demand.

We conduct extensive simulation over hundreds of strategy profiles across parametrically different market environments with and without manipulation. The simulation data is used to estimate normal-form game models, from which we derive empirical equilibria, where every agent chooses its best response to both the market environment and others' behavior. Our goal is to (1) reproduce spoofing and understand its impact on market performance (Section 2.5) and (2) propose and evaluate variations of market designs (Section 2.6) and learning-based trading strategies (Section 2.7) in mitigating manipulation.

2.2 Related Work

2.2.1 Agent-Based Modeling of Financial Markets

Agent-based modeling (ABM) takes a simulation approach to study complex domains with dynamically interacting decision makers. ABM has been frequently applied to modeling and understanding phenomena in financial markets (Lebaron 2006), for example to study the Flash Crash (Paddrik et al. 2012) or to replicate the volatility persistence and leptokurtosis characteristic of financial time series (LeBaron, Arthur, and Palmer 1999). A common goal of agent-based finance studies is to reproduce stylized facts of financial market behavior (Palit, Phelps, and Ng 2012), and to support causal reasoning about market environments and mechanisms. Researchers have also use ABM to investigate the effects of particular trading practices, such as market making (Wah, Wright, and Wellman 2017) and latency arbitrage (Wah and Wellman 2016). ABM advocates argue that simulation is particularly well-suited to study financial markets (Bookstaber 2012), as analytic models in this domain typically require extreme stylization for tractability, and pure data-driven approaches cannot answer questions about changing market and agent designs.

2.2.2 Autonomous Bidding Strategies

There is a substantial literature on autonomous bidding strategies in CDA markets (Wellman 2011). The basic *zero intelligence* (ZI) strategy (Gode and Sunder 1993) submits offers at random offsets from valuation. Despite its simplicity, ZI has been shown surprisingly effective for modeling some cases (Farmer, Patelli, and Zovko 2005). In this study, we adopt an extended and parameterized version of ZI to represent trading strategies that ignore order book information.

Researchers have also extended ZI with adaptive features that exploit observations to tune themselves to market conditions.[2] For example, the *zero intelligence plus* (ZIP) strategy outperforms ZI by adjusting an agent-specific profit margin based on successful and failed trades (Cliff 1997, 2009). Vytelingum, Cliff, and Jennings (2008) introduce another level of strategic adaptation, allowing the agent to control its behavior with respect to short and long time scales.

Gjerstad proposed a more direct approach to learning from market observations, termed *GD* in its original version (Gjerstad and Dickhaut 1998) and named *heuristic belief learning* (HBL) in a subsequent generalized form (Gjerstad 2007). The HBL model estimates a heuristic belief function based on market observations over a specific memory length. Variants of HBL (or GD) have featured prominently in the trading agent literature. For example, Tesauro and Das (2001) adapt the strategy to markets that support persistent orders. Tesauro and Bredin (2002) show how to extend beyond myopic decision making by using dynamic programming to optimize the price and timing of bids.

We adopt HBL as our representative class of agent strategies that exploit order book information. HBL can be applied with relatively few tunable strategic parameters, compared to other adaptive strategies in the literature. We extend HBL to a more complex market environment that supports persistent orders, combined private and fundamental values, noisy observations, stochastic arrivals, and the ability to trade multiple units with buy or sell flexibility. The extended HBL strategy considers the full cycle of an order, including the times an order is submitted, accepted, canceled, or rejected.

[2]To some extent, the adaptive functions of these strategies are implicitly achieved by the game-theoretic equilibration process which we employ to determine the parametric configurations of the (non-adaptive) trading strategies (Wright and Wellman 2018).

2.2.3 Spoofing in Financial Markets

The literature on spoofing and its impact on financial markets is fairly limited. Some empirical research based on historical financial market data has been conducted to understand spoofing. Lee, Eom, and Park (2013) empirically examined spoofing by analyzing a custom data set, which provides the complete intraday order and trade data associated with identified individual accounts in the Korea Exchange. They found investors strategically spoof the stock market by placing orders with little chance to transact to add imbalance to the order book. They also discovered that spoofing usually targets stocks with high return volatility but low market capitalization and managerial transparency. Wang investigated spoofing on the index futures market in Taiwan, identifying strategy characteristics, profitability, and real-time impact (Wang 2019). Martínez-Miranda, McBurney, and Howard (2016) implemented spoofing behavior within a reinforcement learning framework to model conditions where such behavior is effective. Tao, Day, Ling, and Drapeau (2020) presented a micro-structural study of spoofing in a static setting, providing conditions under which a market is more likely to admit spoofing behavior as a function of the characteristics of the market. Beyond traditional financial markets, Chen et al. (2007) studied the equilibrium behavior of informed traders interacting with automated market makers in prediction markets, and examined circumstances when traders can benefit by either hiding or lying about information.

To our knowledge, we provide the first computational model of spoofing a dynamic financial market, and demonstrate the effectiveness of spoofing against approximate-equilibrium traders in this proposed model. Our model provides a way to quantify the effect of spoofing on trading behavior and efficiency, and thus a first step in the design of methods to deter or mitigate market manipulation.

2.3 Market Model

We present the general structure of the agent-based financial market environment in which we model spoofing. Our model comprises agents trading a single security through a continuous double auction (CDA), the mechanism adopted by most financial markets today. We first describe the market mechanism in Section 2.3.1. Our model is designed to capture key features of market microstructure (e.g., fundamental shocks and observation noise), supporting a configurable simulator to understand the effect of spoofing under different market conditions. The market is populated with multiple background traders who represent investors in the market, and in selected treatments, a spoofer who seeks trading profit through manipulative action. We specify the valuation model of background traders in Section 2.3.2 and the two families of background-trader strategies in Section 2.3.3. In Section 2.3.4, we discuss the behavior of the spoofing agent.

2.3.1 Market Mechanism

The market employs a CDA mechanism with a single security traded. Prices are fine-grained and take discrete values at integer multiples of the tick size. Time is also fine-grained and discrete, with trading over a finite horizon T. Agents in the model submit limit orders, which specify the maximum (minimum) price at which they would be willing to buy (sell) together with the number of units to trade. Orders are immediately matched as they arrive: if at any time, one agent's maximum price to buy a unit is greater than or equal to another agent's minimum price to sell a unit, a transaction will occur and the agents trade at the price of the incumbent order.

The CDA market maintains a *limit order book* of outstanding orders, and provides information about the book to traders with zero delay. The buy side of the order book starts with BID_t, the highest-price buy order at time t, and extends to lower prices. Similarly, the sell side starts with ASK_t, the lowest-price sell order at time t, and extends to higher prices. On order cancellation or transaction, the market removes the corresponding orders and updates the order book. Agents may use order book information at their own discretion. In Section 2.6, we investigate how changes made in such order book disclosure may help to mitigate spoofing.

2.3.2 Valuation Model

Each background trader has an individual valuation for the security, which is comprised of a private value and a common value component. The common component is represented as a *fundamental* value, r_t, which changes throughout the trading period according to a mean-reverting stochastic process:

$$r_t = \max\{0, \kappa\bar{r} + (1-\kappa)r_{t-1} + u_t\}; r_0 = \bar{r}. \tag{2.1}$$

Here r_t denotes the fundamental value of the security at time $t \in [0, T]$, and the parameter $\kappa \in [0, 1]$ specifies the degree to which the value reverts back to a fundamental mean $\bar{r}$. A process with $\kappa = 0$ corresponds to a martingale Gaussian fundamental, whereas $\kappa = 1$ specifies a process of i.i.d. Gaussian draws around the fundamental mean. A mean-reverting time series of this sort has been empirically observed in financial markets such as foreign exchange and commodity markets (Chakraborty and Kearns 2011). The perturbation u_t captures a systematic random shock upon the fundamental at time t, and is normally distributed as $u_t \sim N(0, \sigma_s^2)$, where σ_s^2 represents an environment-specific shock variance. The shock variance governs fluctuations in the fundamental time series, and consequently affects the predictability of future price outcomes.

Our time-varying fundamental induces *adverse selection*, a situation where outstanding orders reflect outdated information and thus can be at a disadvantage at the current time. If the fundamental shifts significantly, subsequent agents are more likely to transact with orders on the side opposite to the direction of fundamental change. That is, a positive price shock will tend to trigger transactions with stale sell orders, and a negative shock with stale buys. An agent's exposure to adverse selection in a market is jointly controlled by the fundamental shock variance σ_s^2, the degree of mean reversion κ, and the arrival rate of that agent.

The entries of a background trader follow a Poisson process with an arrival rate λ_{a}. Upon each entry, the trader observes an agent-and-time-specific noisy fundamental $o_t = r_t + n_t$, where the observation noise n_t is drawn from $n_t \sim N(0, \sigma_n^2)$. Just as in real financial markets, investors will never know the true value of the underlying security, such noisy observations represent each trader's assessment of the security's fundamental value at that time. Given its incomplete information about the fundamental, the agent can potentially benefit by considering market information, which is influenced by and therefore reflects the aggregate observations of other agents. When it arrives, the trader withdraws its previous order (if untransacted) and submits a new single-unit limit order, either to buy or sell as instructed with equal probability.

The *private value* of a background trader i represents its individual preferences on holding a long or short position of the security:

$$\Theta_i = (\theta_i^{-q_{\max}+1}, \ldots, \theta_i^0, \theta_i^1, \ldots, \theta_i^{q_{\max}}).$$

The vector has a length of $2q_{\max}$, where $q_{\max}$ is the maximum number of units a trader can be long or short at any time. Element θ_i^q in the vector specifies the incremental private benefit *foregone* by selling one unit of the security given a current net position of q. Alternatively, θ_i^{q+1} can be understood as the marginal private gain from buying an additional unit given current net position q. To capture the diminishing marginal utility, that is $\theta^{q'} \leq \theta^q$ for all $q' \geq q$, we generate Θ_i from a set of $2q_{\max}$ values drawn independently from $N(0, \sigma_{PV}^2)$, sort elements in descending order, and assign θ_i^q to its respective value in the sorted list.

Agent i's incremental surplus for a trade can be calculated based on its position q before the trade, the value of the fundamental at the end of the trading horizon r_T, and the transaction price p:

$$\text{incremental surplus} = \begin{cases} r_T - p + \theta_i^{q+1} & \text{if buying 1 unit,} \\ p - r_T - \theta_i^q & \text{if selling 1 unit.} \end{cases}$$

An agent's total surplus is the sum of the agent's incremental surplus over all transactions. Alternatively, we can also calculate an agent's total surplus by adding its net cash from trading to the *final valuation* of holdings. Specifically, the market's final valuation of trader i with ending holdings H is

$$v_i = \begin{cases} r_T \times H + \sum_{k=1}^{k=H} \theta_i^k & \text{for long position } H > 0, \\ r_T \times H - \sum_{k=H+1}^{k=0} \theta_i^k & \text{for short position } H < 0. \end{cases}$$

We define *background-trader surplus* as the sum of all background agents' surpluses at the end of the trading period T.

2.3.3 Background Trading Agents

Recall that background traders represent investors with actual preferences for holding long or short positions in the underlying security. The limit-order price submitted by a background trader is jointly decided by its *valuation* and *trading strategy*, which we describe in detail below.

2.3.3.1 Estimating the Final Fundamental

As holdings of the security are evaluated at the end of a trading period (i.e., $r_T \times H$), a background trader estimates the final fundamental value based on a series of its noisy observations. We assume the market environment parameters (mean reversion, shock variance, etc.) are common knowledge for background agents.

Given a new noisy observation o_t, an agent estimates the current fundamental by updating its posterior mean $\tilde{r}_t$ and variance $\tilde{\sigma}_t^2$ in a Bayesian manner. Let t' denote the agent's preceding arrival time. We first update the previous posteriors, $\tilde{r}_{t'}$ and

$\tilde{\sigma}^2_{t'}$, by mean reversion for the interval since preceding arrival, denoted $\delta = t - t'$:

$$\tilde{r}_{t'} \leftarrow (1-(1-\kappa)^{\delta})\bar{r} + (1-\kappa)^{\delta}\tilde{r}_{t'} \quad \text{and} \quad \tilde{\sigma}^2_{t'} \leftarrow (1-\kappa)^{2\delta}\tilde{\sigma}^2_{t'} + \frac{1-(1-\kappa)^{2\delta}}{1-(1-\kappa)^2}\sigma^2_s.$$

The estimates for the current arrive at time t are then given by

$$\tilde{r}_t = \frac{\sigma^2_n}{\sigma^2_n + \tilde{\sigma}^2_{t'}}\tilde{r}_{t'} + \frac{\tilde{\sigma}^2_{t'}}{\sigma^2_n + \tilde{\sigma}^2_{t'}}o_t \quad \text{and} \quad \tilde{\sigma}^2_t = \frac{\sigma^2_n \tilde{\sigma}^2_{t'}}{\sigma^2_n + \tilde{\sigma}^2_{t'}}.$$

Based on the posterior estimate of $\tilde{r}_t$, the trader computes $\hat{r}_t$, its estimate at time t of the terminal fundamental r_T, by adjusting for mean reversion:

$$\hat{r}_t = \big(1-(1-\kappa)^{T-t}\big)\bar{r} + (1-\kappa)^{T-t}\tilde{r}_t. \tag{2.2}$$

2.3.3.2 Zero Intelligence (ZI) as a Background Trading Strategy

We consider parameterized trading strategies in the zero intelligence (ZI) family (Gode and Sunder 1993). Background traders who choose to adopt ZI strategies compute limit-order prices solely based on fundamental observations and private values. Specifically, the ZI agent shades its bid from its valuation by a random offset, which is uniformly drawn from $[R_{\min}, R_{\max}]$. Specifically, a ZI trader i arriving at time t with position q generates a limit price

$$p_i(t) \sim \begin{cases} U[\hat{r}_t + \theta^{q+1}_i - R_{\max}, \hat{r}_t + \theta^{q+1}_i - R_{\min}] & \text{if buying,} \\ U[\hat{r}_t - \theta^{q}_i + R_{\min}, \hat{r}_t - \theta^{q}_i + R_{\max}] & \text{if selling.} \end{cases} \tag{2.3}$$

Our version of ZI further considers the market's current best quotes, and can choose to immediately trade to get a certain fraction of its requested surplus. This option is governed by a strategic threshold parameter $\eta \in [0, 1]$: if the agent could achieve a fraction η of its requested surplus at the current price quote, it would simply take that quote rather than submitting a new limit order. Setting η to 1 is equivalent to the strategy without a threshold.

2.3.3.3 Heuristic Belief Learning (HBL) as a Background Trading Strategy

The second background trading strategy family we consider is heuristic belief learning (HBL). Background traders who choose to adopt HBL go beyond their own observations and private values by also considering order book information. We make a set of changes to adapt the strategy to our dynamic market environment, supporting multiple-unit trading with a flexible buy or sell role.

The strategy is centered on the belief function that a background trader forms on the basis of its observed market data. The agent uses the belief function to estimate the probability that orders at various prices would be accepted in the market, and then chooses a limit price that maximizes its expected surplus at current valuation estimates.

Specifically, an HBL agent constructs its belief function based on a dataset $\mathcal{D}$ that records accepted and rejected buy and sell orders during the last L trades. The strategic parameter L represents the agent's memory length, which controls the size of $\mathcal{D}$. Upon an arrival at time t, the HBL agent builds a belief function $f_t(P)$, designed to represent the probability that an order at price P will result in a transaction. Specifically, the belief function is defined for any encountered price P as the following:

$$f_t(P \mid \mathcal{D}) = \begin{cases} \dfrac{\mathrm{TBL}_t(P \mid \mathcal{D}) + \mathrm{AL}_t(P \mid \mathcal{D})}{\mathrm{TBL}_t(P \mid \mathcal{D}) + \mathrm{AL}_t(P \mid \mathcal{D}) + \mathrm{RBG}_t(P \mid \mathcal{D})} & \text{if buying,} \\ \\ \dfrac{\mathrm{TAG}_t(P \mid \mathcal{D}) + \mathrm{BG}_t(P \mid \mathcal{D})}{\mathrm{TAG}_t(P \mid \mathcal{D}) + \mathrm{BG}_t(P \mid \mathcal{D}) + \mathrm{RAL}_t(P \mid \mathcal{D})} & \text{if selling.} \end{cases} \tag{2.4}$$

Here, T and R specify *transacted* and *rejected* orders respectively; A and B represent *asks* and *bids*; L and G describe orders with prices *less* than or equal to and *greater* than or equal to price P respectively. For example, $\mathrm{TBL}_t(P \mid \mathcal{D})$ is the number of transacted bids found in the memory with price less than or equal to P up to time t. An HBL agent updates its dataset $\mathcal{D}$ whenever the market receives new order submissions, transactions, or cancellations, and computes the statistics in Eq. (2.4) upon each arrival.

Since our market model supports persistent orders and cancellations, the classification of an order as rejected is non-obvious and remains to be defined. To address this, we associate orders with a grace period τ_{gp} and an alive period τ_{al}. We define the grace period as the average time interval per arrival, that is $\tau_{\mathrm{gp}} = 1/\lambda_a$, and the alive period τ_{al} of an order as the time interval from submission to transaction or withdrawal if it is inactive, or to the current time if active. An order is considered as rejected only if its alive period τ_{al} is longer than τ_{gp}, otherwise it is partially rejected by a fraction of $\tau_{\mathrm{al}}/\tau_{\mathrm{gp}}$. As the belief function Eq. (2.4) is defined only at encountered prices, we further extend it over the full price domain by cubic spline interpolation. To speed the computation, we pick knot points and interpolate only between those points.

After formulating the belief function, an agent i with the arrival time t and current holdings q searches for the optimal price $P_i^*(t)$ that maximizes its expected surplus:

$$\mathrm{P}_i^*(t) = \begin{cases} \arg\max_P (\hat{r}_t + \theta_i^{q+1} - P) f_t(P \mid \mathcal{D}) & \text{if buying,} \\ \arg\max_P (P - \theta_i^q - \hat{r}_t) f_t(P \mid \mathcal{D}) & \text{if selling.} \end{cases} \tag{2.5}$$

Under the special cases when there are fewer than L transactions at the beginning of a trading period or when one side of the order book is empty, HBL agents behave the same as ZI agents until enough information is gathered to form the belief function. As those cases are rare, the specific ZI strategy that HBL agents adopt does not materially affect the overall performance. In Section 2.7, we explore variations of the HBL strategy to improve its learning robustness in the face of market manipulation.

2.3.4 The Spoofing Agent

The spoofing agent seeks profits only through manipulating prices. Unlike background traders, the spoofer has no private value for the security. We design a simple spoofing strategy which maintains a large volume of buy orders at one tick behind the best bid. Specifically, upon arrival at $T_{\mathrm{sp}} \in [0, T]$, the spoofing agent submits a buy order at price $\mathrm{BID}_{\mathrm{T_{sp}}} - 1$ with volume $Q_{\mathrm{sp}} \gg 1$. Whenever there is an update on the best bid, the spoofer cancels its original spoof order and submits a new one at price $\mathrm{BID}_t - 1$ with the same volume. Since in our model, background traders submit only single-unit orders, they cannot transact with the spoof order, which is always shielded by the order at a higher price $\mathrm{BID}_{\mathrm{T_{sp}}}$. If that higher-price order gets executed, the spoofer will immediately cancel and replace its spoof orders before another background trader arrives. Here, we assume in effect that the spoofing agent can react infinitely fast, in which case its spoof orders are guaranteed never to transact.

By continuously feigning buy interest in the market, this spoofing strategy specifically aims to raise market beliefs. To profit from such manipulation practice, a spoofing agent may first buy some shares of the security, manipulate the market to push prices up, and later sell those previously bought shares at higher prices. Other spoofing strategies such as adding sell pressure or alternating between buy and sell pressure can be extended from the basic version.

2.4 Empirical Game-Theoretic Analysis

To reproduce spoofing and understand its effect, we employ a computational approach that combines agent-based modeling, simulation, and equilibrium computation. The point of identifying equilibria of the agent-based model is to focus on the most relevant strategic contexts, where agents are making the best choices among their available strategies, given others' choices. To derive Nash equilibria, we employ empirical game-theoretic analysis (EGTA), a methodology that finds approximate equilibria in games defined by heuristic strategy space and simulated payoff data (Wellman 2016). We conduct systematic EGTA studies over a range of parametrically defined market environments, based on the market model described in Section 2.3.

We model the market as a game with players in two *roles*: N background traders, treated *symmetrically*, and a single spoofer. In most of our games, the spoofing agent, when present, implements a fixed policy so is not considered a strategic player. Symmetry of the background traders means that each has the same set of available strategies (from the ZI and/or HBL families) to choose from, and their payoffs depend on their own strategy and the number of players choosing each of the other strategies (i.e., it does not matter which other-agent plays which other-strategy). For each game, we evaluate a wide variety of *strategy profiles* (i.e., agent-strategy assignments), and for each profile, we conduct thousands of simulation runs to account for stochastic effects such as the market fundamental series, agent arrival patterns, and private valuations. Given background trader symmetry, the payoff of a specific strategy in a profile can be taken as the average payoff over all agents playing that strategy in the profile. From the payoff data accumulated from these simulated samples of explored strategy profiles, we induce an *empirical game model*, and from that derive an approximate Nash equilibrium.

EGTA employs an iterative process: find candidate equilibria in *subgames* (i.e., games over strategy subsets), confirm or refute candidate solutions by examining deviations, and incrementally extend subgames, until termination criteria are satisfied. We use the EGTAOnline infrastructure (Cassell and Wellman 2013) to conduct and manage experiments. Below, we describe two key components of the EGTA process we follow: profile search (Section 2.4.1) and game reduction (Section 2.4.2).

2.4.1 Profile Search

We apply EGTA iteratively to guide the profile search over the strategy space. Exploration starts with singleton subgames, and incrementally considers each strategy outside the subgame strategy set. Specifically, the singleton subgames are profiles where the same strategy is adopted by all background agents. Starting from this base, we extend evaluation to neighboring profiles with single-agent deviations. Following such a procedure, we systematically explore profiles and incorporate their payoff estimates into the partial payoff matrix corresponding to the empirical game model.

A subgames are completed (all profiles explored for strategy subsets), we compute their equilibria, and consider these as candidate solutions of the full game. We attempt to *refute* these candidates by evaluating deviations outside the subgame strategy set, constructing a new subgame when a beneficial deviation is found. If we examine all deviations without refuting, the candidate is *confirmed*. We continue to refine the empirical subgame with additional strategies and corresponding simulations until at least one equilibrium is confirmed and all non-confirmed candidates are refuted (up to a threshold support size).

The procedure aims to confirm or refute promising equilibrium candidates found throughout our exploration of the strategy space. Since it is often not computationally feasible to search the entire profile space, additional distinct equilibria (e.g., equilibria of large support sizes) are possible. In addition, equilibria identified in empirical games must generally be viewed as provisional, as they are subject to refutation by strategies outside the restricted set considered in the analysis.

2.4.2 Game Reduction

As the *game size* (i.e., number of possible strategy profiles) grows exponentially in the number of players and strategies, it is computationally prohibitive to directly analyze games with more than a moderate number of players. We therefore apply aggregation methods to approximate a many-player game by a game with fewer players. The specific technique we employ, called *deviation-preserving reduction* (DPR) (Wiedenbeck and Wellman 2012), defines reduced-game payoffs in terms of payoffs in the full game as follows. Consider an N-player symmetric game, which we want to reduce to a k-player game. The payoff for playing strategy s_1 in the reduced game, with other agents playing strategies $(s_2, \ldots, s_k)$, is given by the payoff of playing s_1 in the full N-player game when the other $N-1$ agents are evenly divided among the $k-1$ strategies $s_2, \ldots, s_k$. To facilitate DPR, we choose values for N and k to ensure that the required aggregations come out as integers. For example, in one of the market environment, we reduce games with 28 background traders to games with four background traders. With one background player deviating to a new strategy, we can reduce the remaining 27 players to three. For games that vary smoothly with the number of other players choosing any particular strategy, we can expect DPR to produce reasonable approximations of the original many-player games with exponential reduction in simulation.

2.5 Spoofing the Limit Order Book

This section reproduces spoofing in the agent-based market model, and studies its effect on background trading behavior and market outcomes. We start in Section 2.5.1 by exploring a range of market environments that can affect the effectiveness of both learning and spoofing. Section 2.5.2 addresses agents' choices among ZI and HBL strategies in markets without spoofing. This is an important step, as spoofing can be effective only if some fraction of background traders choose to learn from the order book information. Section 2.5.3 investigates games with spoofing. We first illustrate that a market populated with HBL traders is susceptible to spoofing: a simple spoofing strategy can cause a rise in market prices and a redistribution of surplus between ZI and HBL traders. We finally re-equilibrate the game with spoofing to investigate the impact of spoofing on HBL adoption and market surplus. Details of the HBL adoption rates and market surpluses of all found equilibria in games with and without spoofing are provided in Appendix A.1.

2.5.1 Market Environments

Based on the defined market model, we conduct preliminary explorations over a range of market settings, and include the most salient and meaningful ones for our study. We consider nine market environments that differ in fundamental shock, $\sigma_s^2 \in \{10^5, 5 \times 10^5, 10^6\}$, and in observation noise, $\sigma_n^2 \in \{10^3, 10^6, 10^9\}$. Recall that shock variance controls fluctuations in the fundamental time series, and observation variance governs the quality of information agents get about the true fundamental. The nine environments cover representative market conditions that can affect an agent's ability and need to learn from market information. For example, when the market shock is high, prices fluctuate more and market history may become less predictive; when observation noise is high, agents can glean only limited information from their own observations and may gain more from the market's aggregated order book information. We label the low, medium, and high shock variances as $\{LS, MS, HS\}$ and noisy observation variances as $\{LN, MN, HN\}$ respectively. For instance, the label $LSLN$ refers to a market with low shock, $\sigma_s^2 = 10^5$, and low observation noise, $\sigma_n^2 = 10^3$.

Strategy	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	ZI_6	ZI_7	HBL_1	HBL_2	HBL_3	HBL_4
L	–	–	–	–	–	–	–	2	3	5	8
$R_{\min}$	0	0	0	0	0	250	250	–	–	–	–
$R_{\max}$	250	500	1000	1000	2000	500	500	–	–	–	–
η	1	1	0.8	1	0.8	0.8	1	–	–	–	–

Table 2.1: Background trading strategies included in EGTA.

The global fundamental time series is generated according to Eq. (2.1) with fundamental mean $\bar{r} = 10^5$, mean reversion $\kappa = 0.05$, and a specified shock variance σ_s^2. The minimum tick size is fixed at one. Each trading period lasts $T = 10,000$ time steps. For each environment, we consider markets populated with $N \in \{28, 65\}$ background traders and in selected treatments, a spoofer. Background traders arrive at the market according to a Poisson distribution with a rate $\lambda_a = 0.005$ and upon each arrival, the trader observes a noisy fundamental $o_t = r_t + n_t$, where $n_t \sim N(0, \sigma_n^2)$. The maximum number of units background traders can hold at any time is $q_{\max} = 10$. Private values are drawn from a Gaussian distribution with zero mean and a variance of $\sigma_{\text{PV}}^2 = 5 \times 10^6$. The spoofing agent starts to manipulate at time $\text{T}_{\text{sp}} = 1000$ by submitting a large buy order at price $\text{BID}_{\text{T}_{\text{sp}}} - 1$ with volume $Q_{\text{sp}} = 200$, and later maintains spoofing orders at price $\text{BID}_t - 1$ throughout the trading period.

To provide a benchmark for market surplus, we calculate the social optimum—the expected total possible gains from trade, which depends solely on the trader population size and valuation distribution. From 20,000 samples of the joint valuations, we estimate mean social optima of 18,389 and 43,526 for markets with 28 and 65 background traders respectively. We further calculate the average order book depth (on either buy or sell side) in markets without spoofing. Throughout the trading horizon, the $N = 28$ market has a relatively thin order book with an average depth of 12 per side, whereas the $N = 65$ market has a thicker one with an average depth of 30.

The background trading strategy set (see Table 2.1) includes seven versions of ZI and four versions of HBL. Agents are allowed to choose from this restricted set of strategies. We have also explored ZI strategies with larger shading ranges and HBL strategies with longer memory lengths, but they fail to appear in equilibrium in games where they were explored.

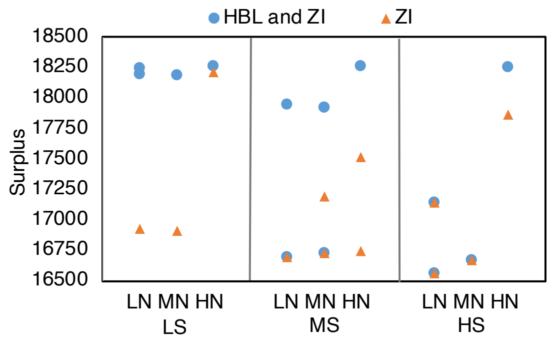
2.5.2 Games without Spoofing

Since spoofing targets the order book and can be effective only to the extent traders exploit order book information, we investigate whether background agents adopt the HBL strategy in markets without spoofing. Applying EGTA to the eleven background strategies in Table 2.1, we found at least one equilibrium for each market environment.

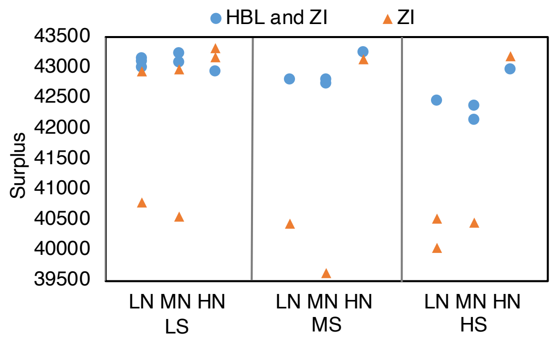
Figure 2.5 (blue circles) depicts the proportion of background traders who choose trading strategies in the HBL family. In most non-spoofing environments, HBL is adopted with positive probability, suggesting that investors generally have incentives to make bidding decisions based on order book information. We find that HBL is robust and widely preferred in markets with more traders, low fundamental shocks, and high observation noise. Intuitively, a larger population size implies a thick order book with more learnable aggregated data; low shocks in fundamental time series increase the predictability of future price outcomes; and high observation noise limits what an agent can glean about the true fundamental from its own information. This is further confirmed in the two exceptions where all agents choose ZI: *HSLN* and *HSMN* with $N = 28$, the environments with fewer traders, high fundamental shocks, and at most medium observation noise.

We further quantify how learning from market information may benefit overall market performance. We conduct EGTA in games where background traders are restricted to strategies in the ZI family (ZI_1–ZI_7 in Table 2.1). This is tantamount to disallowing learning from order book information. We compare equilibrium outcomes for each environment, with and without HBL available to background traders, on two measures: surplus (Figure 2.2) and price discovery (Figure 2.3). Recall that we define background-trader surplus as the sum of all background agents' surpluses at time T, the end of trading. Price discovery is defined as the root-mean-squared deviation (RMSD) of the transaction price from the estimate of the true fundamental in Eq. (2.2) over the trading period. It reflects how well transactions reveal the true value of the security. Lower RMSD means better price discovery. We calculate the two measures by averaging the outcomes of 20,000 simulations of games with strategy profiles sampled according to each equilibrium mixture.

Overall, background traders achieve higher surplus (Figure 2.2) and better price discovery (Figure 2.3) when the market provides order book information and enables the HBL strategy option. When HBL exists in the equilibrium, we find transactions reveal fundamental estimates well, especially in markets with lower shock and observation variances (i.e., *LSLN*, *LSMN*, *MSLN*, *MSMN*). We also notice small exceptions in scenarios with high observation variance and more background traders (environments *LSHN* and *HSHN* with 65 players) where ZI-only equilibria exhibit higher surplus than equilibria combining HBL and ZI.

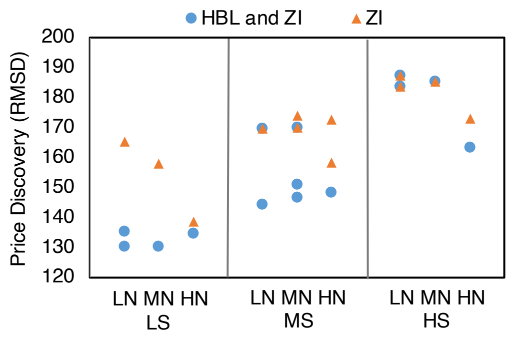


(a) $N = 28$

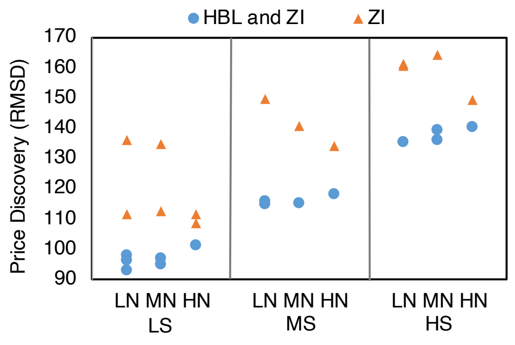


(b) $N = 65$

Figure 2.2: Comparisons of background-trader surplus for equilibria in each environment, with and without the HBL strategies available to background traders. Blue circles represent equilibrium outcomes when agents can choose both HBL and ZI strategies; orange triangles represent equilibrium outcomes when agents are restricted to ZI strategies. Overlapped markers are outcomes from the same equilibrium mixture, despite the availability of HBL. The market generally achieves higher surplus when HBL is available.



(a) $N = 28$



(b) $N = 65$

Figure 2.3: Comparisons of price discovery for equilibrium in each environment, with and without the HBL strategies available to background traders. Blue circles represent equilibrium outcomes when agents can choose both HBL and ZI strategies; orange triangles represent equilibrium outcomes when agents are restricted to ZI strategies. Overlapped markers are outcomes where the equilibrium mixture is ZI only, despite the availability of HBL. The market generally achieves better price discovery when HBL is available.

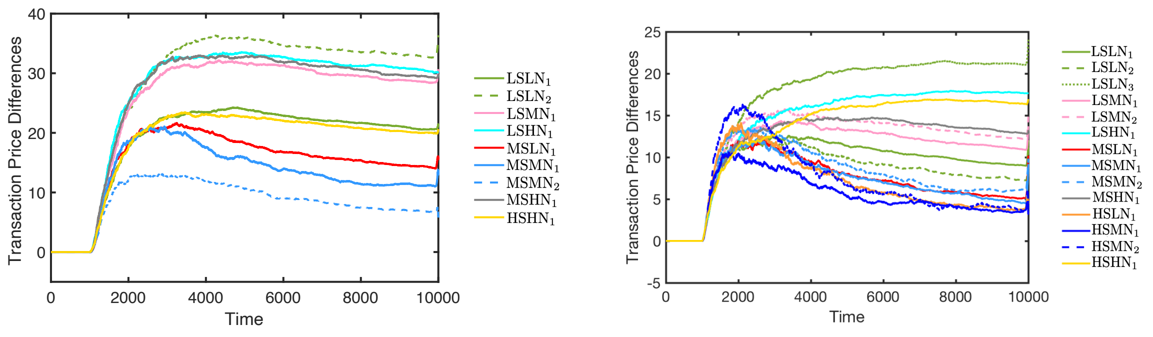
2.5.3 Games with Spoofing

2.5.3.1 Comparing across Fixed Strategy Profiles

We examine the effectiveness of our designed spoofing strategy (Section 2.3.4) by playing a spoofer against each HBL-and-ZI equilibrium found in Section 2.5.2. As ZI agents are oblivious to spoofing, we ignore the ZI-only equilibria in this analysis. We perform controlled comparisons on these games with and without spoofing. In the paired instances, background agents play identical strategies, and are guaranteed to arrive at the same time, receive identical private values, and observe the same fundamental values. Therefore, any change in behavior is an effect of spoof orders on HBL traders. For every setting, we simulate 20,000 paired instances, evaluate transaction price differences (Figure 2.4), and compare surplus attained by HBL and ZI traders. Transaction price difference at a specific time is defined as the most recent transaction price in the run with spoofing minus that of the paired instance without spoofing. Similarly, surplus difference of HBL or ZI is the aggregated surplus obtained in an environment with spoofing minus that of the corresponding environment without spoofing.

Figure 2.4 shows positive changes in transaction prices across all environments, subsequent to the arrival of a spoofing agent at $\mathrm{T_{sp}} = 1000$. This suggests that HBL traders are tricked by the spoof buy orders: they believe the underlying security should be worth more, and therefore submit or accept limit orders at higher prices. Though ZI agents do not change their bidding behavior directly, they may transact at higher prices due to the increased bids of HBL traders.

Several other interesting findings are revealed by the transaction-price difference series. First, the average price rise caused by spoofing the market with 28 background traders is higher than for $N = 65$. This indicates that a market with fewer background traders can be more susceptible to spoofing, due to the limited pricing information a thin market could aggregate. Second, for markets populated with more HBLs than ZIs in the equilibrium mixture, the transaction price differences tend to increase throughout the trading period. This amplification can be explained by HBLs consistently submitting orders at higher prices and confirming each other's spoofed belief. However, for markets with more ZIs, the spoofing effect diminishes as ZIs who do not change their limit-order pricing can partly correct the HBLs' illusions. Third, we notice that differences in transaction prices first increase, and then tend to stabilize or decrease over time. As time approaches the end of the trading period, spoofing wears off in the face of accumulated observations and mean reversion.



(a) $N = 28$ (b) $N = 65$

Figure 2.4: Transaction price differences throughout the trading horizon with and without a spoofer against each HBL-and-ZI equilibrium found in non-spoofing games (Section 2.5.2). Multiple curves for the same environment represent different equilibria. The designed spoofing tactic clearly raises market prices when HBL are present. The effect attenuates over time, generally more quickly in the thicker market environments.

We further compare background-trader payoffs attained in environments with and without spoofing. We find a redistribution of surplus between HBL and ZI agents: HBL aggregated surplus decreases, while that for ZI increases compared to the non-spoofing baselines. Specifically, across 28-trader market environments, HBL traders suffer an average surplus decrease of 184 across all equilibrium profiles, whereas the ZI traders have an average surplus gain of 19. For the 65-trader markets, the average surplus decrease for HBL traders is 238, and the average increase for ZI is 40. This suggests that the ZI agents benefit from the HBL agents' spoofed beliefs. Since the decreases in HBL surplus are consistently larger than the increases for ZI, the overall market surplus decreases. We leave further discussion of spoofing's impact on market surplus to Section 2.5.3.2, where background traders can choose other strategies to adjust to the presence of spoofing.

To examine the potential to profit from a successful price manipulation, we extend the spoofing agent with an *exploitation strategy*: buying, (optionally) spoofing to raise the price, and then selling. The exploiting spoofer starts by buying when there is a limit sell order with price less than the fundamental mean in the market. It then optionally runs the spoofing trick, or alternatively waits, for 1000 time steps. Finally, the agent sells the previously bought unit (if any) when it finds a limit buy order with price more than fundamental mean. Note that even without spoofing, this single-unit exploitation strategy is profitable in expectation due to the mean reversion captured by the fundamental process, and the reliable arrivals of background traders with private preferences.

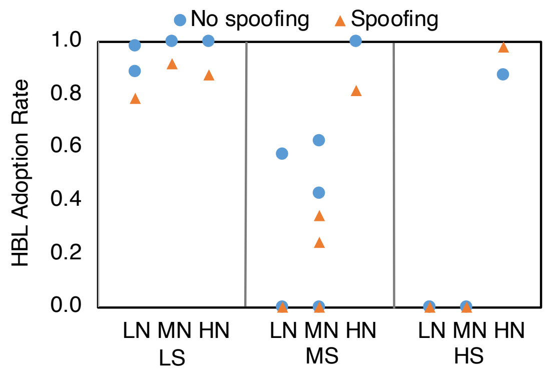
In controlled experiments, we find that exploitation profits are consistently increased when the spoof action is also deployed. Across 28-trader market environments, the exploiter makes an average profit of 206.1 and 201.8 with and without spoofing, and the increases in profit range from 1.2 to 11.5. For the 65-trader market, the average profits of this exploitation strategy with and without spoofing are 50.5 and 46.3 respectively, with the increases in profit varying from 1.7 to 9.4 across environments.[3]

2.5.3.2 Re-Equilibrating Games with Spoofing

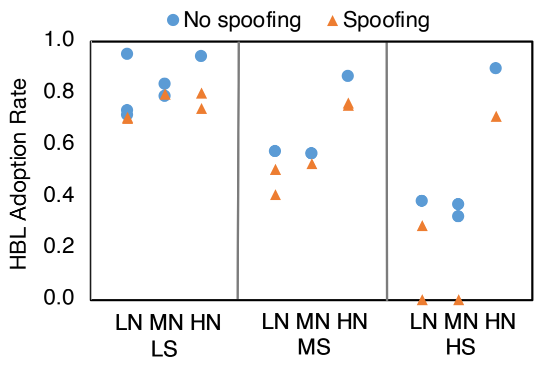
To understand how spoofing changes background-trading behavior, we conduct EGTA again to identify Nash equilibria, allowing background traders to choose any strategy in Table 2.1, in games with spoofing. As indicated in Figure 2.5 (orange triangles), after re-equilibrating games with spoofing, HBL is generally adopted by a smaller fraction of traders, but still persists in equilibrium in most market environments. HBL's existence after re-equilibration indicates a consistently spoofable market: the designed spoofing tactic fails to eliminate HBL agents and in turn, the persistence of HBL may incentivize a spoofer to continue effectively manipulating the market.

We characterize the effect of spoofing on market surplus. Figure 2.6 compares the total surplus achieved by background traders in equilibrium with and without spoofing. Given the presence of HBL traders, spoofing generally decreases total surplus (as in Figure 2.6, most filled orange triangles are below the filled blue circles). However, spoofing has ambiguous effect in the thicker market with large observation variance (environments *LSHN* and *HSHN* with 65 background agents). This may be because noise and spoofing simultaneously hurt the prediction accuracy of the HBL agents and therefore shift agents to other competitive ZI strategies with higher payoffs. Finally, we find the welfare effects of HBL strategies persist regardless of spoofing's presence: markets populated with HBL agents in equilibrium achieve higher total surplus than those markets without HBL (as in Figure 2.6, the hollow markers are below the filled markers).

[3]Statistical tests show all increases in profit are significantly larger than zero. Regardless of spoofing, the exploitation strategy profits more in the thinner market due to the greater variance in transaction prices.

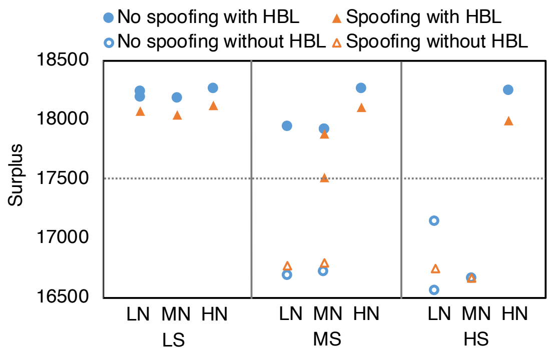


(a) $N = 28$

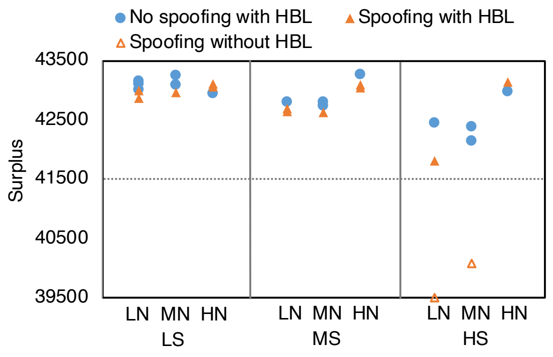


(b) $N = 65$

Figure 2.5: HBL adoption rates at equilibria in games with and without spoofing. Each blue (orange) marker specifies the HBL proportion at one equilibrium found in a specific game environment without (with) spoofing.



(a) $N = 28$



(b) $N = 65$

Figure 2.6: Background-trader surplus achieved at equilibria in games with and without spoofing. Each blue (orange) marker specifies the surplus at one equilibrium found in a specific game environment without (with) spoofing. Surplus achieved at equilibria combining HBL and ZI and equilibria with pure ZI are indicated by markers with and without fills respectively.

2.5.4 Discussion

Our agent-based model of spoofing aims to capture the essential logic of manipulation through influencing belief about market demand. In our model, the order book reflects aggregate information about the market fundamental, and learning traders can use this to advantage in their bidding strategies. The presence of such learning traders benefits price discovery and social welfare, but also renders the market vulnerable to manipulation. As we demonstrate, simple spoofing strategies can effectively mislead learning traders, thereby distorting prices and reducing surplus compared to the non-spoofing baseline. Moreover, the persistence of learning traders in equilibrium with manipulation suggests that the elimination of spoofing requires active measures.

We acknowledge several factors that can limit the accuracy of our equilibrium analysis in individual game instances; these include sampling error, reduced-game approximation, and restricted strategy coverage. Despite such limitations (inherent in any complex modeling effort), we believe the model offers a constructive basis to evaluate manipulation practices and any preventive or deterrent proposals to mitigate manipulation under strategic settings. In the rest of the chapter, we build on this model and conduct comprehensive analysis to investigate the following questions:

- Are there more robust ways for exchanges to disclose order book information (Section 2.6)?
- Are there strategies by which individual traders can adopt to exploit market information but in less vulnerable ways (Section 2.7)?

2.6 A Cloaking Mechanism to Mitigate Spoofing

Despite regulatory enforcement and detection efforts, an individual spoofing episode is hard to catch in high-volume, high-velocity data streams. Legal definitions cannot be easily translated to computer programs to direct detection, and the lack of datasets with labeled manipulation cases makes training a reliable detector infeasible with supervised machine learning techniques. Based on its definition, to determine that a pattern of activity constitutes spoofing requires establishing the manipulation intent behind submission and cancellation of placed orders. However, this is not easy, as order cancellation is in itself common and legitimate: according to one study, 95% of NASDAQ limit orders are canceled, with a median order lifetime less than one second (Hautsch and Huang 2012). Given difficulties in robustly detecting manipulation, we study systematic approaches to deter spoofing, by rendering manipulative practices difficult or uneconomical.

Along these lines, Prewit (2012) and Biais and Woolley (2012) advocated the imposition of cancellation fees to disincentivize manipulative strategies that rely on frequent cancellations of orders. Others argue that cancellation fees could discourage the beneficial activity of liquidity providers, and in the event of a market crash, such a policy may lengthen the recovery process (Leal and Napoletano 2019).

We propose here *a cloaking mechanism* to deter spoofing via the selective disclosure of order book information. The mechanism extends the traditional CDA market with a cloaking parameter K, which specifies the number of price levels to hide symmetrically from inside of the limit order book. The idea is to make it more difficult for the spoofer who relies on the instant order book information to post misleading bids, while not unduly degrading the general usefulness of market information. We focus on deterministic cloaking (i.e., a constant K throughout the trading period), as a stochastic mechanism may raise issues regarding verification of faithful market operations.

We extend our agent-based model of spoofing to support order book cloaking, and conduct simulations to evaluate and find the optimal cloaking parameter under strategic settings, where both the learning traders and the spoofer adapts to the new mechanism. Section 2.6.1 formally defines the cloaking mechanism, and describes how we modify the background trading and spoofing strategies accordingly. In Section 2.6.2, we present an EGTA study conducted to understand agents' strategic responses to the proposed mechanism. Section 2.6.3 reports results from performing *empirical mechanism design* (Vorobeychik, Kiekintveld, and Wellman 2006) to set cloaking parameters that maximize efficiency. Finally, in Section 2.6.4, we explore and evaluate sophisticated spoofing strategies that use probing to reveal cloaked information. Details of all found equilibria in markets with and without cloaking and games with and without spoofing are provided in Appendix A.2.

2.6.1 A Cloaking Market Mechanism

The cloaking mechanism maintains a full limit order book just as the regular CDA market, but discloses only a selective part of the book to traders. Let BID_t^k denote the kth-highest buy price in the book at time t, and ASK_t^k the kth-lowest sell price. In a standard order book, at any given time t, the buy side of the book starts with the best bid, BID_t^1, and extends to lower values; the sell side starts with the best ask, ASK_t^1, and extends to higher ones. The cloaking mechanism works by symmetrically hiding a deterministic number of price levels K from inside of the order book. Thus, the *disclosed* order book in a K-level cloaking mechanism starts with BID_t^{K+1} and ASK_t^{K+1}, and extends to lower and higher values respectively. Upon order submissions, cancellations, and transactions, the market updates the full order book and then cloaks the K inside levels. Therefore, an order hidden in the past can be revealed later due to the arrival of new orders at more competitive prices, or it can be hidden throughout its lifetime due to a cancellation. The market discloses all the transaction information at zero delay.

Example 2.1 (A K-level Cloaking Mechanism)**.** When $K = 0$, the market acts as a standard CDA, disclosing the full limit order book with zero delay. When $K = 1$, the mechanism conceals orders at the best quotes, that is BID_t^1 and ASK_t^1. When $K = \infty$, the market does not reveal any part of the book, and thus disallows learning from order book information.

Cloaking operates to deter spoofing in two ways. First, it mitigates the effect of spoof orders, pushing them further from the inside of the book. Second, it increases the spoofer's transaction risks, as it cannot as easily monitor the quantity of orders ahead of the spoof. On the other hand, the information hiding also affects the non-manipulative traders, for instance in our model it may degrade the HBL traders' learning capability. To quantify this tradeoff, we start by exploring a range of cloaking parameters, $K \in \{0, 1, 2, 4\}$, which control the amount of information being concealed at any given time. We compare trading behavior and outcomes in markets with cloaking to that of a standard CDA. Among the nine market environments defined in Section 2.5.1, we consider three representatives that are increasingly challenging for the learning traders: *LSHN* with $\{\sigma_s^2 = 10^5, \sigma_n^2 = 10^9\}$, *MSMN* with $\{\sigma_s^2 = 5 \times 10^5, \sigma_n^2 = 10^6\}$, and *HSLN* with $\{\sigma_s^2 = 10^6, \sigma_n^2 = 10^3\}$. Together with the four cloaking parameters, this gives us a total of 12 market settings, or 24 games with and without spoofing.

The market is populated with 64 background traders and one exploitation agent. Therefore, when adopting DPR to approximate this many-player game, we use simulation data from the (64, 1)-agent environments to estimate reduced (4, 1)-player games, where four players are used to aggregate and represent the background traders. In each game, we consider background trading strategies and spoofing practice similar to those of Section 2.3, but slightly modified to adapt to order book cloaking. Below, we describe changes made to each strategy.

2.6.1.1 Zero Intelligence

Recall that our ZI strategy uses a threshold parameter $\eta \in [0, 1]$ to immediately transact with an existing order to grasp a portion of desired surplus. That is, if the agent could achieve a fraction η of its requested surplus at the market best quotes, it would simply take that quote rather than posting a limit order for a future transaction. Under a cloaking mechanism, however, ZI may take into account only the current *visible* best quotes that are less competitive compared to the hidden quotes. To adjust to cloaking, we explore a range of more aggressive (smaller) η values to ensure that ZI traders may still transact with incumbent orders to lock a certain fraction of surplus. Besides the seven ZI strategies in Table 2.1, we further include three ZI strategies with $\eta = 0.4$ (Table 2.2), which are competitive enough to appear in at least one equilibrium of our explored environments.

2.6.1.2 Heuristic Belief Learning

We modify HBL to consider only the *revealed* order book information under the corresponding cloaking markets. Orders at competitive price levels will be missed in the belief function (Eq. 2.4) if they are hidden throughout order lifetime; or they may be considered with delay if later exposed at visible levels. This reduction in bid information would naturally be expected to degrade HBL's learning effectiveness and thus its trading performance.

2.6.1.3 Spoofing Strategy

We extend the original spoofing strategy (Section 2.3.4) to cloaking markets. The strategy includes three stages. At the beginning of a trading period $[0, T_{\text{spoof}}]$, the agent buys by accepting any sell order at price lower than the fundamental mean $\bar{r}$. In a cloaking market, this can be achieved by placing a one-unit limit buy order at price $\bar{r}$ and immediately withdrawing it if does not transact with an existing order.

During the second stage $[T_{\text{spoof}}, T_{\text{sell}}]$, the agent submits spoof buy orders at a tick behind the first *visible* bid $\text{BID}^{K+1}_{\text{T}_{\text{spoof}}} - 1$ with volume $Q_{\text{sp}} \gg 1$. Whenever there is an update on the first visible bid, the spoofer replaces its original spoof with new orders at price $\text{BID}^{K+1}_{t} - 1$. This spoofing strategy aims to boost price, in the hope that the units purchased in stage one can be later sold at higher prices. In controlled experiments, when the agent is not manipulating, it waits until the selling stage.

During the last stage $[T_{\text{sell}}, T]$, the agent starts to sell the units it previously bought by accepting any buy orders at a price higher than $\bar{r}$. Inverse to the first stage, this operates by placing one-unit limit sell orders at price $\bar{r}$, followed by immediate cancellation if not filled. The agent who also manipulates continues to spoof until all the bought units are sold or the trading period ends. The pure exploitation strategy can be considered as a baseline for the spoofing strategy, allowing us to quantify how much more the agent may profit from spoofing the market.

We refer to the agent who employs the above strategy, whether places spoof orders or not, as an *exploitation agent* or *exploiter*. An exploiter who also spoofs is referred

Strategy	ZI_8	ZI_9	ZI_{10}
$R_{\min}$	0	0	250
$R_{\max}$	1000	2000	500
η	0.4	0.4	0.4

Table 2.2: Additional background trading strategies included in EGTA for cloaking mechanisms.

to as a *spoofing agent* or *spoofer*. Note that the spoofing strategy considered here does not face any execution risk on its spoof orders, under the assumption it can immediately respond to quote changes. A more sophisticated strategy could *probe* the market to reveal the cloaked bids, and then spoof at a visible price higher than $\text{BID}_t^{K+1} - 1$. We leave discussion of such probing strategies to Section 2.6.4.

2.6.2 Tradeoff Faced by Cloaking Mechanisms

We start by separately investigating the impact of cloaking on background traders and on the spoofer. Our first set of games cover the range of cloaking environments without spoofing (i.e., markets populated with background traders and the non-manipulative exploiter).

Figure 2.7 displays the HBL adoption rate (i.e., total probability over HBL strategies) at equilibrium across cloaking mechanisms, $K \in \{0, 1, 2, 4\}$. We find that the competitiveness of HBL generally persists when the mechanism hides one or two price levels, but at higher cloaking levels the HBL fraction can drastically decrease. The information loss caused by cloaking weakens HBL's ability to make predictions. The effect is strongest in environments with high fundamental shocks (e.g., *HSLN*), as previous hidden orders can become uninformative or even misleading by the time they are revealed. Given the decreasing HBL prevalence and effectiveness, background-trader surplus achieved at equilibrium also decreases, as we see in Figure 2.9(b) (blue diamonds).

Next, we examine whether cloaking can effectively mitigate manipulation. We perform controlled experiments by letting the exploitation agent also execute the spoofing strategy against each found equilibrium, and compare the impact of spoofing under the cloaking mechanism to the standard fully revealed order book ($K = 0$). For every equilibrium, we simulate at least 10,000 paired instances, and evaluate their differences on transaction price and agents' payoffs.

From these controlled experiments, we find that cloaking can considerably dimin-

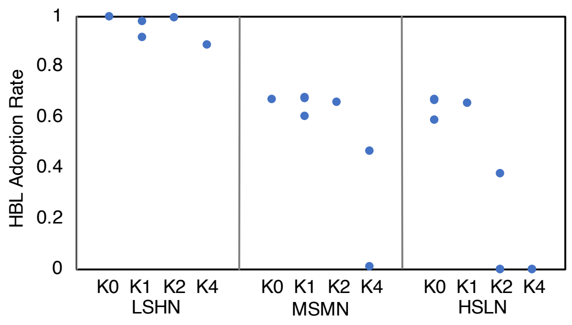
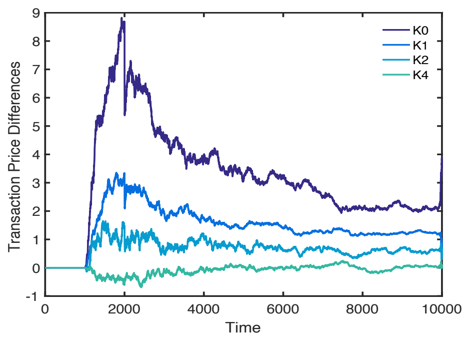
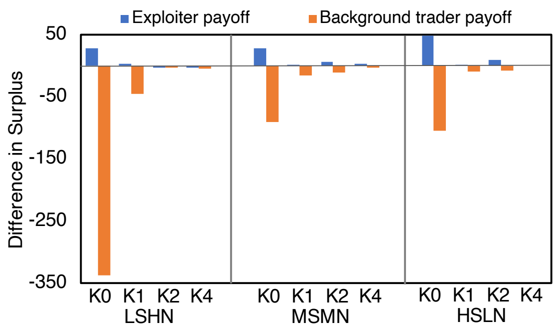


Figure 2.7: HBL adoption rate in equilibrium across different cloaking markets without spoofing.



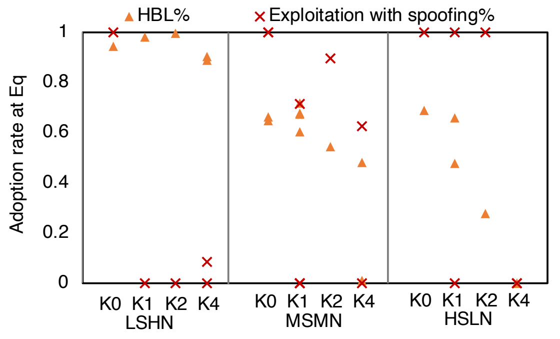
(a) Cloaking mitigates price rise.



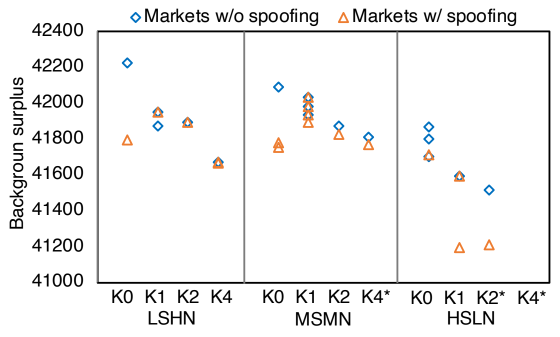
(b) Cloaking reduces spoofing profits.

Figure 2.8: The impact of cloaking on spoofing effectiveness. Cloaking mitigates price rise and the decrease in background surplus caused by spoofing.

ish price distortion caused by spoofing across environments. Recall that we measure price distortion as the transaction price series in a market with spoofing minus that of its paired market without spoofing. Figure 2.8(a) demonstrates the case in a specific environment *MSMN*: without cloaking ($K = 0$), transaction prices significantly rise subsequent to the execution of spoofing at $T_{\text{sp}} = 1000$, as HBL traders are tricked by the spoof buy orders; in cloaked markets, this price rise is effectively mitigated. Figure 2.8(b) further illustrates the surplus change in background traders and the exploiter when it also spoofs. We find the exploiter can robustly profit from manipulating the learning agents in the no-cloaking case. In contrast, partially hiding the order book can significantly reduce spoofing profits, and prevent background traders from losing much. These findings indicate the cloaking mechanism may deter or even eliminate the exploiter's incentive to spoof.



(a) HBL and spoofing adoption rates in equilibrium.



(b) Background-trader surplus in equilibrium.

Figure 2.9: Equilibrium outcomes in games with and without cloaking. Each marker represents one equilibrium of the environment.

2.6.3 Finding the Optimal Cloaking

Given the tradeoff between preserving order book informativeness and mitigating manipulation, the question becomes: under what circumstances do the deterrence benefits of cloaking exceed its efficiency costs? To answer this, we *re-equilibrate* games allowing the exploiter to strategically choose whether to spoof, with background traders able to execute any strategy in Tables 2.1 or 2.2. This allows background traders and the exploitation agent to strategically respond to each other under a certain level of information cloaking.

Our findings are presented in Figure 2.9.[4] We compare market outcomes with and without cloaking on two metrics: the probability of spoofing and total background-trader surplus in equilibrium. As shown in Figure 2.9(a), the cloaking mechanism effectively decreases the probability of spoofing under most environment settings—completely eliminating spoofing in some cases. Moreover, we find moderate cloaking can preserve the prevalence of HBL at equilibrium, which otherwise would be decreased by spoofing as we saw in Section 2.5.

This weakened spoofing effect is further confirmed by Figure 2.9(b), which compares the total background-trader surplus achieved in equilibrium under mechanisms with and without cloaking. Without cloaking (i.e., $K0$ columns), background surplus achieved in equilibrium where the exploiter strategically chooses to spoof (orange triangles) is much lower than the surplus attained when the exploiter is prohibited from spoofing (blue diamonds). We find the decrease in surplus due to spoofing can

[4]Due to the welfare benefits of HBL, equilibria with pure ZIs usually achieve much lower surplus than those with HBLs. For presentation simplicity, we omit all-ZI equilibria from Figure 2.9(b). Environments with such cases are marked with asterisks.

be considerably mitigated by order book cloaking. As shown in Figure 2.9(b), the vertical distances between the blue diamonds and orange triangles get smaller with $K > 0$. Moreover, we find the benefit of this improved robustness to spoofing can outweigh its associated efficiency costs in markets with moderate fundamental shocks (e.g., *LSHN* and *MSMN*). In those environments, background traders in mechanisms that cloak one or two price levels achieve higher surplus than those without cloaking. However, in a market with high shocks (e.g., *HSLN*), hiding or delaying even a little market information degrades learning to such a degree as to render cloaking counter-productive.

2.6.4 Probing the Cloaking Mechanism to Spoof

To this point, we have considered only spoofers who are unwilling to risk execution of their spoof orders. A more sophisticated manipulator could *probe* the market, submitting a series of orders at slightly higher prices, in an attempt to reveal the cloaked bids and spoof at a visible price higher than $\mathrm{BID}_{\mathrm{t}}^{K+1} - 1$. In this section, we study the feasibility of such probing to the spoofing agent.

We design and evaluate parameterized versions of the spoofing strategy combined with probing. The strategy is governed by two parameters: the step size δ, which controls probing aggressiveness, and the maximum attempts allowed per time step l, which limits the probing effort.

The spoofer probes by submitting a unit buy order at $\mathrm{BID}_{\mathrm{t}}^{K+1}+\delta$, a price inside the visible quotes, in the hopes of exposing $\mathrm{BID}_{\mathrm{t}}^{K}$. If the probe succeeds, it immediately cancels the probe order, and places a new spoof order at $\mathrm{BID}_{\mathrm{t}}^{K} - 1$, right behind the lowest hidden bid level. If probing fails because the price is too conservative, the spoofer re-probes by raising the price at a decreasing rate (as a function of δ and the attempt number), until a higher price is revealed or the number of probing attempts reaches l. If probing causes a transaction, the spoofer halves the price increment and re-probes. Algorithm 1 describes the detailed probing procedure.

Table 2.3 reports, for cloaking-beneficial environments, the minimum l required for step sizes $\delta \in \{1, 2, 4, 8\}$ to achieve higher payoffs than the equilibrium performance we found for the exploiter in Section 2.6.3. Multiple rows for the same cloaking parameter correspond to the multiple equilibria found in that market setting. Dashes in the table indicate that an exploiter cannot beat the equilibrium performance with the corresponding δ. We find in order to achieve higher payoffs, the spoofer has to probe with multiple attempts per time step, and conservative probing strategy with smaller δ usually requires more effort. In practice, such frequent cancellation

Algorithm 1 Spoofing with probing in a cloaking market with $K > 0$.

```
Input: The probing step size $\delta$ and the attempt limit $l$.
       The spoofer's time to place spoof orders $T_{\text{spoof}}$, and its current holding $H$.

1:  while $t \geq T_{\text{spoof}}$ and $H > 0$ do
2:      if no active spoof orders then
3:          $c \leftarrow 1$, $\Delta \leftarrow \delta$        $\triangleright$ track probing attempts and the price increment
4:          submit a single-unit probe buy order at price $\text{BID}_{\text{t}}^{K+1} + \Delta$
5:          while the visible $\text{BID}_{\text{t}}^{K+1}$ remains unchanged and $c < l$ do
6:              $c \leftarrow c + 1$
7:              if the probe buy order gets transacted then
8:                  $\Delta \leftarrow \Delta/2$
9:                  submit a single-unit probe buy order at price $\text{BID}_{\text{t}}^{K+1} + \Delta$
10:             else
11:                 $\Delta \leftarrow \Delta + \max\{0.9^{c-1}\delta, 1\}$
12:                 substitute the probe order with a new one at price $\text{BID}_{\text{t}}^{K+1} + \Delta$
13:         submit spoof orders at price $\text{BID}_{\text{t}}^{K+1} - 1$
14:         cancel the probe order
15:     else
16:         if spoof orders become hidden then
17:             substitute spoof orders with new ones at price $\text{BID}_{\text{t}}^{K+1} - 1$
18:         else if spoof orders are no longer one tick behind $\text{BID}_{\text{t}}^{K+1}$ then
19:             withdraw spoof orders
```

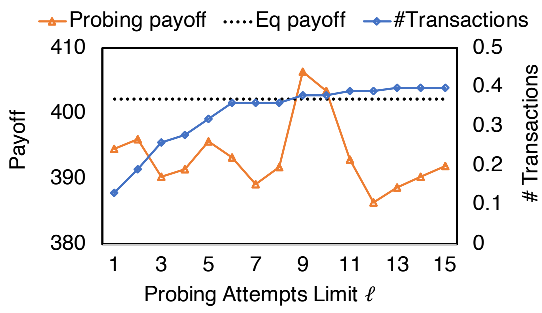
and placement of orders may not be feasible, and can largely increase the risk of associated probing and spoofing intent being identified.

Figure 2.10 further quantifies the change in exploitation payoff and transaction risk (measured as the number of transactions caused by probing), as we vary the probing step δ and the attempt limit l. As we see from Figure 2.10(a), relaxing the maximum number of probing attempts steadily increases the transaction risk, but does not necessarily improve payoff. Moreover, the spikiness of the exploiter's payoff indicates optimizing (δ, l) to maximize profit is a challenging task. Figure 2.10(b) further demonstrates that an exploiter can probe aggressively with larger step sizes to reduce effort, but usually at the cost of a higher transaction risk, and consequently a lower payoff. In highly dynamic markets with frequently updated quotes, finding an appropriate δ to successfully probe a cloaking mechanism within a reasonable number of attempts would be challenging.

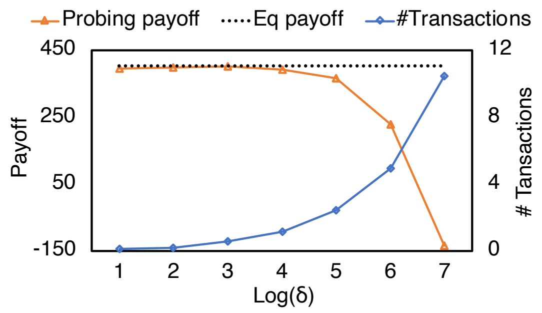
We have explored other more aggressive probing strategies, where the spoofer probes to expose multiple hidden levels and spoofs at even higher prices. To accomplish that, the spoofer is forced to keep at least one order in the cloaked levels to

Env		(δ, l)			
LSHN	K1	(1, 16)	(2, 9)	–	–
	K2	(1, 8)	(2, 5)	(4, 3)	(8, 3)
	K4	(1, 19)	(2, 3)	–	–
	K4	(1, 10)	(2, 5)	(4, 3)	–
MSMN	K1	(1, 7)	(2, 5)	(4, 4)	(8, 3)
	K1	(1, 7)	(2, 4)	(4, 2)	(8, 1)
	K1	(1, 5)	(2, 3)	(4, 2)	–
	K1	(1, 9)	(2, 4)	(4, 2)	–
	K2	(1, 11)	(2, 3)	(4, 4)	(8, 3)
	K4	(1, 5)	(2, 3)	(4, 3)	(8, 3)

Table 2.3: Least number of probing attempts required to beat equilibrium performance.



(a) Fix $\delta = 2$.



(b) Fix $l = 2$.

Figure 2.10: Exploitation payoff and transaction risk as we vary price increment δ and probing limit l.

guarantee that its spoof orders are *visible*. However, according to our experiments, such aggressive probing strategies fail to beat the equilibrium performance, as orders kept in hidden levels are often accepted by background traders due to adverse selection. Those transactions tend to accumulate the spoofer's position, and consequently impose losses at the end of the trading period.

2.7 Learning-Based Trading Strategies under the Presence of Market Manipulation

We next consider how individual traders may construct strategies that are more robust to manipulation. In realistic market scenarios, traders are aware of potential manipulation, but unable to reliably detect spoofing behavior in real time. In the absence of manipulation, traders submit orders that reflect their private observations and preferences, and so learning from others' actions enables more informed decisions. Indeed as shown above, learning as implemented by HBL agents is effective in a realistic market model, and provides benefits to the learning agent as well as to market efficiency. HBL is vulnerable to spoofing, however, and agents adopting such learning are harmed by spoofing compared to non-learning strategies that are oblivious to spoofers and thus non-manipulable. The question we investigate in this section is whether learning-based strategies can be designed to be similarly robust to spoofing. We seek strategies by which individual traders can learn from market information, but in less vulnerable ways.

We treat the original HBL described in Section 2.3.3.3 as a baseline strategy, and propose two variations that aim to reasonably trade off learning effectiveness in non-manipulated markets for robustness against manipulation. The first variation works by selectively ignoring orders at certain price levels, particularly where spoof orders are likely to be placed. The second variation considers the full order book, but has the flexibility to adjust the offer price by a stochastic offset. The adjustment serves to correct biases in learned price beliefs either caused by manipulation or the intrinsic limitation built in the belief function. We formally define the two variations in Section 2.7.1, and then evaluate the proposed strategies in terms of the effectiveness in non-manipulated markets and robustness against manipulation in Section 2.7.2.

We adopt the standard CDA market mechanism as described in Section 2.3.1. The market is populated with 64 background traders and one profitable exploiter. Background traders can choose from a select set of strategies that covers ZI, original HBL, and the two proposed variations of HBL. The exploiter follows the three-stage exploitation strategy specified in Section 2.6.1, and executes spoofing in selected treatments. As in our study of cloaking mechanisms, we consider three representative market settings for our experiments, namely *LSHN*, *MSMN*, and *HSLN*. Details of all found equilibria in this section are provided in Appendix A.3.

2.7.1 Two Variations of HBL

2.7.1.1 HBL with Selective Price Level Blocking

Our first HBL variation is inspired by the success of our cloaking mechanism. It takes advantage of the common placement of spoof orders closely behind the market best quotes. Instead of including all observed trading activities in its memory to construct the belief function just as the standard HBL, the idea is to neglect limit orders at a specified price level when assembling the dataset $\mathcal{D}$ to learn from. We extend standard HBL with a blocking parameter χ, which specifies the index of a single price level to ignore symmetrically from inside of the limit order book. For example, when $\chi = 1$, the trading agent constructs a dataset, $\mathcal{D} \setminus O_{\chi=1}$, by considering only orders strictly outside the best bid and ask. The goal of this additional strategic parameter is to exclude price levels where spoof orders are likely to appear. However, ignoring orders may come at the cost of less effective learning, especially when information that conveys true insight is blocked from the belief function.

2.7.1.2 HBL with Price Offsets

Our second HBL variation considers all orders in its memory, but translates the target price $\mathrm{P}_i^*(t)$ derived by surplus maximization in Eq. (2.5) with an offset uniformly drawn from $[R_{\min}, R_{\max}]$. Specifically, a background trader i who arrives the market at time t with the optimized price $\mathrm{P}_i^*(t)$, submits a limit order for a single unit of the security at price

$$p_i(t) \sim \begin{cases} U[\mathrm{P}_i^*(t) - R_{\max}, \mathrm{P}_i^*(t) - R_{\min}] & \text{if buying,} \\ U[\mathrm{P}_i^*(t) + R_{\min}, \mathrm{P}_i^*(t) + R_{\max}] & \text{if selling.} \end{cases} \tag{2.6}$$

A positive offset can be viewed as a hedge against misleading information, effectively shading the bid to compensate for manipulation risk. A negative offset increases the probability of near-term transaction, which may have benefits in reducing exposure to future spoofing. Offsets (positive or negative) may also serve a useful correction function even when manipulation is absent. In particular, negative offsets may compensate for the myopic nature of HBL optimization Eq. (2.5), which considers only the current bid, ignoring subsequent market arrivals and opportunities to trade additional units. Our design here is in line with prior literature (Tesauro and Bredin 2002; Tesauro and Das 2001) that refines the original HBL to become more competitive.

2.7.2 Empirical Evaluation

2.7.2.1 Standard HBL

We start with our baseline market environments where background traders are restricted to choose from the standard HBL strategies and five parametrically different ZI strategies in Table 2.4(a). Figure 2.15 (dark grey columns) verifies what

Strategy	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	HBL_1	HBL_2
L	-	-	-	-	-	2	5
$R_{\min}$	0	0	0	0	0	-	-
$R_{\max}$	1000	1000	1000	500	250	-	-
η	0.4	0.8	1	0.8	0.8	-	-

(a) A set of basic background trading strategies.

Strategy	HBL_3	HBL_4	HBL_5	HBL_6
L	2	2	5	5
χ	1	2	1	2

(b) A set of first HBL variations with price level blocking.

Strategy	HBL_7	HBL_8	HBL_9	HBL_{10}	HBL_{11}	HBL_{12}	HBL_{13}	HBL_{14}
L	2	2	2	2	5	5	5	5
$R_{\min}$	-10	-20	-40	-80	-10	-20	-40	-80
$R_{\max}$	0	0	0	0	0	0	0	0

(c) A set of second HBL variations with price offsets.

Table 2.4: Background trading strategies included to evaluate the two HBL variations.

we observed in Section 2.5 within this restrictive set of background-trading strategies: (1) the learning-based trading strategy is more widely preferred in environments where fundamental shock is low and observation noise is high (e.g., *LSHN* is the most learning-friendly environment); (2) the presence of spoofing generally hurts the learning-based strategy and reduces background-trader surplus. We next evaluate the two HBL variations.

2.7.2.2 HBL with Selective Price Level Blocking

Learning traders who choose to ignore certain orders face a natural tradeoff between losing useful information and correctly blocking spoof orders to avoid manipulation. We first examine, under *non-spoofing* environments, how learning effectiveness may be compromised by excluding orders at each price level. Starting with the equilibrium strategy profile of each non-spoofing market environment found in Section 2.7.2.1,[5] we perform controlled experiments by letting background traders who adopt the standard HBL strategy ignore orders from a selected price level throughout the trading period. Table 2.5 compares the payoffs obtained by HBL in its standard form and variations that respectively block orders at the first, second, and third price

[5]We arbitrarily select one if there are multiple equilibria found in a certain environment.

Env	HBL	$\text{HBL}_{\chi=1}$	$\text{HBL}_{\chi=2}$	$\text{HBL}_{\chi=3}$	$\text{SP}_{\psi=1}$	$\text{SP}_{\psi=2}$	$\text{SP}_{\psi=3}$	EXP
LSHN	658	650^{*}	658	658	525	$494^{*,**}$	488^{*}	483^{*}
MSMN	655	645^{*}	655	655	356	312^{*}	299^{*}	295^{*}
HSLN	649	641^{*}	649	649	295	264^{*}	$268^{*,**}$	253^{*}

Table 2.5: Average payoffs of learning-based background traders and the exploiter, as they deviate from the equilibrium strategy profiles found in Section 2.7.2.1. We deviate either background traders or the exploiter to its corresponding strategy variation. We refer to the exploiter who spoofs as SP, and the one who only executes trades as EXP. Asterisks denote statistical significance at the 1% level for the paired t-test in payoffs compared to the standard HBL(*), $\text{SP}_{K=1}$(*), and EXP(**).

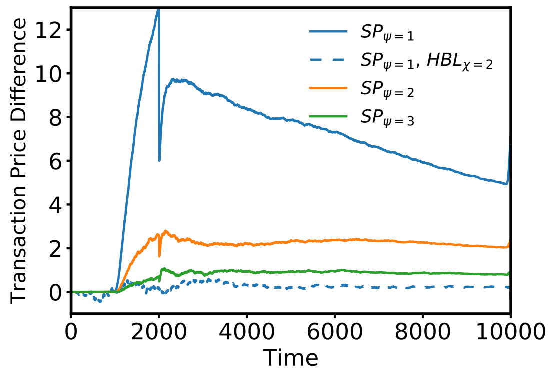


Figure 2.11: Price deviations caused by spoof orders placed behind different price levels in the order book.

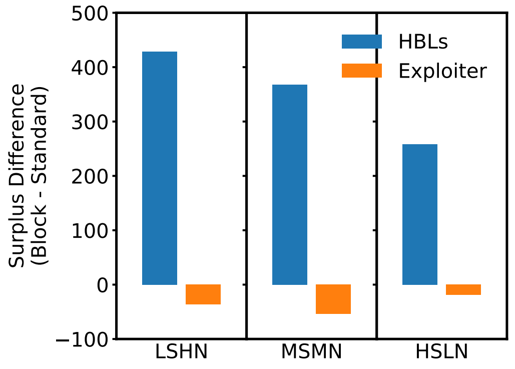


Figure 2.12: Correctly blocking spoof orders increases background-trader surplus and decreases manipulation profits.

level in the order book. We find that consistently across market settings, HBL agents benefit the most by learning from market best bids and asks, and can achieve fairly similar performance even when orders at a selected level beyond the market best quotes are ignored.

In response to the HBL variation that ignores price levels, we extend the exploiter to be able to place spoof orders behind a chosen price level, denoted by ψ. For example, when $\psi = 2$, the exploiter injects spoof orders at one tick behind the second-best bid. We start with the same set of equilibrium strategy profiles, and conduct controlled experiments to evaluate how injecting spoof orders at different levels can change the manipulation effect, even when learning traders are considering the full order book (i.e., adopting standard HBL). We measure the effectiveness of each spoofing strategy by profits from trade as well as the price deviation caused by spoof orders. Experimental results (Table 2.5) show that the exploiter benefits the most by placing spoof orders behind the best bid (i.e., $\psi = 1$), and moving spoof orders to less competitive levels reduces exploitation profit. We further confirm this weakened manipulation effect in Figure 2.11, which showcases market price deviations caused by different spoofing strategies in the *MSMN* environment. We find the price rise diminishes as spoof orders are placed further away from the best bid.

Though our exploration of possible spoofing strategies here is limited, the results suggest that spoof orders near the market quotes tend to maximize manipulation effect. In response, HBL traders who adapt to the presence of spoofing may naturally block orders around such levels. Figure 2.12 shows that when blocking the correct level, HBL traders can significantly increase their payoffs, and reduce the amount the exploiter could profit via manipulation. This mitigated manipulation effect is verified by the dashed blue line in Figure 2.11, which shows price deviations close to zero.[6]

Given these beneficial payoff deviations, in the final set of experiments, we conduct EGTA to find approximate Nash equilibria in games where background traders may choose trading strategies from the ZI family and HBLs that block a selected price level (any strategy from Table 2.4(a) or 2.4(b)). As shown in Figure 2.15 (light grey columns), we find that (1) adding the blocking strategic parameter does not affect the competitiveness of learning-based strategies with respect to ZI (HBL adoption rates in equilibrium remain in similar ranges as those of markets where only the standard HBL strategy is provided); and (2) the extended order blocking ability improves the learning robustness of HBL traders (compared to surplus decreases caused by manipulation in markets where background agents are restricted to the standard HBL, background-trader surpluses are no longer significantly reduced when agents can strategically block orders in the face of manipulation). In other words, background traders who learn from market information but also strategically ignore orders can achieve robustness against manipulation and retain comparable effectiveness in non-manipulated markets.

2.7.2.3 HBL with Price Offsets

Our second HBL variation relies on a price adjustment rather than information selection to adapt to different market conditions. We start by exploring a set of price offset intervals $[R_{\min}, R_{\max}]$, ranging from positive values that understate the learned offer prices (e.g., similar to price shading) to negative values that adjust prices to become more competitive. As in Section 2.7.2.2, we conduct controlled

[6]Price differences are not strictly zero before spoofing (time 1000), as traders who adopt $\text{HBL}_{\chi=2}$ consistently block orders throughout the trading period.

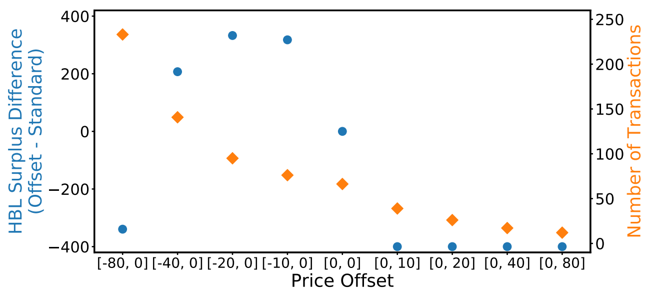


Figure 2.13: Average HBL surplus differences and total number of transactions in non-spoofing markets where HBL traders use different price offsets.

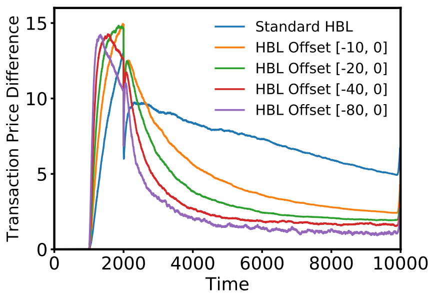


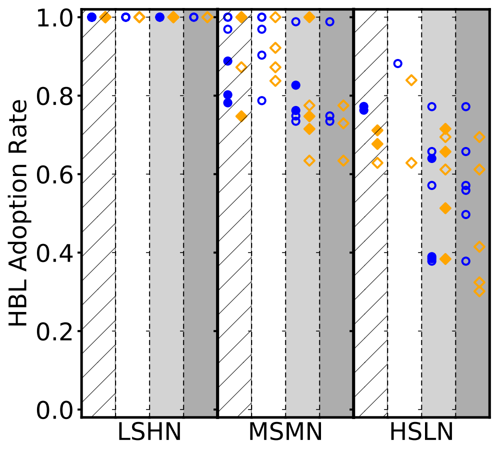
Figure 2.14: Market price deviations caused by spoofing in markets where HBL traders use different price offsets.

experiments starting from equilibrium profiles found in Section 2.7.2.1, then deviating from standard HBL to allow price offsets. Figure 2.13 shows for the *MSMN* non-spoofing environment how HBL surplus and number of transactions vary in markets where HBL traders adopt different offset intervals.[7] We find adjusting learned prices with a range of negative offsets can be generally beneficial in our setting where agents have reentry opportunities. It increases HBL payoff and facilitates transactions, thus improving overall price convergence in markets.

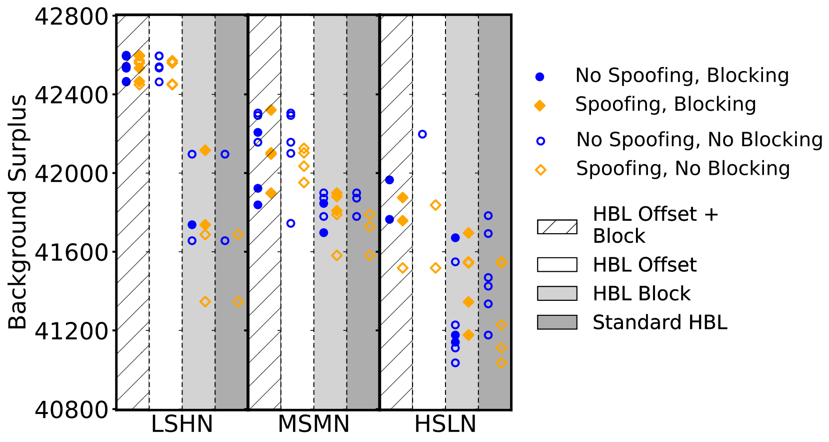
To test the effectiveness of spoofing against the new HBL variation, we further have the $SP_{\psi=1}$ spoof in markets where the learning background traders respectively adopt the standard HBL, $HBL_{[-10,0]}$, $HBL_{[-20,0]}$, $HBL_{[-40,0]}$. Figure 2.14 compares market price deviations caused by spoof orders in those markets. We find that though all markets experience initial price rise as a result of misled pricing beliefs, the spoofing effect tends to wear off faster in markets where HBL traders adopt negative price offsets. This may be because negative offsets promote near-term transaction: as more transactions happen, HBL traders can glean true information from the transaction prices to construct more accurate belief functions, and the $SP_{\psi=1}$ places spoof orders at lower prices due to the widened bid-ask spreads. Indeed, we find that markets populated with the standard HBL, $HBL_{[-10,0]}$, $HBL_{[-20,0]}$, and $HBL_{[-40,0]}$ respectively have average *spoof-order* prices of 99972, 99951, 99945, and 99950.

Finally, we conduct EGTA in games with and without spoofing to find Nash equilibria where background traders can choose from ZI strategies and HBL variations that adjust learned prices with certain offsets (Table 2.4(a) and 2.4(c)). Equilib-

[7]HBL with positive offset usually generates much lower payoff. For presentation simplicity, we cropped the surplus decrease at –400 in Figure 2.13.



(a) HBL (and its variations) adoption rates in equilibrium.



(b) Background-trader surpluses achieved in equilibrium.

Figure 2.15: Total background-trader surpluses and HBL strategy adoption rates achieved at equilibria across different market settings. For each market environment, we compare four settings where background traders are respectively provided with the standard HBL strategy (dark grey), HBL with selective price blocking (light grey), HBL with price offsets (white), and HBL that combines the two variations (striped). Each marker specifies one equilibrium outcome in markets with spoofing (orange) and without spoofing (blue). Filled markers represent that some proportion of HBL with price blocking exists in the equilibrium strategy profiles, whereas hollow ones represent equilibrium strategy profiles without HBL price blocking.

rium results (Figure 2.15 white columns) show that the extended price offsets tend to largely improve HBL's profitability and background-trader surpluses, in both markets with and without manipulation. Such price adjustments can especially help learning traders to better adapt to high shock environments where prices are less predictable from past observations. However, the extended offsets may not directly address manipulation and improve learning robustness against spoofing.

2.7.3 Combine Order Blocking and Price Offsets

We observe that HBL with price offsets is overall competitive across different market settings, but its performance still degrades in markets with spoofing (refer to Figure 2.15 white columns). Since the second HBL variation demonstrates a general improvement in both settings with and without manipulation, we augment this variation with price level blocking to reduce vulnerability to spoofing. Specifically, we extend the background trading strategy set in Table 2.4 with six strategies: $\text{HBL}^{\chi=2}_{[-10,0]}$, $\text{HBL}^{\chi=2}_{[-20,0]}$, and $\text{HBL}^{\chi=2}_{[-40,0]}$ for the respective two memory lengths $L = 2$ and $L = 5$.

We conduct EGTA in a similar manner across market environments with and without spoofing. Equilibrium outcomes (Figure 2.15 striped columns) show that (1) compared to markets where only the standard and the price-blocking HBL are provided, HBL that combines the two variations is more widely preferred and can help to increase overall background-trader surplus in equilibrium; and (2) across all environments, background-trader surpluses in markets with and without spoofing fall roughly into the same ranges. These suggest that by combining the two proposed variations, HBL traders can enjoy both improved competitiveness and robustness against manipulation.

2.8 Conclusions

In this chapter, we construct a computational model of spoofing: the tactic of manipulating market prices by targeting the order book. To do so, we designed an HBL strategy that uses order book information to make pricing decisions. Since HBL traders use the order book, they are potentially spoofable, which we confirmed in simulation analysis. We demonstrate that in the absence of spoofing, HBL is generally adopted in equilibrium and benefits price discovery and social welfare. Though the presence of spoofing decreases the HBL proportion in background traders, HBL's persistence in equilibrium indicates a robustly spoofable market. By comparing equilibrium outcomes with and without spoofing, we find spoofing tends to decrease market surplus. Comparisons across parametrically different environments reveal factors that may influence the adoption of HBL and the impact of spoofing.

We further propose a cloaking mechanism to deter spoofing. The mechanism discloses a partially cloaked order book by symmetrically concealing a deterministic number of price levels from the inside. Our results demonstrate the proposed cloaking mechanism can significantly diminish the efficacy of spoofing, but at the cost of a reduced HBL proportion and surplus in equilibrium. With the goal of maximizing background-trader surplus, we perform EGTA across parametrically different mechanisms and environments, and find in markets with moderate shocks, the benefit of cloaking in mitigating spoofing outweighs its efficiency cost. By further exploring sophisticated spoofing strategies that probe to reveal cloaked information, we demonstrate the associated effort and risk exceed the gains, and verified that the proposed cloaking mechanism cannot be circumvented.

Two strategy variations based on the standard HBL strategy are explored. The first variation considers common characteristics of spoofing activities, and works by offering agents the flexibility to neglect limit orders at a specified price level when assembling a dataset to learn from. The second variation learns from full order book information, and later adjusts the target price derived from surplus maximization with a random offset to correct any biases in the learning process. Our analysis show that the first HBL variation offers learning traders a way to strategically block orders to improve robustness against spoofing, while achieving similar competitiveness in non-manipulated markets. Our second HBL variation exhibits a general improvement over baseline HBL, in both markets with and without manipulation. Further explorations suggest that traders can enjoy both improved profitability and robustness by combining the two HBL variations.

CHAPTER 3

Modeling the Evasion of Manipulation Detection: An Adversarial Learning Framework

This chapter proposes an adversarial learning framework to capture the evolving game between a regulator who develops tools to detect market manipulation and a manipulator who obfuscates actions to evade detection. The model includes three main parts: (1) a generator that learns to adapt original manipulation order streams to resemble trading patterns of a normal trader while preserving the manipulation intent; (2) a discriminator that differentiates the adversarially adapted manipulation order streams from normal trading activities; and (3) the agent-based model of spoofing described in Chapter 2 that evaluates the manipulation effect of adapted outputs.

Experiments are conducted on simulated order streams associated with a manipulator and a market-making agent respectively. The specific goal is to adapt manipulation order streams to resemble market-making, a legitimate trading role with generally positive influence on market efficiency. We show examples of adapted manipulation order streams that mimic a specified market maker's quoting patterns and appear qualitatively different from the original manipulation strategy implemented in the simulator. These results demonstrate the possibility of automatically generating a diverse set of (unseen) manipulation strategies that can serve as a training course for more robust detection algorithms.

3.1 Introduction

The work in Chapter 2 has proposed several deterrent mechanisms and trading strategies that aim to render manipulation strategies uneconomical. A more direct approach is to detect any manipulation activity. Rule-based methods that look for certain trading activities known as manipulation signatures (e.g., frequent order cancellations and modifications), however, may not be enough. They often result in high false positive rates, as these activities can also be legitimate actions for many non-manipulative participants, such as market makers and other liquidity providers (Foucault, Röell, and Sandås 2003; Hautsch and Huang 2012). Therefore, developing a reliable detector requires identifying or learning the manipulation intent from series of observed actions associated with individual trading agents across time, as they reveal agents' interactions with different market states and subsequent market outcomes.

Along this line of efforts, Nasdaq launched an AI-based surveillance system trained with historical trading data and existing patterns of market-abuse techniques to detect suspect equities trading practices (Rundle 2019). Despite promising advances, developing high-fidelity systems to detect manipulation even ex post faces two major challenges. First, the amount of labeled data identifying manipulation is quite small and may not be diverse enough to reflect all manipulation strategies. Second, given any launched detector, agents who seek to manipulate the market may obfuscate their strategies adversarially to evade detection (e.g., manipulating in a way that appears as normal trading activity). This causes regulators to play a costly game of cat-and-mouse with manipulators who constantly innovate to escape.

This chapter proposes using an adversarial learning framework to address these challenges; it reasons about how a manipulator might mask its behavior to evade the detection of a given discriminative model. Traders interact with the market by submitting orders to buy or sell, and throughout this chapter, we refer to the sequence of such actions taken by an individual trader over a period of time as the trader's *order stream*. It is a realization of its associated strategy, which reflects an agent's trading intent. The idea is to let a generative model learn to adapt existing manipulation strategies (represented as order streams) to resemble characteristics of normal trading, while preserving a comparable manipulation effect. A history of adapted order streams that effectively manipulate are further used to improve the robustness of the detector. We apply such adversarial reasoning recursively, updating the generator and the discriminator level-by-level, and characterize the evolution of adapted manipulation strategies.

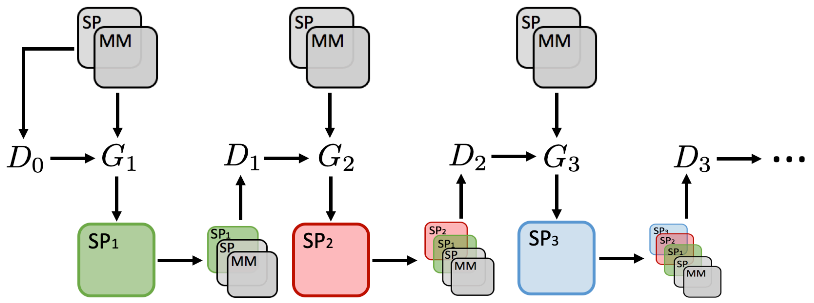
The generative model adopts a sequence-to-sequence paradigm (Sutskever, Vinyals, and Le 2014), and takes a manipulation order stream as source and a paired benign trader's order stream as target. It learns to adapt the source by minimizing the combination of an adversarial loss and a self-regularization loss. The adversarial loss is calculated by a discriminator that classifies an order stream as adapted from manipulation or target, minimized as the output becomes indistinguishable from a benign trader's order stream. The self-regularization loss is a feature-wise distance between the source and the adapted stream, penalizing large changes between the two to preserve the manipulation effect.

We conduct experiments and evaluate the proposed approach using order streams generated by the agent-based market simulator described in Chapter 2.[1] The simulator models simple manipulation strategies (i.e., spoofing), and can practically produce a large set of order streams associated with each agent across a variety of market conditions (e.g., different market shock and observation noise parameters). Controlled simulations are conducted to acquire (1) *source* order streams (SP) associated with a manipulation agent who deploys a variant of the spoofing strategy and (2) *target* order streams (MM) that a market-making agent would have placed under the corresponding market conditions. To help quantify the manipulation effect, we decompose the SP behavior into manipulation and exploitation components, and define *baseline* order streams (EXP) as those that only include a series of transacted exploitation orders. The goal here is to adapt manipulation order streams to resemble market-making, a legitimate trading role with generally positive influence on market efficiency (Schwartz and Peng 2013; Wah, Wright, and Wellman 2017). Figure 3.1 gives an overview of the approach.

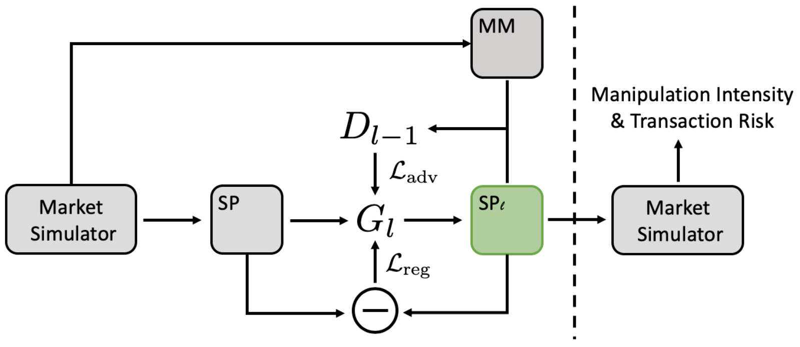
Experimental results show that the proposed framework can generate adapted manipulation order streams that resemble quoting patterns of a market maker and appear qualitatively different from the original spoofing strategy prescribed in the simulator. This adaptation evades detection, but at the cost of compromising effectiveness in manipulation. After a few iterations of evolving and evading the detector, the strategy has sacrificed almost all of its manipulation capability. Though it is likely impossible to develop a detector immune from adversarial attacks, modeling the evasion can be a useful step toward more robust detection of market manipulation.

The rest of this chapter is structured as follows. Section 3.2 provides background and discuss related work. We describe the trading strategies, data representations, and adversarial learning model in Section 3.3. Section 3.4 evaluates the proposed method and presents experiment findings. Section 3.5 concludes with discussions.

[1]Learning from real market data is infeasible, as actual order streams identified as manipulation do not exist in any substantial quantity.



(a) Update the generator and the discriminator level-by-level.


MM
D_{l-1}
$\mathcal{L}_{\text{adv}}$
Market Simulator
SP
G_l
SP$_l$
$\mathcal{L}_{\text{reg}}$
Manipulation Intensity & Transaction Risk
Market Simulator

(b) Given a fixed detector D_{l-1}, train G_l to generate SP$_l$.

Figure 3.1: Overview of the proposed adversarial learning framework that reasons about evading a manipulation detector. The process starts with a classifier D_0 that discriminates between SP and MM order streams. In response, a generator G_1 learns to adapt SP order streams, producing SP$_1$ that can evade detection by D_0. SP$_1$ order streams are then incorporated to train the next-level discriminator D_1. Such adversarial reasoning is applied recursively, producing a sequence of adapted manipulators and corresponding increasingly robust detectors.

3.2 Related Work

3.2.1 Agent-Based Modeling of Trading Roles

To study the effects of particular trading practices, researchers classify market participants into different roles based on their trading intent and activity patterns (e.g., trading volume, frequency, position). An agent-based market model designs agents around such roles, and reproduces "stylized facts" observed in real financial markets through simulating the strategic interactions of these agents (Kirilenko, Kyle, Samadi, and Tuzun 2017; Lebaron 2006).

This chapter builds on the developed agent-based model of spoofing, in which a manipulation agent can effectively deceive approximately rational background traders. Specifically, in markets populated with background learning traders who bid based on beliefs induced from market observations including the malicious activities, the manipulator is able to push prices significantly higher than they would be otherwise, and profit from this manipulation. Since background trading agents react to different market conditions according to their codified strategies, the model can be used to verify manipulation intent and quantify its impact by conducting controlled experiments of markets with and without a spoofing agent.

3.2.2 Learning via Adversarial Training

There is a substantial body of work on *adversarial training* (Goodfellow, Shlens, and Szegedy 2015; Sinha, Namkoong, and Duchi 2018; Tzeng, Hoffman, Saenko, and Darrell 2017; Volpi et al. 2018), investigating a variety of training procedures designed to learn models robust to (adversarial) perturbations in the input. Many of these approaches involve augmenting training dataset with examples from a target domain that is considered "hard" under the current model. A key issue addressed in some but not all of this work is to preserve specified properties of the source domain while generating adversarial examples to improve robustness.

Our proposed approach draws particular inspiration from Shrivastava et al. (2017), who proposed Simulated + Unsupervised (S+U) learning. The idea is to train a generative model to improve the realism of simulated images using unlabeled real ones, while preserving the annotation information from the simulator. A pixel-level loss is further imposed between the simulated input and the generated image to enforce annotation. Experimental results show that S+U learning enables the generation of highly realistic images with reliable labels and helps to improve learning models' performance on classification tasks, including gaze estimation and hand pose estimation. A similar idea was also employed by Bousmalis et al. (2018) for a robotics grasping problem. They extended pixel-level domain adaptation to improve the realism of synthetic data generated by the off-the-shelf grasp simulators. This chapter extends the approach to adapt simulated order streams while preserving the intent behind the original sequence of actions.

3.3 Problem Formulation

3.3.1 Trading Strategies and Representations

We follow prior work (Wah, Wright, and Wellman 2017; Wang and Wellman 2017) in the design of manipulation and market-making strategies, extending each with a bit of flexibility to reduce overfitting to artifacts. We describe the trading strategies and their representations as order streams below. Since an order stream is a sequence of actions incurred by a strategy, in this chapter, we refer to a strategy and order streams associated with that strategy interchangeably.

Manipulation Strategy (SP) During each simulation run, the manipulator maneuvers prices either up or down as instructed by the system with equal probability. We elaborate the case of manipulating prices up, and the other applies vice versa. The strategy includes three stages, similar to the one described in Section 2.6.1.3. During the first execution stage, the agent buys by accepting any sell order at price lower than the fundamental mean $\bar{r}$ (as formulated in Eq. 2.1). In the next manipulation stage, it stops buying and instead maintains large manipulation buy limit orders at price one tick below the best bid. The goal is to falsely signal demand to push price up so that the units bought earlier can be sold at higher prices later. During the last stage, the manipulator starts to sell the units by accepting any buy orders at a price higher than $\bar{r}$. The agent continues to manipulate until the trading period ends or all the bought units are sold.

Market-making Strategy (MM) Upon each arrival, the market maker submits a quote ladder centered around an estimate of the terminal fundamental value of the underlying security, denoted by $\hat{r}_t$. Specifically, the quote ladder is decided by three strategic parameters ω, K, ζ that respectively control the quote spread, number of price levels, and the number of ticks between two adjacent prices:

$$\begin{cases} [B_t - K\zeta, \ldots, B_t - (K-\beta)\zeta] & \text{for buy orders} \\ [S_t + (K-\alpha)\zeta, \ldots, S_t + K\zeta] & \text{for sell orders,} \end{cases} \tag{3.1}$$

where $B_t = \hat{r}_t - \omega/2$, $S_t = \hat{r}_t + \omega/2$, and α and β truncate the price ladder such that limit orders do not immediately transact with the market's current best bid and ask. To mitigate certain artifacts (e.g., prices separated by an equal distance), we add Gaussian noise around each price in Eq. (3.1) and its associated quantity. Since quote ladders are symmetrically centered around unbiased estimations of the terminal fundamental value, the MM orders in expectation do not distort learning traders' pricing beliefs. The MM agent follows the same arrival schedule as the manipulator to produce a paired target order stream, which records orders that would have placed under market conditions encountered by the manipulator.

Exploitation Strategy (EXP) The exploitation order streams serve as the control group to measure the effect of manipulation orders. The strategy executes the same buy and sell scheme as the SP strategy during the first and last stage without placing any manipulation order.

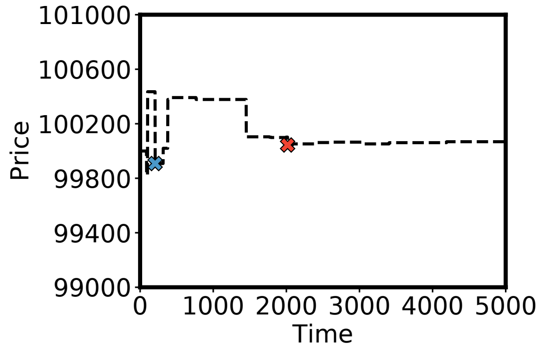
Order Stream Representation An order stream records a sequence of (hypothetical) actions associated with an agent. It is represented by a variable-length sequence with an element corresponding to each time an agent arrives and submits a *bid schedule*. A bid schedule comprises a set of limit orders, each specifying a price (expressed by distance to the market best bid or ask) and a quantity. Figure 3.2 shows order streams respectively associated with EXP, SP, and MM in a set of controlled simulations where we set $\bar{r} = 10^5$.

3.3.2 An Adversarial Learning Framework

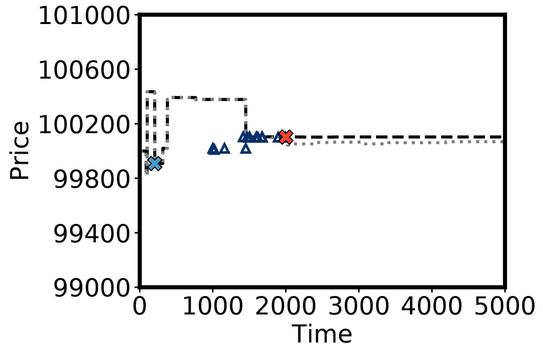
We use the market simulator to generate a dataset of labeled order streams $\mathcal{D} = \{(w_i, \mathrm{EXP}), (x_i, \mathrm{SP}), (y_i, \mathrm{MM})\}_{i=1}^{N}$, where w_i, x_i, and y_i denote order streams incurred by their respective strategies under one set of controlled simulations (like those in Figure 3.2). The goal here is to adapt the simulated SP order streams to become indistinguishable from the MM ones while preserving some manipulation effect.

The generator adopts the sequence-to-sequence paradigm (Sutskever, Vinyals, and Le 2014), which considers the interconnection between bid schedules within a sequence (e.g., a manipulator who buys first is more likely to manipulate price up and later sell). It has an encoder-decoder structure $G_\theta = (G_{\mathrm{enc}}, G_{\mathrm{dec}})$, where θ denotes the function parameters. This encoder-decoder model has been widely used in tasks that require sequence-to-sequence learning, such as the statistical machine translation (Cho et al. 2014; Sutskever, Vinyals, and Le 2014) and sentence generation (Logeswaran, Lee, and Bengio 2018). The encoder adopts a recurrent neural network (RNN) that takes an order stream x as input and produces a fixed-length latent representation vector $z_x := G_{\mathrm{enc}}(x)$. The vector contains compressed information of the input (e.g., manipulate prices up or down), and is decoded by G_{dec}, a second RNN that generates $x' \sim p_{G_{\mathrm{dec}}(\cdot|z_{\boldsymbol{x}})}$ to resemble characteristics of the target domain y. The discriminator D_ϕ also uses an RNN component followed by a linear layer, and outputs the probability of an input being an adapted order stream.

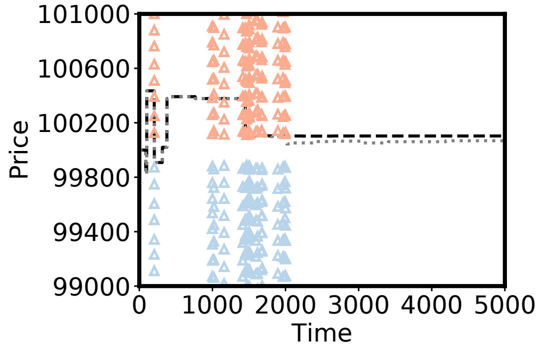
We propose a recursive training procedure of the generator and the detector (depicted in Figure 3.1(a)), designed to mimic the adversarial reasoning between a manipulator and a regulator. The manipulator starts by playing the SP strategy that is



(a) EXP (baseline).



(b) SP (source).



(c) MM (target).

Figure 3.2: Order streams associated with EXP, SP, and MM in a set of controlled simulations. During the execution stage (time before 1000), both EXP and SP bought one share of the security at price $\bar{r}-92$. Then, SP maintained manipulation buy orders at a tick behind the best bid to push the price up. As a result, SP managed to sell the share at price $\bar{r}+102$, whereas EXP sold the share at $\bar{r}+44$.

codified in the market simulator, and the regulator develops detector D_0 to distinguish manipulation order streams from MM streams. The manipulator then constructs its next-level strategy SP_1 by learning a generator G_1 to adapt SP, such that the adapted order streams can evade the detection of D_0 and preserve a comparable manipulation effect. To achieve both aims, the generator is trained to minimize a combination of adversarial loss and regularization loss (depicted in Figure 3.1(b)), which we describe in detail below. In response, a new detector D_1 is trained to identify both the original manipulation strategy SP and the evolved one SP_1. We apply such reasoning recursively to generate adversarial manipulation activities, so as to improve the robustness of a detector.

Adversarial Loss The generator works to bridge the gap between the source (i.e., SP) and the target (i.e., MM) by minimizing the adversarial loss. We follow the GAN setup (Goodfellow et al. 2014) which models the generator and the discriminator as a two-player minimax game. During training, the level-l discriminator network D_l updates its parameters ϕ_l to minimize the following loss:

$$\mathcal{L}_D(\phi_l) = -\sum_i \log(D(x'_i; \phi_l)) - \sum_i \log(1 - D(y_i; \phi_l)), \tag{3.2}$$

where x'_i represents some learned (or identity) transformation of x_i, and $D(\cdot)$ denotes the probability of the input order stream either associated with or adapted from SP.

We fix the discriminator D_{l-1} and train the level-l generator G_l to maximize the probability of D_{l-1} making a mistake. Specifically, it learns θ_l by minimizing the adversarial loss:

$$\mathcal{L}_G^{\text{adv}}(\theta_l) = -\sum_i \log(1 - D_{l-1}(G(x_i; \theta_l))). \tag{3.3}$$

Self-Regularization Loss To preserve the manipulation effect, we combine the adversarial loss with a self-regularization loss that penalizes any difference between the adapted and original order stream. This can be interpreted as a manipulator preference to adapt its original manipulation strategy as little as possible to evade detection. We define regularization loss as the mean squared error between the input and the adapted order stream:

$$\mathcal{L}_G^{\text{reg}}(\theta_l) = \frac{1}{N} \sum_i \|G(x_i; \theta_l) - x_i\|_2^2, \tag{3.4}$$

where $\|\cdot\|_2$ is the L2 norm. The overall loss for G is $\mathcal{L}_G = \mathcal{L}_G^{\text{adv}} + \lambda \mathcal{L}_G^{\text{reg}}$, where λ is a hyperparameter.

Measuring Manipulation Effects We evaluate the manipulation effects of the adapted order stream $x'_i := G_l(x_i)$ by feeding it back to the market simulator under the same set of experimental controls. That is, we compare the effects under scenarios where background traders are guaranteed to arrive at the same time, receive identical private values, and observe the same fundamental values as in simulations that generate w_i, x_i, and y_i. Any change in background bidding behavior can therefore be attributed to the adapted order stream.

We compare market outcomes incurred by the adapted order stream to those of markets with SP and EXP, and measure the *manipulation intensity* and *transaction* *risk*. The manipulation intensity of x_i', denoted by $\delta_{x_i'}$, is defined as the fraction of the price deviation realized by x_i' in that of the SP order stream:

$$\delta_{x_i'} = \begin{cases} \min\left\{ \max\left\{\frac{P_{x_i'} - P_{w_i}}{P_{x_i} - P_{w_i}}, 0\right\}, 1\right\} & \text{if } P_{x_i} > P_{w_i} \\ \\ \min\left\{ \max\left\{\frac{P_{w_i} - P_{x_i'}}{P_{w_i} - P_{x_i}}, 0\right\}, 1\right\} & \text{otherwise,} \end{cases} \tag{3.5}$$

where P_{w_i}, P_{x_i}, and $P_{x_i'}$ denote the average transaction price in respective markets since the start of the manipulation stage. The higher the manipulation intensity is, the better x_i' preserves the manipulation effect. Transaction risk is defined as the ratio between the number of transactions and the number of arrivals during the manipulation phase. By definition, SP and EXP have manipulation intensity one and zero, respectively, and both exhibit transaction risk zero. Algorithm 2 describes the detailed procedure of training G and D in one specific level.

3.4 Experimental Results

We follow the proposed framework and generate adversarial order streams by adapting the simulated SP order streams to look like quoting patterns of a market maker. We visualize examples of adapted manipulation activities, and demonstrate the competing improvement between the adapted manipulation strategies and the detectors.

3.4.1 Dataset and Implementation Details

We conduct simulations using the agent-based market simulator, and generate 10,944 groups of labeled order streams $\{(w_i, \text{EXP}), (x_i, \text{SP}), (y_i, \text{MM})\}$.[2] The order streams respectively record (hypothetical) trading activities of a manipulator, a market maker, and an exploitation agent. Each trading session lasts 5000 time steps, and the generated order streams have lengths varying from 4 to 91. The first execution stage is from time 200 to 1000, after which the manipulation agent starts to spoof. At time 2000, it begins to liquidate previously accumulated positions. The underlying security has a fundamental mean $\bar{r} = 10^5$. Based on estimations of the final funda-

[2]We first conducted 30,000 controlled simulation runs, yielding 30,000 groups of labeled order streams. We kept those groups in which the manipulator successfully trades during the first stage (so that there is an incentive to spoof later), and pushes prices to its desired direction by at least ten ticks. This gives us 10,944 groups of labeled order streams, which meet the described filter standard.

Algorithm 2 Adversarial Training Procedure of G_θ^l and D_ϕ^l

```
Input: $\mathcal{D}\{(x_i, \mathrm{SP}), (y_i, \mathrm{MM}), (w_i, \mathrm{EXP}), s_i\}_{i=1}^N$. Data buffer $\mathcal{B}\{x_i^*\}_{i=1}^N$ with a mixture of $\boldsymbol{x}$, $\boldsymbol{x}^1$, ..., $\boldsymbol{x}^{l-1}$. $D_\phi^{l-1}$.
Output: $G_\theta^l$ and $D_\phi^l$.
1: for $t = 1, ..., \mathrm{epoch}$ do
2:     for $j = 1, ..., \mathrm{batch}$ do
3:         Generate $\boldsymbol{x}'_j = G_\theta(\boldsymbol{x}_j)$
4:         Update $\theta$ on the batch loss $\mathcal{L}_G(\theta)$
5: Get optimal $G_\theta^l$
6: for $i = 1, ..., N$ do
7:     Feed $x'_i := G_\theta^l(x_i)$ back to simulator with seed $s_i$
8:     Replace $x_i^*$ with $x'_i$ with probability 0.9 if $P_{x'_i} \geq \delta P_{x_i}$
9: for $t = 1, ..., \mathrm{epoch}$ do
10:    for $j = 1, ..., \mathrm{batch}$ do
11:        Sample $\boldsymbol{x}_j^* \in \mathcal{B}$ and $\boldsymbol{y}_j \in \mathcal{D}$
12:        Update $\phi$ on the batch loss $\mathcal{L}_D(\phi)$
13: Get optimal $D_\phi^l$
```

mental value, the MM submits a quote ladder with $\omega = 256$, $K = 8$, $\zeta \sim N(128, 10)$, and quantity $q \sim N(5, 2)$. We use 8896 groups of order streams for training (with a 80/20 train-validation split) and the rest 2048 groups for testing.

We use a bi-directional Gated Recurrent Unit (GRU) RNN (Cho et al. 2014) with a hidden state size of 64, followed by a linear layer for both G_{enc}, G_{dec}, and D in the experiments. Since order streams are of variable lengths, we pad them to the maximum length for forward passes, and cut them back to original lengths for loss calculations and evaluations. Model parameters are initialized with the uniform distribution between –0.08 and 0.08. We use batches of 64 order streams to train the discriminator and the generator, and pick weight of the self-regularization loss $\lambda = 1$ based on the validation performance.

3.4.2 Generating Adapted Manipulation Examples

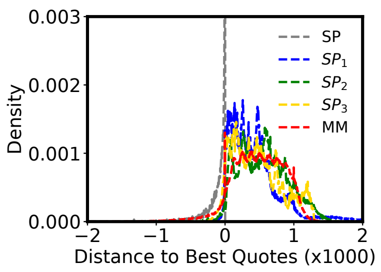
We evaluate the adversarially adapted order streams from three main aspects: (1) similarity to the MM quoting patterns, (2) preservation of manipulation effect, and (3) effectiveness in evading the detection of an existing discriminator. Table 3.1 presents summary statistics of order streams associated with their corresponding trading strategies (or generative models). Each aspect is discussed in detail below.

	Payoff	Manipulation Effect	Transaction Risk	D_{l-1} (%)	D_l (%)
SP	$411^{*,**}$	1	0	-	100
SP_1	$362^{*,**}$	0.50	0.14	0.59	100
SP_2	310^{*}	0.30	0.26	0	100
SP_3	303^{*}	0.22	0.59	0	100
MM	121	0.15	0.85	100	100
EXP	324^{*}	0	0	-	-

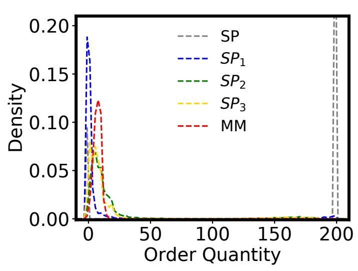
Table 3.1: Summary statistics of the respective trading strategy on test dataset. Asterisks denote statistical significance at 5% level of the paired t-test for payoffs compared to MM(*) and EXP(**).

Comparing to MM We follow prior work (Li et al. 2020) in using price and quantity distributions to measure how well the generated order streams resemble the target MM streams. We further propose a domain-specific measure, the *order imbalance distribution*, defined as the ratio between the numbers of buy and sell orders submitted over a trading period (whichever value is larger on the numerator). This captures a trader's imbalance in preference between long and short positions. Figure 3.3 presents comparisons of the respective distributions. Results show that the adapted manipulation order streams produce distributions similar to that of the MM, and are able to overcome certain artifacts codified in the SP strategy (e.g., large-quantity orders always at one tick behind the best quotes and severe order imbalance to deceive the market). Specifically, orders are gradually adapted to cover a wider range of prices with relative small quantities, and order balance is roughly maintained throughout the trading period.

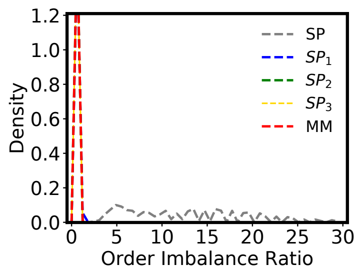
Preserving Manipulation Effect In the final step, we feed adapted order streams back to the market simulator under the same set of experimental controls, and measure their manipulation effect by the manipulation intensity and transaction risk. Figure 3.4 shows the two-dimensional cumulative density over the 2,048 adapted outputs with respect to the two proposed metrics. We find that SP_1 can preserve a comparable manipulation intensity under a reasonable transaction risk; however, as the generator adapts in response to a more robust discriminator, the adapted streams begin to suffer a large degradation in manipulation intensity and an increase in transaction risk (e.g., SP_3 has a similar performance to MM). This weakened manipulation effect is further confirmed in Table 3.1.



(a) Price distribution.

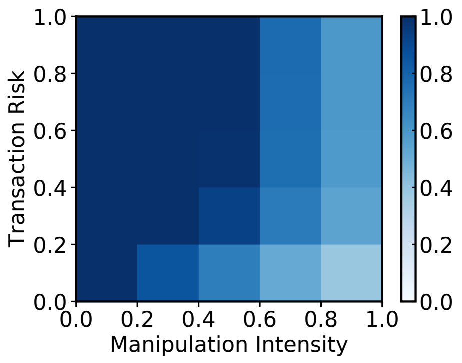


(b) Quantity distribution.

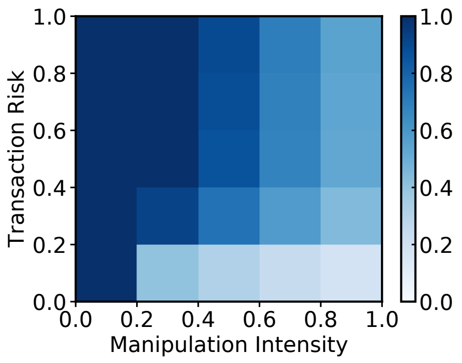


(c) Order imbalance distribution.

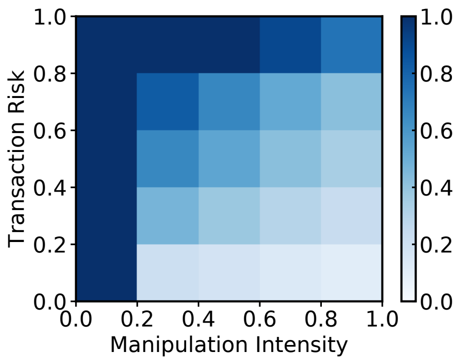
Figure 3.3: Comparisons of the respective statistics on the SP order streams, adapted outputs, and MM order streams.



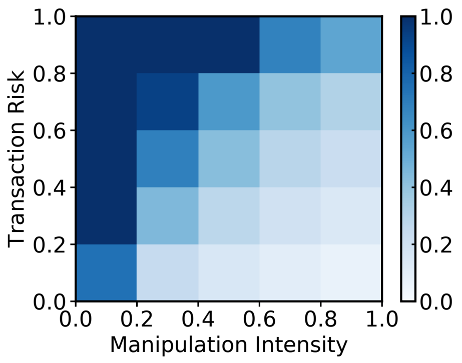
(a) SP$_1$.



(b) SP$_2$.



(c) SP$_3$.



(d) MM.

Figure 3.4: The manipulation effect of order streams associated with the corresponding level of SP strategy. Each color of a cell encodes the cumulative density of order streams that achieve a certain manipulation intensity and transaction risk. The closer dark blue is to the bottom right, the better adapted order streams are able to preserve higher manipulation intensity with lower transaction risk.

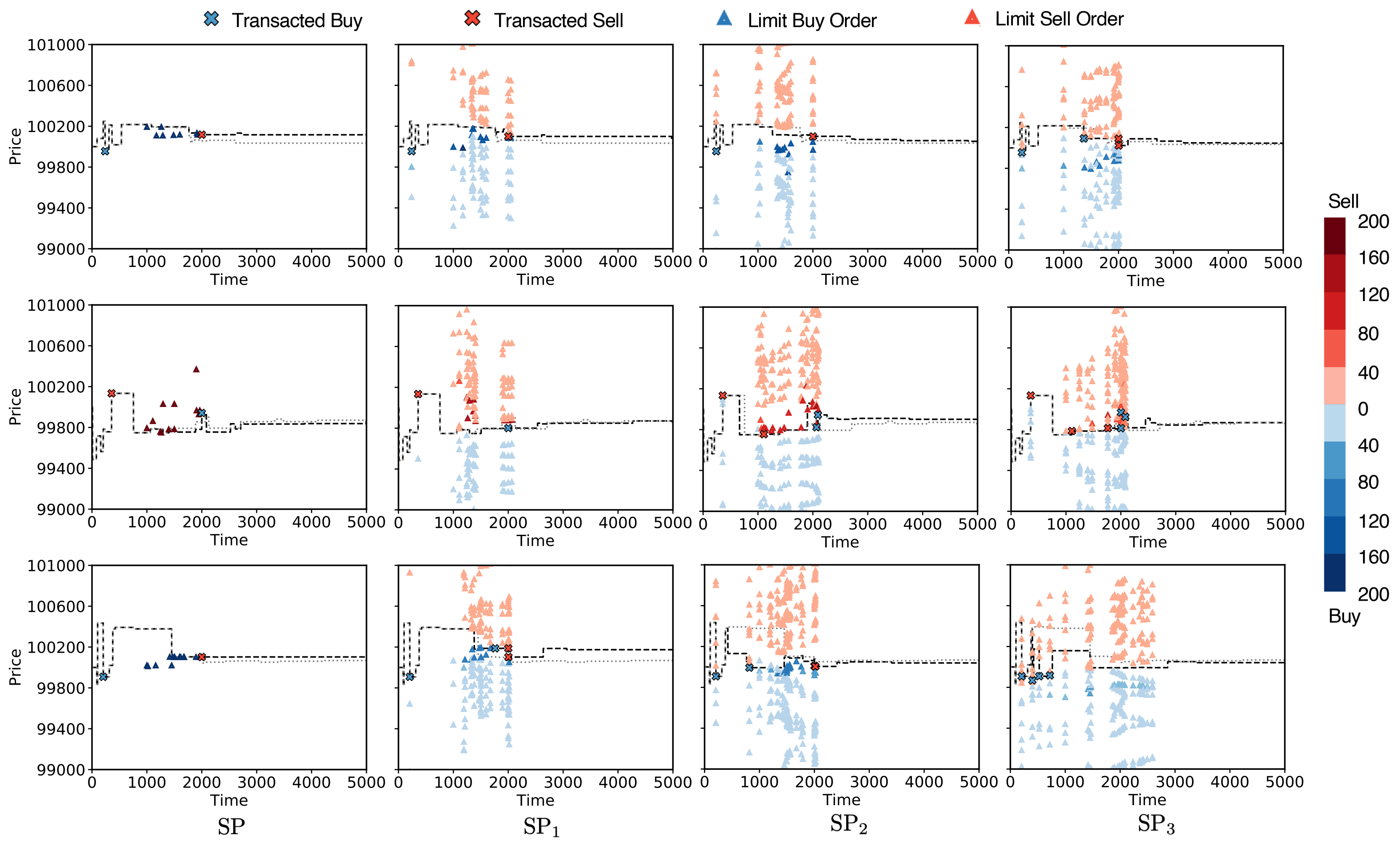


Figure 3.5: Examples of adapted manipulation order streams. Dashed black lines represent the latest transaction prices, whereas dashed grey lines the transaction prices if no manipulation exists.

Evading the Detection Table 3.1 shows that a generator can easily fool an existing detector with adversarially generated order streams. By learning from a history of adapted order streams, the discriminator is able to detect manipulation streams from all previous levels, and in the meantime ensures the training stability of the next-level generator.

Qualitative Evaluation Figure 3.5 demonstrates examples of the original and its corresponding adapted manipulation order streams. These examples demonstrate that the adapted streams become qualitatively similar to the trading patterns of a MM, and such simultaneous quoting behavior on both sides of the market has indeed been suggested as a good strategy for high frequency traders to mask their manipulative intent (Levens 2015). We note several other findings from the evolution of adapted manipulation strategies. First, SP_1 remains to place large orders close to the market best quote, whereas SP_2 and SP_3 choose to either largely decrease the order quantity or place large orders behind smaller ones to avoid being detected. Second, SP_2 and SP_3 tend to submit orders at more aggressive prices across market quotes, and this may cause unintended transactions during the manipulation phase.

3.5 Conclusions

This chapter employs an adversarial learning framework to model the evolving game between a regulator and a manipulator, in which the regulator deploys algorithms to detect manipulation and the manipulator masks actions to evade detection. Evasion is represented by a generative model, trained by augmenting manipulation order streams with examples of market making activity traces. The intent is to produce adapted streams that are hard to distinguish from a market maker's behavior. We visualize examples of adapted manipulation order streams, and show they resemble quoting patterns of a market maker and appear qualitatively different from the original manipulation strategy we implemented in the simulator. This adaptation evades detection, but only at the cost of compromising effectiveness in market manipulation. After a few iterations of evolving and evading the detector, the strategy has sacrificed almost all of its manipulation capability.

Results presented here reflect the specific simulation choices adopted, and thus it remains to be seen whether a more clever form of adaptation can evade detection while retaining more effectiveness in manipulation. Whether or not it is possible to ultimately craft successful adversarial attacks, the generation and evasion process modeled here provides a way to anticipate the evolution of evasive adversaries. Such anticipation capacity provides a way to develop more robust detection methods, for market manipulation as well as other fraudulent behaviors.

CHAPTER 4

Designing a Combinatorial Financial Options Market

This chapter studies expressive mechanisms for *financial options market*. Options are contracts that specify the right to buy or sell an underlying asset at a *strike price* by an expiration date. Standard exchanges offer options of predetermined strike values and trade options of different strikes independently, even for those written on the same underlying asset. Such independent market design can introduce arbitrage opportunities, and lead to the thin market problem. This chapter proposes a mechanism that consolidates and matches orders on standard options related to the same underlying asset, while providing the flexibility of specifying any custom strike value. The mechanism runs in time polynomial to the number of orders, and poses no risk to the exchange, regardless of the value of the underlying asset at expiration. Empirical analysis on real-market options data shows that the mechanism can find new matches for options of different strike prices and reduce bid-ask spreads.

Extending standard options written on a *single* asset, this chapter proposes *combinatorial financial options* that offer contract holders the right to buy or sell any linear combination of multiple underlying assets. We generalize the proposed mechanism to match options written on different combinations of assets, and prove that optimal clearing of combinatorial financial options is coNP-hard. To facilitate market operations, we propose a heuristic algorithm that finds the exact optimal match through iterative constraint generation, and evaluate its performance on synthetically generated combinatorial options markets of different scales. As option prices reveal the market's collective belief of an underlying asset's future value, a combinatorial options market enables the expression of aggregate belief about future correlations among assets.

4.1 Introduction

Financial options are securities that provide the holder with rights to conduct specific trades in the future. For example, a S&P 500 *call option* with *strike price* 3500 and *expiration date* December 31, 2020 provides the right to buy one share of S&P 500 at \$3500 on that day.[1] A standard financial option is a derivative instrument of a *single* underlying asset or index, and its payoff is a function of the underlying variable. The example contract pays $\max\{S - 3500, 0\}$, where S is the value of the S&P 500 index on the expiration date.[2] Investors trade options to hedge risks and achieve certain return patterns, or to speculate about the movement of the underlying asset price, buying an option when its price falls below their estimate of its expected value. Thus, option prices reveal the collective risk-neutral belief distribution of the underlying asset's future value.

Despite the significant volume of trade and interest in financial options, the standard options market has two limitations that compromise its expressiveness and efficiency. First, markets for options of a specific underlying asset only feature a selective set of predetermined strike prices and expiration dates, thus limiting the ability to elicit and recover a full and continuous distribution of the underlying variable. As of this writing, the Chicago Board Options Exchange (CBOE) offers 276 distinct strike prices, ranging from 700 to 5000, for S&P 500 options expiring by the end of 2020. Second, each market independently aggregates and matches orders in regard to a single contract with a specified option type, strike price, and expiration date, despite its interconnectedness to other options and their common dependency on the underlying asset. Such independent market design fails to match options with different strike prices, and may introduce arbitrage opportunities. Moreover, investments get diluted across independent markets even when participants are interested in the same underlying asset, and this can lead to the problem of thin markets, where few trades happen and bid-ask spreads become wide. Even for some of the most actively traded option families, empirical evidence has shown that liquidity can vary much across option types and strikes. Cao and Wei (2010) studied eight years of options trading data and find consistently lower liquidity in puts and deep in-the-money options.

[1] Throughout this study, I restrict to *European options* which can be exercised only at expiration. For simplicity, examples are given without the typical 100x contract multiplier.

[2] The settlement value is calculated as the opening value of the index on the expiration date or the last business day (usually a Friday) before the expiration date. In cash-settled markets, instead of actual physical delivery of the underlying asset, the option holder gets a cash payment that is equivalent to the value of the asset.

We propose a linear program that computationally efficiently matches orders across markets that are logically related to the same underlying asset, while enabling traders to specify any custom option contract (e.g., a S&P call option with a strike price of 3502 that expires a week from today). It runs in time polynomial to the number of orders and poses no risk to the exchange regardless of the value of the underlying security at expiration. Empirical analysis conducted on real-market options data shows that our mechanism, by consolidating independently-traded options markets, can indeed find arbitrage and substantially reduce bid-ask spreads. The improved efficiency and expressiveness may help to aggregate more fine-grained information and recover a complete and fully general probability distribution of the underlying asset's future value.

In the second part of this chapter, we further generalize standard financial options to design a *combinatorial financial options market* in which agents can bid or offer options written on any linear combination of underlying assets, thus enabling the elicitation and recovery of future correlations among assets—in general, their full joint distribution. For example, a call option written on "1AAPL $+$1MSFT" with strike price 300 specifies the right to buy one share of Apple *and* one share of Microsoft at \$300 total price on the expiration date, whereas a call on "1AAPL $-$1MSFT" with strike price 50 confers the ability to buy one share of Apple and sell one share of Microsoft at expiration for a net cost of \$50.

Combinatorial options offer traders to conveniently and precisely hedge their exact portfolio, replicating any payoff functions that standard options can achieve. Standard options on mutual funds and stock indices like the S&P 500 and Dow Jones Industrial Average (DJI) operate on one specific predefined portfolio. The CBOE has recently launched options on eleven Select Sector Indices,[3] each of which can be considered as a pre-specified linear combination of stocks. The goal of this work is to support products like these and generalizes to any custom combinations.

As traders are offered the expressiveness to specify shares (or weights) for each underlying asset, new challenges arise and the thin market problem exacerbates. Opening a separate exchange for each combination of stocks and weights would rapidly grow intractable. Naively matching only buy and sell orders for the exact same portfolio may yield few or no trades at all, despite plenty of acceptable trades among orders.

Extending the model that consolidates standard options on a single underlying se-

[3]https://markets.cboe.com/tradable_products/sp_500/cboe_select_sectors_index_options/

curity, we propose an optimization formulation that can match combinatorial options written on different linear combinations of underlying assets. It works by maximizing net profit subject to no risk to the exchange, regardless of the values of all assets at expiration. We show that the proposed mechanism with increased expressiveness, however, comes at the cost of a higher computational complexity: determining the optimal clearing of a combinatorial options market is coNP-hard.

We demonstrate that the proposed mechanism can be equivalently formulated as a bilevel mixed-integer linear program; it finds the exact optimal-matching solution by satisfying an increasing set of constraints generated from different future values of the underlying assets. In experiments on synthetic combinatorial orders generated from real-market standard options prices, we show that in practice the bilevel optimization terminates quickly, with its running time growing linearly with the number of orders and the size of underlying assets.

4.2 Related Work

4.2.1 Rational Option Pricing

The proposed matching mechanisms are closely related to and built on arbitrage conditions that have been studied extensively in financial economics (Modigliani and Miller 1958; Varian 1987). In short, an arbitrage describes the scenario of "free lunches"—configurations of prices such that an individual can get something for nothing. The matching operation of the exchange (i.e., the auctioneer function) can be considered as arbitrage elimination: matching orders in effect works by identifying combinations of orders that reflect a risk-free surplus (i.e., gains from trade).

Merton (1973a) first investigated the no-arbitrage pricing for options, stating the necessity of convexity in option prices. Other relevant works examine no-arbitrage conditions for options under different scenarios, such as modeling the stochastic behavior of the underlying asset (e.g., the Black-Scholes model), and considering the presence of other types of securities (e.g., bonds and futures). The most relevant work to our proposed approach is by Herzel (2005), who makes little assumption on the underlying process and other financial instruments. The paper proposes a linear program to check the convexity between every strike-price pair; it finds arbitrage opportunities that yield positive cash flows now and no liabilities in the future on European options written on the same underlying security. Our contribution on consolidating standard options generalizes Herzel's, by also allowing a temporary deficit (i.e., a negative cash flow) now, if it is guaranteed to earn it back at the time of option expiration. To our knowledge, no prior work has defined general combinatorial options or investigated the matching mechanism design and complexity for such a market.

4.2.2 Combinatorial Market Design

Much prior work examines the design of combinatorial markets, both exchanges and market makers (Chen et al. 2008; Hanson 2003), for different applications including prediction markets with Boolean combinations (Fortnow, Kilian, Pennock, and Wellman 2005), permutations (Chen, Fortnow, Nikolova, and Pennock 2007), hierarchical structures (Guo and Pennock 2009), tournaments (Chen, Goel, and Pennock 2008), and electronic sourcing (Sandholm 2007). Dudík, Lahaie, Pennock, and Rothschild (2013) show how to employ constraint generation in linear programming to keep complex related prices consistent. Kroer, Dudík, Lahaie, and Balakrishnan (2016) generalize this approach using integer programming. Designing combinatorial markets faces the tradeoff between expressiveness and computational complexity: giving participants greater flexibility to express preferences can help to elicit better information and increase economic efficiency, but leads to a more intricate mechanism that is computationally harder. Several works have formally described and quantified such tradeoff (Benisch, Sadeh, and Sandholm 2008; Golovin 2007), and studied how to balance it by exploiting the outcome space structure and limiting expressivity (Chen et al. 2008; Dudík, Wang, Pennock, and Rothschild 2020; Laskey et al. 2018; Xia and Pennock 2011). This chapter contributes to the rich literature by extending to the popular financial options market and designing mechanisms to operate such markets.

4.3 Background and Notations

There are two types of options, referred to as *call* and *put* options. We denote a call option as $C(S, K, T)$ and a put option as $P(S, K, T)$, which respectively gives the option buyer the right to buy and sell an underlying asset S at a specified *strike price* K on the *expiration date* T. In the rest of this chapter, we omit T from the tuples for simplicity, as the mechanism aggregates options within the same expiration.

The option buyer decides whether to exercise an option. Suppose that a buyer spends \$8 and purchases a call option, $C(S\&P\,500, 3500, 20201231)$. If the S&P 500 index is \$3700 at expiration, the buyer will pay the agreed strike \$3500, receive the index, and get a *payoff* of \$200 and a *net profit* of \$192 (assuming no time value). If the S&P 500 price is \$3200, the buyer will walk away without exercising the option. Therefore, the payoff of a purchased option is

$$\Psi := \max\{\chi(S-K), 0\},$$

where S is the value of underlying asset at expiration and $\chi \in \{-1, 1\}$ equals 1 for calls and -1 for puts. As the payoff for a buyer is always non-negative, the seller receives a *premium* now (e.g., \$8) to compensate for future obligations.

Option contracts written on the same underlying asset, type, strike, and expiration are referred to as an *option series.* Consider options of a single security offering both calls and puts, ten expiration dates and fifty strike prices. All option series render a total of a thousand markets, with each maintaining a separate *limit order book.* In such a market, deciding the existence of a transaction takes $\mathcal{O}(1)$ time by comparing the best bid and ask prices, and matching an incoming order can take up to $\mathcal{O}(N)$ time depending on its quantity, where N is the number of orders on the opposite side of the order book.

4.4 Consolidating Standard Financial Options

A linear program is proposed to consolidate and match options written on the same underlying asset across different types and strike prices. The model is simple without making any assumptions on the option's pricing model or the stochastic behavior of the underlying security.

4.4.1 Match Orders on Standard Options

Let us consider an options market in regard to a single underlying asset S with an expiration date T. It has a set of buy orders indexed $m \in \{1, 2, ..., M\}$ and a set of sell orders indexed $n \in \{1, 2, ..., N\}$. Buy orders are represented by a type vector $\boldsymbol{\phi} \in \{-1, 1\}^M$, a strike price vector $\boldsymbol{p} \in \mathbb{R}_+^M$, and bid prices $\boldsymbol{b} \in \mathbb{R}_+^M$. Sell orders are denoted by a separate type vector $\boldsymbol{\psi} \in \{-1, 1\}^N$, a strike vector $\boldsymbol{q} \in \mathbb{R}_+^N$, and ask prices $\boldsymbol{a} \in \mathbb{R}_+^N$.

The exchange aims to match buy and sell orders submitted by traders. Specifically, it decides the fraction $\boldsymbol{\gamma} \in [0, 1]^M$ to sell to buy orders and the fraction $\boldsymbol{\delta} \in [0, 1]^N$ to buy from sell orders. The objective is to maximize the net profit, taking into account the potential deficit or gain in the future, denoted by a decision variable $L \in \mathbb{R}$:

$$\begin{aligned} \max_{\boldsymbol{\gamma},\boldsymbol{\delta},L} \quad & \boldsymbol{b}^\top \boldsymbol{\gamma} - \boldsymbol{a}^\top \boldsymbol{\delta} - L && \text{(M.1)} \\ \text{s.t.} \quad & \underbrace{\sum_m \gamma_m \max\{\phi_m(S-p_m),0\}}_{\Psi_\Gamma} - \underbrace{\sum_n \delta_n \max\{\psi_n(S-q_n),0\}}_{\Psi_\Delta} \leq L, \quad \forall S \in [0,\infty) \\ & && (4.1) \end{aligned}$$

Here, the first term $\sum_m \gamma_m \max\{\phi_m(S-p_m),0\}$ in the constraint (4.1) calculates the total payoff of sold options as a function of S, which is the obligation or liability of the exchange at the time of option expiration. The second term $\sum_n \delta_n \max\{\psi_n(S-q_n),0\}$ computes the total payoff of bought options, by which the exchange has the right to exercise. The constraint guarantees that the difference between the liability and the payoff of the exchange will be bounded by L, regardless of the value of S on the expiration date. We denote options bought by the exchange as Portfolio Δ and options sold as Portfolio Γ, and describe their relationship below.

Definition 4.1 (Payoff Dominance. Adapted from Merton (1973b))**.** Portfolio Δ *(weakly) dominates* Portfolio Γ with an offset L, if the payoff of portfolio Δ plus a constant L is greater than or equal to that of Portfolio Γ for all possible states of the underlying variable at expiration. Portfolio Γ is said to be *(weakly) dominated* by portfolio Δ with an offset L.

We analyze the complexity of running Mechanism M.1. The left-hand side of constraint (4.1) is a linear combination of max functions, and thus is a piecewise linear function of S. Therefore, it suffices to solve M.1 by satisfying constraints defined by S at each breakpoint. In our case, breakpoints of the constraint (4.1) are the defined strike values in the market, plus two endpoints: $\boldsymbol{p} \cup \boldsymbol{q} \cup \{0,\infty\}$. Let n_K denotes the number of distinct strike values in the market, which is bounded above by $n_{\text{orders}} = M + N$, the total number of orders in the market. Therefore, M.1 is a linear program that has $n_K + 2$ payoff constraints, and requires time polynomial in the size of the problem instance to solve. Complete proofs from this chapter are deferred to Appendix B.

Theorem 4.1. *Mechanism M.1 matches options written on the same underlying asset and expiration date across all types and strike prices in time polynomial in the number of orders.*

We give two motivating examples based on real-market options data below to illustrate the economic meaning and the usefulness of a flexible L. In short, the decision variable L allows the exchange to take a (worst-case) deficit at the future time of option expiration if it is preemptively covered by a surplus (i.e., revenue) at the time of contract transaction (Example 4.1) or to take a temporary deficit (i.e., expense) at the time of contract transaction if it is guaranteed to earn it back later at the time of option expiration (Example 4.2). Therefore, the mechanism guarantees no loss (as L is incorporated in the objective) for each match, and has an extra degree of freedom to match orders.

Example 4.1 (A match with a positive L). We use the proposed Mechanism M.1 to consolidate options of Walt Disney Co. (DIS) that are priced on January 23, 2019 and expire on June 21, 2019. We find the following match, where each order would not transact in its corresponding independent market. The exchange can

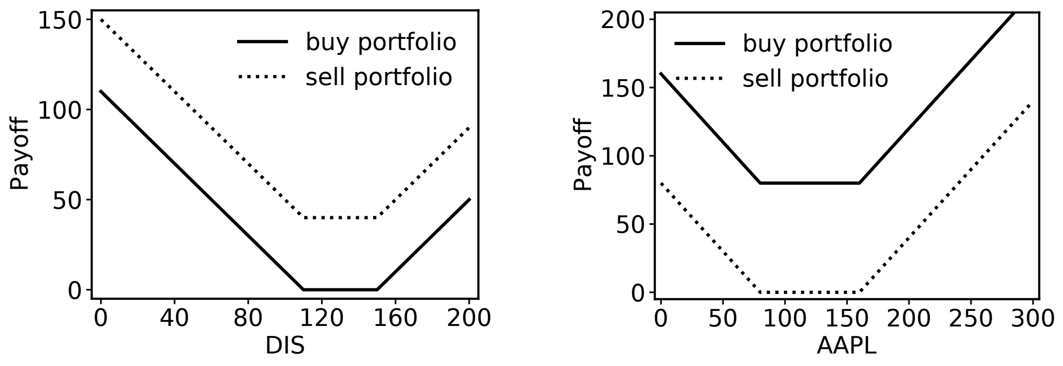
- sell to the buy order on $C(\texttt{DIS}, 110)$ at bid \$7.2,
- sell to the buy order on $P(\texttt{DIS}, 150)$ at bid \$38.75,
- buy from the sell order on $C(\texttt{DIS}, 150)$ at ask \$0.05,
- buy from the sell order on $P(\texttt{DIS}, 110)$ at ask \$5.1,

and get an immediate gain of \$40.8 ($7.2 + 38.75 - 5.1 - 0.05$). Figure 4.1(a) plots the payoffs of bought and sold options as a function of DIS, showing that the exchange will have a net liability of \$40 (i.e., $L = 40$) regardless of the DIS value at expiration. The exchange makes a net profit of \$0.80 from the match at no risk. $\square$

Example 4.2 (A match with a negative L). We consolidate options of Apple Inc. (AAPL) that are priced on January 23, 2019 and expire on January 17, 2020. We find the following match, where the exchange can

- sell to the buy order on $C(\texttt{AAPL}, 160)$ at bid \$14.1,
- sell to the buy order on $P(\texttt{AAPL}, 80)$ at bid \$0.62,
- buy from the sell order on $C(\texttt{AAPL}, 80)$ at ask \$74.2,
- buy from the sell order on $P(\texttt{AAPL}, 160)$ at ask \$19.1.

The match incurs an expense of \$78.58 ($14.1 + 0.62 - 74.2 - 19.1$) and yields a guaranteed payoff of \$80 (i.e., $L = -80$) at expiration. Figure 4.1(b) depicts the respective payoffs of bought and sold options. From the match, we can infer an interest rate of 1.82%, calculated by $78.58e^{r\Delta t} = 80$. $\square$



(a) Payoffs of options bought and sold in Example 4.1 as a function of DIS value.

(b) Payoffs of options bought and sold in Example 4.2 as a function of AAPL value.

Figure 4.1: Payoffs of the matched options as a function of the value of the underlying asset at expiration. Fig. 4.1(a) shows the case of $L > 0$, and Fig. 4.1(b) the case of $L < 0$.

Remarks. Several extensions can be directly applied to the mechanism:

(1) The time value of investments can be incorporated by multiplying L by a (discount) rate in the objective of Mechanism M.1.

(2) We can adapt constraints on decision variables $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$ to reflect different quantities specified in orders.

(3) By restricting L equal to 0, Mechanism M.1 matches orders by finding a common form of arbitrage where the exchange may profit at the time of order transaction subject to zero loss in the future.

4.4.2 Quote Prices for Standard Options

A standard exchange maintains the best quotes (i.e., the highest bid and lowest ask) for each independent options market. This section extends Mechanism M.1 to quote the most competitive prices for a *custom* option of any type and strike by considering other options related to the same underlying security. We describe the price quote procedure in an arbitrage-free market $(\boldsymbol{\phi}, \boldsymbol{p}, \boldsymbol{b}, \boldsymbol{\psi}, \boldsymbol{q}, \boldsymbol{a})$ below, and defer the proof of correctness to the Appendix B.1.2.

(1) The best bid b^* for a custom option (χ, S, K, T) is the maximum gain of selling a portfolio of options that is *weakly dominated* by (χ, S, K, T) for some L.

We derive b^* by adding (χ, S, K, T) to the sell side of the market indexed $N+1$ (as the exchange buys from sell orders), initializing its price a_{N+1} to 0, and solving for M.1. The best bid b^* is then the returned objective.

(2) The best ask a^* for a custom option (χ, S, K, T) is the minimum cost of buying a portfolio of options that *weakly dominates* (χ, S, K, T) for some L.

We derive a^* by adding (χ, S, K, T) to the buy side of the market indexed $M+1$, initializing its price b_{M+1} to a large number (i.e., 10^6), and solving for M.1. The best ask a^* is then b_{M+1} minus the returned objective.

In the case of matching orders with multiple units, it is necessary to consider all orders in the market. For quoting prices and deciding the existence of a match, however, we only need to consider a set of orders that have the most competitive prices. We define these orders as a frontier set $\mathcal{F}$.

Definition 4.2 (A Frontier Set of Options Orders). An option order is in the *frontier set* $\mathcal{F}$ if its bid or ask cannot be improved by any other orders in the market. That is, the bid price of a buy order is no less than the maximum gain of selling a weakly dominated portfolio of options for some offset L; the ask price of a sell order is no larger than the minimum cost of buying a weakly dominant portfolio of options for some offset L.

Corollary 4.1.1. Mechanism M.1 determines price quotes and the existence of a match in time polynomial in $|\mathcal{F}|$.

The complete proof for Corollary 4.1.1 is deferred to Appendix B.1.3, which shows that in order to quote the most competitive prices (i.e., the highest bid and the lowest ask) for any target option (χ, S, K, T), it suffices to consider options orders in $\mathcal{F}$. The runtime complexity follows immediately from Theorem 4.1.

4.5 Combinatorial Financial Options

This section proposes *combinatorial financial options*, which extend standard financial options to more general derivative contracts that can be written on any linear combination of U underlying assets. We formally define a combinatorial option and its specifications.

Definition 4.3 (Combinatorial Financial Options). Combinatorial financial options are contracts that specify the right to buy or sell a linear combination of underlying assets at a strike price by an expiration date. Each contract specifies a call or put type $\chi \in \{1, -1\}$, a weight vector $\boldsymbol{\omega} \in \mathbb{R}^U$, a strike price $K \geq 0$, and an expiration date T. It has a payoff of $\max\{\chi(\boldsymbol{\omega}^\top \boldsymbol{S} - K), 0\}$, where $\boldsymbol{S} \in \mathbb{R}^U_{\geq 0}$ is a vector of the underlying assets' values at T.

Consider a combinatorial option $C(\texttt{MSFT} - \texttt{AAPL}, 0)$ that has weight 1 for MSFT, weight -1 for AAPL, and a strike price of zero. An investor who buys the option bets on the event that Microsoft outperforms Apple Inc., and will exercise it if $S_{\texttt{MSFT}} > S_{\texttt{AAPL}}$. Thus, unlike standard options that will pay off due to price changes of a single security, combinatorial options bet on relative movements between assets or groups of assets, thus enabling the expression of future correlations among different underlying assets. We note that the distinction between a call and a put for combinatorial options depends on the strike price and coefficients that one specifies a contract. For instance, in the above example, $C(\texttt{MSFT}-\texttt{AAPL}, 0)$ is identical to $P(\texttt{AAPL}-\texttt{MSFT}, 0)$, as they have the same payoff function $\max\{S_{\texttt{MSFT}} - S_{\texttt{AAPL}}, 0\}$. Despite the different interpretations and expressions, we follow the convention of standard options and have the strike price always be non-negative.

The increased expressiveness in combinatorial options brings new challenges in market design: only matching buy and sell orders on options related to the same assets and weights may yield few or no trades, despite plenty of profitable trades among options written on different portfolios. We start by giving the following motivating examples to illustrate such scenarios.

Example 4.3 (Matching combinatorial option orders)**.** Consider a combinatorial options market with four orders

- o_1: buy one $C(1\texttt{AAPL} + 2\texttt{MSFT}, 300)$ at bid \$110;
- o_2: buy one $C(1\texttt{AAPL} + 1\texttt{MSFT}, 300)$ at bid \$70;
- o_3: sell one $C(1\texttt{AAPL} + 3\texttt{MSFT}, 300)$ at ask \$160;
- o_4: sell one $C(1\texttt{AAPL}, 250)$ at ask \$5.

The exchange returns no match if it only considers options related to the same combination of assets. However, a profitable match does exist. The exchange can sell to o_1 and o_2 and simultaneously buy from o_3 and o_4 to get an immediate gain of \$15 (110+70-160-5). Figure 4.2 plots the overall payoff

$$\begin{aligned}\Psi := &\max\{S_{\texttt{AAPL}} + 3S_{\texttt{MSFT}} - 300, 0\} + \max\{S_{\texttt{AAPL}} - 250, 0\} - \\ &\max\{S_{\texttt{AAPL}} + 2S_{\texttt{MSFT}} - 300, 0\} - \max\{S_{\texttt{AAPL}} + S_{\texttt{MSFT}} - 300, 0\},\end{aligned}$$

as a function of $S_{\texttt{AAPL}}$ and $S_{\texttt{MSFT}}$. The trade cannot subtract from the \$15 immediate gain, but could add to it, depending on the future prices of the two stocks. □

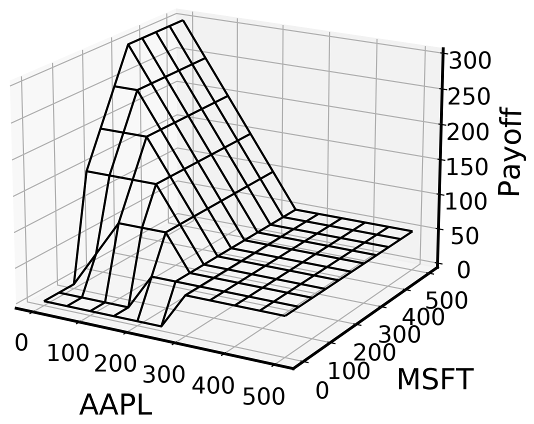


Figure 4.2: Payoff of combinatorial options matched in Example 4.3 as a function of $S_{\texttt{AAPL}}$ and $S_{\texttt{MSFT}}$. The example demonstrates the case of $L = 0$.

In the above example, the exchange can consider matching each individual buy order (selling to o_1 and buying $\frac{2}{3}o_3$ and $\frac{1}{3}o_4$; selling to o_2 and buying $\frac{1}{3}o_3$ and $\frac{2}{3}o_4$). Both are profitable trades, leading to the same match as in the example. The next example shows that matching individual buy order to multiple sell orders may fail to find valid trades.

Example 4.4 (Matching combinatorial option orders)**.** Consider the following four combinatorial options orders

- o_1: buy one $C(\text{A}+\text{B}, 10)$ at bid \$6;
- o_2: buy one $C(\text{B}+\text{C}, 7)$ at bid \$6;
- o_3: sell one $C(\text{A}+\text{B}+\text{C}, 7)$ at ask \$10;
- o_4: sell one $C(\text{B}, 3)$ at ask \$2.

No match will be found if we consider each buy order individually: covering a sold o_1 or o_2 requires buying the same fraction of o_3, which is at a higher price and will incur a net loss. However, a valid match does exist by selling to o_1 and o_2 and buying from o_3 and o_4. It costs the exchange \$0, and can yield a positive payoff in the future: the exchange has

$$\begin{aligned}&\max\{S_A + S_B - 10, 0\} + \max\{S_B + S_C - 7, 0\}\\ &\leq \max\{S_A + S_B + S_C - 7, 0\} + \max\{S_B - 3, 0\},\end{aligned}$$

meaning the liability will always be no larger than the payoffs of bought options, for all non-negative S_A, S_B, S_C. □

We extend the proposed matching mechanism M.1 for standard options to facilitate matching combinatorial options written on different combinations of underlying assets.

4.5.1 Match Orders on Combinatorial Options.

A combinatorial financial options market is a two-sided market with a set of buy orders indexed by $m \in \{1, 2, ..., M\}$ and a set of sell orders indexed by $n \in \{1, 2, ..., N\}$. Buy orders are represented by a type vector $\boldsymbol{\phi} \in \{1, -1\}^M$, a weight matrix $\boldsymbol{\alpha} \in \mathbb{R}^{U \times M}$, a strike vector $\boldsymbol{p} \in \mathbb{R}^M_{\geq 0}$, and a bid price vector $\boldsymbol{b} \in \mathbb{R}^M_+$. Sell orders are defined by a separate type vector $\boldsymbol{\psi} \in \{1, -1\}^N$, a weight matrix $\boldsymbol{\beta} \in \mathbb{R}^{U \times N}$, a strike vector $\boldsymbol{q} \in \mathbb{R}^N_{\geq 0}$, and an ask price vector $\boldsymbol{a} \in \mathbb{R}^N_+$. Similar to a standard options market, the exchange decides the fraction $\boldsymbol{\gamma} \in [0, 1]^M$ to sell to buy orders and the fraction $\boldsymbol{\delta} \in [0, 1]^N$ to buy from sell orders to maximize net profit.

$$\max_{\boldsymbol{\gamma},\boldsymbol{\delta},L} \quad \boldsymbol{b}^\top \boldsymbol{\gamma} - \boldsymbol{a}^\top \boldsymbol{\delta} - L \tag{M.2}$$

$$\text{s.t.} \quad \sum_m \gamma_m \max\{\phi_m(\boldsymbol{\alpha}_m^\top \boldsymbol{S} - p_m), 0\} - \sum_n \delta_n \max\{\psi_n(\boldsymbol{\beta}_n^\top \boldsymbol{S} - q_n), 0\} \leq L \quad \forall \boldsymbol{S} \in \mathbb{R}^U_{\geq 0} \tag{4.2}$$

However, unlike M.1, due to the combinatorial nature, it is no longer feasible to solve the optimization problem M.2 by iterating every combination of breakpoint values, and the number of constraints can grow exponentially as $\mathcal{O}(2^{M+N})$. We analyze the complexity of finding the optimal match in a combinatorial options market, showing that given a market instance, it is NP-complete to decide if a certain matching assignment, $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$, violates the constraint (4.2) for a fixed L. We defer the detailed proof to the Appendix B.2.1, which shows a reduction from the Vertex Cover problem. Using a slightly stronger version of Theorem 4.2, we show that optimal clearing of a combinatorial options market is coNP-hard.

Theorem 4.2. *Consider all combinatorial options in the market* $(\boldsymbol{\phi}, \boldsymbol{\alpha}, \boldsymbol{p}, \boldsymbol{\psi}, \boldsymbol{\beta}, \boldsymbol{q})$. *For any fixed L, it is NP-complete to decide*

- *Yes:* $\boldsymbol{\gamma} = \boldsymbol{\delta} = \mathbf{1}$ *violates the constraint in M.2 for some* $\boldsymbol{S}$,
- *No:* $\boldsymbol{\gamma} = \boldsymbol{\delta} = \mathbf{1}$ *satisfies the constraint in M.2 for all* $\boldsymbol{S}$,

even assuming that each combinatorial option is written on at most two underlying assets.

Theorem 4.3. *Optimal clearing of a combinatorial options market* $(\boldsymbol{\phi}, \boldsymbol{\alpha}, \boldsymbol{p}, \boldsymbol{b}, \boldsymbol{\psi}, \boldsymbol{\beta}, \boldsymbol{q}, \boldsymbol{a})$ *is coNP-hard, even assuming that each combinatorial option is written on at most two underlying assets.*

Since it is no longer practical to solve M.2 by identifying all constraints defined by different combinations of underlying asset values, we propose Algorithm 3 that finds the exact optimal match through iterative constraint generation. At the core of Algorithm 3 is a bilevel optimation formulation: the upper level M.3U is a linear program that computes the optimal solution that satisfies all generated constraints (i.e., realized payoffs w.r.t. different $\boldsymbol{S}$), and the lower level M.3L is a mixed-integer linear program which in each iteration, generates an $\boldsymbol{S}$ that violates the upper-level constraint (i.e., constraint 4.2) the most. The generated $\boldsymbol{S}$ is then included in the constraint set.

The exact optimal match is returned when the lower level M.3L gives an objective value of zero, meaning there exists no $\boldsymbol{S}$ that violates the upper-level constraint. The algorithm trivially terminates finitely, but similar to the simplex method, it has no guarantee on the rate of convergence. We later demonstrate in experiments that Algorithm 3 converges quickly and the number of iterations grows linearly in the size of a problem instance for synthetic options data.

We prove that Algorithm 3 returns the same optimal clearing as M.2, by first claiming in the following Lemma that M.3L finds the $\boldsymbol{S}$ which violates the constraint (4.2) of M.2 the most.

Lemma 4.4. *Given fixed* $\boldsymbol{\gamma}$, $\boldsymbol{\delta}$, *and* L *for a combinatorial options market* $(\boldsymbol{\phi}, \boldsymbol{\alpha}, \boldsymbol{p}, \boldsymbol{b}, \boldsymbol{\psi}, \boldsymbol{\beta}, \boldsymbol{q}, \boldsymbol{a})$, *M.3L returns the value of underlying assets* $\boldsymbol{S}$ *that violates the constraint of M.2 the most.*

Proof. First, it is easy to see that the formulation below returns the $\boldsymbol{S}$ that violates constraint (4.2) the most, since we will have the largest feasible $\boldsymbol{f}$ and the smallest feasible $\boldsymbol{g}$ at the optimum. That is, $f_m = \max\{\phi_m(\boldsymbol{\alpha}_m^\top \boldsymbol{S} - p_m), 0\}$ and $g_n = \max\{\psi_n(\boldsymbol{\beta}_n^\top \boldsymbol{S} - q_n), 0\}$.

$$\begin{aligned}
\max_{\boldsymbol{S},\boldsymbol{f},\boldsymbol{g}} \quad & \boldsymbol{\gamma}^\top \boldsymbol{f} - \boldsymbol{\delta}^\top \boldsymbol{g} - L \\
\text{s.t.} \quad & f_m \le \max\{\phi_m(\boldsymbol{\alpha}_m^\top \boldsymbol{S} - p_m), 0\} && \forall m \in \{1, ..., M\} \\
& g_n \ge \psi_n(\boldsymbol{\beta}_n^\top \boldsymbol{S} - q_n) \\
& g_n \ge 0 && \forall n \in \{1, ..., N\}
\end{aligned}$$

Algorithm 3 Match orders in a combinatorial options market.

```
    Input: A combinatorial options market defined by
        $(\boldsymbol{\phi}, \boldsymbol{\alpha}, \boldsymbol{p}, \boldsymbol{b}, \boldsymbol{\psi}, \boldsymbol{\beta}, \boldsymbol{q}, \boldsymbol{a})$.
    Output: An optimal clearing that matches $\boldsymbol{\gamma}^*$ buy orders to $\boldsymbol{\delta}^*$
        sell orders.
1: Initialize $z \leftarrow \infty$, $\boldsymbol{S} \leftarrow \boldsymbol{0}$, $\boldsymbol{f} \leftarrow \max\{\boldsymbol{\phi}(\boldsymbol{\alpha}^\top \boldsymbol{S} - \boldsymbol{p}), \boldsymbol{0}\}$,
        $\boldsymbol{g} \leftarrow \max\{\boldsymbol{\psi}(\boldsymbol{\beta}^\top \boldsymbol{S} - \boldsymbol{q}), \boldsymbol{0}\}$, $\mathcal{C} \leftarrow \{(\boldsymbol{f}, \boldsymbol{g})\}$.
2: while $z > 0$ do
3:     Solve the following upper level optimization problem and
       get an optimal solution $(\boldsymbol{\gamma}^*, \boldsymbol{\delta}^*, L^*)$
```

$$\begin{aligned} \max_{\boldsymbol{\gamma},\boldsymbol{\delta},L} \quad & \boldsymbol{b}^\top \boldsymbol{\gamma} - \boldsymbol{a}^\top \boldsymbol{\delta} - L && \text{(M.3U)} \\ \text{s.t.} \quad & \boldsymbol{\gamma}^\top \boldsymbol{f} - \boldsymbol{\delta}^\top \boldsymbol{g} \le L && \forall (\boldsymbol{f}, \boldsymbol{g}) \in \mathcal{C} \end{aligned}$$

```
4:     Given $(\boldsymbol{\gamma}^*, \boldsymbol{\delta}^*, L^*)$, solve the following lower level MILP
       and get an optimal solution $(\boldsymbol{S}^*, \boldsymbol{f}^*, \boldsymbol{g}^*, z^*)$
```

$$\begin{aligned} \max_{\boldsymbol{S},\boldsymbol{f},\boldsymbol{g},\boldsymbol{I}} \quad & z := \boldsymbol{\gamma}^\top \boldsymbol{f} - \boldsymbol{\delta}^\top \boldsymbol{g} - L && \text{(M.3L)} \\ \text{s.t.} \quad & \phi_m(\boldsymbol{\alpha}_m^\top \boldsymbol{S} - p_m) \ge \mathcal{M}(\mathcal{I}_m - 1) \\ & \phi_m(\boldsymbol{\alpha}_m^\top \boldsymbol{S} - p_m) \le \mathcal{M}\mathcal{I}_m \\ & f_m \le \phi_m(\boldsymbol{\alpha}_m^\top \boldsymbol{S} - p_m) - \mathcal{M}(\mathcal{I}_m - 1) \\ & f_m \le \mathcal{M}\mathcal{I}_m \\ & \mathcal{I}_m \in \{0, 1\} && \forall m \in \{1, ..., M\} \\ & g_n \ge \psi_n(\boldsymbol{\beta}_n^\top \boldsymbol{S} - q_n) \\ & g_n \ge 0 && \forall n \in \{1, ..., N\} \end{aligned}$$

```
5:     $\mathcal{C} \leftarrow \mathcal{C} \cup (\boldsymbol{f}^*, \boldsymbol{g}^*)$, $z \leftarrow z^*$
6: return $\boldsymbol{\gamma}^*$ and $\boldsymbol{\delta}^*$
```

It remains to show the set of constraints related to any buy order m in M.3L is equivalent to $f_m \le \max\{\phi_m(\boldsymbol{\alpha}_m^\top \boldsymbol{S} - p_m), 0\}$. This set of constraints implements the big-$\mathcal{M}$ trick (where $\mathcal{M}$ is a large constant, say 10^6) on a binary decision variable $\mathcal{I}_m$ to linearize the max function. Consider each case of $\mathcal{I}_m \in \{0, 1\}$. We have $\phi_m(\boldsymbol{\alpha}_m^\top \boldsymbol{S} - p_m) \ge 0 \iff \mathcal{I}_m = 1$ and $f_m = \phi_m(\boldsymbol{\alpha}_m^\top \boldsymbol{S} - p_m) \iff \mathcal{I}_m = 1$. □

Therefore, when M.3L returns an objective value of zero, the constraint 4.2 is satisfied for all $\boldsymbol{S}$, and Algorithm 3 returns a valid match that optimizes for overall profit.

Theorem 4.5. *Given a combinatorial options market instance* $(\boldsymbol{\phi}, \boldsymbol{\alpha}, \boldsymbol{p}, \boldsymbol{b}, \boldsymbol{\psi}, \boldsymbol{\beta}, \boldsymbol{q}, \boldsymbol{a})$, *Algorithm 3 returns the optimal clearing defined in M.2.*

4.6 Experiments: OptionMetrics Data

We first show on real options data that our mechanism finds matches that the current independent-market design cannot and provides more competitive bid and ask prices.

We conduct empirical analyses on the OptionMetrics dataset provided by the Wharton Research Data Services (WRDS), which contains real-market option prices (i.e., the best bid and ask) for each options market defined by an underlying asset, an option type, a strike price, and an expiration date.[4] We choose options data on 30 stocks that compose the DJI, as these stocks have actively traded options that cover a wide range of strike values. There are a total of 25,502 distinct options markets for stocks in DJI on January 23, 2019, yielding an average of 850 separate markets for each security. The offered options cover around 12 expiration dates for each stock, and thus about 70 markets that have different combinations of types and strikes within the same security and expiration date.

We use the proposed mechanism M.1 to consolidate options markets, reducing the original 25,502 markets to a total of 366 markets, each associated with one underlying security and expiration. We are interested in matching orders (i.e., finding arbitrage opportunities) that fail to transact under the independent market design, computing new option quotes implied by our consolidated arbitrage-free markets, and comparing the case of restricting L to 0 to the case of having L as a decision variable. Detailed statistics for options of each stock are available in the Appendix B.3.

Out of the 366 consolidated options markets, we spot arbitrage opportunities in 150 markets, among which 94 cases make profits at contract transaction (e.g., Example 4.1) and the remaining 56 incur expenses for higher payoffs upon option expiration (e.g., Example 4.2). Matches with non-negative L make an average profit of \$1.03, with a maximum of \$9.64, and matches with negative L imply an average interest rate of 0.7%, with a maximum at 2.02%. When restrict L to 0, we are able to find arbitrage in 74 markets.

The remaining 216 markets are arbitrage-free (for the flexible L case), which capture a total of 16,088 option series and 32,176 orders. We find that approximately 49% of the orders belong to the frontier set. Using these orders to derive the most competitive bids and asks, we find that the bid-ask spreads can be reduced by 73%, from an average of 80 cents for each option series in the independent markets to 21

[4]Our data includes American options that allow exercise before expiration. In practice, American options are almost always more profitable to sell than to exercise early (Singh 2019). In experiments, we ignore early exercise and treat them as European options.

cents in consolidated options markets. For the case of L set to 0, the bid-ask spreads can be reduced by 52%. These results show that aggregating independently-traded options leads to a more efficient market, with tightened bid-ask spreads and matches of options across types and strikes.

4.7 Experiments: Synthetic Combinatorial Options Market

Since there is no combinatorial option traded in financial markets, we evaluate the proposed algorithm on synthetic combinatorial options, with prices calibrated using real-market standard options written on each related underlying security.[5] We are interested in quantifying the performance of Algorithm 3 in parametrically different markets that vary in the likelihood of matching, the number of orders, and the number of underlying assets.

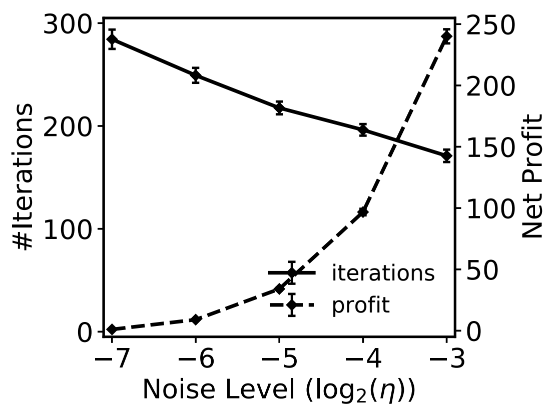
4.7.1 Generate Synthetic Orders

We generate combinatorial options markets of U underlying assets. Each combinatorial option is written on a combination of two stocks, S_i and S_j, randomly selected from the U underlying assets. This gives a total number of $\binom{U}{2}$ asset pairs. Weights for the selected assets, w_i and w_j, are picked uniformly randomly from $\{\pm 1, \pm 2, \ldots, \pm 9\}$ and are processed to be relatively prime.

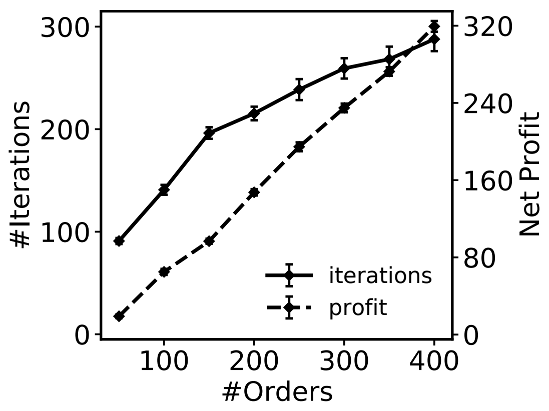
We generate strikes and premium prices using real-market standard options data related to each individual asset to realistically capture the value of the synthetic portfolio. Let $\mathcal{K}_i$ and $\mathcal{K}_j$ respectively denote the set of strike prices offered by standard options on each selected asset. We generate the strike K by first sampling two strike values, $k_i \sim \mathcal{K}_i$ and $k_j \sim \mathcal{K}_j$, and scaling them by the associated weights to get $K = w_i k_i + w_j k_j$. If K is positive, we have a call option. Instead, if K is negative, we generate a put option and update the strike to $-K$ and weights to $-w_i$ and $-w_j$ to comply with the representation and facilitate payoff computations.

We randomly assign each option to the buy or sell side of the market, and generate a bid or an ask price accordingly. Similar to the price quote procedure in Section 4.4, we derive the bid b by calculating the maximum gain of selling a set of standard options whose payoff is dominated by the combinatorial option of interest and the ask a by calculating the minimum cost of buying a set of standard options whose payoff dominates the generated option. We add noises to the derived prices to control the

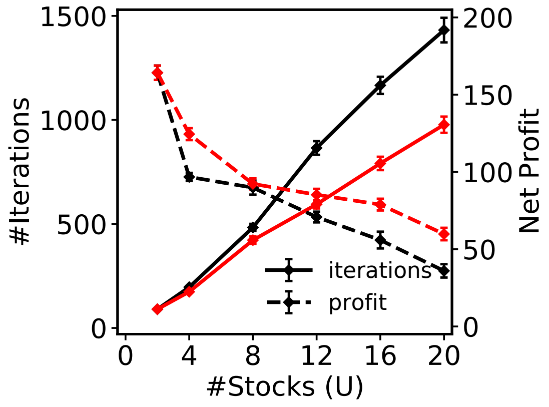
[5]We adopt the same dataset as Section 4.6, and use standard options that expire on Febuary 1, 2019, to calibrate order prices for generated combinatorial options.



(a) Vary noise η added to order prices in markets with $U = 4$ and $n_{\text{orders}} = 150$.



(b) Vary number of orders n_{orders} in markets with $U = 4$ and $\eta = 2^{-4}$.



(c) Vary size of underlying assets U in markets with $n_{\text{orders}} = 150$ and $\eta = 2^{-4}$.

Figure 4.3: Results of using Mechanism M.2 to match orders in synthetic combinatorial options markets. The number of generated constraints (solid lines) and the net profits (dashed line), as the markets vary in price noise, the number of orders, and the size of underlying assets. Red lines represent markets that offer a restrictive set of asset pairs, which covers all U underlying assets.

likelihood of matching in a market, and set final prices to $b(1+\zeta)$ or $a(1-\zeta)$, where $\zeta \sim [0, \eta]$ and η is a noise parameter.

4.7.2 Evaluation

We explore a range of markets that vary in price noise, the number of orders, and the number of underlying assets. For each market, we measure the number of iterations (i.e., the number of constraints generated) that Algorithm 3 runs to find an exact optimal clearing and the net profit made from the trade. For all experiments, we show results averaged over 40 simulated markets, with the error bars denoted one standard error around the means.

We first validate that as noises added to the derived bids and asks increase, the likelihood of matching in our simulated combinatorial options market becomes higher. We generate markets with four underlying assets (arbitrarily selected from the 30 stocks in DJI) and 150 synthetic combinatorial options orders, and vary the noise $\eta \in \{2^{-7}, 2^{-6}, 2^{-5}, 2^{-4}, 2^{-3}\}$. Figure 4.3(a) plots the averaged results. As expected, the net profit made from the optimal match increases, as η increases. Moreover, we find that as the matching probability increases, the number of iterations that Algorithm 3 takes to find the optimal solution consistently decreases. This makes sense as intuitively, in thin markets where few trades are likely to occur, the lower-level MILP will keep coming up with $\boldsymbol{S}$ values to refute a large number of matching proposals until convergence.

Figure 4.3(b) further quantifies the change in iteration numbers and net profits, as we vary the number of combinatorial options orders. We fix these markets to have four underlying assets with a price noise of 2^{-4}. As we see from Figure 4.3(b), as a market aggregates more orders, transactions are more likely to happen, leading to larger net profits. We also find that the number of generated constraints grows (sub)linear in the number of orders. Since different $\boldsymbol{S}$ values are generated to define payoffs of *distinct* options and the number of distinct options increases sublinear in the number of total orders, the rate of increase in the number of iterations tends to decrease as a market aggregates more orders.

Finally, we evaluate how Algorithm 3 scales to markets with increasingly larger numbers of underlying assets. In this case, as the dimension of $\boldsymbol{S}$ becomes large, the number of asset-value combinations grows exponentially. Figure 4.3(c) (black lines) demonstrates a much faster increase in the number of iterations and a steady decrease in the net profit.[6] It suggests that as the market provides a large set of underlying assets (e.g., all 30 stocks in DJI), the thin market problem may still arise even when the mechanism facilitates matching options written on different combinations of underlying assets. Here, we make the assumption that every asset pair in the $\binom{U}{2}$ is equally likely to be traded. In real markets, investors may be more interested in certain asset pairs, trading them more frequently than the others. Based on such observations, a market can specify a prescriptive set of asset pairs, $\mathcal{P}$, which covers the U assets, for

[6]We report average runtimes to quantify the impact of increasing constraints. The average times (in seconds) that Algorithm 3 computes the optimal match are 20, 153, 233, 297, 344, and 358 for the respective markets with $U \in \{2, 4, 8, 12, 16, 20\}$.

traders to choose from and specify custom weights. For the experiments, we choose $|\mathcal{P}| = U$. Figure 4.3(c) (red lines) shows that such prescriptive design may indeed attenuate the thin market problem.

4.8 Discussion

When related financial markets run independently, traders remove arbitrage and close bid-ask spreads themselves. Our OptionMetrics experiments show that they do so suboptimally. Profits can flow to agents with computational power and no information. Our design instead rewards informed agents only and reduces the arms race among traders, by putting computational power into the exchange.

This chapter has examined a fully expressive combinatorial options market that allows all linear combinations of assets. One next step is to explore naturally structured markets where combinations are limited to components in a graph of underlying assets. One special case is a hierarchical graph (Guo and Pennock 2009), for example, the S&P 500, sectors like travel and technology, subsectors like airlines and internet within those, etc.

CHAPTER 5

Log-time Prediction Markets for Interval Securities

This chapter studies the design of a prediction market to recover a complete and fully general probability distribution over a random variable. Traders bet on outcomes by buying and selling *interval securities* that pay \$1 if the outcome falls into an interval and \$0 otherwise. The market takes the form of a central *automated market maker* and allows traders to express interval endpoints of arbitrary precision.

This chapter presents two designs in both of which market operations take time logarithmic in the number of intervals (that traders distinguish), providing the first computationally efficient market for a continuous variable. The first design replicates the popular *logarithmic market scoring rule* (LMSR), but operates exponentially faster than a standard LMSR by exploiting its modularity properties to construct a balanced binary tree and decompose computations along the tree nodes. The second design features two or more parallel LMSR market makers that mediate submarkets of increasingly fine-grained outcome partitions. This design remains computationally efficient for all operations, including arbitrage removal across submarkets. It adds two additional benefits for the *market designer*: (1) the ability to express utility for information at various resolutions by assigning different liquidity values, and (2) the ability to guarantee a true constant bounded loss by geometrically decreasing the liquidity in each submarket.

5.1 Introduction

Consider a one-dimensional random variable, such as the opening value of the S&P 500 index on December 17, 2021. We design a market for trading *interval securities* corresponding to predictions that the outcome will fall into some specified interval, say between 2957.60 and 3804.59, implemented as binary contracts that pay out \$1 if the outcome falls in the interval and \$0 otherwise. We are interested in designing *automated market makers* to facilitate a fully expressive market computationally efficiently. Traders can select custom interval endpoints of arbitrary precision corresponding to a continuous outcome space, whereas the market maker will always offer to buy or sell any interval security at some price.

A form of interval security called the *condor spread* is common in financial options markets, with significant volume of trade. Each condor spread involves trading four different options,[1] and financial options offered by the market may only support a limited subset of approximate intervals. For example, as of this writing, S&P 500 options expiring on December 17, 2021, distinguish 56 strike prices, allowing the purchase of around 1500 distinct intervals of minimum width 25. Moreover, as discussed in Chapter 4, each strike price trades independently despite the logical constraints on their relative values, and thus it will require time linear in the number of offered strike prices to remove arbitrage.

Outside traditional financial markets, the *logarithmic market scoring rule* (LMSR) market maker (Hanson 2003, 2007) has been used to elicit information through the trade of interval securities. The Gates Hillman Prediction Market at Carnegie Mellon University operated LMSR on 365 outcomes, representing 365 days of one year, to forecast the opening time of the new computer science building (Othman and Sandholm 2010). Traders could bet on different intervals by choosing a start and an end date. A similar market[2] was later launched at the University of Texas at Austin, using a liquidity-sensitive variation of LMSR (Othman, Pennock, Reeves, and Sandholm 2013). Moreover, LMSR has been deployed to predict product-sales levels (Plott and Chen 2002), instructor ratings (Chakraborty et al. 2013), and political events (Hanson 1999).

LMSR has two limitations that prevent its scaling to markets with a continuous outcome space. First, LMSR's worst-case loss can grow unbounded if traders select intervals with prior probability approaching zero (Gao, Chen, and Pennock 2009). Second, standard implementations of LMSR operations run in time linear in the number of outcomes or distinct future values traders define—in our case, arbitrarily many. The constant-log-utility and other barrier-function-based market makers (Chen and Pennock 2007; Othman and Sandholm 2012) feature constant bounded loss,

[1]A *call option* written on an underlying stock with *strike price* K and expiration date T pays $\max\{S-K,0\}$, where S is the opening value of the stock on T. For example, 25 shares of "\$1 iff [2650,2775]" $\approx$ $\max\{S-2650,0\}-\max\{S-2675,0\}-\max\{S-2750,0\}+\max\{S-2775,0\}$.

[2]`www.cs.utexas.edu/news/2012/research-corner-gates-building-prediction-market`

but still suffer the second limitation regarding computational intractability. Thus, previous markets feature a relatively small set of predetermined intervals and run in time linear in the number of supported outcomes, limiting the ability to aggregate high-precision trades and elicit the full distribution of a continuous random variable.

In this chapter, we propose two automated market makers that perform exponentially faster than the standard LMSR and previous designs. Market operations (i.e., price, cost, and buy) can be executed in time *logarithmic* in the number of distinct intervals traded, or linear in the number of bits describing the outcome space. The first market maker calculates LMSR exactly, but employs a balanced binary tree to implement interval queries and trades. We show that the normalization constant of LMSR—a key quantity in its price and cost function—can be calculated recursively via local computations on the balanced tree. The work here contributes to the rich literature that aims to overcome the worst-case #P-hardness of LMSR pricing (Chen et al. 2008) by exploiting the outcome space structure and limiting expressivity (Chen, Fortnow, Nikolova, and Pennock 2007; Chen, Goel, and Pennock 2008; Guo and Pennock 2009; Laskey et al. 2018; Xia and Pennock 2011).

The second market maker works by maintaining parallel LMSR submarkets that adopt different liquidity parameters and offer interval securities at various resolutions. We show that liquidity parameters can be chosen to guarantee a *constant* bounded loss independent of market precision, and prices can be kept coherent efficiently by removing arbitrages across submarkets. We demonstrate through agent-based simulation that our second design enjoys more flexible liquidity choices to facilitate the information-gathering objective: it can get close to the "best of both worlds" displayed by coarse and fine LMSR markets, with prices converging fast at both resolutions regardless of the traders' information structure.

The two proposed designs, to our knowledge, are the first to simultaneously achieve expressiveness and computational efficiency. As both market makers facilitate trading intervals at arbitrary precision, they can elicit any probability distribution over a continuous random variable that can be practically encoded by a machine. Throughout this chapter, we use the S&P 500 index value as a running example, but the framework is generic and can handle any one-dimensional discrete or continuous variable, for example, the number of coronavirus infections by the end of the year, the date when a vaccine will be released, the landfall point of a hurricane along a coastline, or the number of tickets sold in the first week of a new movie release.

5.2 Formal Setting

This section first reviews cost-function-based market making (Abernethy, Chen, and Vaughan 2011; Chen and Pennock 2007), and then introduces interval markets.

5.2.1 Cost-Function-Based Market Making

Let Ω denote a finite set of *outcomes*, corresponding to mutually exclusive and exhaustive states of the world. We are interested in eliciting expectations of binary random variables $\phi_i \colon \Omega \to \{0, 1\}$, indexed by $i \in \mathcal{I}$, which model the occurrence of various events, such as "*S&P 500 will open between 2957.60 and 3804.59 on December 17, 2021*". Each variable ϕ_i is associated with a *security* that pays out $\phi_i(\omega)$ when the outcome $\omega \in \Omega$ occurs, and thus ϕ_i is also called the *payoff function.* Binary securities pay out \$1 if the specified event occurs and \$0 otherwise. The vector $(\phi_i)_{i\in\mathcal{I}}$ is denoted $\boldsymbol{\phi}$. Traders trade *bundles* $\boldsymbol{\delta} \in \mathbb{R}^{|\mathcal{I}|}$ of security with a central market maker, where positive entries in $\boldsymbol{\delta}$ correspond to purchases and negative entries short sales. A trader holding a bundle $\boldsymbol{\delta}$ receives a payoff of $\boldsymbol{\delta} \cdot \boldsymbol{\phi}(\omega)$, when ω occurs.

Following Abernethy, Chen, and Vaughan (2011) and Chen and Pennock (2007), we assume that the market maker determines security prices using a convex and differentiable potential function $C \colon \mathbb{R}^{|\mathcal{I}|} \to \mathbb{R}$, called a *cost function.* The state of the market is specified by a vector $\boldsymbol{\theta} \in \mathbb{R}^{|\mathcal{I}|}$, listing the number of shares of each security *sold* by the market maker so far. A trader who wants to buy a bundle $\boldsymbol{\delta}$ in the market state $\boldsymbol{\theta}$ must pay $C(\boldsymbol{\theta} + \boldsymbol{\delta}) - C(\boldsymbol{\theta})$ to the market maker, after which the new state becomes $\boldsymbol{\theta} + \boldsymbol{\delta}$.

The vector of instantaneous prices in the corresponding state $\boldsymbol{\theta}$ is $\boldsymbol{p}(\boldsymbol{\theta}) := \nabla C(\boldsymbol{\theta})$. Its entries can be interpreted as the market's collective estimates of $\mathbb{E}[\phi_i]$: a trader can make an expected profit by buying (at least a small amount of) the security i if she believes that $\mathbb{E}[\phi_i]$ is larger than the instantaneous price $p_i(\boldsymbol{\theta}) = \partial C(\boldsymbol{\theta})/\partial\theta_i$, and by selling if she believes the opposite. Therefore, risk neutral traders with sufficient budgets maximize their expected profits by moving the price vector to match their expectation of $\boldsymbol{\phi}$. Any expected payoff must lie in the convex hull of the set $\{\boldsymbol{\phi}(\omega)\}_{\omega\in\Omega}$, which we denote $\mathcal{M}$ and call a *coherent price space* with its elements referred to as *coherent price vectors.*

We assume that the cost function satisfies two standard properties: *no arbitrage* and *bounded loss.* The *no-arbitrage* property requires that as long as all outcomes ω are possible, there be no market transaction with a guaranteed profit for a trader. In this study, we use the fact that C is arbitrage-free if and only if it yields price vectors $\boldsymbol{p}(\boldsymbol{\theta})$ that are always coherent (Abernethy, Chen, and Vaughan 2011). The *bounded-loss* property is defined in terms of the worst-case loss of a market maker, $\sup_{\boldsymbol{\theta}\in\mathbb{R}^{|\mathcal{I}|}}\big(\sup_{\omega\in\Omega}\big(\boldsymbol{\theta}\cdot\boldsymbol{\phi}(\omega)\big) - C(\boldsymbol{\theta}) + C(\mathbf{0})\big)$, meaning the largest difference, across all possible trading sequences and outcomes, between the amount that the market maker has to pay the traders (once the outcome is realized) and the amount that the market maker has collected (when securities were traded). The property requires that this worst-case loss be a priori bounded by a constant.

5.2.2 Complete Markets and LMSR

In a complete market, we have $\mathcal{I} = \Omega$. Securities are indicators of individual outcomes, $\phi_i(\omega) = 1\{\omega = i\}$, where $1\{\cdot\}$ denotes the binary indicator. We denote each market security as ϕ_ω. A risk-neutral trader is incentivized to move the price of each security ϕ_ω to her estimate of $\mathbb{E}[\phi_\omega] = \mathbb{P}[\omega]$, which is her subjective probability of ω occurring. Thus, traders can express arbitrary probability distributions over Ω.

This chapter considers variants of LMSR (Hanson 2003) for a complete market. It has the cost function and prices of the form

$$C(\boldsymbol{\theta}) = b\log\left(\sum_{\omega\in\Omega} e^{\theta_\omega/b}\right), \quad p_\omega(\boldsymbol{\theta}) = \partial C(\boldsymbol{\theta})/\partial\theta_\omega = \frac{e^{\theta_\omega/b}}{\sum_{\nu\in\Omega} e^{\theta_\nu/b}}, \tag{5.1}$$

where b is the liquidity parameter, controlling how fast the price moves in response to trading and the worst-case loss of the market maker, which equals $b\log|\Omega|$ (Hanson 2003).

The securities in a complete market can be used to express bets on any event E. Specifically, one share of a security for the event E can be represented by the indicator bundle $\mathbf{1}_E \in \mathbb{R}^\Omega$ with entries $1_{E,\omega} = 1\{\omega \in E\}$. We refer to this bundle as the *bundle security for event E*. The immediate price of the bundle $\mathbf{1}_E$ in the state $\boldsymbol{\theta}$ is

$$p_E(\boldsymbol{\theta}) := \mathbf{1}_E\cdot\boldsymbol{p}(\boldsymbol{\theta}) = \sum_{\omega\in E} p_\omega(\boldsymbol{\theta}) = \frac{\sum_{\omega\in E} e^{\theta_\omega/b}}{\sum_{\nu\in\Omega} e^{\theta_\nu/b}}. \tag{5.2}$$

The cost of buying the bundle $s\mathbf{1}_E$, or sometimes referred to as “the cost of s shares of $\mathbf{1}_E$", is a function of $p_E(\boldsymbol{\theta})$ and s

$$\begin{aligned} & C(\boldsymbol{\theta} + s\mathbf{1}_E) - C(\boldsymbol{\theta}) \qquad (5.3) \\ &= b \log \left(\sum_{\omega \notin E} e^{\theta_\omega / b} + \sum_{\omega \in E} e^{(\theta_\omega + s)/b} \right) - b \log \left(\sum_{\omega \in \Omega} e^{\theta_\omega / b} \right) \\ &= b \log \left(p_{E^c}(\boldsymbol{\theta}) + e^{s/b} p_E(\boldsymbol{\theta}) \right) = b \log \left(1 - p_E(\boldsymbol{\theta}) + e^{s/b} p_E(\boldsymbol{\theta}) \right). \end{aligned}$$

We write E^c for the complementary event $E^c = \Omega \backslash E$, and use the fact $p_E(\boldsymbol{\theta}) + p_{E^c}(\boldsymbol{\theta}) = 1$, which follows from Eq. (5.2).

5.2.3 Interval Securities over $[0, 1)$

This chapter considers betting on outcomes within an interval $[0, 1)$. Our approach generalizes to outcomes that are in any $[\alpha, \beta) \subseteq [-\infty, \infty)$ by applying any increasing transformation $F : [\alpha, \beta) \to [0, 1)$. We assume that the outcome ω is specified with K bits, meaning that there are $N = 2^K$ outcomes with $\Omega = \{j/N : j \in \{0, 1, \ldots, N-1\}\}$. Sections 5.3 and 5.4 will discuss how the assumption of pre-specified bit precision can be removed.

Example 5.1 (Complete market for S&P 500)**.** Consider a complete market for the S&P 500 opening price on December 17, 2021, by setting $N = 2^{19} = 524{,}288$. The resulting complete market is $\mathcal{I} = \{0,\ 0.01,\ \ldots,\ 5242.86,\ 5242.87\}$, where we cap prices at \$5242.87 (i.e., larger prices are treated as \$5242.87). The transformed outcome is then $\omega = \omega'/N$, where ω' is the $S\&P\,500$ price in cents.

In the outcome space Ω, we would like to enable price and cost queries as well as buying and selling of *bundle securities* for the interval events $I = [\alpha, \beta)$ for any $\alpha, \beta \in \Omega \cup \{1\}$.[3] For cost-based markets, sell transactions are equivalent to buying a negative amount of shares, so we design algorithms for three operations: $\mathbf{price}(I)$, $\mathbf{cost}(I, s)$, and $\mathbf{buy}(I, s)$, where I is the interval event and s the number of shares. A naive implementation of **price** and **cost** following Eqs. (5.2) and (5.3) would be *linear* in N. In this chapter, we propose to implement these operations in time that is *logarithmic* in N.

[3]Throughout this study, we operate in the outcome space discretized to events $\omega \in \Omega$ specified with K bits, but would like to indeed discuss interval events $[a, b)$ that include reals of arbitrary precisions. Since, in measure theory, events are subsets of the outcome space, what we mean here are events of form $[a, b) \cap \Omega$.

5.3 A Log-time LMSR Market Maker

This section designs a data structure, referred to as an *LMSR tree*, which resembles an *interval tree* (Cormen, Leiserson, and Rivest 1999, Section 15.3), but includes additional annotations to support LMSR calculations. We first define the LMSR tree, and show that it can facilitate market operations in time logarithmic in the number of distinct intervals that traders define.

5.3.1 An LMSR Tree for $[0,1)$

An LMSR tree T is represented by a *full binary tree*, where each node z has either no children (when z is a leaf) or exactly two children, denoted $\mathit{left}(z)$ and $\mathit{right}(z)$ (when z is an inner node).

Definition 5.1 (LMSR Tree)**.** An *LMSR tree* is a full binary tree, where each node z is annotated with an interval $I_z = [\alpha_z, \beta_z)$ with $\alpha_z, \beta_z \in \Omega \cup \{1\}$, a height $h_z \geq 0$, a quantity $s_z \in \mathbb{R}$ that records the number of bundle securities sold associated with I_z, and a partial normalization constant $S_z \geq 0$. An LMSR tree satisfies

- *Binary-search property*: $I_{\mathit{root}} = [0,1)$, and for inner node z,

$$\alpha_z = \alpha_{\mathit{left}(z)} \;<\; \beta_{\mathit{left}(z)} = \alpha_{\mathit{right}(z)} \;<\; \beta_{\mathit{right}(z)} = \beta_z.$$

- *Height balance*: $h_z = 0$ for leaves, and for inner node z,

$$h_z = 1 + \max\{h_{\mathit{left}(z)}, h_{\mathit{right}(z)}\}, \quad |h_{\mathit{left}(z)} - h_{\mathit{right}(z)}| \leq 1.$$

- *Partial-normalization correctness*: $S_z = e^{s_z/b} \cdot (\beta_z - \alpha_z)$ for leaves, and for inner node z,

$$S_z = e^{s_z/b} \cdot \left(S_{\mathit{left}(z)} + S_{\mathit{right}(z)}\right).$$

The *binary-search property* helps to find the unique leaf that contains any $\omega \in \Omega$ by descending from *root* and choosing left or right in each node based on whether $\omega < \beta_{\mathit{left}(z)}$ or $\omega \geq \beta_{\mathit{left}(z)}$. The node heights serve to maintain the *height-balance property*, ensuring that the path length from root to any leaf is at most $\mathcal{O}(\log n)$ where n is the number of leaves of the tree (Knuth 1998). We adopt an *AVL tree* (Adel′son-Vel′skiĭ and Landis 1962) at the basis of our LMSR tree, but other balanced binary-search trees (e.g., red-black trees or splay trees) could also be used.

To facilitate LMSR computations, we maintain a scalar quantity $s_z \in \mathbb{R}$ for each node z, which records the number of *bundle securities* associated with I_z sold by the market maker. Therefore, the market state and its components for each individual outcome ω represented by the LMSR tree T are:[4]

$$\boldsymbol{\theta}(T) = \sum_{z \in T} s_z \mathbf{1}_{I_z}; \quad \theta_\omega(T) = \sum_{z \in T} s_z 1_{I_z,\omega} = \sum_{z \ni \omega} s_z. \tag{5.4}$$

The normalization constant in the LMSR price (Eq. 5.2) is then

$$\sum_{\omega \in \Omega} e^{\theta_\omega / b} = \sum_{\omega \in \Omega} e^{\sum_{\omega \in z} s_z / b} = \sum_{\omega \in \Omega} \prod_{z \ni \omega} e^{s_z / b}. \tag{5.5}$$

We decompose the computation of the above normalization constant along the nodes of an LMSR tree, by defining a *partial normalization constant* S_z in each node:[5]

$$S_z := \frac{1}{N} \sum_{\omega \in z} \prod_{z' \colon z \supseteq z' \ni \omega} e^{s_{z'} / b}. \tag{5.6}$$

It enables the following recursive relationship, which we refer to as *partial-normalization correctness*—a key property that is at the core of implementing **price** and **buy**:

$$S_z = \begin{cases} e^{s_z/b} \cdot (\beta_z - \alpha_z) & \text{if } z \text{ is a leaf,} \\ e^{s_z/b} \cdot \left(S_{\textit{left}(z)} + S_{\textit{right}(z)} \right) & \text{otherwise.} \end{cases} \tag{5.7}$$

Based on the LMSR tree construction, we implement the following operations for any interval $I = [\alpha, \beta)$:

- **price**(I, T) returns the price of bundle security for I;
- **cost**(I, s, T) returns the cost of s shares of bundle security for I;
- **buy**(I, s, T) updates T to reflect the purchase of s shares of bundle security for I.

In order to implement **cost**, it suffices to implement **price** by Eq. (5.3). Since the price of $[\alpha, \beta)$ can be obtained from prices for $[\alpha, 1)$ and $[\beta, 1)$, i.e., $p_{[\alpha,\beta)}(\boldsymbol{\theta}) = p_{[\alpha,1)}(\boldsymbol{\theta}) - p_{[\beta,1)}(\boldsymbol{\theta})$, we implement **price** for intervals of the form $[\alpha, 1)$. Similarly, buying s shares of $[\alpha, \beta)$ is equivalent to first buying s shares of $[\alpha, 1)$ and then buying $(-s)$ shares of $[\beta, 1)$, as the market ends up in the same state $\boldsymbol{\theta} + s\mathbf{1}_{[\alpha,\beta)}$. We implement

[4]For simplicity, we write $\omega \in z$ to mean $\omega \in I_z$ and $z' \subseteq z$ to mean $I_{z'} \subseteq I_z$. Thus, $z' \subseteq z$ corresponds to z' being a descendant of z in T.

[5]The $1/N$ scale in Eq. (5.6) leads to a natural interpretation of S_z, when z is a leaf.

price and **buy** for one-sided intervals $I = [\alpha, 1)$, and the remaining operations will follow.

5.3.2 Price Queries

Consider price queries for $I = [\alpha, 1)$. Let $\mathit{vals}(T) = \{\alpha_z : z \in T\}$ denote the set of distinct left endpoints in the tree nodes. We start by assuming that $\alpha \in \mathit{vals}(T)$, and later relax this assumption. We proceed to calculate $p_I(\boldsymbol{\theta})$ in two steps. *First*, we construct a set of nodes $\mathcal{Z}$ whose associated intervals I_z are disjoint and cover I. To achieve this, we conduct a binary search for α, putting in $\mathcal{Z}$ all of the right children of the visited nodes that have $\alpha_z > \alpha$, as well as the final node with $\alpha_z = \alpha$. Recall that n is the number of leaves of the LMSR tree, and the height balance implies that $\mathcal{Z}$ has a cardinality of $\mathcal{O}(\log n)$. The resulting set $\mathcal{Z}$ satisfies $p_I(\boldsymbol{\theta}) = \sum_{z \in \mathcal{Z}} p_{I_z}(\boldsymbol{\theta})$.

Second, we determine $p_{I_z}(\boldsymbol{\theta})$ for each node $z \in \mathcal{Z}$. Starting from the LMSR price in Eq. (5.2), we take advantage of the defined partial normalization constants S_z to calculate $p_{I_z}(\boldsymbol{\theta})$:

$$p_{I_z}(\boldsymbol{\theta}) = \frac{1}{NS_{root}} \sum_{\omega \in z} e^{\theta_\omega / b} = \frac{1}{S_{root}} \cdot \frac{1}{N} \sum_{\omega \in z} \prod_{z' \ni \omega} e^{s_{z'}/b} \tag{5.8}$$

$$= \frac{1}{S_{root}} \cdot \frac{1}{N} \sum_{\omega \in z} \left[\left(\prod_{z' :\, z \supseteq z' \ni \omega} e^{s_{z'}/b} \right) \left(\prod_{z' \supset z} e^{s_{z'}/b} \right) \right] \tag{5.9}$$

$$= \frac{S_z}{S_{root}} \underbrace{\left(\prod_{z' \supset z} e^{s_{z'}/b} \right)}_{P_z} . \tag{5.10}$$

In Eq. (5.8), we expand θ_ω using Eq. (5.4). In Eq. (5.9), we use the fact that any node z' with a non-empty intersection with z (i.e., $I_z \cap I_{z'} \neq \emptyset$) must be either a descendant or an ancestor of z as a direct consequence of the binary-search property. The product P_z in Eq. (5.10) iterates over z' on the path from root to z, and thus can be calculated along the binary-search path.

We now handle the case when $\alpha \notin \mathit{vals}(T)$. After the leaf z on the search path is reached, we have $\alpha_z < \alpha < \beta_z$. Instead of expanding the tree, we conceptually create two children of z: z' and z'' with $I_{z'} = [\alpha_z, \alpha)$ and $I_{z''} = [\alpha, \beta_z)$, and add z'' in $\mathcal{Z}$. Since θ_ω is constant across $\omega \in I_z$, we obtain $p_{I_{z''}}(\boldsymbol{\theta}) = \frac{\beta_z - \alpha}{\beta_z - \alpha_z} \cdot p_{I_z}(\boldsymbol{\theta})$ by Eq. (5.2).

Summarizing the foregoing procedures yields Algorithm 4, which simultaneously constructs the set $\mathcal{Z}$ and calculates the prices $p_{I_z}(\boldsymbol{\theta})$. Since it suffices to go down a single path and only perform constant-time computation in each node, the resulting algorithm runs in time $\mathcal{O}(\log n_{vals})$, where n_{vals} denotes the number of distinct values appeared as endpoints of intervals in all the executed transactions. We defer complete proofs from this chapter to Appendix C.1.

Theorem 5.1. *Algorithm 4 implements* **price**(I, T) *in time* $\mathcal{O}(\log n_{vals})$.

Algorithm 4 Query price of an interval $I = [\alpha, 1)$.

Input: Interval $I = [\alpha, 1)$ with $\alpha \in \Omega$. LMSR tree T, with nodes z annotated with $I_z = [\alpha_z, \beta_z)$, h_z, s_z and S_z.

Output: Price of bundle security for I.

```
1: Initialize $z \leftarrow root$, $P \leftarrow 1$, $price \leftarrow 0$
2: while $\alpha_z \neq \alpha$ and $z$ is not a leaf do
3:     $P \leftarrow Pe^{s_z/b}$
4:     if $\alpha < \alpha_{right(z)}$ then
5:         $price \leftarrow price + PS_{right(z)}/S_{root}$
6:         $z \leftarrow left(z)$
7:     else
8:         $z \leftarrow right(z)$
9: return $price + \frac{\beta_z - \alpha}{\beta_z - \alpha_z} \cdot PS_z/S_{root}$
```

5.3.3 Buy Transactions

We next implement **buy**$([\alpha, 1), s, T)$ while maintaining the LMSR tree properties. The main challenge here is to simultaneously maintain *partial-normalization correctness* and *height balance*. We address this by adapting AVL-tree rebalancing.

We begin by considering the case $\alpha \in vals(T)$. Similar to price queries, we conduct binary search for α to obtain the set of nodes $\mathcal{Z}$ that covers $I = [\alpha, 1)$. We update the values of s_z across $z \in \mathcal{Z}$ by adding s, and obtain T' that has the same structure as T with the updated share quantities

$$s'_z = \begin{cases} s_z + s & \text{if } z \in \mathcal{Z} \\ s_z & \text{otherwise.} \end{cases}$$

Thus, the resulting market state is

$$\boldsymbol{\theta}(T') = \sum_{z \in T'} s'_z \mathbf{1}_{I_z} = \sum_{z \in T} s_z \mathbf{1}_{I_z} + \sum_{z \in \mathcal{Z}} s \mathbf{1}_{I_z} = \boldsymbol{\theta}(T) + s\mathbf{1}_I.$$

We then rely on the recursive relationship defined in Eq. (5.7) to update the partial normalization constants S_z. It suffices to update the ancestors of the nodes $z \in \mathcal{Z}$, all of which lie along the search path to α, and each update requires constant time.

When $\alpha \notin \mathit{vals}(T)$, we split the leaf z that contains $\alpha \in [\alpha_z, \beta_z)$ before adding shares to $\mathit{right}(z)$. However, this may violate the *height-balance property*. Similar to the AVL insertion algorithm (Knuth 1998, Section 6.2.3), we fix any imbalance by means of *rotations*, as we go back along the search path. Rotations are operations that modify small portions of the tree, and at most two rotations are needed to rebalance the tree (Adelʹson-Velʹskiĭ and Landis 1962).

We next show that in each rotation, only a constant number of nodes will require updates to preserve the *partial-normalization correctness.* There are two kinds of rotations, depicted in Figure 5.1. The *left rotation* takes as input a node z, with children denoted z_1 and z_{23}, and children of z_{23} denoted z_2 and z_3, and rearranges these relationships by removing the node z_{23} and creating a node z_{12}, such that z now has children z_{12} and z_3, and z_{12} has children z_1 and z_2. The *right rotation* is the symmetric operation.

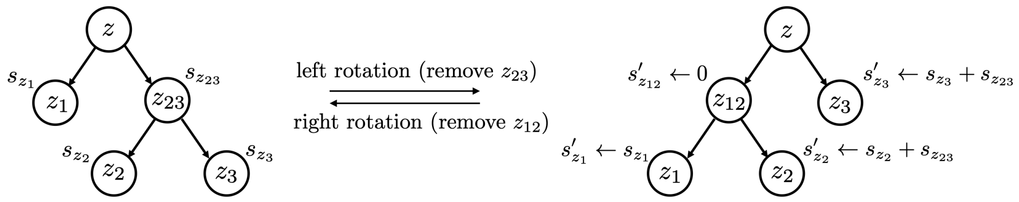

z
s_{z_1}
z_1
$s_{z_{23}}$
z_{23}
s_{z_2}
z_2
s_{z_3}
z_3
left rotation (remove z_{23})
right rotation (remove z_{12})
$s'_{z_{12}} \leftarrow 0$
z_{12}
$s'_{z_3} \leftarrow s_{z_3} + s_{z_{23}}$
$s'_{z_1} \leftarrow s_{z_1}$
$s'_{z_2} \leftarrow s_{z_2} + s_{z_{23}}$

Figure 5.1: Left and right rotations with node z as an input. Depicted update corresponds to the left rotation.

When performing rotations, we need to ensure that the node removal (i.e., removal of z_{23} in left rotation and of z_{12} in right rotation) does not impact the market state. We achieve this by moving the shares from the removed node into its children, so at the time of removal it holds zero shares. The full procedure of `RotateLeft` is described in Appendix Algorithm 8.

Lemma 5.2. *A rotation operation preserves partial-normalization correctness.*

Algorithm 5 describes the **buy** operation, which takes time $\mathcal{O}(\log n_{vals})$ thanks to the height balance.

Theorem 5.3. *Algorithm 5 implements* **buy**(I, s, T) *in time* $\mathcal{O}(\log n_{vals})$.

Algorithm 5 Buy s shares of bundle security for an interval $I = [\alpha, 1)$.

```
Input: Quantity $s \in \mathbb{R}$ and an interval $I = [\alpha, 1)$ with $\alpha \in \Omega$. LMSR tree $T$,
with nodes $z$ annotated with $I_z = [\alpha_z, \beta_z)$, $h_z$, $s_z$ and $S_z$.
Output: Tree $T$ updated to reflect the purchase of $s$ shares of bundle security
for $I$.
 1: Define subroutines:
        NewLeaf($\alpha_0, \beta_0$): return a new leaf node $z$ with
            $I_z = [\alpha_0, \beta_0)$, $h_z = 0$, $s_z = 0$, $S_z = (\beta_0 - \alpha_0)$
        ResetInnerNode($z$): reset $h_z$ and $S_z$ based on the children of $z$
            $h_z \leftarrow 1 + \max\{h_{left(z)}, h_{right(z)}\}$, $S_z \leftarrow e^{s_z/b}(S_{left(z)} + S_{right(z)})$
        AddShares($z, s$): increase the number of shares held in $z$ by $s$
            $s_z \leftarrow s_z + s$, $S_z \leftarrow e^{s/b} S_z$
 2: Initialize $z \leftarrow root$
 3: while $\alpha_z \neq \alpha$ and $z$ is not a leaf do          ▷ add $s$ shares to $z \in \mathcal{Z}$
 4:     if $\alpha < \alpha_{right(z)}$ then
 5:         AddShares($right(z), s$)
 6:         $z \leftarrow left(z)$
 7:     else
 8:         $z \leftarrow right(z)$
 9: if $\alpha_z < \alpha$ then                                     ▷ split the leaf $z$
10:     $left(z) \leftarrow$ NewLeaf($\alpha_z, \alpha$), $right(z) \leftarrow$ NewLeaf($\alpha, \beta_z$)
11:     $z \leftarrow right(z)$
12: AddShares($z, s$)
13: while $z$ is not a $root$ do                    ▷ trace the binary-search path back
14:     $z \leftarrow par(z)$
15:     if $|h_{left(z)} - h_{right(z)}| \geq 2$ then             ▷ restore height balance
16:         Rotate $z$ and possibly one of its children
             (details in Appendix C.1.2 Algorithms 8)
17:     ResetInnerNode($z$)                                  ▷ update $h_z$ and $S_z$
```

Remarks. We have shown that **price**, **cost** and **buy** operations can all be implemented in time $\mathcal{O}(\log n_{vals})$, which is bounded above by the log of the number of **buy** transactions $\mathcal{O}(\log n_{buy})$ as well as the bit precision of the outcome $\mathcal{O}(\log N) = \mathcal{O}(K)$.[6] We note that none of the operations require the knowledge of K, so the market in fact supports queries with arbitrary precision. However, the market precision does affect the worst-case loss bound for the market maker, which is $\mathcal{O}(\log N) = \mathcal{O}(K)$. The next section presents a different construction, which achieves a *constant* worst-case loss independent of the market precision.

[6]Clearly, $n_{vals} \leq 2n_{buy}$ with each **buy** transaction introducing at most two new endpoint values. The value of n_{vals} is also bounded above by $N+1$ since the interval endpoints are always in $\Omega \cup \{1\}$.

5.4 A Multi-Resolution Linearly Constrained Market Maker

This section introduces the second design, referred to as the *multi-resolution linearly constrained market maker* (multi-resolution LCMM). The design is based on the LMSR, but it enables more flexibility by assigning two or more parallel LMSRs with different liquidity parameters to orchestrate submarkets that offer interval securities at different resolutions. However, running submarkets independently can create arbitrage opportunities, as any interval expressible in a coarser market can also be expressed in a finer one. To maintain coherent prices, we design a matrix that imposes linear constraints to tie market prices among different submarkets to support the efficient removal of any arbitrage opportunity. We first define the multi-resolution LCMM and its properties, and show that **price**, **cost** and **buy** can be implemented in time $\mathcal{O}(\log N)$.

5.4.1 A Multi-Resolution LCMM for $[0, 1)$

5.4.1.1 A Multi-Resolution Market

A binary search tree remains at the core construction of our multi-resolution market. Unlike a log-time LMSR that uses a self-balancing tree, it builds upon a *static* one, where each level of the tree represents a submarket of intervals, forming a finer and finer partition of $[0, 1)$. We start with an example of a market that offers interval securities at two resolutions.

Example 5.2 (Two-level market for $[0, 1)$)**.** Consider a market composed of two submarkets, indexed by $\mathcal{Z}_1 = \{11, 12\}$ and $\mathcal{Z}_2 = \{21, 22, 23, 24\}$, which partition $[0, 1)$ into interval events at two levels of coarseness:

$$\mathcal{I}_1 : I_{11} = \left[0, \tfrac{1}{2}\right), I_{12} = \left[\tfrac{1}{2}, 1\right);$$

$$\mathcal{I}_2 : I_{21} = \left[0, \tfrac{1}{4}\right), I_{22} = \left[\tfrac{1}{4}, \tfrac{1}{2}\right), I_{23} = \left[\tfrac{1}{2}, \tfrac{3}{4}\right), I_{24} = \left[\tfrac{3}{4}, 1\right).$$

The market provides six interval securities associated with the corresponding interval events (i.e., $\mathcal{I} = \mathcal{I}_1 \biguplus \mathcal{I}_2$ and $|\mathcal{I}| = 6$). We index the securities by $z \in \mathcal{Z}$, where $\mathcal{Z} = \mathcal{Z}_1 \biguplus \mathcal{Z}_2 = \{11, 12, 21, 22, 23, 24\}$.

We extend Example 5.2 to multiple resolutions. We represent the initial independent submarkets with a *complete binary tree* T^* of depth K, which corresponds to the bit precision of the outcome ω. Let $\mathcal{Z}^*$ denote the set of nodes of T^* and $\mathcal{Z}_k$ for $k \in \{0, 1, \ldots, K\}$ the set of nodes at each level. $\mathcal{Z}_0$ contains the root associated with

$I_{root} = [0, 1)$, and each *consecutive* level contains the children of nodes from the previous level, which split their corresponding parent intervals in half. Thus, level k partitions $[0, 1)$ into 2^k intervals of size 2^{-k} and the final level $\mathcal{Z}_K$ contains $N = 2^K$ leaves.

We index interval securities by nodes, with their payoffs defined by $\phi_z(\omega) = 1\{\omega \in I_z\}$. We partition securities into submarkets corresponding to levels, i.e., $\mathcal{I}_k = \mathcal{Z}_k$ for $k \leq K$, where $|\mathcal{I}_k| = 2^k$ and $\mathcal{I} = \biguplus_{k \leq K} \mathcal{I}_k$. For each submarket, we define the LMSR cost function C_k with a *separate* liquidity parameter $b_k > 0$:

$$C_k(\boldsymbol{\theta}_k) = b_k \log\left(\sum_{z \in \mathcal{Z}_k} e^{\theta_z / b_k}\right) \tag{5.11}$$

5.4.1.2 A Linearly Constrained Market Maker

Following the above multi-resolution construction, the overall market has a *direct-sum cost* $\tilde{C}(\boldsymbol{\theta}) = \sum_{k \leq K} C_k(\boldsymbol{\theta}_k)$, which corresponds to pricing securities in each block $\mathcal{I}_k$ independently using C_k. However, as there are logical dependencies between securities in different levels, independent pricing may lead to incoherent prices among submarkets and create arbitrage opportunities.

Example 5.3 (Arbitrage in a two-level market)**.** Continuing Example 5.2, we define separate LMSR costs, where $b_1 = 1$ and $b_2 = 1$:

$$C_1(\boldsymbol{\theta}_1) = \log\left(e^{\theta_{11}} + e^{\theta_{12}}\right); \quad C_2(\boldsymbol{\theta}_2) = \log\left(e^{\theta_{21}} + e^{\theta_{22}} + e^{\theta_{23}} + e^{\theta_{24}}\right).$$

The direct-sum market $\tilde{C}(\boldsymbol{\theta}) = C_1(\boldsymbol{\theta}_1) + C_2(\boldsymbol{\theta}_2)$ gives rise to incoherent prices. For example, after buying some shares of security ϕ_{21} associated with $I_{21} = \left[0, \frac{1}{4}\right)$ in submarket $\mathcal{I}_2$, the market can have

$$\tilde{p}_{I_{11}=[0,\frac{1}{2})}(\boldsymbol{\theta}) = 0.5; \quad \tilde{p}_{I_{21}=[0,\frac{1}{4})}(\boldsymbol{\theta}) + \tilde{p}_{I_{22}=[\frac{1}{4},\frac{1}{2})}(\boldsymbol{\theta}) = 0.6.$$

This violates the *no-arbitrage* property that requires $\mathbb{P}[I_{11}] = \mathbb{P}[I_{21}] + \mathbb{P}[I_{22}]$ and $\mathbb{P}[I_{12}] = \mathbb{P}[I_{23}] + \mathbb{P}[I_{24}]$ under any probability distribution over Ω. We specify the linear price constraints $\mu_{11} - \mu_{21} - \mu_{22} = 0$ and $\mu_{12} - \mu_{23} - \mu_{24} = 0$ by the following vectors

$$\boldsymbol{a}_1 = (1, 0, -1, -1, 0, 0)^\top \quad \text{and} \quad \boldsymbol{a}_2 = (0, 1, 0, 0, -1, -1)^\top,$$

and refer $\mathbf{A} = (\boldsymbol{a}_1\, \boldsymbol{a}_2) \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{J}|}$ as the constraint matrix of the two-level market, where $\mathcal{J}$ denotes the set of interval events associated with inner nodes, i.e., $\mathcal{J} = \mathcal{I} \backslash \mathcal{I}_K$.

We extend Example 5.3 to specify price constraints in a multi-resolution market to achieve *no arbitrage*. Recall that $\mathcal{M}$ denotes a *coherent price space*, where any expected payoff lies in the convex hull of $\{\boldsymbol{\phi}(\omega)\}_{\omega\in\Omega}$. It is always polyhedral and can be described by a set of linear inequalities (Dudík, Lahaie, and Pennock 2012). Arbitrage opportunities arise whenever prices fall outside the set of coherent prices $\mathcal{M}$ (Abernethy, Chen, and Vaughan 2011). For the multi-resolution market, we specify a set of *homogeneous linear equalities* describing a superset of $\mathcal{M}$. Equalities are indexed by $j \in \mathcal{J}$ and are described by a matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{J}|}$, such that

$$\mathcal{M} \subseteq \{\boldsymbol{\mu} \in \mathbb{R}^{|\mathcal{I}|} : \mathbf{A}^\top \boldsymbol{\mu} = \mathbf{0}\}. \tag{5.12}$$

We design the constraint matrix $\mathbf{A}$ to ensure that any pair of submarkets is price coherent, meaning that any interval event $I \subseteq \Omega$ gets the same price on all levels that can express it. Therefore, for each *inner node* $y \in \mathcal{Z}_l$ where $l < K$, we have

$$\mu_y = \sum_{z\in\mathcal{Z}_k:\, z\subset y} \mu_z \qquad \text{for any } l < k \leq K.$$

To facilitate the implementation in a binary tree, we further tie the price of y to the prices of *all* of y's descendants and weight each level by its liquidity parameter b_k:

$$\underbrace{\Big(\sum_{k>\ell} b_k\Big)}_{B_\ell} \mu_y = \sum_{k>\ell}\Big(b_k \sum_{z\in\mathcal{Z}_k:\, z\subset y} \mu_z\Big). \tag{5.13}$$

This design turns out to be more algorithmically convenient to restore price consistency (as we will see in Section 5.4.3).

Now we can formally define the constraint matrix $\mathbf{A}$. Let $\mathcal{Y}^* = \mathcal{Z}^*\backslash\mathcal{Z}_K$ be the set of inner nodes of T^* and let *level*(z) denote the level of a node z. The matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{Z}^*|\times|\mathcal{Y}^*|}$ contains the constraints from Eq. (5.13) across all $y \in \mathcal{Y}^*$:

$$A_{zy} = \begin{cases} B_{\mathit{level}(z)} & \text{if } z = y, \\ -b_{\mathit{level}(z)} & \text{if } z \subset y, \\ 0 & \text{otherwise.} \end{cases} \tag{5.14}$$

We refer to the *linearly constrained market maker* with the matrix $\mathbf{A}$ as the *multi-resolution LCMM.* The derivation above shows that the constraints in the matrix $\mathbf{A}$ are *necessary* to assure no arbitrage. The next theorem shows that they are also sufficient. The proof shows that consecutive levels are coherent, which by transitivity implies that the overall price vector is coherent (see Appendix C.1.3).

Theorem 5.4. *A multi-resolution LCMM is arbitrage-free.*

We next show that the multi-resolution LCMM also enjoys the *bounded-loss* property. For a suitable choice of liquidities, such as $b_k = \mathcal{O}(1/k^{2.01})$, it can achieve a *constant* worst-case loss bound. The proof uses the fact that the overall loss is bounded by the sum of losses of level markets, which are at most $b_k \log |\mathcal{Z}_k| = kb_k \log 2$.

Theorem 5.5. *Let $\{b_k\}_{k=1}^{\infty}$ be a sequence of positive numbers such that $\sum_{k=1}^{\infty} kb_k = B^*$ for some finite B^*. Then the multi-resolution LCMM with liquidity parameters b_k for $k \leq K$ guarantees the worst-case loss of the market maker of at most $B^* \log 2$, regardless of the outcome precision K.*

Arbitrage opportunities appear if the price of bundle $\boldsymbol{a}_j$ differs from zero, where $\boldsymbol{a}_j$ denotes the jth column of $\mathbf{A}$. Traders profit by buying a positive quantity of $\boldsymbol{a}_j$ if its price is negative, and selling otherwise. Thus, the constraint matrix $\mathbf{A}$ gives a recipe on arbitrage removals. We provide some intuition via the example below.

Example 5.4 (Arbitrage removal by a two-level LCMM)**.** Continue Example 5.3. we have prices $\tilde{\boldsymbol{p}}(\boldsymbol{\theta})$ violate the constraint matrix $\mathbf{A}$, i.e., $\boldsymbol{a}_1^\top \tilde{\boldsymbol{p}}(\boldsymbol{\theta}) = \tilde{p}_{11}(\boldsymbol{\theta}) - \tilde{p}_{21}(\boldsymbol{\theta}) - \tilde{p}_{22}(\boldsymbol{\theta}) = 0.5 - 0.6 \neq 0$. The constraint vector $\boldsymbol{a}_1$ reveals a profitable arbitrage opportunity: buy the security ϕ_{11} (at the initial price 0.5) and simultaneously sell securities ϕ_{21} and ϕ_{22} (at the initial price 0.6). This will increase the price of ϕ_{11} and decrease the prices of ϕ_{21} and ϕ_{22}. After a sufficiently large quantity s shares of ϕ_{11} is bought (and the same quantities of ϕ_{21} and ϕ_{22} are sold), $\boldsymbol{a}_1^\top \tilde{\boldsymbol{p}}(\tilde{\boldsymbol{\theta}}) = 0$ is achieved in a new state $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} + s\boldsymbol{a}_1 = \boldsymbol{\theta} + \mathbf{A}\boldsymbol{\eta}$, where $\boldsymbol{\eta} := (s, 0)^\top$.

Therefore, after each update of $\boldsymbol{\theta}$, an LCMM who follows the matrix $\mathbf{A}$ can automatically find a new state $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} + s\boldsymbol{a}_1 + t\boldsymbol{a}_2 = \boldsymbol{\theta} + \mathbf{A}\boldsymbol{\eta}$ where $\boldsymbol{\eta} := (s, t)^\top$, such that $\boldsymbol{a}_1^\top \tilde{\boldsymbol{p}}(\tilde{\boldsymbol{\theta}}) = 0$ and $\boldsymbol{a}_2^\top \tilde{\boldsymbol{p}}(\tilde{\boldsymbol{\theta}}) = 0$ hold.

We generalize Example 5.4 to the multi-resolution market. Formally, an LCMM is described by the cost function

$$C(\boldsymbol{\theta}) = \inf_{\boldsymbol{\eta} \in \mathbb{R}^{|\mathcal{J}|}} \tilde{C}(\boldsymbol{\theta} + \mathbf{A}\boldsymbol{\eta}). \tag{5.15}$$

It relies on the direct-sum cost $\tilde{C}$, but with each trader purchase $\boldsymbol{\delta}$ that causes inconsistent prices, an LCMM automatically seeks the most advantageous cost for the trader by buying bundles $\mathbf{A}\boldsymbol{\delta}_{\text{arb}}$ on the trader's behalf to remove arbitrage. Trader purchases are accumulated as the state $\boldsymbol{\theta}$, and automatic purchases made by the LCMM are accumulated as $\mathbf{A}\boldsymbol{\eta}$.

We note that the purchase of bundle $\mathbf{A}\boldsymbol{\delta}_{\text{arb}}$ has no effect on the trader's payoff, since $(\mathbf{A}\boldsymbol{\delta}_{\text{arb}})^\top \boldsymbol{\phi}(\omega) = 0$ for all $\omega \in \Omega$ thanks to Eq. (5.12) and the fact that $\boldsymbol{\phi}(\omega) \in \mathcal{M}$. However, the purchase of $\mathbf{A}\boldsymbol{\delta}_{\text{arb}}$ can lower the cost, so optimizing over $\boldsymbol{\delta}_{\text{arb}}$ benefits the traders, while maintaining the same worst-case loss guarantee for the market maker as $\tilde{C}$ (Dudík, Lahaie, and Pennock 2012). Consider a fixed $\boldsymbol{\theta}$ and the corresponding $\boldsymbol{\eta}^\star$ minimizing Eq. (5.15). We calculate prices as $\boldsymbol{p}(\boldsymbol{\theta}) = \nabla C(\boldsymbol{\theta}) = \nabla \tilde{C}(\boldsymbol{\theta}+\mathbf{A}\boldsymbol{\eta}^\star)$. By the first order optimality, $\boldsymbol{\eta}^\star$ minimizes Eq. (5.15) if and only if $\mathbf{A}^\top\big(\nabla\tilde{C}(\boldsymbol{\theta}+\mathbf{A}\boldsymbol{\eta}^\star)\big) = \mathbf{0}$. This means that $\mathbf{A}^\top \boldsymbol{p}(\boldsymbol{\theta}) = \mathbf{0}$, and thus arbitrage opportunities expressed by $\mathbf{A}$ are completely removed by the LCMM cost function C.

To implement an LCMM, we maintain the state $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} + \mathbf{A}\boldsymbol{\eta}$ in the direct-sum market $\tilde{C}$. After updating $\boldsymbol{\theta}$ to a new value $\boldsymbol{\theta}' = \boldsymbol{\theta} + \boldsymbol{\delta}$, we seek to find $\boldsymbol{\eta}' = \boldsymbol{\eta} + \boldsymbol{\delta}_{\text{arb}}$ that removes all the arbitrage opportunities expressed by $\mathbf{A}$. The resulting cost for the trader is

$$\tilde{C}(\boldsymbol{\theta}' + \mathbf{A}\boldsymbol{\eta}') - \tilde{C}(\boldsymbol{\theta} + \mathbf{A}\boldsymbol{\eta}) = \tilde{C}(\tilde{\boldsymbol{\theta}} + \boldsymbol{\delta} + \mathbf{A}\boldsymbol{\delta}_{\text{arb}}) - \tilde{C}(\tilde{\boldsymbol{\theta}}).$$

5.4.1.3 A Multi-Resolution LCMM Tree

We can now formally define the multi-resolution LCMM tree. The market state of a multi-resolution LCMM is represented by vectors $\boldsymbol{\theta} \in \mathbb{R}^{|\mathcal{Z}^*|}$ and $\boldsymbol{\eta} \in \mathbb{R}^{|\mathcal{Y}^*|}$, whose dimensions can be intractably large (e.g., on the order of $2^K = N$). However, since each LCMM operation involves only a small set of coordinates of $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$, we keep track of these coordinates accessed so far by organizing an annotated subtree T of T^*, referred to as an *LCMM tree.*

Definition 5.2 (LCMM Tree)**.** An *LCMM tree* T is a full binary tree, where each node z is annotated with $I_z = [\alpha_z, \beta_z)$, $\theta_z \in \mathbb{R}$, $\eta_z \in \mathbb{R}$, such that $I_{root} = [0,1)$, and for every inner node z:

$$\alpha_z = \alpha_{left(z)}, \quad \beta_{left(z)} = \alpha_{right(z)} = \frac{\alpha_z + \beta_z}{2}, \quad \beta_{right(z)} = \beta_z.$$

The tree T contains the coordinates of $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ accessed so far. Since $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ are initialized to zero, their remaining entries are zero. We write $\boldsymbol{\theta}(T) \in \mathbb{R}^{|\mathcal{Z}^*|}$ and $\boldsymbol{\eta}(T) \in \mathbb{R}^{|\mathcal{Y}^*|}$ for the vectors represented by T. To calculate prices, we maintain $\boldsymbol{\eta}(T)$

that minimizes Eq. (5.15), or equivalently $\boldsymbol{\eta}(T)$ that satisfies $\mathbf{A}^\top \tilde{\boldsymbol{p}}\big(\boldsymbol{\theta}(T) + \mathbf{A}\boldsymbol{\eta}(T)\big) = \mathbf{0}$. If this property holds, we say that an LCMM tree T is *coherent*.

5.4.2 Price Queries

There are many ways to decompose an interval I in a multi-resolution market, but they all yield the same price thanks to coherence. The *no-arbitrage* property also guarantees that the price of $[\alpha, \beta)$ can be obtained by subtracting the price of $[\beta, 1)$ from $[\alpha, 1)$. Therefore, we focus on pricing one-sided intervals of the form $I = [\alpha, 1)$.

Let T be a coherent LCMM tree and $\boldsymbol{\theta} := \boldsymbol{\theta}(T)$ and $\boldsymbol{\eta} := \boldsymbol{\eta}(T)$ the vectors represented by T. Let $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} + \mathbf{A}\boldsymbol{\eta}$ be the corresponding state in $\tilde{C}$, so the current security prices are $\boldsymbol{\mu} := \tilde{\boldsymbol{p}}(\tilde{\boldsymbol{\theta}})$. As before, we identify a set of nodes $\mathcal{Z}$ that covers I, and then rely on price coherence to calculate each μ_z along the search path.

Assume that z is not a root node and we know the price of its parent. Let $\mathit{sib}(z)$ denote the sibling of z and $k = \mathit{level}(z)$. We can then relate the price of z to the price of $\mathit{par}(z)$:

$$\mu_z = \frac{\mu_z}{\mu_{par(z)}} \cdot \mu_{par(z)} = \frac{\mu_z}{\mu_z + \mu_{sib(z)}} \cdot \mu_{par(z)} \tag{5.16}$$

$$= \frac{e^{\tilde{\theta}_z/b_k}}{e^{\tilde{\theta}_z/b_k} + e^{\tilde{\theta}_{sib(z)}/b_k}} \cdot \mu_{par(z)}. \tag{5.17}$$

Eq. (5.16) follows by price coherence and Eq. (5.17) follows by the price calculation in Eq. (5.1). Thus, we descend the search path to calculate each price μ_z, beginning with $\mu_{root} = 1$. It remains to obtain $\tilde{\theta}_z$, which we follow the construction of $\mathbf{A}$ in Eq. (5.14) to compute

$$\tilde{\theta}_z = \theta_z + \sum_{y \in \mathcal{Y}^*} A_{zy}\eta_y = \theta_z + B_k\eta_z - b_k \sum_{y \supset z} \eta_y. \tag{5.18}$$

Plugging the above equation back in Eq. (5.17), we obtain[7]

$$\mu_z = \frac{\exp\big\{(\theta_z + B_k\eta_z)/b_k\big\}}{\exp\big\{(\theta_z + B_k\eta_z)/b_k\big\} + \exp\big\{(\theta_{sib(z)} + B_k\eta_{sib(z)})/b_k\big\}} \cdot \mu_{par(z)}. \tag{5.19}$$

Combining the described procedure yields Algorithm 6. The final line of the algorithm addresses the case when the search ends in the leaf z with $\alpha_z < \alpha < \beta_z$.

[7]The factor $\exp\{-\sum_{y \supset z} \eta_y\} = \exp\{-\sum_{y \supset sib(z)} \eta_y\}$ appears in both the numerator and the denominator after plugging Eq. (5.18) to Eq. (5.17), so it cancels out.

Rather than expanding the tree to its lowest level K, we use price coherence again: since any strict descendant $z' \subset z$ on the path from z to a leaf node $u \in \mathcal{Z}_K$ has $\theta_{z'} = \eta_{z'} = 0$ by market initialization, all leaf nodes have the same price. Therefore, the price of $[\alpha, \beta_z)$ equals $\frac{\beta_z - \alpha}{\beta_z - \alpha_z} \cdot \mu_z$.

The length of search path for α is $prec(\alpha)$, which denotes the bit precision of α, defined as the smallest integer k such that α is an integer multiple of 2^{-k}. As the computation at each node only requires constant time, the time to price $I = [\alpha, 1)$ is $\mathcal{O}(prec(\alpha))$, which is bounded above by $\mathcal{O}(K)$.

0 Let $I = [\alpha, 1)$, $\alpha \in \Omega$. Algorithm 6 implements **price**(I, T) in time $\mathcal{O}(prec(\alpha))$.

Algorithm 6 Query price of an interval $I = [\alpha, 1)$.

Input: Interval $I = [\alpha, 1)$ with $\alpha \in \Omega$. Coherent LCMM tree T, with nodes z annotated with $I_z = [\alpha_z, \beta_z)$, θ_z, η_z.

Output: Price of bundle security for I.

```
1: Initialize $z \leftarrow root$, $\mu_z \leftarrow 1$, $price \leftarrow 0$
2: while $\alpha_z \neq \alpha$ and $z$ is not a leaf do
3:     $z_l \leftarrow left(z)$, $z_r \leftarrow right(z)$, $k \leftarrow level(z_l)$
4:     $e_l \leftarrow \exp\{(\theta_{z_l} + B_k \eta_{z_l})/b_k\}$, $e_r \leftarrow \exp\{(\theta_{z_r} + B_k \eta_{z_r})/b_k\}$,
       $\mu_{z_l} \leftarrow \frac{e_l}{e_l + e_r} \mu_z$, $\mu_{z_r} \leftarrow \frac{e_r}{e_l + e_r} \mu_z$        ▷ calculate prices by Eq. (5.19)
5:     if $\alpha < \alpha_{right(z)}$ then
6:         $z \leftarrow z_l$,    $price \leftarrow price + \mu_{z_r}$
7:     else
8:         $z \leftarrow z_r$
9: return $price + \frac{\beta_z - \alpha}{\beta_z - \alpha_z} \cdot \mu_z$
```

5.4.3 Buy and Cost Operations

Different from LMSR, the cost query for a multi-resolution LCMM cannot be directly derived from prices. We instead augment **buy** to implement **cost** by executing **buy** and then reverting all the changes. We focus on **buy**(I, s, T) for $I = [\alpha, 1)$. By buying s shares of $[\alpha, 1)$ and then $(-s)$ shares of $[\beta, 1)$, we obtain buying $[\alpha, \beta)$.

We summarize the procedure in Algorithm 7, which performs **buy**(I, s, T) and keeps track of **cost**(I, s, T). Similar to price queries, we start with a set of nodes $\mathcal{Z}$ that partition I, by searching for α and simultaneously calculating prices μ_z along the way (lines 3–6).

Algorithm 7 Buy s shares of bundle security for an interval $I = [\alpha, 1)$.

```
Input: Quantity $s \in \mathbb{R}$ and an interval $I = [\alpha, 1)$ with $\alpha \in \Omega$. Coherent LCMM
tree $T$, with nodes $z$ annotated with $I_z = [\alpha_z, \beta_z)$, $\theta_z$, $\eta_z$.
Output: Cost of $s$ shares bundle security for $I$ and the updated tree $T$.
1: Define subroutines:
       NewLeaf($\alpha_0, \beta_0$): return a new leaf node $z$ with
           $I_z = [\alpha_0, \beta_0)$, $\theta_z = 0$, $\eta_z = 0$
       RemoveArbitrage($y, \mu_{other}$): restore price coherence among
        submarkets $k \geq level(y)$ following Eq. (5.20) and update cost
           Let $\ell = level(y)$, $y' = sib(y)$, $t = \frac{b_\ell}{B_{\ell-1}} \log\left(\frac{1-\mu_y}{\mu_y} \cdot \frac{\mu_{other}}{1-\mu_{other}}\right)$
           $S = \mu_y e^{tB_\ell/b_\ell} + 1 - \mu_y$, $S_{other} = \mu_{other} e^{-t} + 1 - \mu_{other}$
           $\eta_y \leftarrow \eta_y + t$, $\mu_y \leftarrow \mu_y e^{tB_\ell/b_\ell}/S$, $\mu_{y'} \leftarrow \mu_{y'}/S$
           $cost \leftarrow cost + (b_\ell \log S) + (B_\ell \log S_{other})$
       AddShares($z, s$): increase shares held in $z$ by $s$, update cost, and
        restore price coherence among submarkets $k \geq level(z)$
           Let $\ell = level(z)$, $z' = sib(z)$, $\mu_{other} = \mu_z$, $S = \mu_z e^{s/b_\ell} + 1 - \mu_z$
           $\theta_z \leftarrow \theta_z + s$
           $cost \leftarrow cost + (b_\ell \log S)$
           $\mu_z \leftarrow \mu_z e^{s/b_\ell}/S$, $\mu_{z'} \leftarrow \mu_{z'}/S$
           RemoveArbitrage($z, \mu_{other}$)
2: Initialize $z \leftarrow root$, $\mu_z \leftarrow 1$, a global variable $cost \leftarrow 0$
3: while $\alpha_z \neq \alpha$ do
4:     if $z$ is a leaf then
5:         $left(z) \leftarrow$ NewLeaf($\alpha_z$, $\frac{1}{2}(\alpha_z + \beta_z)$),
            $right(z) \leftarrow$ NewLeaf($\frac{1}{2}(\alpha_z + \beta_z)$, $\beta_z$)
6:     Search for $\alpha$ and calculate $\mu_z$ (same as Algorithm 6 lines 3-8)
7: AddShares($z, s$)
8: while $z$ is not a $root$ do                    ▷ remove arbitrage up the search path
9:     $z' \leftarrow sib(z)$, $y \leftarrow par(z)$
10:    if $z' = right(y)$ then
11:        AddShares($z', s$)                                ▷ add shares to $z \in \mathcal{Z}$
12:    RemoveArbitrage($y$, $\mu_z + \mu_{z'}$)
13:    $z \leftarrow y$
14: return $cost$
```

We then proceed back up the search path, adding s shares to nodes within the cover $\mathcal{Z}$ (lines 7–13). Consider one of such nodes $y \in \mathcal{Z}$ at level $\ell := level(y)$. Increasing θ_y by s creates price incoherence between the submarket at level ℓ and submarkets at all the other levels. We design RemoveArbitrage to remove any arbitrage opportunity appeared between level ℓ and *all lower levels* with $k > \ell$. We show in Appendix C.1.6 Lemma C.2 that in order to restore coherence, it suffices to update η_y by a closed-form amount:

$$t = \frac{b_\ell}{B_{\ell-1}} \log\left(\frac{1-\mu_y}{\mu_y} \cdot \frac{\mu_{other}}{1-\mu_{other}}\right), \tag{5.20}$$

where $\mu_{other} = \mu_{left(y)} + \mu_{right(y)}$ records the price of y in all the lower levels. Similar to Example 5.4, this key algorithmic step is enabled by the arbitrage bundle $\boldsymbol{a}_y$, which corresponds to buying ϕ_y on the level ℓ while selling securities associated with all descendants of y, with their shares appropriately weighted by the respective liquidity values as specified in the constraint matrix $\mathbf{A}$.

The market remains incoherent between ℓ and *all upper levels* $k < \ell$. Since the updates have been localized to the subtree rooted at y, we use Lemma C.2 again to update $\eta_{par(y)}$ and restore coherence among all levels $k \geq \ell - 1$ (line 12). We continue in this manner back along the path to root to restore a coherent market.

The algorithm also tracks the total cost of the **buy** transaction by evaluating Eq. (5.3) in the component submarkets. Note that costs in all submarkets with $k > \ell$ can be evaluated simultaneously thanks to the restored coherence. As the design of the constraint matrix $\mathbf{A}$ enables a gradual bottom-up removal of arbitrage (and a closed-form solution for t), Algorithm 7 runs in time $\mathcal{O}(prec(\alpha))$.

Theorem 5.6. *Let $I = [\alpha, 1)$, $\alpha \in \Omega$. Algorithm 7 implements a simultaneous* **buy**(I, s, T) *and* **cost**(I, s, T) *in time $\mathcal{O}(prec(\alpha))$.*

Remarks. In Algorithms 6 and 7, we assume that each node z can store a scalar μ_z, which can be modified during the run to support price calculations but is disposed afterwards. The only part of the proposed algorithms that depends on K are the cumulative liquidities $B_\ell = \sum_{k=\ell+1}^{K} b_k$.

To remove such dependence, we can use $B'_\ell = \sum_{k=\ell+1}^{\infty} b_k = B^* - \sum_{k=1}^{\ell} b_k$, where $B^* = \sum_{k=1}^{\infty} b_k$. This has no impact on the correctness of our algorithms: if at a given time the largest level in the tree T is L, we can simply view T as a multi-resolution LCMM with $K = L + 1$ and liquidities $b_1, b_2, \ldots, b_L, B'_L$. The last level $K = L + 1$ then corresponds to infinitely many mutually coherent markets $\{C_k\}_{k=L+1}^{\infty}$. Thus, a multi-resolution LCMM can achieve a constant loss bound regardless of K and support market operations for $I = [\alpha, \beta)$ in time $\mathcal{O}(prec(\alpha) + prec(\beta))$.

5.5 Discussion

This chapter has proposed two cost-function-based market makers that support trading interval securities of arbitrary precision and execute market operations exponentially faster than previous designs. This section discusses when we expect each market maker to be empirically appropriate.

In short, the log-time LMSR enjoys better storage and runtime efficiency, whereas the multi-resolution LCMM has more flexibility in its pricing strategies. While both market makers gradually grow the trees upon each interval trade, the LMSR tree enables a shorter search path thanks to its height-balance property. Thus, the log-time LMSR would be more preferable especially when the designer expects betting interest to be concentrated on a small set of intervals (i.e., the number of distinct intervals traded would be much smaller than the size of nodes in an LCMM tree). Despite its advantage in storage and runtime efficiency, the log-time LMSR faces similar challenges as a standard LMSR in making design choices, such as setting a suitable liquidity value or choosing a proper market resolution. Correctly setting these parameters often requires a good estimate of trader interest even before trading in the market starts.

The multi-resolution LCMM, on the other hand, grants more pricing flexibility and liquidity attenuation to facilitate the designer's information-gathering objective. For example, an LMSR that operates at precision $k = 4$ with liquidity b can be represented by an LCMM with the level liquidity values $\boldsymbol{b} = (0, 0, 0, b, 0, 0, \dots)$. Moreover, if the market expects most of the information at precision 4 but also wants to support bets up to precision 8, one could run an LCMM with the liquidity placed at two levels as $\boldsymbol{b} = (0, 0, 0, b_4, 0, 0, 0, b_8)$. By choosing different values b_4 and b_8, the market designer can express utility for information at different precision levels.

We next empirically highlight such flexibility by showing how LCMM can interpolate between LMSRs at different resolutions, allowing the market to match the coarseness of traders' information.

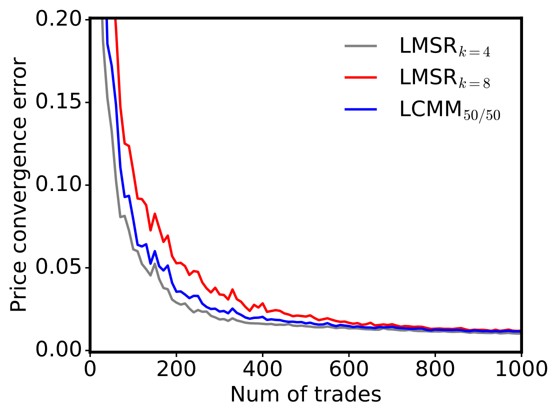
We conduct agent-based simulation using the trader model with exponential utility and exponential-family beliefs (Abernethy, Kutty, Lahaie, and Sami 2014). Agents trade with a market maker (either a LMSR or a multi-resolution LCMM) to bet on intervals within $[0, 1)$, following the dynamics described below. We note that while our market makers support agents with any beliefs and utility functions, the exponential trader model is convenient, because it allows a closed-form derivation of *market-clearing price*, meaning the clearing price reached when agents only trade among themselves, without a market maker (Abernethy, Kutty, Lahaie, and Sami 2014; Dudík, Lahaie, Rogers, and Wortman Vaughan 2017). This can be viewed as a "ground truth" for the information elicitation. We evaluate market makers in terms of their *price convergence error*, calculated as the relative entropy between the market-clearing price and the price maintained by the market maker.

Trading Dynamics We simulate a market consisting of ten traders. The outcome space is $[0, 1)$, discretized at the precision $K = 10$. Traders, indexed as $i \in \{1, \ldots, 10\}$, have noisy access to the underlying true signal $p = 0.4$. Trader i's belief takes form of a beta distribution $\text{Beta}(a_i, b_i)$ with $a_i \sim \text{Binomial}(p, n_i)$, $b_i = n_i - a_i$, and $n_i = 16i$ representing the quality of the agent's observation of the signal p. Each trader i has an exponential utility $u_i(W) = -e^{-W}$, where W is the trader's wealth. We consider budget-limited cost-based market makers, whose worst-case loss may not exceed a budget constraint B. For LMSR at precision k, this means setting the liquidity parameter to $b = B/\log(2^k)$. In our experiments, we consider two LMSR markets at precision levels 4 and 8, denoted as $\texttt{LMSR}_{k=4}$ and $\texttt{LMSR}_{k=8}$. On the other hand, a multi-resolution LCMM has an infinite number of choices for its liquidity at each precision level. To showcase its interpolation ability, we consider LCMM that evenly splits its budget to precision levels 4 and 8, and denote it as $\texttt{LCMM}_{50/50}$.

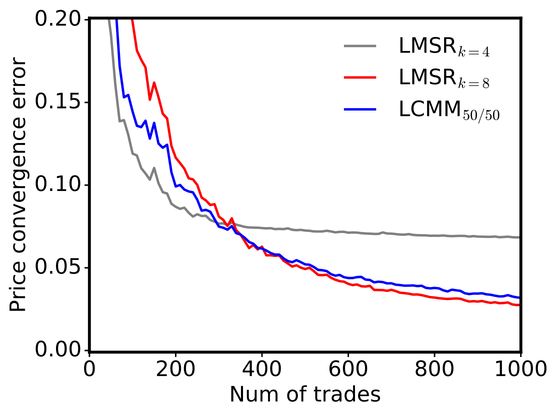
Each market starts with the uniform prior, i.e., the initial market prices for all outcomes are equal. In each time step, a uniformly random agent is picked to trade. The selected agent considers a set of 50 interval securities, with endpoints randomly sampled according to the agent's belief. The candidate intervals are rounded to the precision of the corresponding market. The agent considers trading the expected-utility-optimizing number of shares for each interval, and ultimately picks the best interval and executes the trade.

Figure 5.2 shows the price convergence as a function of the number of trades, averaged over 40 simulated markets mediated by $\texttt{LMSR}_{k=4}$, $\texttt{LMSR}_{k=8}$, and $\texttt{LCMM}_{50/50}$ respectively under the budget constraint $B = 1$ (see Appendix C.2 for results at different budget levels). As one may expect, $\texttt{LMSR}_{k=4}$ achieves a faster price convergence at the coarser precision level $k = 4$ compared to $\texttt{LMSR}_{k=8}$ (Fig. 5.2a), but fails to elicit information at any finer granularity by design.[8] The proposed $\texttt{LCMM}_{50/50}$, by equally splitting the budget between $k = 4$ and $k = 8$, is able to interpolate between the

[8]In Figure 5.2b, to facilitate comparisons, we assume that $\texttt{LMSR}_{k=4}$ equally splits the price of a coarse interval into finer intervals.



(a) $k = 4$.



(b) $k = 8$.

Figure 5.2: The price convergence error as a function of the number of trades, measured at two resolution levels.

performance of $\texttt{LMSR}_{k=4}$ and $\texttt{LMSR}_{k=8}$ and achieves the "best of both worlds": it can elicit forecasts at the finer level $k = 8$ similarly to $\texttt{LMSR}_{k=8}$, but also obtain a fast convergence at the coarser level $k = 4$, almost matching the convergence speed of $\texttt{LMSR}_{k=4}$.

Two natural questions arise from the two proposed designs. First, do our constructions generalize to two- or higher-dimensional outcomes? One promising avenue is to combine the ideas from our log-time LMSR market maker with multi-dimensional segment trees (Mishra 2016) to obtain an efficient multi-dimensional LMSR based on a static tree. However, it is not clear how to generalize our balanced LMSR tree construction or the multi-resolution LCMM. Second, does our approach extend to non-interval securities, such as call options?

CHAPTER 6

Conclusion

This dissertation focuses on addressing two categories of problems present in today's financial markets: *the vulnerability to manipulation* and *the lack of expressiveness*. The first part examines a form of market manipulation and proposes deterrent solutions, combining techniques from agent-based modeling, game-theoretic analysis, and adversarial learning. The second part investigates expressive designs and explores efficient implementations for such mechanisms, using tools from optimization, data structure design, and complexity analysis. This chapter concludes with a summary of contributions and a discussion of limitations and future directions.

Spoofing the Limit Order Book: A Strategic Agent-Based Analysis Chapter 2 models a form of order-based market manipulation, *spoofing*, and proposes market mechanisms and trading strategies to mitigate such manipulation practice. Main contributions of this chapter include:

(1) A computational agent-based model of spoofing prices in a limit-order market (Section 2.5);

The model illustrates the strategic interactions between a manipulator and two groups of background traders, namely *heuristic belief learning* (HBL) and *zero intelligence* (ZI). The former uses market information to trade, whereas the later does not. We demonstrate through *empirical game-theoretic analysis* (EGTA) that in the absence of spoofing, HBL is generally adopted in equilibrium and benefits price discovery and social welfare. Their existence, however, renders a market vulnerable to manipulation: simple spoofing strategies can effectively mislead traders, distort prices, and reduce market surplus. After re-equilibrating, we show that learning traders persist even with manipulators, suggesting that the elimination of spoofing requires active measures.

(2) A *cloaking mechanism* that systematically deters spoofing through disclosing a partially cloaked order book (Section 2.6);

    The mechanism works by symmetrically concealing a deterministic number of price levels from the inside of an order book. The design presents a tradeoff between preserving order book informativeness and mitigating manipulation. We perform *empirical mechanism design* with the goal of maximizing background-trader surplus, and demonstrate in markets with moderate shocks, the benefit of cloaking in deterring spoofing outweighs its efficiency cost. We demonstrate the robustness of cloaking mechanisms by exploring sophisticated spoofing strategies that probe to reveal cloaked information and showing that their associated costs exceed the gains.

(3) Two variations of HBL that intend to improve the robustness of learning-based strategies against spoofing (Section 2.7);

    The first variation offers agents the flexibility to exclude limit orders at a certain price level from the dataset they learn from. We show that this variation can improve robustness against spoofing, while retain a comparable competitiveness in non-manipulated markets. The second variation considers the full order book, but adjusts its learned order price by an offset to correct for bias. It exhibits a general improvement over the baseline HBL, and when combined with the first proposal, it enjoys both improved profitability and robustness against manipulation.

The proposed agent-based model aims to capture the complex essence of real-world financial markets, and EGTA the strategic interactions among agents. However, as discussed in Chapter 2, these studies have several limitations. First, results presented reflect the specific modeling and simulation choices we adopt. Second, several factors can affect our equilibrium analysis, including sampling error, reduced-game approximation, and restricted bidding strategy coverage.

Despite these limitations that are inherent in any complex modeling effort, we believe the agent-based model and deterrent proposals can serve as a constructive basis to study and prevent other forms of manipulation. For instance, a manipulator who learns to spoof the market by optimizing defined objectives (e.g., profits, price deviations) under certain constraints (e.g., order sizes, arrival frequencies). The model can also facilitate identifying practical considerations (e.g., agent strategic responses) that should be regarded when making regulatory decisions.

Modeling the Evasion of Manipulation Detection: An Adversarial Learning Framework Chapter 3 proposes an adversarial learning framework to proactively reason about how a manipulator might mask its behavior to evade a manipulation detector. The framework includes three main components: (1) a generative model that is trained to adapt encoded manipulation order streams to resemble trading patterns of a normal trader, while preserving the manipulation intent; (2) a discriminative model that differentiates the adversarially adapted manipulation order streams from normal trading activities; and (3) an agent-based simulator that generates the source (i.e., manipulation) and target (i.e., market making) order streams, and evaluates the manipulation effect of adapted outputs. The framework is able to generate adapted manipulation examples that resemble the target distribution and appear qualitatively different from the original manipulation strategy. We find that this adaptation evades detection, but at the cost of compromising effectiveness in market manipulation.

One limitation in the current framework is that the adversary only learns to evade detection, but does not assess the cost of such adaptations. A smarter form of adaptation can evade detection, and simultaneously optimizes for trading profits and effectiveness in manipulation.

Several extensions can be made based on the current framework. First, generated examples can be classified into two groups—those that preserve certain manipulation effects and those do not—to support the training of detection algorithms that focus on intent (or effect) rather than patterns. This is somewhat equivalent to training a black-box approximator of the agent-based model we developed. Second, the target distribution can be substituted or removed: substituting with other trading activities as the target enables to generate a more diverse sets of synthetic manipulation order streams; substituting with real market order streams enables to calibrate simulated strategies to real trading practices.

Designing a Combinatorial Financial Options Market Chapter 4 examines current design of financial options market, and proposes a new derivative contract, *combinatorial financial options.* Main contributions of this chapter include:

(1) A mechanism that consolidates and matches orders on standard options related to the same underlying asset (Section 4.4);

The mechanism uses a linear program to aggregate options markets of different strike prices but logically related to the same underlying asset, providing traders the flexibility to define any custom strike value. It runs in time polynomial to the number of orders and poses no risk, regardless of the value of the underlying asset at expiration. Experiments on real options data show that the proposed mechanism finds matches that the current independent-market design cannot, and provides more competitive bid and ask prices.

(2) A combinatorial financial option that offers the contract holder the right to buy or sell any linear combination of multiple underlying assets (Section 4.5);

Combinatorial options markets enable the expression of aggregate belief about future correlations among assets. This increased expressiveness comes at the cost of a higher computational complexity: optimal clearing of a combinatorial financial options market is coNP-hard. We show that the optimal clearing problem can be equivalently formulated as a bilevel mixed-integer linear program, which computes the exact solution by satisfying an increasing set of constraints generated from different future values of the underlying assets. Experiments on synthetic combinatorial options orders demonstrate its practicability.

An immediate next step is to investigate the use of different clearing rules to run a combinatorial options market and quantify the tradeoffs among them. Continuous clearing facilitates instantaneous matching and information disclosing, whereas batch clearing tends to yield efficient matches and higher market surplus, as suggested in our experiments on synthetic markets. Another interesting direction is to explore structured markets where combinations are limited to components in a graph of underlying assets: by limiting aspects of expressivity, we may find computationally tractable mechanisms to clear the market.

Log-time Prediction Markets for Interval Securities Chapter 5 investigates the design of prediction markets to recover a complete and fully general probability distribution over a random variable, through trading interval securities. Main contributions of this chapter include:

(1) A log-time *logarithmic market scoring rule* (LMSR) market maker (Section 5.3);

The log-time LMSR exploits the modularity properties of LMSR to construct a balanced binary tree data structure and decompose computations along the tree nodes. It expedites market operations (i.e., buy operations, price and cost queries) to time logarithmic in the number of distinct intervals that traders define.

(2) A multi-resolution linearly constrained market maker (LCMM) (Section 5.4);

> The multi-resolution LCMM adopts a different binary tree data structure that assigns two or more parallel LMSRs with different liquidity parameters to orchestrate submarkets that offer interval securities at different resolutions. It uses a constraint matrix to tie prices among submarkets, supporting the computationally efficient removal of any arbitrage opportunity. This design remains log-time market operations, and adds two additional benefits for the market designer: (1) the ability to express utility for information at various resolutions, and (2) the ability to guarantee a true constant bounded loss. It opens up the possibilities to elicit arbitrarily fine-grained information (up to the machine precision).

Both proposals restrict to a one-dimensional continuous variable (plus any form of its discretization). An interesting and useful future direction is to extend either binary tree data structure to support disjoint exhaustive outcomes that correspond to some hierarchical structure, or higher-dimensional outcomes.

APPENDICES

APPENDIX A

Detailed Equilibrium Results for Chapter 2

A.1 Spoofing the Limit Order Book

Env	surplus	HBL%
LSLN	18198*	88
LSLN	18246*	98
LSMN	18189*	100
LSHN	18265*	100
MSLN	17947*	58
MSLN	16693*	0
MSMN	17923*	62
MSMN	17927*	43
MSMN	16726	0
MSHN	18266*	100
HSLN	16565	0
HSLN	17143*	0
HSMN	16667	0
HSHN	18253*	87

(a) N = 28 without spoofing

Env	surplus	HBL%
LSLN	43157*	71
LSLN	43102*	73
LSLN	43010*	95
LSMN	43249*	83
LSMN	43086*	79
LSHN	42946	94
MSLN	42804*	57
MSMN	42807*	56
MSMN	42745*	56
MSHN	43265*	86
HSLN	42455*	37
HSMN	42383*	37
HSMN	42144*	32
HSHN	42981	89

(b) N = 65 without spoofing

Table A.1: Background-trader surplus and HBL proportion in equilibrium of markets without spoofing. Each row describes one Nash equilibrium found in a game (rounded to the nearest integer). Surpluses marked with asterisks indicate statistically significantly higher surpluses than those achieved in their corresponding markets with spoofing (see Table A.2).

Env	surplus	HBL%
LSLN	18076	78
LSMN	18040	91
LSHN	18125	87
MSLN	16774	0
MSMN	17883	34
MSMN	17517	24
MSMN	16796	0
MSHN	18108	81
HSLN	16749	0
HSMN	16667	0
HSHN	17999	97

(a) N = 28 with spoofing

Env	surplus	HBL%
LSLN	42868	70
LSLN	42993	70
LSMN	42961	80
LSHN	43061	80
LSHN	43103	74
MSLN	42639	41
MSLN	42698	50
MSMN	42624	52
MSHN	43038	75
MSHN	43101	76
HSLN	41815	29
HSLN	39502	0
HSMN	40091	0
HSHN	43143	71

(b) N = 65 with spoofing

Table A.2: Background-trader surplus and HBL proportion in equilibrium of markets with spoofing. Each row describes one Nash equilibrium found in a game (rounded to the nearest integer).

Env	surplus	HBL	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	ZI_6	ZI_7	HBL_1	HBL_2	HBL_3	HBL_4
LSLN	18198	0.88	0	0.12	0	0	0	0	0	0.88	0	0	0
LSLN	18246	0.98	0	0	0	0	0.02	0	0	0	0.92	0.06	0
LSMN	18189	1	0	0	0	0	0	0	0	0.82	0	0.18	0
LSHN	18265	1	0	0	0	0	0	0	0	1	0	0	0
MSLN	17947	0.58	0	0	0	0.40	0.02	0	0	0	0.40	0.18	0
MSLN	16693	0	0	0	0	0	0	0.74	0.26	0	0	0	0
MSMN	17923	0.62	0	0	0	0	0.38	0	0	0.44	0.18	0	0
MSMN	17927	0.43	0	0.04	0.53	0	0	0	0	0.43	0	0	0
MSMN	16726	0	0	0	0	0	0	0.80	0.20	0	0	0	0
MSHN	18266	1	0	0	0	0	0	0	0	0.74	0.24	0	0.02
HSLN	16565	0	0	0	0	0	0	0.73	0.27	0	0	0	0
HSLN	17143	0	0	0	0.53	0	0	0	0.47	0	0	0	0
HSMN	16667	0	0	0	0	0	0	1	0	0	0	0	0
HSHN	18253	0.87	0	0	0.13	0	0	0	0	0.84	0	0	0.03

Table A.3: Equilibria for games without spoofing, $N = 28$, calculated from the 4-player DPR approximation. Each row of the table describes one equilibrium found with its corresponding surplus, HBL adoption rate and the equilibrium mixture probabilities of strategies included.

Env	surplus	HBL	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	ZI_6	ZI_7	HBL_1	HBL_2	HBL_3	HBL_4
LSLN	43157	0.71	0	0.29	0	0	0	0	0	0.59	0	0.12	0
LSLN	43102	0.73	0	0	0.27	0	0	0	0	0.73	0	0	0
LSLN	43010	0.95	0	0	0.05	0	0	0	0	0.22	0	0.73	0
LSMN	43249	0.83	0	0	0.17	0	0	0	0	0.57	0	0.26	0
LSMN	43086	0.79	0	0.05	0.16	0	0	0	0	0	0.79	0	0
LSHN	42946	0.94	0	0.04	0.02	0	0	0	0	0.75	0.19	0	0
MSLN	42804	0.57	0	0	0.43	0	0	0	0	0.31	0.26	0	0
MSMN	42807	0.56	0	0	0.44	0	0	0	0	0.31	0.25	0	0
MSMN	42745	0.56	0.01	0	0	0.43	0	0	0	0	0.56	0	0
MSHN	43265	0.86	0	0.06	0	0.08	0	0	0	0.67	0.19	0	0
HSLN	42455	0.37	0	0	0.63	0	0	0	0	0	0.18	0.19	0
HSMN	42383	0.37	0	0	0.63	0	0	0	0	0.26	0	0.11	0
HSMN	42144	0.32	0	0	0	0.54	0.14	0	0	0	0	0.32	0
HSHN	42981	0.89	0	0.08	0	0	0	0.03	0	0.89	0	0	0

Table A.4: Equilibria for games without spoofing, $N = 65$, calculated from the 5-player DPR approximation. Each row of the table describes one equilibrium found with its corresponding surplus, HBL adoption rate and the equilibrium mixture probabilities of strategies included.

Env	surplus	HBL	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	ZI_6	ZI_7	HBL_1	HBL_2	HBL_3	HBL_4
LSLN	18076	0.78	0	0	0.22	0	0	0	0	0.78	0	0	0
LSMN	18040	0.91	0	0	0	0.09	0	0	0	0.91	0	0	0
LSHN	18125	0.87	0	0	0.13	0	0	0	0	0.87	0	0	0
MSLN	16774	0	0	0	0	0	0	1	0	0	0	0	0
MSMN	17883	0.34	0	0	0.11	0.55	0	0	0	0	0.34	0	0
MSMN	17517	0.24	0	0	0.54	0	0	0	0.21	0.24	0	0	0
MSMN	16796	0	0	0	0	0	0	1	0	0	0	0	0
MSHN	18108	0.81	0	0	0.12	0.07	0	0	0	0.81	0	0	0
HSLN	16749	0	0	0	0	0	0.04	0.96	0	0	0	0	0
HSMN	16667	0	0	0	0	0	0	1	0	0	0	0	0
HSHN	17999	0.97	0	0	0	0.03	0	0	0	0.75	0	0.22	0

Table A.5: Equilibria for games with spoofing, $N = 28$, calculated from the 4-player DPR approximation. Each row of the table describes one equilibrium found with its corresponding surplus, HBL adoption rate and the equilibrium mixture probabilities of strategies included.

Env	surplus	HBL	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	ZI_6	ZI_7	HBL_1	HBL_2	HBL_3	HBL_4
LSLN	42868	0.70	0.21	0	0.09	0	0	0	0	0.70	0	0	0
LSLN	42993	0.70	0	0.30	0	0	0	0	0	0.54	0.16	0	0
LSMN	42961	0.80	0	0	0.20	0	0	0	0	0.51	0.29	0	0
LSHN	43061	0.80	0	0	0.20	0	0	0	0	0.80	0	0	0
LSHN	43103	0.74	0	0.26	0	0	0	0	0	0.74	0	0	0
MSLN	42639	0.41	0	0	0	0 .59	0	0	0	0	0.41	0	0
MSLN	42698	0.50	0	0	0.50	0	0	0	0	0.32	0	0.18	0
MSMN	42624	0.52	0	0	0.48	0	0	0	0	0	0.38	0.14	0
MSHN	43038	0.75	0	0.25	0	0	0	0	0	0.48	0.27	0	0
MSHN	43101	0.76	0	0.24	0	0	0	0	0	0.41	0.35	0	0
HSLN	41815	0.29	0	0	0.50	0	0	0.21	0	0	0.29	0	0
HSLN	39502	0	0	0	0	0	0	0	1	0	0	0	0
HSMN	40091	0	0	0	0	0	0	0.77	0.23	0	0	0	0
HSHN	43143	0.71	0.10	0	0.19	0	0	0	0	0.71	0	0	0

Table A.6: Equilibria for games with spoofing, $N = 65$, calculated from the 5-player DPR approximation. Each row of the table describes one equilibrium found with its corresponding surplus, HBL adoption rate and the equilibrium mixture probabilities of strategies included.

Env	surplus	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	ZI_6	ZI_7
LSLN	16929	0	0	0	0	0	0.98	0.02
LSMN	16914	0	0	0	0	0	0.89	0.11
LSHN	18213	0.22	0.78	0	0	0	0	0
MSLN	16693	0	0	0	0	0	0.74	0.26
MSMN	17192	0	0	0.42	0	0	0	0.58
MSMN	16726	0	0	0	0	0	0.80	0.20
MSHN	16746	0	0	0.09	0	0	0	0.91
MSHN	17516	0.38	0	0	0	0	0.62	0
HSLN	16565	0	0	0	0	0	0.73	0.27
HSLN	17143	0	0	0.53	0	0	0	0.47
HSMN	16667	0	0	0	0	0	1	0
HSHN	17861	0.31	0.39	0	0	0	0.30	0

Table A.7: Equilibria for games where agents are restricted to ZI strategies, $N = 28$, calculated from the 4-player DPR approximation. Each row of the table describes one equilibrium found with its corresponding surplus and the equilibrium mixture probabilities of strategies included.

Env	surplus	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	ZI_6	ZI_7
LSLN	42938	0	1	0	0	0	0	0
LSLN	40779	0	0	0	0	0	1	0
LSMN	42972	0	0.97	0	0	0	0	0.03
LSMN	40557	0	0	0	0	0	0.83	0.17
LSHN	43327	0.44	0.56	0	0	0	0	0
LSHN	43173	0.11	0.89	0	0	0	0	0
MSLN	40444	0	0	0	0	0	1	0
MSMN	39622	0	0	0	0	0	1	0
MSHN	43140	0	0.73	0.27	0	0	0	0
HSLN	40523	0	0	0.28	0	0	0	0.72
HSLN	40038	0	0	0	0	0	0.60	0.40
HSMN	40458	0	0	0	0.08	0	0.73	0.19
HSHN	43197	0	0.88	0	0.12	0	0	0

Table A.8: Equilibria for games where agents are restricted to ZI strategies, $N = 65$, calculated from the 5-player DPR approximation. Each row of the table describes one equilibrium found with its corresponding surplus and the equilibrium mixture probabilities of strategies included.

A.2 A Cloaking Mechanism to Mitigate Spoofing

Strategy	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	ZI_6	ZI_7	ZI_8	ZI_9	HBL_1	HBL_2	HBL_3	HBL_4
L	-	-	-	-	-	-	-	-	-	2	3	5	8
$R_{\min}$	0	0	0	0	0	0	250	250	250	250	250	250	250
$R_{\max}$	1000	1000	1000	2000	2000	2000	500	500	500	500	500	500	500
η	0.4	0.8	1	0.4	0.8	1	0.4	0.8	1	1	1	1	1

Table A.9: Background trading strategies included in EGTA for cloaking mechanisms.

Env	K	95% CI background surplus	95% CI total surplus	HBL fraction
LSHN	K0	[42121, 42329]	[42548, 42694]	1.00
LSHN	K1	[41848, 42048]	[42254, 42396]	0.98
LSHN	K1	[41769, 41977]	[42264, 42406]	0.92
LSHN	K2	[41788, 42000]	[42205, 42347]	0.997
LSHN	K4	[41572, 41772]	[42046, 42188]	0.89
MSMN	K0	[41958, 42220]	[42274, 42388]	0.67
MSMN	K1	[41902, 42164]	[42210, 42324]	0.67
MSMN	K1	[41849, 42107]	[42170, 42284]	0.60
MSMN	K1	[41801, 42067]	[42167, 42281]	0.68
MSMN	K2	[41742, 42000]	[42123, 42237]	0.66
MSMN	K4	[41693, 41924]	[42116, 42230]	0.47
MSMN	K4	[38809, 39025]	[39367, 39485]	0.012
HSLN	K0	[41529, 41871]	[41974, 42088]	0.59
HSLN	K0	[41698, 42040]	[42102, 42216]	0.67
HSLN	K0	[41625, 41973]	[42021, 42135]	0.67
HSLN	K1	[41417, 41769]	[41869, 41983]	0.66
HSLN	K2	[41377, 41655]	[41776, 41890]	0.38
HSLN	K2	[39728, 39972]	[40484, 40594]	0
HSLN	K2	[38691, 38965]	[39419, 39537]	0
HSLN	K4	[39557, 39803]	[40256, 40374]	0
HSLN	K4	[39558, 39804]	[40290, 40408]	0

Table A.10: Equilibria for games where the exploiter does not spoof. Each row of the table describes one equilibrium found with its corresponding background surplus, total surplus and HBL adoption rate. Results reported are based on at least 20,000 simulation runs.

Env	K	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	ZI_6	ZI_7	ZI_8	ZI_9	HBL_1	HBL_2	HBL_3	HBL_4
LSHN	K0	0	0	0	0	0	0	0	0	0	0.70	0	0	0.30
LSHN	K1	0	0	0.02	0	0	0	0	0	0	0.73	0	0	0.25
LSHN	K1	0.05	0	0.03	0	0	0	0	0	0	0	0.88	0	0.04
LSHN	K2	0	0	0	0	0	0	0.003	0	0	0	0.856	0	0.141
LSHN	K4	0	0	0.11	0	0	0	0	0	0	0.29	0.60	0	0
MSMN	K0	0	0	0.33	0	0	0	0	0	0	0.51	0.16	0	0
MSMN	K1	0.11	0.01	0.21	0	0	0	0	0	0	0	0.14	0.53	0
MSMN	K1	0	0.20	0.20	0	0	0	0	0	0	0.20	0.15	0.25	0
MSMN	K1	0	0	0.32	0	0	0	0	0	0	0.14	0.39	0.03	0.12
MSMN	K2	0	0.15	0	0.19	0	0	0	0	0	0.11	0.40	0.15	0
MSMN	K4	0.20	0	0.33	0	0	0	0	0	0	0.40	0	0.07	0
MSMN	K4	0	0	0	0	0	0	0.674	0.312	0.002	0	0	0.012	0
HSLN	K0	0	0	0.12	0	0	0	0.29	0	0	0.49	0.10	0	0
HSLN	K0	0	0	0	0.33	0	0	0	0	0	0.66	0	0	0.01
HSLN	K0	0	0	0	0.19	0.14	0	0	0	0	0	0.50	0.17	0
HSLN	K1	0.05	0	0	0.29	0	0	0	0	0	0	0.09	0.57	0
HSLN	K2	0.27	0.35	0	0	0	0	0	0	0	0.08	0	0.30	0
HSLN	K2	0.03	0.29	0.13	0	0	0	0	0	0.55	0	0	0	0
HSLN	K2	0	0	0	0	0	0	0.25	0.34	0.41	0	0	0	0
HSLN	K4	0	0.35	0	0	0	0	0.65	0	0	0	0	0	0
HSLN	K4	0	0.36	0	0	0	0	0.64	0	0	0	0	0	0

Table A.11: Detailed equilibria for games where the exploiter does not spoof. Each row of the table describes one equilibrium found with its corresponding mixture of background strategies.

Env	K	95% CI background surplus	95% CI total surplus	HBL fraction	Spoofing fraction
LSHN	K0	[41693, 41893]	[42243, 42389]	0.95	1.00
LSHN	K1	[41848, 42048]	[42254, 42396]	0.98	0.00
LSHN	K2	[41788, 42000]	[42205, 42347]	0.997	0.00
LSHN	K4	[41564, 41764]	[42010, 42152]	0.90	0.08
LSHN	K4	[41572, 41772]	[42046, 42188]	0.89	0.00
MSMN	K0	[41652, 41902]	[42151, 42265]	0.65	1.00
MSMN	K0	[41622, 41884]	[42106, 42220]	0.66	1.00
MSMN	K1	[41902, 42164]	[42210, 42324]	0.67	0.00
MSMN	K1	[41849, 42107]	[42170, 42284]	0.60	0.00
MSMN	K1	[41801, 42067]	[42167, 42281]	0.68	0.00
MSMN	K1	[41749, 42031]	[42146, 42260]	0.72	0.71
MSMN	K2	[41700, 41946]	[42109, 42223]	0.54	0.90
MSMN	K4	[41655, 41883]	[42111, 42225]	0.48	0.62
MSMN	K4	[38809, 39025]	[39367, 39485]	0.012	0.00
HSLN	K0	[41538, 41882]	[42047, 42161]	0.69	1.00
HSLN	K1	[41417, 41769]	[41869, 41983]	0.66	0.00
HSLN	K1	[41039, 41345]	[41593, 41707]	0.48	1.00
HSLN	K2	[41080, 41342]	[41719, 41833]	0.28	1.00
HSLN	K4	[39557, 39803]	[40256, 40374]	0	0.00
HSLN	K4	[39558, 39804]	[40290, 40408]	0	0.00

Table A.12: Equilibria for games where the exploiter strategically chooses to spoof. Each row of the table describes one equilibrium found with its corresponding background surplus, total surplus, HBL and spoofing adoption rate. Results reported are based on at least 20,000 simulation runs.

Env	K	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	ZI_6	ZI_7	ZI_8	ZI_9	HBL_1	HBL_2	HBL_3	HBL_4
LSHN	K0	0	0	0	0.05	0	0	0	0	0	0.95	0	0	0
LSHN	K1	0	0	0.02	0	0	0	0	0	0	0.73	0	0	0.25
LSHN	K2	0	0	0	0	0	0	0.003	0	0	0	0.856	0	0.141
LSHN	K4	0	0	0.10	0	0	0	0	0	0	0.38	0.52	0	0
LSHN	K4	0	0	0.11	0	0	0	0	0	0	0.29	0.60	0	0
MSMN	K0	0	0.19	0	0.16	0	0	0	0	0	0.65	0	0	0
MSMN	K0	0	0.20	0	0	0	0	0.14	0	0	0.61	0.05	0	0
MSMN	K1	0.11	0.01	0.21	0	0	0	0	0	0	0	0.14	0.53	0
MSMN	K1	0	0.20	0.20	0	0	0	0	0	0	0.20	0.15	0.25	0
MSMN	K1	0	0	0.32	0	0	0	0	0	0	0.14	0.39	0.03	0.12
MSMN	K1	0.28	0	0	0	0	0	0	0	0	0	0.51	0.21	0
MSMN	K2	0	0	0.46	0	0	0	0	0	0	0.35	0	0.19	0
MSMN	K4	0	0.52	0	0	0	0	0	0	0	0.46	0	0	0.02
MSMN	K4	0	0	0	0	0	0	0.674	0.312	0.002	0	0	0.012	0
HSLN	K0	0	0	0	0.31	0	0	0	0	0	0.69	0	0	0
HSLN	K1	0.05	0	0	0.29	0	0	0	0	0	0	0.09	0.57	0
HSLN	K1	0	0	0	0.33	0	0	0.19	0	0	0.08	0.40	0	0
HSLN	K2	0	0.72	0	0	0	0	0	0	0	0	0	0.28	0
HSLN	K4	0	0.35	0	0	0	0	0.65	0	0	0	0	0	0
HSLN	K4	0	0.36	0	0	0	0	0.64	0	0	0	0	0	0

Table A.13: Detailed Equilibria for games where the exploiter strategically chooses to spoof. Each row of the table describes one equilibrium found with its corresponding mixture of background strategies.

A.3 Learning-Based Trading Strategies under the Presence of Market Manipulation

Strategy	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	HBL_1	HBL_2	HBL_3	HBL_4	HBL_5	HBL_6	HBL_7	HBL_8	HBL_9	HBL_{10}
χ	NA	NA	NA	NA	NA	NA	1	2	NA	NA	NA	NA	2	2	2
$R_{\min}$	0	0	0	0	0	0	0	0	-10	-20	-40	-80	-10	-20	-40
$R_{\max}$	1000	1000	1000	500	250	0	0	0	0	0	0	0	0	0	0
η	0.4	0.8	1	0.8	0.8	1	1	1	1	1	1	1	1	1	1

Table A.14: Background trading strategies used in EGTA for HBL variations. HBL_{n-L} in tables below means HBL_n with memory length of L.

Env	Baseline	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	HBL_{1-2}	HBL_{1-5}	95% CI Background Surplus
LSHN	✓	0	0	0	0	0	0	1.00	[42050, 42142]
		0	0	0	0	0	1.00	0	[41609, 41703]
LSHN - Spoof	✓	0	0	0	0	0	0	1.00	[41641, 41733]
		0	0	0	0	0	1.00	0	[41300, 41393]
MSMN	✓	0.0116	0	0	0	0	0	0.9884	[41820, 41978]
		0	0	0	0	0.2516	0.7484	0	[41779, 41965]
		0	0	0.2652	0	0	0.7348	0	[41693, 41866]
MSMN - Spoof	✓	0	0	0.3656	0	0	0.6344	0	[41493, 41669]
		0.2250	0	0	0	0	0.7750	0	[41702, 41876]
		0	0.2705	0	0	0	0.7295	0	[41642, 41814]
HSLN	✓	0.2280	0	0	0	0	0	0.7720	[41659, 41907]
		0.3424	0	0	0	0	0.6576	0	[41568, 41816]
		0	0	0	0.4288	0	0	0.5712	[41339, 41599]
		0	0	0.6218	0	0	0	0.3782	[41071, 41281]
		0	0.5029	0	0	0	0.4971	0	[41218, 41452]
		0	0.4413	0	0	0	0	0.5587	[41304, 41546]
HSLN - Spoof	✓	0.3054	0	0	0	0	0	0.6946	[41427, 41670]
		0	0	0.6985	0	0	0	0.3015	[40944, 41127]
		0	0.5851	0	0	0	0.4149	0	[41120, 41335]
		0.3882	0	0	0	0	0.6118	0	[41420, 41665]
		0	0	0.6758	0	0	0.3242	0	[41014, 41208]

Table A.15: Equilibria for games where the learning-based trading strategy set is restricted to standard HBL. Each row describes an equilibrium found for the game described by the *Env* column, detailing the adoption rate of each strategy considered and the corresponding background surplus. The equilibrium strategy profiles with checkmarks in the "Baseline" column indicates those used as baseline strategy profiles for controlled experiments.

Env	ZI_1	ZI_2	ZI_3	ZI_4	ZI_5	HBL_{1-2}	HBL_{1-5}	HBL_{3-2}	HBL_{3-5}	95% CI Background Surplus
LSHN	0	0	0	0	0	1.00	0	0	0	[41609, 41703]
	0	0	0	0	0	0	1.00	0	0	[42050, 42142]
	0	0	0	0	0	0	0	1.00	0	[41690, 41784]
LSHN -	0	0	0	0	0	1.00	0	0	0	[41300, 41393]
Spoof	0	0	0	0	0	0	1.00	0	0	[41641, 41733]
	0	0	0	0	0	0	0	1.00	0	[41690, 41784]
	0	0	0	0	0	0	0	0	1.00	[42093, 42139]
MSMN	0.0116	0	0	0	0	0	0.9884	0	0	[41820, 41978]
	0	0	0	0	0.2516	0.7484	0	0	0	[41779, 41965]
	0	0	0.2652	0	0	0.7348	0	0	0	[41693, 41866]
	0.2378	0	0	0	0	0	0	0.7622	0	[41651, 41743]
	0	0.1733	0	0	0	0	0	0	0.8267	[41801, 41890]
MSMN -	0	0	0.3656	0	0	0.6344	0	0	0	[41493, 41669]
Spoof	0.2250	0	0	0	0	0.7750	0	0	0	[41702, 41876]
	0	0	0	0	0	0	0	0	1.00	[41841, 41920]
	0.2526	0	0	0	0	0	0.7474	0	0	[41808, 41988]
	0	0.2848	0	0	0	0	0	0.7152	0	[41764, 41853]
HSLN	0.2280	0	0	0	0	0	0.7720	0	0	[41659, 41907]
	0.3424	0	0	0	0	0.6576	0	0	0	[41568, 41816]
	0	0	0	0.4288	0	0	0.5712	0	0	[41339, 41599]
	0	0	0.6218	0	0	0	0.3782	0	0	[41071, 41281]
	0.36	0	0	0	0	0	0	0.64	0	[41608, 41734]
	0	0	0.6155	0	0	0	0	0	0.3845	[41087, 41194]
	0	0	0.6103	0	0	0	0	0.3897	0	[41122, 41231]
HSLN -	0.3054	0	0	0	0	0	0.6946	0	0	[41427, 41670]
Spoof	0.3882	0	0	0	0	0.6118	0	0	0	[41420, 41665]
	0	0.4868	0	0	0	0	0	0.5132	0	[41285, 41405]
	0.3428	0	0	0	0	0	0	0.6572	0	[41632, 41759]
	0	0	0.6163	0	0	0	0	0.3837	0	[41123, 41230]
	0.2846	0	0	0	0	0	0	0	0.7154	[41720, 41848]

Table A.16: Equilibria for games where the learning-based trading strategy set is comprised of standard HBL and HBL with price level blocking. Each row describes an equilibrium found for the game described by the *Env* column, detailing the adoption rate of each strategy considered and the corresponding background surplus.

Env	ZI_1	ZI_2	ZI_3	ZI_4	HBL_{4-2}	HBL_{5-2}	HBL_{6-2}	HBL_{4-5}	HBL_{5-5}	HBL_{6-5}	95% CI Background Surplus
LSHN	0	0	0	0	0	0	0	0	1.00	0	[41518, 42562]
	0	0	0	0	0	0	0	1.00	0	0	[41512, 42556]
	0	0	0	0	0	0	1.00	0	0	0	[42420, 42507]
	0	0	0	0	0	1.00	0	0	0	0	[42551, 42640]
	0	0	0	0	1.00	0	0	0	0	0	[42551, 42639]
LSHN -	0	0	0	0	0	0	0	1.00	0	0	[42430, 42474]
Spoof	0	0	0	0	0	0	1.00	0	0	0	[42406, 42492]
	0	0	0	0	0	1.00	0	0	0	0	[42527, 42614]
	0	0	0	0	1.00	0	0	0	0	0	[42516, 42603]
MSMN	0	0	0	0	0	0	1.00	0	0	0	[42085, 42229]
	0.0307	0	0	0	0	0.9693	0	0	0	0	[42227, 42383]
	0	0	0	0	1.00	0	0	0	0	0	[42219, 42366]
	0	0	0.2127	0	0	0	0	0	0	0.7873	[41702, 41787]
	0.0968	0	0	0	0	0	0	0	0.9032	0	[42058, 42142]
MSMN -	0	0	0	0.0781	0	0	0	0	0	0.9219	[41912, 41991]
Spoof	0	0	0	0	0	0	1.00	0	0	0	[42054, 42197]
	0	0.1623	0	0	0	0.8377	0	0	0	0	[41951, 42119]
	0	0.1276	0	0	0	0.8724	0	0	0	0	[42021, 42185]
HSLN	0.1181	0	0	0	0	0.8819	0	0	0	0	[42140, 42255]
HSLN -	0.1601	0	0	0	0	0	0	0	0	0.8394	[41782, 41893]
Spoof	0	0.3713	0	0	0	0	0	0	0.6287	0	[41457, 41578]

Table A.17: Equilibria for games where the learning-based trading strategy set is comprised of standard HBL and HBL with price offsets. Each row describes an equilibrium found for the game described by the *Env* column, detailing the adoption rate of each strategy considered and the corresponding background surplus.

Env	ZI_1	ZI_2	ZI_3	HBL_{3-2}	HBL_{4-2}	HBL_{5-2}	HBL_{6-2}	HBL_{8-2}	HBL_{9-2}	HBL_{10-2}	HBL_{4-5}	HBL_{5-5}	HBL_{6-5}	HBL_{8-5}	HBL_{9-5}	HBL_{10-5}	95% CI
LSHN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	0	[42520, 42565]
	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	0	0	[42511, 42556]
	0	0	0	0	0	0	0	0	0	0	0	1.00	0	0	0	0	[41518, 42562]
	0	0	0	0	0	0	0	0	0	0	1.00	0	0	0	0	0	[41512, 42556
	0	0	0	0	0	0	0	0	0	1.00	0	0	0	0	0	0	[42423, 42509]
	0	0	0	0	0	0	0	0	1.00	0	0	0	0	0	0	0	[42550, 42638]
	0	0	0	0	0	0	0	1.00	0	0	0	0	0	0	0	0	[42555, 42642]
	0	0	0	0	0	0	1.00	0	0	0	0	0	0	0	0	0	[42420, 42507]
	0	0	0	0	0	1.00	0	0	0	0	0	0	0	0	0	0	[42551, 42640]
	0	0	0	0	1.00	0	0	0	0	0	0	0	0	0	0	0	[42551, 42639]
LSHN - Spoof	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	0	0	[42511, 42556]
	0	0	0	0	0	0	0	0	0	1.00	0	0	0	0	0	0	[42422, 42509]
	0	0	0	0	0	0	0	0	1.00	0	0	0	0	0	0	0	[42551, 42639]
	0	0	0	0	0	0	0	1.00	0	0	0	0	0	0	0	0	[42554, 42641]
	0	0	0	0	0	0	1.00	0	0	0	0	0	0	0	0	0	[42406, 42492]
	0	0	0	0	0	1.00	0	0	0	0	0	0	0	0	0	0	[42527, 42614]
	0	0	0	0	1.00	0	0	0	0	0	0	0	0	0	0	0	[42516, 42603]
MSMN	0.2182	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7818	0	[41877, 41967]
	0.1118	0	0	0	0	0	0	0	0.8882	0	0	0	0	0	0	0	[42123, 42291]
	0	0	0.1978	0	0	0	0	0	0	0.8022	0	0	0	0	0	0	[41755, 41921]
	0	0	0	0	0	0	1.00	0	0	0	0	0	0	0	0	0	[42085, 42229]
	0.0307	0	0	0	0	0.9693	0	0	0	0	0	0	0	0	0	0	[42227, 42383]
	0	0	0	0	1.00	0	0	0	0	0	0	0	0	0	0	0	[42219, 42366]
MSMN - Spoof	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.00	[42060, 42133]
	0	0	0	0	0	0	0	0	1.00	0	0	0	0	0	0	0	[42246, 42395]
	0	0.1276	0	0	0.8724	0	0	0	0	0	0	0	0	0	0	0	[42021, 42185]
	0.2526	0	0	0.7474	0	0	0	0	0	0	0	0	0	0	0	0	[41808, 41988]
HSLN	0.2275	0	0	0	0	0	0	0	0.7725	0	0	0	0	0	0	0	[41845, 42085]
	0.2370	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7630	[41705, 41824]
HSLN - Spoof	0	0.3713	0	0	0	0	0	0	0	0	0	0.6287	0	0	0	0	[41457, 41578]
	0.2884	0	0	0	0	0	0	0.7116	0	0	0	0	0	0	0	0	[41754, 41997]
	0	0.3231	0	0	0	0	0	0	0.6769	0	0	0	0	0	0	0	[41639, 41877]

Table A.18: Equilibria for games where the learning-based trading strategy set is comprised of standard HBL, HBL with price level blocking, HBL with price offsets, and HBL with both price offsets and price level blocking (HBL_1 and HBL_2 are not shown because they do not appear in any equilibrium). Each row describes an equilibrium found for the game described by the *Env* column, detailing the adoption rate of each strategy considered and the corresponding background surplus.

APPENDIX B

Additional Proofs and Results for Chapter 4

B.1 Deferred Proofs from Section 4.4

B.1.1 Proof of Theorem 4.1

As the left-hand side of the constraint in M.1 is a piecewise linear function of S, it suffices to solve M.1 by including constraints defined by S at breakpoint values. In our case, breakpoints of the constraint are the strike values specified in the market, and the number of distinct strikes n_K grows sublinear in the number of orders n_{orders}. Therefore, we can specify the constraint in M.1 as $n_K + 2$ payoff constraints for each stock value $S \in \boldsymbol{p} \cup \boldsymbol{q} \cup \{0, \infty\}$. Moreover, there are at most n_{orders} quantity constraints, if we consider the quantity specified in each order. Thus, M.1 is a linear program with $n_{\text{orders}}+1$ decision variables and $\mathcal{O}(n_{\text{orders}})$ constraints. Mechanism M.1 matches options written on the same underlying asset and expiration across all types and strikes in time polynomial in the number of orders.

B.1.2 Proof of Price Quote Procedure

We reiterate the procedure of using mechanism M.1 to price a target options (χ, S, K, T) with existing options orders in the market defined by $(\boldsymbol{\phi}, \boldsymbol{p}, \boldsymbol{b}, \boldsymbol{\psi}, \boldsymbol{q}, \boldsymbol{a})$.

(1) The best bid b^* for an option (χ, S, K, T) is the maximum gain of selling a portfolio of options that is *weakly dominated* by (χ, S, K, T) for some constant L.

We derive b^* by adding (χ, S, K, T) to the sell side of the market indexed $N + 1$, initializing its price a_{N+1} to 0, and solving for M.1. The best bid b^* is the returned objective of M.1.

Proof. We show that the above procedure finds the best bid, by returning the maximum gain of selling a *weakly dominated* portfolio of options. After adding (χ, S, K, T) to the market, we have the updated M.1 as the following:

$$\begin{aligned}
\max_{\boldsymbol{\gamma},\boldsymbol{\delta},L} \quad & \boldsymbol{b}^\top \boldsymbol{\gamma} - \boldsymbol{a}^\top \boldsymbol{\delta} - a_{N+1}\delta_{N+1} - L \\
\text{s.t.} \quad & \sum_m \gamma_m \max\{\phi_m(S - p_m), 0\} - \sum_n \delta_n \max\{\psi_n(S - q_n), 0\} \\
& - \delta_{N+1} \max\{\chi(S - K), 0\} \leq L \qquad \forall S \in [0, \infty)
\end{aligned}$$

Since we set $a_{N+1} = 0$, it is always optimal to buy option (χ, S, K, T) and have $\delta_{N+1} = 1$. Therefore, we have the following optimization problem extended from M.1:

$$\begin{aligned}
\max_{\boldsymbol{\gamma},\boldsymbol{\delta},L} \quad & z := \boldsymbol{b}^\top \boldsymbol{\gamma} - \boldsymbol{a}^\top \boldsymbol{\delta} - L \qquad \text{(M.1-bid)} \\
\text{s.t.} \quad & \underbrace{\sum_m \gamma_m \max\{\phi_m(S - p_m), 0\} - \sum_n \delta_n \max\{\psi_n(S - q_n), 0\}}_{\text{Portfolio } (*)} \\
& \leq \max\{\chi(S - K), 0\} + L \quad \forall S \in [0, \infty)
\end{aligned}$$

Following Definition 4.1, the left-hand side of the above inequality describes the payoff of a portfolio of existing options that is weakly dominated by the target options (χ, S, K, T) with an offset L, and the objective z maximizes the gain of selling such a portfolio. One is willing to buy (χ, S, K, T) at the highest price (i.e., the best bid) $b^* = z$, as one can always sell the weakly dominated Portfolio $(*)$ and get z back without losing anything in the future. $\square$

(2) The best ask a^* for an option (χ, S, K, T) is the minimum cost of buying a portfolio of options that *weakly dominates* (χ, S, K, T) for some constant L.

We derive a^* by adding (χ, S, K, T) to the buy side of the market indexed $M + 1$, initializing its price b_{M+1} to a large number (i.e., 10^6), and solving for M.1. The best ask a^* is then b_{M+1} minus the returned objective.

Proof. We show that the above procedure finds the best ask, by returning the minimum cost of buying a *weakly dominant* portfolio of options. After adding

(χ, S, K, T) to the market, we have the updated M.1 as the following:

$$\begin{aligned}
\max_{\boldsymbol{\gamma},\boldsymbol{\delta},L} \quad & \boldsymbol{b}^\top\boldsymbol{\gamma} + b_{M+1}\gamma_{M+1} - \boldsymbol{a}^\top\boldsymbol{\delta} - L \\
\text{s.t.} \quad & \sum_m \gamma_m \max\{\phi_m(S-p_m),0\} + \gamma_{M+1}\max\{\chi(S-K),0\} \\
& - \sum_n \delta_n \max\{\psi_n(S-q_n),0\} \le L \qquad\qquad \forall S \in [0,\infty)
\end{aligned}$$

We set b_{M+1} to a sufficiently large number, say $b_{M+1} = 10^6$, so that it is always optimal to sell option (χ, S, K, T) and thus have $\gamma_{M+1} = 1$. Therefore, we have the following optimization problem:

$$\begin{aligned}
\max_{\boldsymbol{\gamma},\boldsymbol{\delta},L} \quad & z := \boldsymbol{b}^\top\boldsymbol{\gamma} + b_{M+1} - \boldsymbol{a}^\top\boldsymbol{\delta} - L \qquad \text{(M.1-ask)} \\
\text{s.t.} \quad & \max\{\chi(S-K),0\} \le \\
& \underbrace{-\sum_m \gamma_m \max\{\phi_m(S-p_m),0\} + \sum_n \delta_n \max\{\psi_n(S-q_n),0\}}_{\text{Portfolio } (*)} + L \\
& \qquad\qquad \forall S \in [0,\infty)
\end{aligned}$$

Following Definition 4.1, the right-hand side of the above inequality describes the payoff of a portfolio of existing options that weakly dominates the target options (χ, S, K, T) with an offset L. Since b_{M+1} is a fixed constant, the objective that maximizes for z is equivalent to minimizing for $-\boldsymbol{b}^\top\boldsymbol{\gamma} + \boldsymbol{a}^\top\boldsymbol{\delta} + L$, which is the net cost of buying Portfolio $(*)$ plus L. One is willing to sell (χ, S, K, T) at the lowest price (i.e., the best ask) $a^* = b_{M+1} - z$, as one can always pay a^* and buy back a weakly dominant Portfolio $(*)$ without losing anything in the future. □

B.1.3 Proof of Corollary 4.1.1

We start by showing that in order to quote the most competitive prices (i.e., the highest bid and the lowest ask) for any target option (χ, S, K, T), it suffices to consider options orders in $\mathcal{F}$. We prove by contradiction and consider the following two cases:

(1) Suppose that there exists a bid order $o \notin \mathcal{F}$ with $\gamma_o > 0$ in the Portfolio $(*)$, which is the optimal portfolio constructed to derive the highest bid or lowest ask for (χ, S, K, T).

Since $o \notin \mathcal{F}$, then by the contrapositive of Definition 4.2, the bid of o can be improved by a portfolio of other orders, denoted Portfolio o^*, which is weakly dominated by o with some constant offset L^*. We denote b_o the bid price specified in order o and Π_{o^*} the revenue of selling portfolio o^*. Then, we have

$$z^* := \Pi_{o^*} - L^* > b_o \quad \text{and} \quad \Psi_{o^*} \leq \Psi_o + L^*.$$

This means that we can replace o with Portfolio o^* and the L^* without violating the constraints in M.1-bid and M.1-ask (since $\gamma_o \Psi_o \geq \gamma_o(\Psi_{o^*} - L^*)$), and improve the objective by $\gamma_o(z^* - b_o) > 0$. This contradicts our premises, and thus to derive the most competitive prices for any option (χ, S, K, T), we have $\gamma_o = 0$ for all $o \notin \mathcal{F}$.

(2) Similarly, suppose that there exists an ask order $o \notin \mathcal{F}$ with $\delta_o > 0$ in the Portfolio $(*)$, which is the optimal portfolio constructed to derive the highest bid or lowest ask for (χ, S, K, T).
Since $o \notin \mathcal{F}$, then by the contrapositive of Definition 4.2, the ask of o can be improved by a portfolio of other orders, denoted Portfolio o^*, which weakly dominates o with some constant offset L^*. We denote a_o the ask price specified in order o and Π_{o^*} the cost of buying portfolio o^*. Then, we have

$$z^* := \Pi_{o^*} - L^* < a_o \quad \text{and} \quad \Psi_o \leq \Psi_{o^*} + L^*.$$

This means that we can replace o with Portfolio o^* and the L^* without violating the constraints in M.1-bid and M.1-ask (since $\delta_o \Psi_o \leq \delta_o(\Psi_{o^*} + L^*)$), and improve the objective by $\delta_o(a_o - z^*) > 0$. This contradicts our premises, and thus to derive the most competitive prices for any option (χ, S, K, T), we have $\delta_o = 0$ for all $o \notin \mathcal{F}$.

Therefore, to quote the most competitive prices for any target option (χ, S, K, T), it suffices to consider options orders in $\mathcal{F}$. Similar proofs hold for deciding the existence of matching: if an order $o \notin \mathcal{F}$ appears in the matched portfolio, we can always improve the objective by substituting o with Portfolio o^*. Thus, to determine the price quotes and the existence of a match, it suffices to consider options orders in $\mathcal{F}$. Following Theorem 4.1, our proposed mechanism M.1 determines price quotes and the existence of a match in time polynomial in the size of the frontier set.

B.2 Deferred Proofs from Section 4.5

B.2.1 Proof of Theorem 4.2

The decision problem described in Theorem 4.2 is in NP. Given a certificate which is a value vector $\boldsymbol{S} \in \mathbb{R}_{+}^{U}$, we plug $\boldsymbol{S}$ into constraint in Mechanism M.2 to compute the payoff and check whether it is less than L. This takes time $\mathcal{O}(U(M+N))$. For the NP-hardness, we prove the following stronger statement, which we will later use directly to prove Theorem 4.3.

Theorem B.1 (Variation of Theorem 4.2)**.** *Consider all combinatorial options in the market* $(\boldsymbol{\phi}, \boldsymbol{\alpha}, \boldsymbol{p}, \boldsymbol{\psi}, \boldsymbol{\beta}, \boldsymbol{q})$*. For any fixed L, it is NP-hard to decide*

- *Yes:* $\boldsymbol{\gamma} = \boldsymbol{\delta} = \mathbf{1}$ *violates constraint* (4.2) *in M.2 for some* $\boldsymbol{S}$*. Moreover, there exists a function* $\varepsilon : \mathbb{Z} \to \mathbb{R}^{+}$ *such that given the fixed L, for any* $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ *that satisfies*

$$\sum_{m} \gamma_m \max\{\phi_m(\boldsymbol{\alpha}_m^{\top}\boldsymbol{S} - p_m), 0\} - \sum_{n} \delta_n \max\{\psi_n(\boldsymbol{\beta}_n^{\top}\boldsymbol{S} - q_n), 0\} \le L + 0.25 \quad \forall \boldsymbol{S} \in \mathbb{R}_{\ge 0}^{U}$$

  we have $\frac{|\gamma|}{M} < 1 - \varepsilon(M)$*,*

- *No:* $\boldsymbol{\gamma} = \boldsymbol{\delta} = \mathbf{1}$ *satisfies constraint* (4.2) *for all* $\boldsymbol{S}$*,*

even assuming that each combinatorial option is written on at most two underlying assets.

Proof. We prove by reducing from the Vertex Cover problem: given an undirected graph $G = (V, E)$ and an integer k, decide if there is a subset of vertices $V' \subseteq V$ of size k such that each edge has at least one vertex in V'. Given a Vertex Cover instance (G, k), we construct an instance of the combinatorial options matching problem. Let the set of underlying assets correspond to vertices in G, i.e., $U = |V|$. For each vertex indexed i, we associate four options with it (one on the buy side and three on the sell side), which have payoff functions as follows:

$$f_i = \max\{2K_1 S_i - K_1, 0\}, \qquad g_i^{(1)} = \max\{K_1 S_i, 0\},$$
$$g_i^{(2)} = \max\{K_2 S_i - K_2, 0\}, \qquad g_i^{(3)} = \max\{S_i, 0\},$$

where we choose K_1 and K_2 for some large numbers with $K_2 \gg K_1$. For example, we have $K_1 = 10|E|$ and $K_2 = 100|E|$. For each edge $e = (i, j)$, we define two options (one on the buy side and one on the sell side) that involve its two end-points i and j:

$$f_e = \max\{S_i + S_j, 0\}, \qquad g_e = \max\{S_i + S_j - 1, 0\}.$$

Finally, we include one sell order on an option with payoff $g^\star = \max\{|E| - k - L - 0.5, 0\}$. Since L is fixed in advance, we assume $|E| - k - L - 0.5 > 0$ without loss of generality. Thus, we have $M = |V| + |E|$ buy orders and $N = 3|V| + |E| + 1$ sell orders. The construction takes time polynomial in the size of the Vertex Cover instance.

Suppose the Vertex Cover instance is a *Yes* instance, and $\{v_1, v_2, ..., v_k\}$ is a vertex cover. We show that assigning $S_1, S_2, ..., S_k$ to 1 for the selected underlying assets and 0 for the rest unselected gives an $\boldsymbol{S}$ that violates the constraint (4.2). The left-hand side of the constraint (4.2) is

$$z := \underbrace{\sum_{i \in V} \left(f_i - g_i^{(1)} - g_i^{(2)} - g_i^{(3)} \right)}_{z_v} + \underbrace{\sum_{e \in E} (f_e - g_e)}_{z_e} - g^\star. \tag{B.1}$$

For $S_i \in \{0, 1\}$, it is easy to see that $f_i - g_i^{(1)} - g_i^{(2)} = 0$. Thus, we have $z_v = -\sum_{i \in V} g^{(3)} = -k$ by our assignment. Since at least one of S_i, S_j is 1 for any edge $(i, j) \in E$, we have $f_e - g_e = 1$ and $z_e = |E|$. Therefore, we have

$$z = -k + |E| - (|E| - k - L - 0.5) = L + 0.5 > L.$$

To conclude the proof for the *Yes* instance case, we find the function $\varepsilon(\cdot)$ such that for *any* $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ with $|\boldsymbol{\gamma}| \geq M(1 - \varepsilon(M))$, there exists a $\boldsymbol{S}$ that violates constraint (4.2) even if the L on the right-hand side is changed to $L + 0.25$. We keep the assignment of $\boldsymbol{S}$ as before, and the left-hand side of constraint (4.2) is as the following:

$$\begin{aligned}
z' :=& \sum_{i\in V}\left(\gamma_i f_i - \delta_i^{(1)} g_i^{(1)} - \delta_i^{(2)} g_i^{(2)} - \delta_i^{(3)} g_i^{(3)}\right) + \sum_{e\in E}(\gamma_e f_e - \delta_e g_e) - \delta^\star g^\star \\
\geq& \sum_{i\in V}\left(\gamma_i f_i - g_i^{(1)} - g_i^{(2)} - g_i^{(3)}\right) + \sum_{e\in E}(\gamma_e f_e - g_e) - g^\star \\
&\qquad\qquad\qquad\qquad \text{(since } \delta_n \in [0,1] \text{ and option payoffs are non-negative)} \\
=& z - \sum_{i\in V}(1-\gamma_i) f_i - \sum_{e\in E}(1-\gamma_e) f_e \\
\geq& z - M\cdot 2K_1 + \sum_{i\in V}\gamma_i \cdot 2K_1 + \sum_{e\in E}\gamma_e\cdot 2K_1 \\
&\qquad\qquad\qquad\qquad \text{(since } \forall m: f_m \leq 2K_1 \text{ and } M = |V|+|E|) \\
=& z - M\cdot 2K_1 + |\boldsymbol{\gamma}|\cdot 2K_1 \qquad\qquad (|\boldsymbol{\gamma}| = \textstyle\sum_m \gamma_m) \\
\geq& z - M\varepsilon(M)\cdot 2K_1 \qquad\qquad \text{(since } |\boldsymbol{\gamma}| \geq M(1-\varepsilon(M))) \\
=& L + 0.5 - M\varepsilon(M)\cdot 2K_1
\end{aligned}$$

It suffices to choose ε such that $M\varepsilon(M)\cdot 2K_1 < 0.25$. Recall that $K_1 = 10|E| < 10M$. We can choose, say $\varepsilon = \frac{1}{80M^2}$.

Suppose the Vertex Cover instance is a *No* instance. We aim to show that for the given $\boldsymbol{\gamma}$, $\boldsymbol{\delta}$, and L, there does not exist an $\boldsymbol{S}$ that violates the constraint. We prove by maximizing z and demonstrating $z \leq L$. We start by proving the following claim.

Claim 1. *For an optimal z, we have $S_i \in \{0,1\}$.*

We prove by contradiction, first assuming $S_j > 1$ for some j. Similarly, we have

$$z := \underbrace{f_j - g_j^{(1)} - g_j^{(2)} - g_j^{(3)}}_{z_j} + \underbrace{\sum_{i\in V\setminus j}\left(f_i - g_i^{(1)} - g_i^{(2)} - g_i^{(3)}\right)}_{z_i} + \underbrace{\sum_{e\in E}(f_e - g_e)}_{z_e} - g^\star.$$

We first analyze z_j and have

$$\begin{aligned}
z_j =& \max\{2K_1 S_j - K_1, 0\} - \max\{K_1 S_j, 0\} - \max\{K_2 S_j - K_2, 0\} - \max\{S_j, 0\} \\
=& 2K_1 S_j - K_1 - K_1 S_j - (K_2 S_j - K_2) - S_j \qquad\qquad \text{(by assumption of } S_j > 1) \\
=& K_2 - K_1 - (K_2 - K_1 + 1) S_j.
\end{aligned}$$

Recall that we choose $K_2 \gg K_1$, $K_1 = 10|E|$ and $K_2 = 100|E|$. Thus, we have z_j increase with rate $K_2 - K_1 + 1$ as S_j decreases uniformly. Since z_e decreases at most

$|E|$ and the rest two terms, z_i and $g^\star$, do not depend on S_j, decreasing S_j increases z. It is sub-optimal to have $S_j > 1$ for some j.

Next, we assume $0 \le S_j \le 1$, and have

$$z_j = \begin{cases} -K_1 S_j - S_j & 0 \le S_j \le 0.5, \\ \\ K_1(S_j - 1) - S_j & 0.5 < S_j \le 1. \end{cases}$$

As K_1 is large, by a similar argument analyzing the growth rate of each term, we show that z_j (and also z) increases by assigning S_j to 0 if $0 \le S_j \le 0.5$, and by assigning S_j to 1 if $0.5 < S_j \le 1$.

Now, we have $S_i \in \{0, 1\}$ and aim to maximize z. Following Eq. (B.1), we have

$$z = -\sum_{i \in V} S_i + \sum_{e \in E} (f_e - g_e) - (|E| - k) + L + 0.5.$$

As S_i is an integer, to show $z \le L$, it suffices to show that $\sum_{e\in E}(f_e - g_e) - \sum_{i \in V} S_i < |E| - k$. We prove by contradiction, assuming $\sum_{e\in E}(f_e - g_e) - \sum_{i\in V} S_i \ge |E| - k$. Recall that to have $f_e - g_e = 1$ for $e = (i, j)$, we need at least one of S_i, S_j to be 1. We consider the following two possible cases, and aim to refute them:

(a) $\sum_{e\in E}(f_e - g_e) = |E|$ and $\sum_{i\in V} S_i \le k$.

This means we cover all edges with at most k vertices assigned to 1.

(b) $\sum_{e\in E}(f_e - g_e) < |E|$ and $\sum_{i\in V} S_i < k$.

For any e with $f_e - g_e = 0$, we can assign 1 to one of its end-points, and have $\sum_{e\in E}(f_e - g_e) = |E|$ without changing $\sum_{e\in E}(f_e - g_e) - \sum_{i\in V} S_i$. This leads back to (a).

Both cases contradict to the fact that the Vertex Cover instance is a No instance. We have $\sum_{e\in E}(f_e - g_e) - \sum_{i\in V} S_i < |E| - k$ as desired, and thus $z \le L$ for all $\boldsymbol{S}$. □

B.2.2 Proof of Theorem 4.3

We reduce this decision problem from the decision problem in Theorem B.1 with $L = 0$ and each combinatorial option is written on at most two underlying assets.

The reduction is as follows. The instance of the optimization problem have the same $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{p}, \boldsymbol{q}$ as given in the instance for the decision problem. In other words, the reduction keeps the same for $f_1, \ldots, f_M, g_1, \ldots, g_N$. Set $a_1 = \cdots = a_N = a$, and set

$b_1 = \cdots = b_M = b$, where $a > 0$ is sufficiently small and $b > 0$ is sufficiently large. We will decide both values later.

If the decision problem instance is a *No* instance, we know that the constraint (2) holds for $f_1, \dots, f_M$, $g_1, \dots, g_N$, $\gamma_1 = \cdots = \gamma_M = \delta_1 = \cdots = \delta_N = 1$ and $L = 0$. Under this feasible assignment for $\boldsymbol{\gamma}, \boldsymbol{\delta}$ and L, we have $\boldsymbol{b}^\top \boldsymbol{\gamma} - \boldsymbol{a}^\top \boldsymbol{\delta} - L = Mb - Na$.

If the decision problem instance problem is a *Yes* instance, we aim to show that $\boldsymbol{b}^\top \boldsymbol{\gamma} - \boldsymbol{a}^\top \boldsymbol{\delta} - L < Mb - Na$ for any feasible $\boldsymbol{\gamma}, \boldsymbol{\delta}, L$. We discuss three different cases: $L > 0.25$, $0 \le L \le 0.25$, and $L < 0$.

For $L > 0.25$, we have $\boldsymbol{b}^\top \boldsymbol{\gamma} - \boldsymbol{a}^\top \boldsymbol{\delta} - L < \boldsymbol{b}^\top \boldsymbol{\gamma} - 0.25$. Since each entry of $\boldsymbol{\gamma}$ is at most 1, the maximum of $\boldsymbol{b}^\top \boldsymbol{\gamma}$ is Mb. Putting together, we have $\boldsymbol{b}^\top \boldsymbol{\gamma} - \boldsymbol{a}^\top \boldsymbol{\delta} - L < Mb - 0.25$, which is less than $Mb - Na$ if a is set such that $a < 1/4N$. We will fix $a = 1/8N$ from now on.

For $0 \le L \le 0.25$, Theorem B.1 ensures that there exists an ε which depends only on M such that any feasible $\boldsymbol{\gamma}, \boldsymbol{\delta}$ satisfy $|\boldsymbol{\gamma}| < (1 - \varepsilon)M < M$. Notice that L is set to 0 in the instance we are reducing *from*, and L here is between 0 and 0.25. These make the statement corresponding to the *Yes* case of Theorem B.1 apply. Therefore, the objective $\boldsymbol{b}^\top \boldsymbol{\gamma} - \boldsymbol{a}^\top \boldsymbol{\delta} - L \le \boldsymbol{b}^\top \boldsymbol{\gamma} \le (1 - \varepsilon)Mb$ is strictly less than $Mb - Na$ if b is set such that $b > \frac{Na}{\varepsilon M}$ (notice that ε in Theorem B.1 does not depend on b).

For $L < 0$, notice that substituting $\boldsymbol{\gamma} = \boldsymbol{0}, \boldsymbol{\delta} = \boldsymbol{1}, \boldsymbol{S} = \boldsymbol{0}$ to the left-hand side of (2) gives an upper-bound to $-L$. Let L^* be this upper-bound. Notice that L^* only depends on $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{p}, \boldsymbol{q}$ and is computable in polynomial time. In the case the decision problem instance is a *Yes* instance, we know that any feasible $\boldsymbol{\gamma}, \boldsymbol{\delta}$ satisfy $|\boldsymbol{\gamma}| < (1-\varepsilon)M$ (this is already the case for $L = 0$, and the feasible region for $(\boldsymbol{\gamma}, \boldsymbol{\delta})$ can only be smaller for negative L). Therefore, we have $\boldsymbol{b}^\top \boldsymbol{\gamma} - \boldsymbol{a}^\top \boldsymbol{\delta} - L < Mb(1-\varepsilon) + L^*$, which is less than $Mb - Na$ if b satisfies $b > \frac{Na+L^*}{\varepsilon M}$. By setting $b = \frac{Na+L^*}{\varepsilon M} + 1$, the theorem concludes. Notice that L^* and ε only depend on $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{p}, \boldsymbol{q}$, so our definition of b is valid.

B.3 Deferred Experimental Results

We implement Mechanism M.1 and Algorithm 3 using Gurobi and conduct experiments on an AWS m5a.8xlarge instance. For consolidating standard options related to the same underlying asset and expiration date, we compare the two cases where L is treated as a decision variable and L is fixed to 0. We attach the detailed statistics for options of each stock below. For Algorithm 3, we use $\mathcal{M} = 10^6$ throughout all experiments.

B.3.1 Statistics of options on each stock using M.1 with L as a decision variable

Stock	#markets	#expira-tions	#markets/ expiration	#matches ($L \geq 0$)	ave profit	#matches ($L < 0$)	ave interest rate (%)
AAPL	1452	14	104	1	0.8	3	0.94
AXP	804	11	73	4	0.5	2	0.99
BA	1694	14	121	5	3.35	3	0.66
CAT	968	13	74	6	0.18	2	0.69
CSCO	728	12	61	1	0.02	2	0.44
CVX	742	12	62	7	1.26	2	0.41
DD	778	13	60	1	0.32	3	0.45
DIS	806	13	62	5	0.86	0	0
GS	1110	12	93	1	0.16	3	0.82
HD	908	13	70	1	0.02	2	0.17
IBM	848	12	71	4	3.45	1	0.91
INTC	662	12	55	2	0.54	1	0.86
JNJ	850	12	71	4	0.12	2	0.57
JPM	854	13	66	3	0.4	3	0.7
KO	618	13	48	4	0.2	2	0.34
MCD	692	12	58	1	0.8	3	0.44
MMM	804	12	67	5	2.23	2	0.41
MRK	766	12	64	3	0.18	2	0.42
MSFT	1194	13	92	1	0	0	0
NKE	844	12	70	0	0	2	0.6
PFE	640	12	53	6	0.84	1	0.59
PG	786	12	66	3	0.27	2	0.31
RTX	920	14	66	1	0.01	0	0
TRV	256	6	43	0	0	1	0.4
UNH	964	12	80	1	0.04	2	1.34
V	856	13	66	3	0.03	5	1.47
VZ	508	12	42	6	0.65	1	1.81
WBA	808	11	73	0	0	1	0.14
WMT	810	12	68	3	0.2	3	0.51
XOM	832	12	69	7	2.78	1	0.85
Total	25502	366	69	94	1.03	56	0.70

Table B.1: Summary statistics (matching) of options on each stock in DJI by consolidating options related to the same underlying asset and expiration date.

Stock	#option series	#orders in $\mathcal{F}$	Frontier (%)	ave call spread	ave put spread	improved call spread	improved put spread	% spread reduced
AAPL	1038	787	38	1.46	1.92	0.25	0.29	84
AXP	476	440	46	0.37	0.3	0.15	0.13	58
BA	734	684	47	1.18	0.56	0.34	0.26	65
CAT	482	503	52	0.73	0.41	0.22	0.18	65
CSCO	608	616	51	1.27	0.45	0.1	0.07	90
CVX	254	274	54	0.2	0.2	0.11	0.09	51
DD	564	624	55	0.28	0.17	0.13	0.1	49
DIS	518	515	50	0.91	0.79	0.13	0.12	85
GS	738	631	43	0.91	0.61	0.24	0.16	73
HD	688	675	49	1.69	1.53	0.33	0.32	80
IBM	506	429	42	1.54	0.76	0.31	0.21	77
INTC	462	523	57	1.07	0.62	0.2	0.16	79
JNJ	404	412	51	1.04	1.2	0.14	0.13	88
JPM	524	472	45	0.62	0.57	0.11	0.09	83
KO	370	350	47	0.34	0.17	0.05	0.04	82
MCD	266	285	54	0.48	0.31	0.21	0.16	53
MMM	336	379	56	0.74	0.56	0.22	0.19	68
MRK	488	507	52	0.4	0.34	0.15	0.14	62
MSFT	1120	833	37	2.2	0.96	0.33	0.27	81
NKE	720	766	53	1.78	0.63	0.13	0.08	91
PFE	284	304	54	0.58	0.44	0.05	0.05	90
PG	486	491	51	0.31	0.16	0.15	0.11	44
RTX	822	831	51	1.74	1.5	0.93	0.93	43
TRV	204	216	53	1.18	1.36	0.46	0.49	63
UNH	750	618	41	2.05	1.23	0.54	0.37	72
V	416	413	50	0.51	0.43	0.22	0.2	56
VZ	248	295	59	0.11	0.09	0.06	0.06	43
WBA	744	641	43	1.2	1.46	0.37	0.38	72
WMT	494	475	48	0.47	0.25	0.14	0.11	64
XOM	344	288	42	0.4	0.47	0.1	0.11	75
Total	16088	15277	49	0.93	0.68	0.23	0.2	73

Table B.2: Summary statistics (quoting) of options on each stock in DJI by consolidating options related to the same underlying asset and expiration date.

B.3.2 Statistics of options on each stock using M.1 with $L = 0$

Stock	#markets	#expira-tions	#markets/ expiration	#matches ($L = 0$)	average profit
AAPL	1452	14	104	3	2.14
AXP	804	11	73	4	1.56
BA	1694	14	121	4	3.72
CAT	968	13	74	5	0.48
CSCO	728	12	61	0	0
CVX	742	12	62	5	0.36
DD	778	13	60	2	0.03
DIS	806	13	62	5	0.58
GS	1110	12	93	3	7.3
HD	908	13	70	1	0.29
IBM	848	12	71	2	4.89
INTC	662	12	55	1	0.01
JNJ	850	12	71	2	0.08
JPM	854	13	66	2	0.28
KO	618	13	48	1	0.13
MCD	692	12	58	2	0.43
MMM	804	12	67	1	1.36
MRK	766	12	64	2	0.04
MSFT	1194	13	92	0	0
NKE	844	12	70	1	0.01
PFE	640	12	53	3	0.15
PG	786	12	66	3	0.48
RTX	920	14	66	1	0.01
TRV	256	6	43	0	0
UNH	964	12	80	2	7.54
V	856	13	66	5	3.5
VZ	508	12	42	4	0.49
WBA	808	11	73	0	0
WMT	810	12	68	4	0.18
XOM	832	12	69	6	1.16
Total	25502	366	69	74	1.54

Table B.3: Summary statistics (matching with $L = 0$) of options on each stock in DJI by consolidating options related to the same underlying asset and expiration date.

Stock	#option series	#orders in $\mathcal{F}$	Frontier (%)	ave call spread	ave put spread	improved call spread	improved put spread	% spread reduced
AAPL	1120	902	40	1.61	2.05	0.69	1.1	51
AXP	616	605	49	0.31	0.26	0.17	0.15	44
BA	1208	1399	58	1.03	0.49	0.65	0.35	34
CAT	678	768	57	0.6	0.42	0.25	0.19	57
CSCO	728	783	54	1.34	0.48	0.28	0.14	77
CVX	456	537	59	0.48	0.58	0.23	0.24	56
DD	666	845	63	0.28	0.2	0.16	0.12	42
DIS	518	569	55	0.91	0.79	0.31	0.24	68
GS	826	848	51	0.92	0.65	0.54	0.3	46
HD	828	887	54	1.68	1.51	0.93	0.83	45
IBM	696	668	48	1.62	0.81	0.73	0.34	56
INTC	568	701	62	1.02	0.61	0.32	0.21	67
JNJ	672	779	58	0.99	1.16	0.39	0.38	64
JPM	724	835	58	0.73	0.75	0.26	0.25	66
KO	572	676	59	0.44	0.18	0.1	0.07	73
MCD	558	710	64	0.63	0.34	0.35	0.22	41
MMM	722	934	65	1.03	0.84	0.52	0.44	49
MRK	672	791	59	0.36	0.28	0.17	0.14	52
MSFT	1194	1003	42	2.19	0.97	1	0.47	53
NKE	788	874	55	1.78	0.63	0.34	0.14	80
PFE	480	612	64	0.51	0.32	0.11	0.09	76
PG	582	665	57	0.34	0.16	0.18	0.12	40
RTX	822	931	57	1.74	1.5	1.33	1.24	21
TRV	256	307	60	1.22	1.47	0.78	0.82	41
UNH	806	722	45	2.02	1.22	1.15	0.61	46
V	580	689	59	0.45	0.38	0.27	0.19	45
VZ	384	511	67	0.11	0.09	0.07	0.06	35
WBA	808	783	48	1.24	1.47	0.65	0.77	48
WMT	550	605	55	0.45	0.28	0.2	0.14	53
XOM	440	459	52	0.4	0.5	0.5	0.17	64
Total	20518	22398	56	0.95	0.71	0.44	0.35	52

Table B.4: Summary statistics (quoting with $L = 0$) of options on each stock in DJI by consolidating options related to the same underlying asset and expiration date.

APPENDIX C

Deferred Proofs and Additional Experiments for Chapter 5

C.1 Deferred Proofs

C.1.1 Proof of Theorem 5.1

The binary-search property implies that the nodes z included in the price calculation (lines 5 and 9) form the cover of I, so the algorithm correctly returns the price of I. The running time follows thanks to height balance, which implies the depth of the tree is $\mathcal{O}(\log n_{vals})$.

C.1.2 Proof of Theorem 5.3

Algorithm 8 Buy s shares of bundle security for an interval $I = [\alpha, 1)$.

```
1: Define subroutines:
       ResetInnerNode($z$): reset $h_z$ and $S_z$ based on the children of $z$ and the value $s_z$:
           $h_z \leftarrow 1 + \max\{h_{\mathit{left}(z)}, h_{\mathit{right}(z)}\}$, $S_z \leftarrow e^{s_z/b}(S_{\mathit{left}(z)} + S_{\mathit{right}(z)})$
       AddShares($z, s$): increase the number of shares held in $z$ by $s$:
           $s_z \leftarrow s_z + s$, $S_z \leftarrow e^{s/b} S_z$
2: procedure RotateLeft($z$):
3:     Let $z_1 = \mathit{left}(z)$, $z_{23} = \mathit{right}(z)$, $z_2 = \mathit{left}(z_{23})$, $z_3 = \mathit{right}(z_{23})$
4:     AddShares($z_2, s_{z_{23}}$), AddShares($z_3, s_{z_{23}}$), delete node $z_{23}$
5:     Let $z_{12}$ be a new node with:
           $\mathit{left}(z_{12}) = z_1$, $\mathit{right}(z_{12}) = z_2$, $I_{z_{12}} = I_{z_1} \cup I_{z_2}$, $s_{z_{12}} = 0$
6:     ResetInnerNode($z_{12}$)
7:     Update node $z$:
           $\mathit{left}(z) \leftarrow z_{12}$, $\mathit{right}(z) \leftarrow z_3$, ResetInnerNode($z$)
```

Proof of Lemma 5.2. We prove that the original partial normalization value of node z, S_z, is the same as the updated value, S'_z, after a left rotation. A right rotation follows symetrically.

$$\begin{aligned}
S_z &= e^{s_z/b} \cdot (S_{z_1} + S_{z_{23}}) \\
&= e^{s_z/b} \cdot \left(S_{z_1} + e^{s_{z_{23}}/b} \cdot (S_{z_2} + S_{z_3})\right) \\
&= e^{s_z/b} \cdot \left(S'_{z_1} + S'_{z_2} + S'_{z_3}\right) \\
&= e^{s_z/b} \cdot \left(e^{s'_{z_{12}}/b} \cdot \left(S'_{z_1} + S'_{z_2}\right) + S'_{z_3}\right) && \text{(since } s'_{z_{12}} = 0\text{)} \\
&= e^{s_z/b} \cdot \left(S'_{z_{12}} + S'_{z_3}\right) = S'_z
\end{aligned}$$

□

Proof of Theorem 5.3. The correctness of the **buy** operation follows because the shares are added to the nodes that form the cover of I (lines 5 and 12 in Algorithms 5), and the updates up the search path restore the properties of the LMSR tree (lines 13–17 in Algorithms 5). The running time follows from height balance, which implies that the length of the search path is $\mathcal{O}(\log n) = \mathcal{O}(\log n_{vals})$. □

C.1.3 Proof of Theorem 5.4

We first show that the constraints $\mathbf{A}^\top \boldsymbol{\mu} = \mathbf{0}$ imply that all levels $\ell = 0, 1, \ldots, K$ in $\boldsymbol{\mu}$ are mutually coherent. To do this, it suffices to show that all pairs of consecutive levels ℓ and $\ell+1$ are coherent, i.e., $\mu_y = \mu_{y_l} + \mu_{\mathrm{y}}$ for all $y \in \mathcal{Z}_\ell$ where we let $y_l = \mathit{left}(y)$ and $\mathrm{y} = \mathit{right}(y)$.

We proceed by induction, beginning with $\ell = K - 1$. In this base case, the constraint $\boldsymbol{a}_y^\top \boldsymbol{\mu} = 0$, expressed in Eq. (5.13), states that $b_K \mu_y = b_K \mu_{y_l} + b_K \mu_{\mathrm{y}}$, implying levels $K - 1$ and K are coherent.

Now assume that all the levels $k > \ell$ are mutually coherent. We aim to show that levels l and $l + 1$ are coherent. Pick any $y \in \mathcal{Z}_\ell$. Then the constraint $\boldsymbol{a}_y^\top \boldsymbol{\mu} = 0$, expressed in Eq. (5.13), implies that

$$\begin{aligned}
\left(\sum_{k>\ell} b_k\right) \mu_y &= \sum_{k>\ell} b_k \sum_{z \in \mathcal{Z}_k \colon z \subset y} \mu_z \\
&= \sum_{k>\ell} b_k \left(\sum_{z \in \mathcal{Z}_k \colon z \subseteq y_l} \mu_z + \sum_{z \in \mathcal{Z}_k \colon z \subseteq \mathrm{y}} \mu_z \right) \\
&= \sum_{k>\ell} b_k \left(\mu_{y_l} + \mu_{\mathrm{y}}\right). && \text{(C.1)}
\end{aligned}$$

Eq. (C.1) follows because y_l and y are in level $\ell + 1$, which is coherent with all levels $k \geq \ell + 1$ by the inductive assumption. Thus, we obtain that $\mu_y = \mu_{y_l} + \mu_{\mathrm{y}}$ for all $y \in \mathcal{Z}_\ell$, establishing the coherence between levels ℓ and $\ell + 1$ and completing the induction.

To finish the proof, we note that the LCMM prices at level K are determined by C_K, so they describe a probability distribution over Ω. Since $\mathbf{A}^\top \boldsymbol{p}(\boldsymbol{\theta}) = \mathbf{0}$, all the levels in $\boldsymbol{p}(\boldsymbol{\theta})$ are coherent with level K, which means that they correspond to the expectation of $\boldsymbol{\phi}$ under the probability distribution described by the prices at level K. Thus, $\boldsymbol{p}(\boldsymbol{\theta})$ is a coherent price vector and the multi-resolution LCMM is therefore arbitrage-free.

C.1.4 Proof of Theorem 5.5

The worst-case loss of an LCMM is bounded by the sum of the worst-case losses of the component markets C_k (Dudík, Lahaie, and Pennock 2012). In our case, these are LMSR submarkets with losses bounded by $b_k \log |\mathcal{Z}_k|$, so the worst-case loss of the resulting LCMM is at most

$$\sum_{k=1}^{K} b_k \log(2^k) = \sum_{k=1}^{K} b_k (k \log 2) \leq B^* \log 2,$$

proving the theorem.

C.1.5 Proof of Theorem 5.4.2

Algorithm 6 returns the correct price of I, because prices are coherent among submarkets and the nodes included in price calculations form a cover of I.

C.1.6 Proof of Theorem 5.6 and Additional Deferred Material from Section 5.4.3

We begin by deriving an identity that will be useful in the following analysis. For this derivation, let C be an LMSR with the liquidity parameter b, defined over an outcome space Ω. We will derive a relationship between the price vector in a state $\boldsymbol{\theta}$ and the price vector in a new state $\boldsymbol{\theta}' = \boldsymbol{\theta} + \boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is any bundle restricted to securities in E, i.e., $\delta_\omega = 0$ for $\omega \notin E$. Denoting $\boldsymbol{\mu} = \boldsymbol{p}(\boldsymbol{\theta})$, $\mu_E = p_E(\boldsymbol{\theta})$, and

$\boldsymbol{\mu}' = \boldsymbol{p}(\boldsymbol{\theta}')$, we have

$$\begin{aligned}\mu'_\omega &= \frac{e^{\theta_\omega/b}e^{\delta_\omega/b}}{\sum_{\nu\notin E} e^{\theta_\nu/b} + \sum_{\nu\in E} e^{\theta_\nu/b}e^{\delta_\nu/b}} \\ &= \frac{\mu_\omega e^{\delta_\omega/b}}{1-\mu_E + \sum_{\nu\in E}\mu_\nu e^{\delta_\nu/b}}, \end{aligned} \tag{C.2}$$

where Eq. (C.2) follows by dividing the numerator as well as denominator by $\sum_{\nu\in\Omega} e^{\theta_\nu/b}$.

We next establish correctness of the arbitrage removal procedure from Algorithm 7. The following lemma provides a critical step:

Lemma C.1. *Fix a level $\ell < K$. Let $\tilde{\boldsymbol{\theta}}$ be a market state in $\tilde{C}$ such that the associated prices, $\boldsymbol{\mu} = \tilde{\boldsymbol{p}}(\tilde{\boldsymbol{\theta}})$, are coherent among all levels $k > \ell$. Then, for any $t \in \mathbb{R}$ and any node y with level$(y) \le \ell$, the prices after buying t shares of $\boldsymbol{a}_y$, i.e., $\boldsymbol{\mu}' = \tilde{\boldsymbol{p}}(\tilde{\boldsymbol{\theta}} + t\boldsymbol{a}_y)$, remain coherent among all levels $k > \ell$.*

To use Lemma C.1 for arbitrage removal, we start with a market state $\tilde{\boldsymbol{\theta}}$ where all levels are coherent. When a trader buys some shares of a security ϕ_y, the level $\ell = \mathit{level}(y)$ loses coherence with other levels. By buying a certain number of shares of $\boldsymbol{a}_y$, it is possible to restore coherence between ℓ and $\ell + 1$, and Lemma C.1 then implies that coherence with all further levels $k > \ell + 1$ is also restored. The process of restoring coherence now continues with the parent of y and the bundle $\boldsymbol{a}_{par(y)}$ as implemented in Algorithm 7.

Proof. Consider two arbitrary levels k and m with $\ell < k < m$. Since prices are coherent between levels k and m before buying t shares of $\boldsymbol{a}_y$, we have, for any $z \in \mathcal{Z}_k$,

$$\mu_z = \sum_{u\in\mathcal{Z}_m:\, u\subset z} \mu_u. \tag{C.3}$$

Let π_y denote the price of ϕ_y according to the securities in $\mathcal{Z}_k$ and $\mathcal{Z}_m$, that is, $\pi_y = \sum_{z\in\mathcal{Z}_k:\, z\subset y} \mu_z = \sum_{u\in\mathcal{Z}_m:\, u\subset y} \mu_u$. Note that π_y might differ from μ_y, because level ℓ is not necessarily coherent with levels k and m. Let $\tilde{\boldsymbol{\theta}}' = \tilde{\boldsymbol{\theta}} + t\boldsymbol{a}_y$. From the definition of matrix $\mathbf{A}$, the updated $\tilde{\theta}'_z$ and $\tilde{\theta}'_u$ for any $z \in \mathcal{Z}_k$ and $u \in \mathcal{Z}_m$ are

$$\tilde{\theta}'_z = \begin{cases} \tilde{\theta}_z - tb_k & \text{if } z \subset y, \\ \tilde{\theta}_z & \text{otherwise,} \end{cases} \qquad \tilde{\theta}'_u = \begin{cases} \tilde{\theta}_u - tb_m & \text{if } u \subset y, \\ \tilde{\theta}_u & \text{otherwise.} \end{cases}$$

We calculate the new price μ'_z of any node $z \in \mathcal{Z}_k$ and show it equals to the price derived from its descendants $u \in \mathcal{Z}_m$. First, if $z \subset y$, then by Eq. (C.2) and Eq. (C.3),

$$\mu'_z = \frac{\mu_z e^{-t}}{\pi_y e^{-t} + 1 - \pi_y} = \frac{\sum_{u \in \mathcal{Z}_m \colon u \subset z} \mu_u e^{-t}}{\pi_y e^{-t} + 1 - \pi_y} = \sum_{u \in \mathcal{Z}_m \colon u \subset z} \mu'_u.$$

If $z \not\subset y$, then we similarly have

$$\mu'_z = \frac{\mu_z}{\pi_y e^{-t} + 1 - \pi_y} = \frac{\sum_{u \in \mathcal{Z}_m \colon u \subset z} \mu_u}{\pi_y e^{-t} + 1 - \pi_y} = \sum_{u \in \mathcal{Z}_m \colon u \subset z} \mu'_u.$$

Thus, prices remain coherent among all levels $m > k > \ell$. □

Building upon Lemma C.1, the following lemma provides the precise trade required to restore coherence after an update.

Lemma C.2. *Fix a level $\ell < K$ and a node $y \in \mathcal{Z}_\ell$ and let $y_l =$ left(y) and $\mathrm{y} =$ right(y). Let $\tilde{\boldsymbol{\theta}}^0$ and $\tilde{\boldsymbol{\theta}} = \tilde{\boldsymbol{\theta}}^0 + \boldsymbol{\delta}$ be market states in $\tilde{C}$, with associated prices $\boldsymbol{\mu}^0 = \tilde{\boldsymbol{p}}(\tilde{\boldsymbol{\theta}}^0)$ and $\boldsymbol{\mu} = \tilde{\boldsymbol{p}}(\tilde{\boldsymbol{\theta}})$ such that:*

- *prices $\boldsymbol{\mu}^0$ are coherent among all levels $k \geq \ell$;*
- *$\boldsymbol{\delta}$ is a vector, which is zero outside descendants of y, i.e., $\delta_z = 0$ whenever $z \not\subseteq y$;*
- *prices $\boldsymbol{\mu}$ are coherent among all levels $k > \ell$.*

Let $\tilde{\boldsymbol{\theta}}' = \tilde{\boldsymbol{\theta}} + t\boldsymbol{a}_y$ where

$$t = \frac{b_\ell}{B_{\ell-1}} \log\left(\frac{1-\mu_y}{\mu_y} \cdot \frac{\mu_{y_l} + \mu_{\mathrm{y}}}{1 - \mu_{y_l} - \mu_{\mathrm{y}}}\right).$$

Then the associated prices $\boldsymbol{\mu}' = \tilde{\boldsymbol{p}}(\tilde{\boldsymbol{\theta}}')$ are coherent among all levels $k \geq \ell$.

Proof. By Lemma C.1, adding $t\boldsymbol{a}_y$ to $\tilde{\boldsymbol{\theta}}$ maintains coherence among levels $k > \ell$, so it suffices to show that levels ℓ and $\ell + 1$ are mutually coherent in $\boldsymbol{\mu}'$. Thus, we have to show that $\mu'_z = \mu'_{\mathit{left}(z)} + \mu'_{\mathit{right}(z)}$ for all $z \in \mathcal{Z}_\ell$.

First note that by the assumption on $\boldsymbol{\delta}$ and the definition of $\boldsymbol{a}_y$, we have

$$\begin{aligned} \tilde{\theta}^0_z = \tilde{\theta}_z = \tilde{\theta}'_z \quad & \text{for all } z \in \mathcal{Z}_\ell \setminus \{y\} \\ \tilde{\theta}^0_u = \tilde{\theta}_u = \tilde{\theta}'_u \quad & \text{for all } u \in \mathcal{Z}_{\ell+1} \setminus \{y_l, \mathrm{y}\}. \end{aligned}$$

Therefore, by Eq. (C.2), we have for all $z \in \mathcal{Z}_\ell \backslash \{y\}$

$$\frac{\mu'_z}{1-\mu'_y} = \frac{\mu^0_z}{1-\mu^0_y}, \qquad \text{and} \qquad \frac{\mu'_{\mathit{left}(z)} + \mu'_{\mathit{right}(z)}}{1-\mu'_{y_l}-\mu'_{\mathrm{y}}} = \frac{\mu^0_{\mathit{left}(z)} + \mu^0_{\mathit{right}(z)}}{1-\mu^0_{y_l}-\mu^0_{\mathrm{y}}}. \tag{C.4}$$

Since the vector $\boldsymbol{\mu}^0$ satisfies $\mu^0_z = \mu^0_{\mathit{left}(z)} + \mu^0_{\mathit{right}(z)}$ for all $z \in \mathcal{Z}_\ell \backslash \{y\}$, Eq. (C.4) implies that we also have $\mu'_z = \mu'_{\mathit{left}(z)} + \mu'_{\mathit{right}(z)}$ for all $z \in \mathcal{Z}_\ell \backslash \{y\}$ as long as $\mu'_y = \mu'_{y_l} + \mu'_{\mathrm{y}}$. Thus, in order to show that levels ℓ and $\ell+1$ are coherent in $\boldsymbol{\mu}'$, it suffices to show that $\mu'_y = \mu'_{y_l} + \mu'_{\mathrm{y}}$.

We begin by explicitly calculating $\tilde{\theta}'_z$ and $\tilde{\theta}'_u$ for any $z \in \mathcal{Z}_\ell$ and any $u \in \mathcal{Z}_{\ell+1}$:

$$\tilde{\theta}'_z = \begin{cases} \tilde{\theta}_z + tB_\ell & \text{if } z = y, \\ \tilde{\theta}_z & \text{otherwise,} \end{cases} \qquad \tilde{\theta}'_u = \begin{cases} \tilde{\theta}_u - tb_{\ell+1} & \text{if } u \in \{y_l, \mathrm{y}\}, \\ \tilde{\theta}_u & \text{otherwise.} \end{cases}$$

Therefore,

$$\mu'_y = \frac{\mu_y e^{tB_\ell/b_\ell}}{\mu_y e^{tB_\ell/b_\ell} + 1 - \mu_y} = \frac{1}{1 + \frac{1-\mu_y}{\mu_y} e^{-tB_\ell/b_\ell}}$$

and similarly,

$$\mu'_{y_l} + \mu'_{\mathrm{y}} = \frac{(\mu_{y_l} + \mu_{\mathrm{y}})e^{-t}}{(\mu_{y_l} + \mu_{\mathrm{y}})e^{-t} + 1 - \mu_{y_l} - \mu_{\mathrm{y}}} = \frac{1}{1 + \frac{1-\mu_{y_l}-\mu_{\mathrm{y}}}{\mu_{y_l}+\mu_{\mathrm{y}}} e^t}.$$

Thus, it remains to show that

$$\tfrac{1-\mu_y}{\mu_y} e^{-tB_\ell/b_\ell} = \tfrac{1-\mu_{y_l}-\mu_{\mathrm{y}}}{\mu_{y_l}+\mu_{\mathrm{y}}} e^t,$$

or equivalently:

$$\tfrac{1-\mu_y}{\mu_y} \cdot \tfrac{\mu_{y_l}+\mu_{\mathrm{y}}}{1-\mu_{y_l}-\mu_{\mathrm{y}}} = e^{t(1+B_\ell/b_\ell)}.$$

But this follows from our choice of t and the fact that $B_{\ell-1} = B_\ell + b_\ell$, completing the proof. □

We finish the section with the proof of Theorem 5.6.

Proof of Theorem 5.6. Algorithm 7 correctly updates the tree (and returns the cost), because the shares are added to the nodes that form a cover of I, and coherence is then restored by applying Lemma C.2 up the search path. Running times of both algorithms are proportional to the length of the search path to the first node z with

$\alpha_z = \alpha$, whose level coincides with the precision of α. □

C.2 Trading Dynamics and Additional Results

C.2.1 Detailed Trading Dynamics

We simulate a market consisting of ten traders. The outcome space is $[0, 1)$, discretized at the precision $K = 10$. Traders, indexed as $i \in \{1, \dots, 10\}$, have noisy access to the underlying true signal $p = 0.4$. Trader i's belief takes form of a beta distribution $\text{Beta}(a_i, b_i)$ with $a_i \sim \text{Binomial}(p, n_i)$, $b_i = n_i - a_i$, and $n_i = 16i$ representing the quality of the agent's observation of the signal p. Each trader i has an exponential utility $u_i(W) = -e^{-W}$, where W is the trader's wealth. We consider budget-limited cost-based market makers, whose worst-case loss may not exceed a budget constraint B. For LMSR at precision k, this means setting the liquidity parameter to $b = B/\log(2^k)$. In our experiments, we consider two LMSR markets at precision levels 4 and 8, denoted as $\texttt{LMSR}_{k=4}$ and $\texttt{LMSR}_{k=8}$. On the other hand, a multi-resolution LCMM has an infinite number of choices for its liquidity at each precision level. To showcase its interpolation ability, we consider LCMM that evenly splits its budget to precision levels 4 and 8, and denote it as $\texttt{LCMM}_{50/50}$.

Each market starts with the uniform prior, i.e., the initial market prices for all outcomes are equal. In each time step, a uniformly random agent is picked to trade. The selected agent considers a set of 50 interval securities, with endpoints randomly sampled according to the agent's belief. The candidate intervals are rounded to the precision of the corresponding market.[1] The agent considers trading the expected-utility-optimizing number of shares for each interval, and ultimately picks the best interval and executes the trade. The market maker updates prices accordingly, until the market equilibrium is reached (no trader in the market has the incentive to trade).

Following the described protocol, we run markets mediated by the three respective market makers, $\texttt{LMSR}_{k=4}$, $\texttt{LMSR}_{k=8}$, and $\texttt{LCMM}_{50/50}$, over a range of budget constraints. To decrease variance, we generate 40 controlled simulation traces (described by a sequence of agent arrivals and their draws of the candidate intervals) and run the market makers on those same traces. Therefore, any change in agent behavior and

[1]As the number of available interval securities grows exponentially as the supported precision increases, we assume agents have a computational limit and can only consider a (sub)set of available securities.

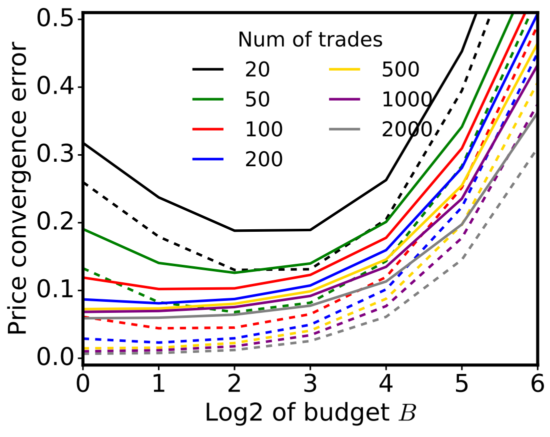
price convergence is caused by the different cost functions that market makers adopt to aggregate trades.

C.2.2 Additional Experiments

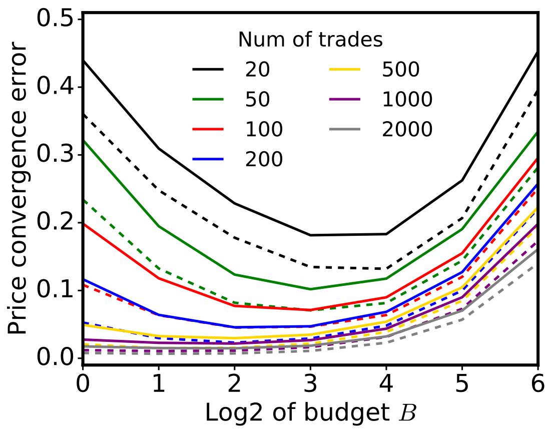
In Section 5.5, we demonstrated that by splitting the budget between submarkets that offer interval securities at different precisions, the multi-resolution LCMM is able to interpolate the performance of LMSR market makers. It can aggregate information at the coarser level efficiently, while achieving accurate belief elicitation at the finer resolution, given enough trading period. Here we provide numerical results over a wider range of budget constraints, validating how the multi-resolution LCMM may balance the price convergence behavior of LMSR markets.

Fig. C.1 shows the price convergence error as a function of budget constraint (thus, the liquidity parameter) and the number of trades for the three respective market makers. Results are averaged over forty random but controlled trading sequences. The solid lines depict the price convergence error at precision level $k = 8$, and the dashed ones for precision level $k = 4$. The minimum point on each curve indicates the optimal budget, or the optimal value of the liquidity parameter to adopt, for the particular cost function and a specific number of trades.

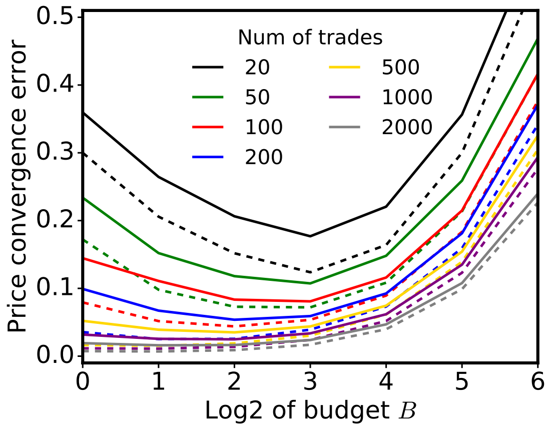
Intuitively, when the budget for running a market is sufficient, a market operator can support interval securities at any fine-grained precision level, or use only a portion of the budget to achieve optimal performance. However, when the budget for running a market is limited, say B less than 8, the market designer can preferably aggregate information faster at a coarser resolution by limiting the precision of interval endpoints (e.g., adopting $\texttt{LMSR}_{k=4}$). However, by design, it can not accurately elicit beliefs at finer resolutions, even when the market is run for a sufficiently long period of time. The $\texttt{LMSR}_{k=8}$, on the other hand, benefits from a larger number of trades to aggregate more fine-grained information. Running the two LMSR markets independently may balance this convergence trade-off, but inevitably results in inconsistent prices between the markets. Given the different convergence properties of separate LMSRs, a multi-resolution LCMM can allocate its budget accordingly to achieve a desired convergence performance, while maintaining coherent prices. For example, a market designer, who considers information at precision levels $k = 4$ and $k = 8$ equally important, may divide the budget between the two levels to enjoy faster price convergence at the coarser resolution, while accurately aggregating a full probability distribution of the continuous variable as trading proceeds.



(a) $\texttt{LMSR}_{k=4}$.



(b) $\texttt{LMSR}_{k=8}$.



(c) $\texttt{LCMM}_{50/50}$.

Figure C.1: The price convergence error as a function of liquidity and the number of trades (indicated by the color of the line) for the three respective market makers. Solid lines record price convergence error at the finer precision level $k = 8$, and dashed ones at the coarser level $k = 4$.

BIBLIOGRAPHY

BIBLIOGRAPHY

Abernethy, Jacob, Yiling Chen, and Jennifer Wortman Vaughan (2011). “An optimization-based framework for automated market-making”. In: *12th ACM Conference on Electronic Commerce.*

Abernethy, Jacob, Sindhu Kutty, Sébastien Lahaie, and Rahul Sami (2014). “Information aggregation in exponential family markets”. In: *15th ACM Conference on Economics and Computation*, pp. 395–412.

Adel′son-Vel′skiĭ, G. M. and E. M. Landis (1962). “An algorithm for the organization of information”. In: *Soviet Mathematics—Doklady* 3, pp. 1259–1263.

Aldrich, Eric M., Joseph Grundfest, and Gregory Laughlin (2017). “The Flash Crash: A new deconstruction”. In: *Available at SSRN 2721922.*

Allen, Franklin and Douglas Gale (1992). “Stock price manipulation”. In: *The Review of Financial Studies* 5.3, pp. 503–529.

Benisch, Michael, Norman Sadeh, and Tuomas Sandholm (2008). “A Theory of expressiveness in mechanisms”. In: *Proceedings of the 23rd National Conference on Artificial Intelligence*, 17–23.

Biais, Bruno and Paul Woolley (2012). *High Frequency Trading.* Tech. rep. Toulouse University.

Biggio, Battista, Giorgio Fumera, and Fabio Roli (2014). “Pattern Recognition Systems Under Attack: Design Issues and Research Challenges”. In: *International Journal of Pattern Recognition and Artificial Intelligence* 28.07, p. 1460002.

Bookstaber, Richard (2012). *Using agent-based models for analyzing threats to financial stability.* Working Paper. Office of Financial Research.

Bousmalis, Konstantinos et al. (2018). “Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping”. In: *IEEE International Conference on Robotics and Automation.*

Cao, Melanie and Jason Wei (2010). “Option market liquidity: Commonality and other characteristics”. In: *Journal of Financial Markets* 13.1, pp. 20 –48.

Cassell, Ben-Alexander and Michael P. Wellman (2013). "EGTAOnline: An Experiment Manager for Simulation-Based Game Studies". In: *Multi-Agent-Based Simulation XIII.* Ed. by Francesca Giardini and Frédéric Amblard. Springer Berlin Heidelberg, pp. 85–100.

Chakraborty, Mithun et al. (2013). "Instructor rating markets". In: *27th AAAI Conference on Artificial Intelligence*, pp. 159–165.

Chakraborty, Tanmoy and Michael Kearns (2011). "Market making and mean reversion". In: *12th ACM Conference on Electronic Commerce*, pp. 307–314.

Chen, Yiling, Lance Fortnow, Evdokia Nikolova, and David M. Pennock (2007). "Betting on Permutations". In: *8th ACM Conference on Electronic Commerce*, 326–335.

Chen, Yiling, Sharad Goel, and David M. Pennock (2008). "Pricing combinatorial markets for tournaments". In: *40th Annual ACM Symposium on Theory of Computing*, 305–314.

Chen, Yiling and David M. Pennock (2007). "A utility framework for bounded-loss market makers". In: *23rd Conference on Uncertainty in Artificial Intelligence*, pp. 49–56.

Chen, Yiling et al. (2007). "Bluffing and Strategic Reticence in Prediction Markets". In: *3rd International Conference on Internet and Network Economics*, 70–81.

Chen, Yiling et al. (2008). "Complexity of combinatorial market makers". In: *9th ACM Conference on Electronic Commerce*, 190–199.

Cho, Kyunghyun et al. (2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734.

Cliff, Dave (1997). *Minimal-intelligence agents for bargaining behaviors in market-based environments.* Tech. rep. Hewlett-Packard Labs.

— (2009). "ZIP60: Further explorations in the evolutionary design of trader agents and online auction-market mechanisms". In: *IEEE Transactions on Evolutionary Computation* 13.1, pp. 3–18.

Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest (1999). *Introduction to Algorithms.* The MIT Press.

Dudík, Miroslav, Sébastien Lahaie, and David M. Pennock (2012). "A tractable combinatorial market maker using constraint generation". In: *13th ACM Conference on Electronic Commerce.*

Dudík, Miroslav, Sébastien Lahaie, David M. Pennock, and David Rothschild (2013). "A combinatorial prediction market for the U.S. elections". In: *14th ACM Conference on Electronic Commerce.*

Dudík, Miroslav, Sébastien Lahaie, Ryan M Rogers, and Jennifer Wortman Vaughan (2017). "A decomposition of forecast error in prediction markets". In: *Advances in Neural Information Processing Systems*, pp. 4371–4380.

Dudík, Miroslav, Xintong Wang, David M. Pennock, and David M. Rothschild (2020). "Log-time Prediction Markets for Interval Securities". In: *20th International Conference on Autonomous Agents and Multiagent Systems, to appear.*

Farmer, J. Doyne, Paolo Patelli, and Ilija I. Zovko (2005). "The predictive power of zero intelligence in financial markets". In: *Proceedings of the National Academy of Sciences* 102.6, pp. 2254–2259.

Fishman, Michael J. and Kathleen M. Hagerty (1995). "The mandatory disclosure of trades and market liquidity". In: *The Review of Financial Studies* 8.3, pp. 637–676.

Fortnow, Lance, Joe Kilian, David M. Pennock, and Michael P. Wellman (2005). "Betting Boolean-style: A framework for trading in securities based on logical formulas". In: *Decision Support Systems* 39.1, 87–104.

Foucault, Thierry, Ailsa Röell, and Patrik Sandås (2003). "Market making with costly monitoring: An analysis of the SOES controversy". In: *The Review of Financial Studies* 16.2, pp. 345–384.

Friedman, Daniel (1993). "The double auction market institution: A survey". In: *The Double Auction Market: Institutions, Theories, and Evidence.* Addison-Wesley, pp. 3–25.

Gao, Xi, Yiling Chen, and David M. Pennock (2009). "Betting on the Real Line". In: *5th Workshop on Internet and Network Economics*, pp. 553–560.

Gjerstad, Steven (2007). "The competitive market paradox". In: *Journal of Economic Dynamics and Control* 31, pp. 1753–1780.

Gjerstad, Steven and John Dickhaut (1998). "Price formation in double auctions". In: 22 (1), pp. 1–29.

Gode, Dhananjay K. and Shyam Sunder (1993). "Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality". In: *Journal of Political Economy*, pp. 119–137.

Golovin, Daniel (2007). "More expressive market models and the future of combinatorial auctions". In: *SIGecom Exchanges* 7.1, 55–57.

Goodfellow, Ian, Jonathon Shlens, and Christian Szegedy (2015). “Explaining and Harnessing Adversarial Examples”. In: *International Conference on Learning Representations.*

Goodfellow, Ian et al. (2014). “Generative adversarial nets”. In: *27th International Conference on Neural Information Processing Systems*, pp. 2672–2680.

Guo, Mingyu and David M. Pennock (2009). “Combinatorial prediction markets for event hierarchies”. In: *Proceedings of the 8th International Conference on Autonomous Agents and Multi-agent Systems*, 201–208.

Hanson, Robin (2003). “Combinatorial information market design”. In: *Information Systems Frontiers* 5.1, pp. 107–119.

— (2007). “Logarithmic Market Scoring Rules for Modular Combinatorial Information Aggregation”. In: *Journal of Prediction Markets* 1.1, pp. 1–15.

Hanson, Robin D. (1999). “Decision markets”. In: *IEEE Intelligent Systems* 14.3, pp. 16–19.

Hautsch, Nikolaus and Ruihong Huang (2012). “Limit Order Flow, Market Impact, and Optimal Order Sizes: Evidence from NASDAQ TotalView-ITCH Data”. In: *Market Microstructure: Confronting Many Viewpoints.* Ed. by Frederic Abergel et al. Wiley.

Herzel, Stefano (2005). “Arbitrage opportunities on derivatives: A linear programming approach”. In: *Dynamics of Continuous, Discrete and Impulsive Systems Series B: Application and Algorithms.*

Hope, Bradley (2015a). “How ‘Spoofing’ traders dupe markets”. In: *Wall Street Journal.*

— (2015b). “Was ‘John Doe’ manipulating Treasury futures? New lawsuit says yes”. In: *Wall Street Journal.*

Huang, Ling et al. (2011). “Adversarial machine learning”. In: *4th ACM Workshop on Security and Artificial Intelligence*, pp. 43–58.

Kirilenko, Andrei A., Albert S. Kyle, Mehrdad Samadi, and Tugkan Tuzun (2017). “The Flash Crash: High frequency trading in an electronic market”. In: *Journal of Finance* 72.3, pp. 967–998.

Knuth, Donald E. (1998). *The Art of Computer Programming, Volume 3: Sorting and Searching.* Addison Wesley.

Kroer, Christian, Miroslav Dudík, Sébastien Lahaie, and Sivaraman Balakrishnan (2016). “Arbitrage-free combinatorial market making via integer programming”. In: *17th ACM Conference on Electronic Commerce.*

Laskey, Kathryn Blackmond et al. (2018). "Graphical model market maker for combinatorial prediction markets". In: *Journal of Artificial Intelligence Research* 63, pp. 421–460.

Leal, Sandrine Jacob and Mauro Napoletano (2019). "Market stability vs. market resilience: Regulatory policies experiments in an agent-based model with low- and high-frequency trading". In: *Journal of Economic Behavior and Organization* 157, pp. 15–41.

Lebaron, Blake (2006). "Agent-based computational finance". In: *Handbook of Computational Economics*. Ed. by Leigh Tesfatsion and Kenneth L. Judd. 1st ed. Vol. 2. Elsevier. Chap. 24, pp. 1187–1233.

LeBaron, Blake, W. Brian Arthur, and Richard Palmer (1999). "Time series properties of an artificial stock market". In: *Journal of Economic Dynamics and Control* 23.9, pp. 1487–1516.

Lee, Eun Jung, Kyong Shik Eom, and Kyung Suh Park (2013). "Microstructure-based manipulation: Strategic behavior and performance of spoofing traders". In: *Journal of Financial Markets* 16.2, pp. 227–252.

Levens, Tara E. (2015). "Too Fast, Too Frequent? High-Frequency Trading and Securities Class Actions". In: *University of Chicago Law Review* 82.3, pp. 1511–1557.

Lewis, Michael (2014). *Flash Boys: A Wall Street Revolt*. W. W. Norton & Company.

Li, Junyi et al. (2020). "Generating realistic stock market order streams". In: *34th AAAI Conference on Artificial Intelligence*, pp. 727–734.

Lin, Tom C. W. (2015). "The new market manipulation". In: *Emory Law Journal* 66, pp. 1253–1314.

Logeswaran, Lajanugen, Honglak Lee, and Samy Bengio (2018). "Content Preserving Text Generation with Attribute Controls". In: *32nd International Conference on Neural Information Processing Systems*, pp. 5108–5118.

Martínez-Miranda, Enrique, Peter McBurney, and Matthew Howard (2016). "Learning unfair trading: A market manipulation analysis from the reinforcement learning perspective". In: *IEEE International Conference on Evolving and Adaptive Intelligent Systems*, pp. 103–109.

Merton, Robert C. (1973a). "Theory of rational option pricing". In: *The Bell Journal of Economics and Management Science* 4.1, pp. 141–183.

— (1973b). "Theory of Rational Option Pricing". In: *The Bell Journal of Economics and Management Science* 4.1, pp. 141–183. ISSN: 00058556. URL: `http://www.jstor.org/stable/3003143`.

Mishra, Pushkar (2016). “On Updating and Querying Sub-arrays of Multidimensional Arrays”. In: *CoRR* abs/1311.6093.

Modigliani, Franco and Merton H. Miller (1958). “The cost of capital, corporation finance and the theory of investment”. In: *The American Economic Review* 48.3, pp. 261–297.

Montgomery, John D. (2016). “Spoofing, market manipulation, and the limit-order book”. In: *Available at SSRN 2780579.* URL: `http://ssrn.com/abstract=2780579`.

Othman, Abraham, David M. Pennock, Daniel M. Reeves, and Tuomas Sandholm (2013). “A practical liquidity-sensitive automated market maker”. In: *ACM Transactions on Economics and Computation* 1.3, 14:1–14:25.

Othman, Abraham and Tuomas Sandholm (2010). “Automated market-making in the large: The Gates Hillman Prediction Market”. In: *11th ACM Conference on Electronic Commerce*, pp. 367–376.

— (2012). “Automated market makers that enable new settings: Extending constant-utility cost functions”. In: *Auctions, Market Mechanisms, and Their Applications*, pp. 19–30.

Paddrik, Mark et al. (2012). “An agent based model of the E-Mini S&P 500 applied to Flash Crash analysis”. In: *IEEE Conference on Computational Intelligence for Financial Engineering and Economics*, pp. 1–8.

Palit, Imon, Steve Phelps, and Wing Lon Ng (2012). “Can a zero-intelligence plus model explain the stylized facts of financial time series data?” In: *11th International Conference on Autonomous Agents and Multiagent Systems*, pp. 653–660.

Patterson, Scott and Jamila Trindle (2013). “CFTC charges high-speed trader under new powers”. In: *Wall Street Journal.*

Plott, Charles R. and Kay-Yut Chen (2002). “Information aggregation mechanisms: Concept, design and implementation for a sales forecasting problem”. Working paper No. 1131, California Institute of Technology.

Prewit, Matt (2012). “High-frequency trading: Should regulators do more”. In: *Michigan Telecommunications and Technology Law Review* 19, pp. 131–161.

Rundle, James (2019). “Nasdaq deploys AI to detect stock-market abuse”. In: *Wall Street Journal.*

Sandholm, Tuomas (2007). “Expressive commerce and its application to sourcing: How we conducted $35 billion of generalized combinatorial auctions”. In: *AI Magazine* 28.3.

Schwartz, Robert A. and Lin Peng (2013). “Market Makers”. In: *Encyclopedia of Finance*. Ed. by Cheng-Few Lee and Alice C. Lee. Boston, MA: Springer US, pp. 487–489.

Shrivastava, Ashish et al. (2017). “Learning from Simulated and Unsupervised Images through Adversarial Training”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2242–2251.

Singh, Charan (2019). “How often do options get exercised early?” In: URL: `https://www.optionsanimal.com/how-often-do-options-get-exercised-early`.

Sinha, Aman, Hongseok Namkoong, and John Duchi (2018). “Certifiable Distributional Robustness with Principled Adversarial Training”. In: *International Conference on Learning Representations*.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). “Sequence to Sequence Learning with Neural Networks”. In: *27th International Conference on Neural Information Processing Systems*, pp. 3104–3112.

Tao, Xuan, Andrew Day, Lan Ling, and Samuel Drapeau (2020). *On Detecting Spoofing Strategies in High Frequency Trading*. arXiv: `2009.14818`.

Tesauro, Gerald and Jonathan L. Bredin (2002). “Strategic sequential bidding in auctions using dynamic programming”. In: *First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2*, pp. 591–598.

Tesauro, Gerald and Rajarshi Das (2001). “High-performance bidding agents for the continuous double auction”. In: *3rd ACM Conference on Electronic Commerce*, pp. 206–209.

Tzeng, Eric, Judy Hoffman, Kate Saenko, and Trevor Darrell (2017). “Adversarial Discriminative Domain Adaptation”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2962–2971.

U.S. SEC (2017). *Division of Enforcement Annual Report*.

— (2018). *Division of Enforcement Annual Report*.

— (2019). *Division of Enforcement Annual Report*.

Varian, Hal R. (1987). “The arbitrage principle in financial economics”. In: *The Journal of Economic Perspectives* 1.2, pp. 55–72.

Volpi, Riccardo et al. (2018). “Generalizing to Unseen Domains via Adversarial Data Augmentation”. In: *32nd International Conference on Neural Information Processing Systems*, pp. 5339–5349.

Vorobeychik, Yevgeniy, Christopher Kiekintveld, and Michael P. Wellman (2006). "Empirical mechanism design: Methods, with application to a supply-chain scenario". In: *7th ACM Conference on Electronic Commerce*, pp. 306–315.

Vytelingum, Perukrishnen, Dave Cliff, and Nicholas R. Jennings (2008). "Strategic bidding in continuous double auctions". In: *Artificial Intelligence* 172.14, pp. 1700–1729.

Wah, Elaine and Michael P. Wellman (2016). "Latency arbitrage in fragmented markets: A strategic agent-based analysis". In: *Algorithmic Finance* 5, pp. 69–93.

Wah, Elaine, Mason Wright, and Michael P. Wellman (2017). "Welfare Effects of Market Making in Continuous Double Auctions". In: *Journal of Artificial Intelligence Research* 59, pp. 613–650.

Wang, Xintong, Christopher Hoang, and Michael P. Wellman (2020). "Learning-Based Trading Strategies in the Face of Market Manipulation." In: *First ACM International Conference on AI in Finance.*

Wang, Xintong, Yevgeniy Vorobeychik, and Michael P. Wellman (2018). "A Cloaking Mechanism to Mitigate Market Manipulation". In: *27th International Joint Conference on Artificial Intelligence*, pp. 541–547.

Wang, Xintong and Michael P. Wellman (2017). "Spoofing the limit order book: An agent-based model". In: *16th International Conference on Autonomous Agents and Multiagent Systems*, pp. 651–659.

— (2020). "Market Manipulation: An Adversarial Learning Framework for Detection and Evasion". In: *29th International Joint Conference on Artificial Intelligence.* Special Track on AI in FinTech, pp. 4626–4632.

Wang, Xintong et al. (2020). "Designing a Combinatorial Financial Options Market". In: *Manuscript submitted.*

Wang, Yun-Yi (2019). "Strategic spoofing order trading by different types of investors in Taiwan Index futures market". In: *Journal of Financial Studies* 27.1, p. 65.

Wellman, Michael P. (2006). "Methods for empirical game-theoretic analysis (Extended abstract)". In: *21st National Conference on Artificial Intelligence*, pp. 1552–1555.

— (2011). *Trading Agents.* Morgan & Claypool.

— (2016). "Putting the agent in agent-based modeling". In: *Autonomous Agents and Multi-Agent Systems* 30.6, pp. 1175–1189.

Wiedenbeck, Bryce and Michael P. Wellman (2012). "Scaling simulation-based game analysis through deviation-preserving reduction". In: *11th International Conference on Autonomous Agents and Multiagent Systems*, pp. 931–938.

Wright, Mason and Michael P. Wellman (2018). “Evaluating the stability of non-adaptive trading in continuous double auctions”. In: *17th International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 614–622.

Xia, Lirong and David M. Pennock (2011). “An efficient Monte-Carlo algorithm for pricing combinatorial prediction markets for tournaments”. In: *22nd International Joint Conference on Artificial Intelligence*, pp. 452–457.